Università degli Studi di Udine

Dipartimento di Matematica, Informatica e Fisica

Corso di Dottorato in
Informatica e Scienze Matematiche e Fisiche

Ciclo XXXI

Model Checking: the Interval Way

Tesi di Dottorato

Relatore
prof. Angelo MONTANARI

Correlatore
prof. Adriano PERON

Dottorando
Alberto MOLINARI

Submission: October 31, 2018
Defense: February 28, 2019

Author's e-mail:
molinari.alberto@gmail.com

Author's address:
via del Prà delle Molle, 13
IT-31029 Vittorio Veneto (TV)
Italy

The **updated version** of this thesis can be found at:

http://albertom.altervista.org/Th.pdf.



© 2019 by Alberto Molinari

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. To view a copy of this license, please visit http://creativecommons.org/licenses/by-nc-sa/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

"Multa non quia difficilia sunt non audemus,
sed quia non audemus sunt difficilia."

L. A. Seneca, Epistulae morales ad Lucilium, 104, 26

Abstract

Formal methods are structured methodologies that support the development of critical systems—whose safety and reliability are fundamental requirements—with the aim of establishing system correctness with mathematical rigor, providing effective verification techniques and tools, and reducing verification time while simultaneously increasing coverage.

Model checking (MC) is a family of formal methods that have been accepted by industry and are becoming integral part of standards and of system development cycles. In MC, some properties of a finite-state transition system are expressed in suitable specification languages and then verified over a model of the system itself (usually a finite Kripke structure) through *exhaustive enumeration of all the reachable states*. This technique is *fully automatic* and every time the design violates a desired property, a *counterexample* is produced, which illustrates a behavior falsifying such a property: this is extremely useful for debugging.

The most famous MC techniques—just to mention a few, *partial order reduction*, *symbolic* and *bounded* MC—were developed starting from the late 80s, bearing in mind the well-known "point-based" temporal logics LTL and CTL. However, while the expressiveness of such logics is beyond doubt, there are some properties we may want to check that are inherently "interval-based" and thus cannot be expressed by point-based temporal logics, e.g., "the proposition p has to hold in at least an average number of system states in a given computation sector". Here *interval temporal logics* (ITLs) come into play, providing an alternative setting for reasoning about time. Such logics deal with intervals, instead of points, as their primitive entities: this feature gives them the ability of expressing temporal properties, such as actions with duration, accomplishments, and temporal aggregations, which cannot be dealt with in standard point-based logics.

The *Halpern and Shoham's modal logic of time intervals* (HS, for short) is one of the most famous ITLs: it features one modality for each of the 13 possible ordering relations between pairs of intervals, apart from equality. In this thesis we focus our attention on MC based on HS, in the role of property specification language, for which a little work has been done if compared to MC for point-based temporal logics. The idea is to evaluate HS formulas on finite Kripke structures, making it possible to check the correctness of the behavior of systems with respect to meaningful interval properties. To this end, we interpret each one of the (possibly infinitely many) finite paths of a Kripke structure as an interval, and we define its atomic properties on the basis of the properties of the states composing it, at first assuming the *homogeneity principle*: the latter enforces an atomic property to hold over an interval if and only if it holds over all its subintervals. We prove that MC for HS interpreted over finite Kripke structures is a *decidable* problem (whose computational complexity has a nonelementary upper bound), and then we show it to be **EXPSPACE**-hard.

Since the problem provably admits no polynomial-time decision procedure, we also focus on HS fragments, which feature considerably better complexities—from **EXPSPACE**, down to **PSPACE** and to low levels of the polynomial hierarchy—yet retaining the ability to capture meaningful interval properties of state transition systems. Several MC algorithms are presented, tailored to the specific fragments being considered, and founded on concepts and techniques different from each other.

Moreover, we study the expressive power of HS in MC, in comparison with that of the standard point-based logics LTL, CTL and CTL*, still under the homogeneity principle, which is then relaxed showing how this impacts on the complexity of MC for HS and its fragments, and on the expressiveness of the logic.

Finally, we consider a possible replacement of Kripke structures by a more expressive model, which allows us to directly describe systems in terms of their *interval-based behaviour and properties*, thus paving the way for a more general interval-based MC.

Keywords: Model checking, interval temporal logics, timelines, computational complexity

2010 MSC: 03B70, 68Q60

ACM classes: F.4.1, D.2.4

Sommario

I *metodi formali* sono metodologie strutturate che supportano lo sviluppo di sistemi critici—la sicurezza ed affidabilità dei quali sono requisiti fondamentali—allo scopo di dimostrare la correttezza di tali sistemi con rigore matematico, fornendo tecniche e strumenti di verifica efficaci, e riducendo il tempo del processo di verifica, aumentando contemporaneamente il grado di copertura.

Il *model checking* (MC) è una famiglia di metodi formali che sono stati accettati dal mondo dell'industria e stanno diventando parte integrante di standard e dei cicli di sviluppo dei sistemi. Nel contesto del MC, alcune proprietà di un sistema di transizione a stati finiti vengono espresse mediante linguaggi di specifica e, successivamente, queste sono verificate su un modello del sistema stesso (di solito una struttura di Kripke finita), tramite *l'enumerazione completa di tutti gli stati raggiungibili.* Questa tecnica è *totalmente automatica* ed ogni volta che viene violata una proprietà desiderata, viene fornito un *controesempio* che illustra un comportamento che falsifica tale proprietà: ciò è estremamente utile per il processo di debugging.

Le più famose tecniche di MC, come la *partial order reduction*, il *symbolic* ed il *bounded* MC, furono sviluppate verso la fine degli anni 80 prendendo in considerazione le famose logiche temporali LTL e CTL, che sono basate su punti. Tuttavia, nonostante l'indubbia espressività di tali logiche, esistono alcune proprietà che potremmo voler verificare che hanno inerentemente una semantica intervallare e quindi non possono essere espresse da logiche puntuali, per esempio: "la proposizione p deve valere in almeno un dato numero medio di stati del sistema, in un settore di computazione specifico". Le logiche temporali intervallari entrano in gioco in questi casi, permettendoci di ragionare su aspetti temporali in modo diverso: esse adottano gli intervalli, invece dei punti, come loro entità primitive. Questa caratteristica dà loro l'abilità di esprimere proprietà intervallari, come azioni con durata, conseguimenti di obiettivi e aggregazioni temporali, che non possono essere trattate nelle logiche puntuali standard.

La *logica modale degli intervalli temporali di Halpern e Shoham* (HS in breve) è una delle più famose logiche intervallari: essa possiede una modalità per ognuna delle 13 possibili relazioni di ordinamento fra coppie di intervalli, eccetto l'uguaglianza. In questa tesi viene considerato il problema del MC basato su HS, come linguaggio di specifica delle proprietà, il quale ha ricevuto ben poca attenzione in letteratura in confronto al MC per logiche temporali puntuali. L'idea è quella di valutare formule di HS su strutture di Kripke finite, per riuscire a verificare la correttezza del comportamento di un sistema rispetto a proprietà intervallari. A questo scopo, ognuno dei percorsi finiti di una struttura di Kripke (i quali possono essere presenti in quantità infinita) è interpretato come un intervallo, e le proprietà atomiche che valgono su quest'ultimo sono definite sulla base di quelle degli stati che lo costituiscono, inizialmente secondo il *principio di omogeneità*: esso prevede che una proprietà atomica valga su un intervallo se e solo se vale su tutti i suoi

sottointervalli. Dimostriamo innanzitutto che il MC per HS interpretata su strutture di Kripke finite è un problema *decidibile* (la sua complessità computazionale ha un upper bound non-elementare); poi mostriamo che esso è **EXPSPACE**-hard.

Poiché il problema non ammette procedure di decisione di complessità polinomiale, consideriamo anche frammenti di HS, i quali si caratterizzano per complessità notevolmente migliori—da **EXPSPACE**, giù fino a **PSPACE** e a livelli bassi della gerarchia polinomiale—pur tuttavia mantenendo l'abilità di esprimere proprietà intervallari significative dei sistemi di transizione. Presentiamo svariati algoritmi di MC, costruiti ad-hoc per gli specifici frammenti considerati, e fondati su concetti e tecniche diversi fra loro.

Inoltre studiamo il potere espressivo di HS in confronto a quello delle logiche puntuali standard LTL, CTL e CTL*, sempre sotto l'ipotesi di omogeneità, la quale viene poi rilassata mostrando quali implicazioni ha questo sulla complessità del MC per HS ed i suoi frammenti, e sull'espressività della logica stessa.

Infine consideriamo una possibile alternativa alle strutture di Kripke: studiamo un modello di sistemi più espressivo, che ci permette di descrivere gli stessi direttamente in termini delle loro proprietà intervallari. Questo apre la strada a un MC basato su intervalli più generale.

Parole chiave: Model checking, logiche temporali intervallari, timeline, complessità computazionale

2010 MSC: 03B70, 68Q60

Classi ACM: F.4.1, D.2.4

Acknowledgements (and some trifles)

This thesis is the result of a three-year period of research on automatic system verification, which actually started with my master thesis (and even before graduating!). This "journey" called PhD has given me the opportunity of enriching myself (from the cultural point of view, clearly!), of knowing outstanding people from this research area, and traveling to places (most of them beautiful!) where, probably, I would have never gone in other situations.

I would like to thank for this experience some people that I affectionately want to call my "travel mates": in primis, Angelo Montanari and Adriano Peron—also for helping me during my (seemingly periodic) "reflection periods"—then Pietro Sala, Laura Bozzelli and Gabriele Puppis. I also express gratitude to the external referees, Alessio Lomuscio and Stéphane Demri, for the reviews of my thesis.

I thank my parents, my grandma, my aunt and uncle for… everything! I bet my words would not be enough to express what I feel like saying, but I am sure they know how grateful I am to them. And, I was forgetting… thank you also to my brother for taking me to the station so many times :)

Contents

I Introduction **1**

1 Introduction **3**

1.1 An overview on MC 5
1.2 Logics for MC 10
1.3 Interval temporal logics and HS 11
1.4 The problem of MC for HS 16
1.5 Organization of the thesis 17

II MC for HS under homogeneity **19**

2 MC for full HS under homogeneity **21**

2.1 Preliminaries: Kripke structures and induced abstract interval models 22
2.2 The fundamental notion of BE_k-descriptor 28
2.3 The decidability proof 37
2.4 **EXPSPACE**-hardness of MC for BE 39
2.5 An overview on MC for HS and its fragments under homogeneity 44
2.6 Related work 48

3 The logic of sub-intervals D **51**
3.1 Preliminaries . . . 52
3.2 A spatial representation of interval models . . . 54
3.3 SAT of D over finite linear orders . . . 57
3.4 MC for D over Kripke structures . . . 68
3.5 An outline of **PSPACE**-completeness of SAT and MC for D . . . 71
3.6 Conclusions . . . 72

4 HS fragments in PSPACE and EXPSPACE **73**
4.1 The fragments $A\overline{A}B\overline{B}$ and $A\overline{A}E\overline{E}$ . . . 74
4.1.1 Polynomial small-model property for $A\overline{A}B\overline{B}$ and $A\overline{A}E\overline{E}$ . . . 74
4.1.2 **PSPACE** MC algorithm for $A\overline{A}E\overline{E}$ . . . 80
4.1.3 **co-NP** MC algorithm for B and E . . . 82
4.2 The fragments $A\overline{A}B\overline{B}E$ and $A\overline{A}E\overline{B}E$: exponential small-model property . . . 84
4.3 Conclusions . . . 90

5 HS fragments at the bottom of the polynomial hierarchy **93**
5.1 Some complexity classes in the polynomial hierarchy . . . 96
5.2 $\mathbf{P}^{\mathbf{NP}}$ MC algorithm for $A\overline{A}B$ and $A\overline{A}E$ . . . 98
5.3 $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$ MC algorithm for $A\overline{A}$, $\overline{A}B$, and AE . . . 103
5.3.1 The problem $\mathrm{TB(SAT)}_{1\times M}$ . . . 103
5.3.2 Reduction of MC for $A\overline{A}$ to $\mathrm{TB(SAT)}_{1\times M}$ . . . 105
5.3.3 Reduction of MC for $\overline{A}B$ (resp., AE) to $\mathrm{TB(SAT)}_{1\times M}$ . . . 109
5.4 $\mathbf{P}^{\mathbf{NP}}$-hardness of MC for AB and $\overline{A}E$ . . . 110
5.5 $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$-hardness of MC for A and $\overline{A}$ . . . 115
5.6 Conclusions . . . 119

6 Interval vs. point logic MC: an expressiveness comparison **121**
6.1 Preliminaries . . . 125
6.1.1 Standard point-based temporal logics . . . 125
6.1.2 Three semantic variants of HS for MC . . . 126
6.1.3 An example: a vending machine . . . 129
6.2 Equivalence between LTL and $\mathrm{HS}_{\mathrm{lin}}$ . . . 132
6.3 A characterization of $\mathrm{HS}_{\mathrm{ct}}$ . . . 135
6.3.1 From finitary CTL* to $\mathrm{HS}_{\mathrm{ct}}$ . . . 135
6.3.2 From $\mathrm{HS}_{\mathrm{ct}}$ to finitary CTL* . . . 139
6.4 Expressiveness comparison of $\mathrm{HS}_{\mathrm{lin}}$, $\mathrm{HS}_{\mathrm{st}}$, $\mathrm{HS}_{\mathrm{ct}}$ . . . 146
6.5 Conclusions . . . 149

III MC for HS relaxing homogeneity 151

7 MC for HS and its fragments with regular expressions 153
7.1 Preliminaries . . . 156
7.2 MC for full HS . . . 160
7.3 The fragments $A\overline{A}B\overline{B}E$ and $A\overline{A}E\overline{B}E$ . . . 165
7.3.1 Exponential small-model property for $A\overline{A}B\overline{B}E$ . . . 165
7.3.2 **AEXP**$_{\text{pol}}$ MC algorithm for $A\overline{A}B\overline{B}E$ . . . 169
7.3.3 **AEXP**$_{\text{pol}}$-hardness of MC for $B\overline{E}$ . . . 174
7.4 The fragments $A\overline{A}B\overline{B}$ and $A\overline{A}E\overline{E}$ . . . 183
7.4.1 Exponential small-model property for $A\overline{A}B\overline{B}$ and $A\overline{A}E\overline{E}$ . . . 183
7.4.2 **PSPACE** MC algorithm for $B\overline{B}$ . . . 186
7.4.3 **PSPACE**-completeness of MC for $A\overline{A}B\overline{B}$ . . . 190
7.5 Conclusions . . . 192

IV Interval-based system models 193

8 Interval-based system models: timelines 195
8.1 The TP Problem . . . 198
8.2 TP over dense temporal domains is an undecidable problem . . . 202
8.3 Decidability of future TP with simple trigger rules . . . 210
8.3.1 Timed automata and the logic MTL . . . 210
8.3.2 Reduction to existential MC for TAs against MTL . . . 213
8.4 Future TP, simple trigger rules, non-singular intervals: hardness . . . 222
8.5 TP with trigger-less rules only is **NP**-complete . . . 225
8.6 MC for MITL over timelines . . . 231
8.7 Conclusions . . . 235

9 Conclusions 237

V Appendices 241

A Proofs and complements of Chapter 3 243
A.1 Proof of Lemma 3.3.3 . . . 243
A.2 Proof of Lemma 3.3.5 . . . 245
A.3 Proof of Lemma 3.3.7 . . . 245
A.4 Proof of Theorem 3.4.3 . . . 246
A.5 **PSPACE**-Hardness of MC for D over finite Kripke structures . . . 246
A.6 **PSPACE**-Hardness of SAT for D over finite linear orders . . . 253

B Proofs and complements of Chapter 4 **255**
B.1 Proof of Lemma 4.1.7 . . . 255
B.2 Proof of Lemma 4.1.8 . . . 256
B.3 **PSPACE**-hardness of MC for $\overline{\text{B}}$ and $\overline{\text{E}}$ . . . 257
B.4 Proof of Lemma 4.2.10 . . . 260

C Proofs of Chapter 5 **263**
C.1 Proof of Lemma 5.2.2 . . . 263
C.2 Proof of Theorem 5.3.3 . . . 264
C.3 Proof of Theorem 5.4.2 . . . 265

D Proofs of Chapter 6 **269**
D.1 Proof of Lemma 6.3.3 . . . 269
D.2 Proof of Lemma 6.3.4 . . . 271
D.3 Proof of Lemma 6.3.10 . . . 272
D.4 Proof of Lemma 6.3.13 . . . 273
D.5 Proof of Lemma 6.4.4 . . . 274

E Proofs and complements of Chapter 7 **279**
E.1 Completion of the proof of Proposition 7.2.3 . . . 279
E.2 Proof of Proposition 7.3.5 . . . 280
E.3 Proof of Proposition 7.3.6 . . . 281
E.4 Proof of Lemma 7.3.11 . . . 282
E.5 Pseudocode of `checkFalse` . . . 283
E.6 Proof of Proposition 7.3.15 . . . 283
E.7 Proof of Theorem 7.3.16 . . . 290
E.7.1 TM alternation problem . . . 290
E.7.2 $\mathbf{AEXP}_{\text{pol}}$-hardness of the alternating multi-tiling problem 294
E.8 Proof of Proposition 7.4.2 . . . 298
E.9 Proof of Theorem 7.4.3 . . . 299
E.10 Proof of Theorem 7.4.5 . . . 301
E.11 Proof of Theorem 7.4.6 . . . 304
E.12 Proof of Theorem 7.4.9 . . . 304

F Proofs and complements of Chapter 8 **307**
F.1 Definition of the value transition function T in the proof of Theorem 8.2.1 . . . 307
F.2 Non-primitive recursive-hardness of future TP . . . 309
F.3 Future TP, simple trigger rules, non-sing. intervals: **EXPSPACE**-hardness . . . 314

Bibliography **317**

Part I

Introduction

1 Introduction

Contents

1.1 An overview on MC 5
1.2 Logics for MC 10
1.3 Interval temporal logics and HS 11
1.4 The problem of MC for HS 16
1.5 Organization of the thesis 17

As the years pass by, information and communication technology (ICT) systems are becoming more and more pervasive in our lives, having a crucial role to play in countless industry, communication, security, cybernetics areas and applications. Embedded systems are employed for critical purposes, such as air traffic and railway control, telephone networks and nuclear plants monitoring. Security protocols are at the basis of e-commerce websites and services, and are exploited in all applications committed to ensure user privacy. Biomedical instruments and equipment are endowed with automatic or proactive functionalities, and are supposed to help humans and prevent human error.

The essential requirements of safety, reliability and correctness for these systems suggest (or, rather, compel) to support their development, throughout the phases of the life cycle, by structured methodologies founded on *formal methods*, with the aim to provide effective verification techniques and tools, and reduce the verification time, while simultaneously increasing coverage.

The typical techniques of system engineering for system validation, namely, *simulation* and *testing*, are clearly not sufficient alone nowadays. They are dynamic

techniques: the former involves performing some experiments with a restrictive set of scenarios or over a model, the latter actually running the system, whose correctness is determined by forcing it to traverse a set of execution paths. However, an exhaustive consideration of all execution paths is practically unfeasible, and thus testing and simulation can never be complete, that is, they can only show the presence of errors, but *not their absence*. Moreover, their effectiveness decreases dramatically as the complexity of design grows, they often discover errors and unpredictable behaviors in systems at late stages of development (or even when they have already been deployed), and are not effective at determining the more subtle and hidden bugs. Just to mention a few of them, some famed examples of bugs which were discovered too late—so to speak—were those affecting, with catastrophic economic, functional, security or life consequences, the Intel Pentium CPU floating-point division unit [Pra95], the Ariane 5 rocket [JM97], the radiation therapy machine Therac 25 [Tho94], the Needham-Schroeder authentication protocol [Low96] and the ISDN protocol [Hol94].

Formal methods are complementary to simulation and testing. They can be considered as "the applied mathematics for modeling and analyzing ICT systems" [BK08], being their aim to establish system correctness with mathematical rigor. As a matter of fact, *all the possible states and scenarios of a system* are considered during formal verification, in order to prove that the system features some desired properties such as deadlock freedom, data integrity, liveness, safety, fairness, responsiveness, interference freedom, and so on. The two most famous approaches to formal verification are *axiomatic reasoning* and *model checking*.

Axiomatic reasoning involves specifying the desired properties of a system by means of *formulas* of some logic; then a *proof system*, consisting of axioms and rules, allows one to formally prove that the system meets the expected behavior. A well-known example is Hoare's tuple-based proof system. However, axiomatic reasoning has several limitations: the proof rules are designed only for an *a posteriori* verification of existing software, not for its systematic development; it is very time-consuming, cumbersome, and can be performed only by experts. As a consequence, it is mostly employed for (parts of) critical systems or security protocols.[1] Additionally, such techniques were developed having in mind *transformational* systems/programs: in the 70s most early computer programs were designed to *compute* something (i.e., transforming initial data to final ones), and correctness was to be assessed by showing that, given some preconditions on the input, desired post-conditions hold on the output of the program. The first tools of *computer-aided verification*, namely, theorem provers, made their debut then, with the objective of automating mathematical logical reasoning. The language of such tools was often indeed Hoare-like. However, proofs could not be completely automated, and thus theorem provers were rather proof assistants and checkers.

[1] Or to verify model checkers!

In the 80s the focus shifted to *reactive systems*, whose goal is continuously interacting with the environment within which they operate, instead of terminating their execution returning some value (e.g., operating systems, communication protocols, programs for process control, embedded systems). One of the most successful techniques for the verification of such systems is *model checking*, on which research actively started during the early 80s, and originated from the independent work of Clarke and Emerson [CE81] and Queille and Sifakis [QS82].

1.1 An overview on MC

In model checking (MC) [CE81; CGP02; VW86; QS82] some properties of a system are expressed in a *temporal logic* (e.g., CTL or LTL) and then verified over a model of the system itself (usually a labeled state-transition graph or Kripke structure) through *exhaustive*—implicit or explicit—enumeration of all the reachable states. This technique is *fully automatic*: MC algorithms either terminate positively (proving that all properties are met), or produce a *counterexample* (namely, a behavior that falsifies a property), which is extremely useful for debugging purposes.

Typical system properties that are to be checked over system models by MC algorithms are:

- *functional correctness* ("the system does what it is supposed to do");
- *safety* ("something bad never happens");
- *liveness* ("something good will eventually happen");
- *fairness* ("under certain conditions, an event occurs repeatedly");
- absence of *deadlock states* ("the system never reaches a state in which it remains indefinitely");
- *real-time properties* ("the system satisfies some specific timing constraints during its operation").

The whole process of MC consists of three major phases [BK08].

1. The *modeling* phase: (*i*) a system is modeled in an accurate and unambiguous way in terms of a labeled state-transition graph or using a *model description language* (e.g., Promela, VHDL, Verilog). Then, (*ii*) the properties of the system meant to be checked are formalized unambiguously in a suitable *property specification language*, typically by formulas of some *temporal logic*.
2. The *running* phase: the model checker is executed on the system model and on the property specifications, checking the validity of the latter over the former, that is—in mathematical terms—the model checker verifies whether the system model is a model of the specifications/formulas (hence the name "model checking").

3. The *analysis* phase: the outcome of the running phase is analyzed. If a property is violated, this may depend on several causes:
   - the presence of a *modeling* error (i.e., the model does not faithfully reflect the behavior/design of the system, thus needs to be corrected);
   - if there is no modeling error, then a *design* error may have been discovered (i.e., the design has to be changed, along with the resulting model), or
   - a *property* error has taken place (i.e., the requirements to be validated have not been correctly "translated" into the specifications).

   It is worth pointing out that, whenever the model gets changed, verification has to be restarted from scratch: the design is verified only when all properties have successfully been checked with respect to the "final version" of the system model.

We now list the strengths of the MC approach.

- First of all, it is a *general* verification approach that has been employed in innumerable areas, such as:
  - planning [GT99],
  - communication and security protocols [Hol94; Low96; ACC07; BCM15],
  - embedded reactive systems [Cim01],
  - computer device drivers [Wit+07],
  - database-backed web applications [GM13],
  - concurrency control and transaction atomicity [MK12],
  - automated verification of UML design of applications [Don+06],
  - verification of air traffic control systems [Cha+98],
  - testing of railway control systems [Ben+17; Nar+16],
  - analysis of complex circuits [Bur+90a],
  - design and development of (components of) CPUs [Fix08], and verification of their microcode [Aro+05], and even
  - verification of clinical guidelines [Gio+06].
- It supports *partial specification*, i.e., no complete requirements/system specifications are needed before information can be obtained on its correctness.
- It supports *partial verification*, i.e., MC allows us to verify also *subsets* of properties at a time; thus one may choose to temporarily neglect properties of secondary importance.
- As we already mentioned, MC is *exhaustive*, meaning that all possible system behaviors are considered, and all (reachable) states are enumerated/visited.
- It provides *diagnostic information* whenever a property is invalidated; this is useful for debugging.

- It has been accepted by industry, commercial model checkers have been available for a long time, and MC-based verification is becoming integral part of standards and of system development cycles.

Conversely, we now highlight weaknesses and limitations of MC.

- First of all, MC algorithms *verify a system model*, and not the actual system! Therefore "any obtained result is as good as the system model" [BK08].
- MC techniques suffer from the so-called *state space explosion problem*: in order to accurately represent complex systems, models with a huge number of states may be necessary, sometimes possibly exceeding the available computation resources (memory and/or processing time). The parallel composition of two concurrent components is modeled by taking the product of the corresponding state spaces. This means that the global state space of a concurrent system has size exponential in the number of concurrent processes running. We will shortly list some effective methods to deal with this problem; however, models of realistic systems may sometimes still be too large to be processed. In such situations MC can be used to find bugs (but not for an exhaustive verification).
- It is worth observing that modeling is a task far from being trivial and automatable: on one hand the model has to capture all the salient features and aspects of the system to verify; on the other hand, details which are useless for the correctness proof have to be removed, as they would just weight verification down, increasing the number of states of the model. However, if the model is too coarse, MC may fail to find bugs existing in the real system. Conversely, if it is too fine-grained, spurious states/counterexamples may be generated, which are not reachable in the real system. Thus, expertise is required in finding suitable abstractions to obtain smaller, yet faithful, system models.
- Simulation and testing are still needed, in support of MC, to find fabrication faults (for hardware) or coding errors (for software), as these arise during implementation, thus falling outside of the application of MC.
- Only stated and formalized requirements can be verified (*coverage* problem); again, complementary techniques, such as testing, may reveal additional problems, that were not considered and checked.
- MC is mainly suited for *control-intensive applications*, and less for data-intensive ones, as the former can usually be represented as finite state systems, whereas data typically feature infinite domains.

MC techniques and tools can also be employed in "reverse way", that is, not during the specification and design phases, but to check already existing systems, by "extracting" a model from them and verifying it. This approach is known as "MC at implementation level". The automated generation of models amenable to

MC from programs written in C, C++ or Java [HD01; God97] has been studied, for instance, at Microsoft [BPR01] and at NASA [HP00].

As we have already said, MC techniques have to cope with the problem of the *state space explosion*, which becomes particularly serious when the system being verified is very complex or many components are working in parallel; this is the case, in particular, of software, which is extremely flexible and tends to get more and more complex (as one of Nathan's "laws" of software states).

- In this respect, *symbolic MC* [Bur+90b; CM90; McM93] allows an exhaustive implicit enumeration of a huge quantity of states (even more than 10^{120}): ordered binary decision diagrams (OBDDs), particular data structures for representing Boolean functions, are used to get concise representations of (sets of states of) transition systems and to efficiently manipulate them through fix-point algorithms. A very successful model checker was developed based on OBDDs, SMV, nowadays exploited as the basis for several commercial model checkers (e.g., CadenceSMV [MBT00] and NuSMV [Cim+00]).
- *Partial order reduction* [Pel93] tries to reduce the size of the state space by making computations that differ only in the ordering of independently executed actions collapse, as they are indistinguishable by the specification (i.e., they can be considered equivalent) and thus only one for each group needs to be tested. This approach works particularly well when applied to, for instance, communication protocols and distributed algorithms, in which several loosely coupled components are working in parallel. The model checker SPIN makes use of partial order reduction.
- Another traditional MC methodology is *bounded MC* [Bie+03]: proposed in [Bie+99], its basic idea is searching for a counterexample in computations whose length is bounded by a fixed integer k. Then, either a bug is found, or one can increase k until the problem becomes intractable, or the so-called *completeness threshold* is reached (i.e., for high enough values of k, this technique is guaranteed to find any existing counterexample). In bounded MC both the specifications of the system and the properties to check have to be translated into a Boolean formula. In this way, it is possible to employ SAT-solvers (extremely efficient procedures that can decide the satisfiability of Boolean formulas) in MC, which are less sensitive to the state space explosion problem than OBDD-based solvers. However, this method is in general *incomplete* if the bound is not high enough, hence it is used as a complementary technique to OBDD-based symbolic MC: the former is usually exploited for falsification, i.e., finding counterexamples and bugs, while the latter for verification.
- *IC3* (acronym of the expression "Incremental Construction of Inductive Clauses of Indubitable Correctness") [Bra11; Bra12] is another successful

SAT-based MC algorithm, mainly used for the verification of safety properties (but it has been adapted to liveness and fairness as well). IC3 keeps a list of formulas F_i representing an over-approximation of the system states reachable in up to i steps from the initial ones. These formulas are iteratively enriched with additional clauses, that strengthen the approximation of the reachable states, and are determined with the help of a SAT-solver. Eventually, either a pair of formulas in the list, say (F_k, F_{k+1}), become identical—which means, by the correctness of IC3, that the property holds—or a counterexample is found. Throughout the process, the *incrementality* feature of SAT-solvers is exploited: IC3 generates many similar formulas, following a common structural pattern; these can be solved quickly by re-using knowledge acquired in previous runs of the SAT-solver.

- *CEGAR* (CounterExample-Guided Abstraction Refinement), a technique typically used for safety/universal properties, minimizes the state space by *hiding system variables* that are irrelevant with respect to the property to check. To this aim, an *abstraction* of the system is produced, which must be *conservative*, namely, if a universal property is true in the abstract system, it will also be true in the concrete system model but, conversely, even if the abstract model invalidates the specification, the concrete one may still satisfy it. In such a case, intuitively, the produced abstraction is too coarse to validate the specification, as there are some abstract computation paths, reaching error states, not corresponding to any real system paths. Thus, CEGAR algorithms iteratively refine the abstraction based on unfeasible counterexamples, until either a real counterexample is obtained (which is reported to the user), or no more spurious ones are generated (hence the property holds in the real system model as well).

  Lazy abstraction [Hen+02; McM06] is an evolution of the CEGAR approach in which refinement of the abstraction (*i*) is performed on-the-fly during MC, and (*ii*) it does not affect already explored parts of the abstract state space, which saves a high computation time.

The techniques seen so far were born to deal with *finite-state* systems. However, while the finite-state assumption is often realistic for hardware systems, this is not the case of software, whose state space can either be really infinite, or it is convenient to assume so. Being able to deal with infinite-state systems is thus essential, but raises the obvious issue of the impossibility of any exhaustive exploration. Such systems are defined in terms of first-order logic predicates and formulas, typically with an underlying theory (e.g., the theory of real arithmetic), and not by Boolean formulas. As a consequence, in this context, SAT-solvers are replaced by Satisfiability Modulo Theory (SMT) solvers, namely, procedures for deciding the satisfiability of formulas of (decidable fragments of) first-order logic.

The *predicate abstraction* technique [DP14] reduces infinite-state MC to finite-state MC, by representing the infinite state space as an abstract one. Abstract computation paths are overapproximations of real paths, thus spurious counterexamples can be generated: abstraction refinement techniques are needed to deal with this. As an example, IC3 has been adapted to handle infinite-state systems by "embedding" predicate abstraction [CG12; Cim+14; Cim+16].

1.2 Logics for MC

The previous section demonstrates that a thorough investigation into MC has been performed over the years: nowadays MC techniques are mature and MC tools have gained acceptance in industry, becoming routinely used for many applications in several companies (e.g., IBM, Motorola, Intel, Nokia,...).

Two temporal logics, LTL and CTL, take the lion's share as the property specification language in MC tools. In 1977, Pnueli [Pnu77] proposed the use of the linear temporal logic LTL for program verification. LTL allows one to reason about changes in the truth value of formulas in system models over a linearly-ordered temporal domain, where each moment in time has a unique possible future. More precisely, one has to consider all possible paths in a model and to analyze, for each of them, how atomic properties (proposition letters), labeling the states, change from one state to the next one along the path. The MC problem for LTL turns out to be **PSPACE**-complete [CGP02; SC85]. This logic has also been investigated with respect to the *satisfiability* (SAT) problem (useful, for instance, in planning and as a *sanity check* for formulas in system verification) which is, again, **PSPACE**-complete [SC85].

Four years later, in 1981, Clarke and Emerson invented the branching time logic CTL [CE81], whose model of time is a tree, i.e., the future is not determined, as there are different paths in the future, any one of which may be realized. The MC problem for CTL is in **P** [CES86], while its SAT is **EXPTIME**-complete [FL79]. CTL and LTL are somewhat complementary, as there are properties expressible in only one of them [EH86].

Since then, several extensions of these logics have been investigated, as well as many of their fragments. Just to mention some, we recall the extension of LTL with *promptness*, that makes it possible to bound the delay with which a liveness request is fulfilled [KPV09], the *metric* extension of LTL called MTL [OW08], and the fragments of CTL investigated in [Mei+08].

Algorithms for other logics, for instance, CTL*, μ-calculus and first-order extensions thereof, have been developed, but those which find application in industrial-level MC are mostly LTL and CTL.

1.3 Interval temporal logics and HS

All the aforementioned logics are *"point-based"*, meaning that they can only predicate properties of system/computation states. Conversely, there are some properties we might want to check that are *inherently "interval-based"* and thus cannot be expressed by point-based temporal logics, e.g., "For every time interval I during which the green light is on for the vehicles on either road at the intersection, for some time interval beginning strictly before and ending strictly after I, the green light must be continuously off and the red light on for the vehicles on the other intersecting road" [Del+11]. Here interval temporal logics (ITLs) come into play, providing an alternative setting for reasoning about time [HS91; Ven91; Ven90]. Such logics feature intervals, instead of points, as their basic ontological temporal entities; this choice gives them the ability to express temporal properties, such as actions with duration, accomplishments, and temporal aggregations, which cannot be dealt with by standard (point-based) logics.

ITLs have been applied in a variety of computer science fields, including:

- *computational linguistics* (for the study of progressive tenses in natural language [Pra05]),
- *artificial intelligence* (for qualitative reasoning, reasoning about action and change, planning, configuration, multi-agent systems [BT03; GT99; LR06]),
- *theoretical computer science* (in formal verification [CH04; Mos83]), and
- *temporal databases* (for the specification of the expected behavior of concurrency control and transaction scheduling [GMS04]).

Moszkowski's Propositional Interval Temporal Logic (PITL) [Mos83] is one of the first ITLs, introduced in 1983 with the aim of specifying and verifying hardware components. In the 90s, Duration Calculus (DC)—an extension of PITL featuring the concept of duration of an event on a time interval—was studied [CHR91] and, since then, developed and applied to the specification and design of time-critical systems [CH04; HV07]. However, its semantics is point-based, being proposition letters interpreted only over time points. In [Ven91], Venema introduced the ITL CDT featuring the binary modality C (chop)—the natural *chopping* operation on an interval involves bisecting it into two consecutive parts/subintervals—and its two residual modalities D and T.

A prominent position among ITLs is occupied by *Halpern and Shoham's modal logic of time intervals* (HS, for short) [HS91]. HS features one modality for each of the 13 possible ordering relations between pairs of intervals (the so-called *Allen's relations* [All83]), apart from equality. As an example, the statement "the current interval meets an interval over which p holds" can be expressed in HS by the formula $\langle \mathrm{A} \rangle p$, where $\langle \mathrm{A} \rangle$ is the (existential) HS modality for Allen's relation *meet*.

Table 1.1: Allen's relations and corresponding HS modalities.

Allen relation	HS modality	Definition w.r.t. interval structures	Example
MEETS	$\langle A \rangle$ (after)	$[x,y]\mathcal{R}_A[v,z] \iff y = v$	
BEFORE	$\langle L \rangle$ (later)	$[x,y]\mathcal{R}_L[v,z] \iff y < v$	
STARTED-BY	$\langle B \rangle$ (begins)	$[x,y]\mathcal{R}_B[v,z] \iff x = v \wedge z < y$	
FINISHED-BY	$\langle E \rangle$ (ends)	$[x,y]\mathcal{R}_E[v,z] \iff y = z \wedge x < v$	
CONTAINS	$\langle D \rangle$ (during)	$[x,y]\mathcal{R}_D[v,z] \iff x < v \wedge z < y$	
OVERLAPS	$\langle O \rangle$ (overlaps)	$[x,y]\mathcal{R}_O[v,z] \iff x < v < y < z$	

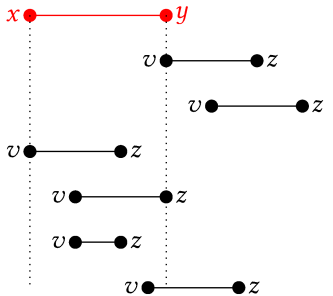


Table 1.1 depicts 6 of the 13 Allen's relations together with the corresponding HS (existential) modalities. The other 7 are equality and the 6 inverse (symmetrical) relations (given a binary relation $\mathcal{R}$, its inverse $\overline{\mathcal{R}}$ is such that $b\overline{\mathcal{R}}a$ iff $a\mathcal{R}b$).

The language of HS features a set of proposition letters $\mathcal{AP}$, the Boolean connectives $\neg$ and $\vee$, and a temporal modality for each of the (non trivial) Allen's relations, namely, $\langle \mathrm{A} \rangle$, $\langle \mathrm{L} \rangle$, $\langle \mathrm{B} \rangle$, $\langle \mathrm{E} \rangle$, $\langle \mathrm{D} \rangle$, $\langle \mathrm{O} \rangle$, $\langle \overline{\mathrm{A}} \rangle$, $\langle \overline{\mathrm{L}} \rangle$, $\langle \overline{\mathrm{B}} \rangle$, $\langle \overline{\mathrm{E}} \rangle$, $\langle \overline{\mathrm{D}} \rangle$ and $\langle \overline{\mathrm{O}} \rangle$. HS formulas are defined by the following grammar:

$$\psi ::= p \mid \neg\psi \mid \psi \vee \psi \mid \langle \mathrm{X} \rangle \psi \mid \langle \overline{\mathrm{X}} \rangle \psi, \quad \text{with } p \in \mathcal{AP},\ X \in \{A, L, B, E, D, O\}.$$

Throughout the thesis, we will make use of the standard abbreviations of propositional logic, e.g., we write $\psi \wedge \phi$ for $\neg(\neg\psi \vee \neg\phi)$, $\psi \rightarrow \phi$ for $\neg\psi \vee \phi$, $\psi \leftrightarrow \phi$ for $(\psi \rightarrow \phi) \wedge (\phi \rightarrow \psi)$, $\top$ (true) for $p \vee \neg p$, and $\bot$ (false) for $\neg \top$.

For all $X \in \{A, L, B, E, D, O\}$, the (dual) universal modalities $[\mathrm{X}]\,\psi$ and $[\overline{\mathrm{X}}]\,\psi$ are defined as $\neg \langle \mathrm{X} \rangle \neg\psi$ and $\neg \langle \overline{\mathrm{X}} \rangle \neg\psi$, respectively. We denote by $\mathrm{X}_1 \cdots \mathrm{X}_n$ the fragment of HS, closed under Boolean connectives, that features (existential and universal) modalities for $\langle X_1 \rangle, \ldots, \langle X_n \rangle$ only.

W.l.o.g., we assume the *non-strict semantics* of HS, which admits intervals consisting of a single point.[2] Under such an assumption, all HS modalities can be expressed in terms of $\langle \mathrm{B} \rangle$, $\langle \mathrm{E} \rangle$, $\langle \overline{\mathrm{B}} \rangle$ and $\langle \overline{\mathrm{E}} \rangle$ [HS91]. As an example, $\langle \mathrm{A} \rangle$ (resp., $\langle \overline{\mathrm{A}} \rangle$) can be expressed by $\langle \mathrm{E} \rangle$ and $\langle \overline{\mathrm{B}} \rangle$ (resp., $\langle \mathrm{B} \rangle$ and $\langle \overline{\mathrm{E}} \rangle$) as

$$\langle \mathrm{A} \rangle \varphi = ([\mathrm{E}] \bot \wedge (\varphi \vee \langle \overline{\mathrm{B}} \rangle \varphi)) \vee \langle \mathrm{E} \rangle ([\mathrm{E}] \bot \wedge (\varphi \vee \langle \overline{\mathrm{B}} \rangle \varphi))$$

$$(\text{resp., } \langle \overline{\mathrm{A}} \rangle \varphi = ([\mathrm{B}] \bot \wedge (\varphi \vee \langle \overline{\mathrm{E}} \rangle \varphi)) \vee \langle \mathrm{B} \rangle ([\mathrm{B}] \bot \wedge (\varphi \vee \langle \overline{\mathrm{E}} \rangle \varphi))).$$

As for all other modalities, the next equivalences hold:

$$\langle \mathrm{L} \rangle \psi = \langle \mathrm{A} \rangle \langle \mathrm{A} \rangle \psi \qquad \langle \overline{\mathrm{L}} \rangle \psi = \langle \overline{\mathrm{A}} \rangle \langle \overline{\mathrm{A}} \rangle \psi$$
$$\langle \mathrm{D} \rangle \psi = \langle \mathrm{B} \rangle \langle \mathrm{E} \rangle \psi = \langle \mathrm{E} \rangle \langle \mathrm{B} \rangle \psi \qquad \langle \overline{\mathrm{D}} \rangle \psi = \langle \overline{\mathrm{B}} \rangle \langle \overline{\mathrm{E}} \rangle \psi = \langle \overline{\mathrm{E}} \rangle \langle \overline{\mathrm{B}} \rangle \psi$$
$$\langle \mathrm{O} \rangle \psi = \langle \mathrm{E} \rangle \langle \overline{\mathrm{B}} \rangle \psi \qquad \langle \overline{\mathrm{O}} \rangle \psi = \langle \mathrm{B} \rangle \langle \overline{\mathrm{E}} \rangle \psi.$$

We also use the auxiliary operator $\langle \mathrm{G} \rangle$ of HS (and its dual $[\mathrm{G}]$), which allows one to select arbitrary subintervals of a given interval, and is defined as: $\langle \mathrm{G} \rangle \psi = \psi \vee \langle \mathrm{B} \rangle \psi \vee \langle \mathrm{E} \rangle \psi \vee \langle \mathrm{B} \rangle \langle \mathrm{E} \rangle \psi$.

HS can thus be seen as a multi-modal logic with four primitive modalities and its semantics can be defined over a multi-modal structure, called *abstract interval model*, where intervals are treated as atomic objects and Allen's relations as binary relations between pairs of intervals. Since later we will focus on some HS fragments not including some of the primitive modalities, we consider explicitly both $\langle \mathrm{A} \rangle$ and $\langle \overline{\mathrm{A}} \rangle$ in addition to $\langle \mathrm{B} \rangle$, $\langle \mathrm{E} \rangle$, $\langle \overline{\mathrm{B}} \rangle$ and $\langle \overline{\mathrm{E}} \rangle$.

[2] All the results we prove in this thesis hold for the strict semantics as well, which disallows point-intervals.

Definition 1.3.1 (Abstract interval model)**.** An *abstract interval model* is a tuple

$$\mathcal{A} = (\mathcal{AP}, \mathbb{I}, A_{\mathbb{I}}, B_{\mathbb{I}}, E_{\mathbb{I}}, \sigma),$$

where:

- $\mathcal{AP}$ is a finite set of proposition letters;
- $\mathbb{I}$ is a possibly infinite set of atomic objects (worlds);
- $A_{\mathbb{I}}$, $B_{\mathbb{I}}$, $E_{\mathbb{I}}$ are three binary relations over $\mathbb{I}$;
- $\sigma : \mathbb{I} \mapsto 2^{\mathcal{AP}}$ is a (total) labeling function, which assigns a set of proposition letters to each world.

Intuitively, in the interval setting, $\mathbb{I}$ is a set of intervals, $A_{\mathbb{I}}$, $B_{\mathbb{I}}$, and $E_{\mathbb{I}}$ are interpreted as Allen's interval relations A (*meets*), B (*started-by*), and E (*finished-by*), respectively, and σ assigns to each interval the set of proposition letters that hold over it.

Definition 1.3.2 (HS semantics)**.** Given an abstract interval model $\mathcal{A} = (\mathcal{AP}, \mathbb{I}, A_{\mathbb{I}}, B_{\mathbb{I}}, E_{\mathbb{I}}, \sigma)$ and a world $I \in \mathbb{I}$, the truth of an HS formula over I is defined by induction on the structural complexity of the formula as:

- $\mathcal{A}, I \models p$ iff $p \in \sigma(I)$, for any proposition letter $p \in \mathcal{AP}$;
- $\mathcal{A}, I \models \neg\psi$ iff it is not true that $\mathcal{A}, I \models \psi$ (also denoted as $\mathcal{A}, I \not\models \psi$);
- $\mathcal{A}, I \models \psi \vee \varphi$ iff $\mathcal{A}, I \models \psi$ or $\mathcal{A}, I \models \varphi$;
- $\mathcal{A}, I \models \langle \mathrm{X} \rangle \psi$, for $X \in \{A, B, E\}$, iff there exists $J \in \mathbb{I}$ such that $I \, X_{\mathbb{I}} \, J$ and $\mathcal{A}, J \models \psi$;
- $\mathcal{A}, I \models \langle \overline{\mathrm{X}} \rangle \psi$, for $\overline{X} \in \{\overline{A}, \overline{B}, \overline{E}\}$, iff there exists $J \in \mathbb{I}$ such that $J \, X_{\mathbb{I}} \, I$ and $\mathcal{A}, J \models \psi$.

Satisfiability (SAT) and *validity* are defined in the usual way: an HS formula ψ is satisfiable if there exists an interval model $\mathcal{A}$ and a world (interval) I such that $\mathcal{A}, I \models \psi$. Moreover, ψ is valid if $\mathcal{A}, I \models \psi$ for all worlds (intervals) I of every interval model $\mathcal{A}$.

In [HS91] it was shown that the *validity problem* for HS interpreted over any class of linear orders that contains at least one linear order with an infinite ascending or descending sequence of points, is *undecidable* (more precisely, r.e. hard). For example, it is undecidable over $\mathbb{N}$ (the natural numbers), $\mathbb{Z}$ (the integers), $\mathbb{Q}$ (the rationals) and $\mathbb{R}$ (the reals), as well as over the classes of all linear orders, all dense linear orders, and all discrete linear orders. Moreover, over any class of *Dedekind complete* ordered structures containing at least an order with an infinite ascending sequence, it is Π^1_1-hard (i.e., not recursively axiomatizable). This is true, e.g., over $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$, being Dedekind complete.

The authors of [Del+11] state that:

> [...] so general undecidability results about HS are given in [HS91] that for a long time it was considered unsuitable for practical applications and attracted little interest among computer scientists.

Since then, a lot of work has been done on the SAT problem for *fragments* of HS, which showed that undecidability rules also over them [Bre+08; Bre+09a; Bre+14; Lod00; MM14]. As an example, the fragment BE (i.e., formulas of HS with $\langle B \rangle$ and $\langle E \rangle$ modalities only) is undecidable, again, over the class of all linear orders, over all dense linear orders, discrete linear orders and finite linear orders [Lod00].

The reason for these undecidability results must be ascribed to the very nature of purely interval-based temporal reasoning, where proposition letters are interpreted on intervals (and not on points): the set-theoretic interpretation of an HS formula in a model is a set of abstract intervals, that is, a set of pairs of points (a binary relation over the underlying linear order). Automata-based methods, founded for instance on Büchi and Rabin theorems (implying decidability of MSO theories of various linear orders and trees), do not apply here, as SAT and validity for ITLs are dyadic, and not monadic. New approaches for proving decidability of HS fragments with genuinely interval-based semantics were needed.

Meanwhile, several modifications or restrictions, essentially reducing the interval-based semantics to a point-based one, were proposed to remedy the problem and obtain decidable systems. As an example, already in [Mos83] Moszkowski showed that the decidability of the (otherwise undecidable) logic PITL can be recovered by constraining proposition letters to be point-wise and defining truth of an interval as truth on its initial point (*locality principle*).

The bleak picture started lightening up later, with the discovery of several non-trivial cases of decidable fragments of HS [Bre+10b; Bre+09b; Bre+11c; MPS10b; Mon+10; MS12; Bre+11b], including the *interval logic of temporal neighbourhood* $A\overline{A}$ and the *temporal logic of sub-intervals* D. In particular, the former was shown to be decidable over the class of all linear orders, of all dense, of all discrete, and of all finite linear orders;[3] as for the latter, the situation is more involved: D is *decidable* over all dense linear orders, *undecidable* over discrete linear orders and finite linear orders [MM14], and it is not known whether it is decidable or not in the case of all linear orders. Some other fragments, such as $B\overline{B}$ and $E\overline{E}$, have actually a *point-based* semantics: one of the endpoints of every interval related to the current one can "move", but the other always remains fixed; as a consequence they can be polynomially translated into a basic point-based temporal logic (namely, the fragment of LTL with only the future F and past P operators): any interval is

[3] It is also worth mentioning the decidable metric extension of the fragment $A\overline{A}$ called MPNL, studied in [Bre+13; Bre+10a; Bre+11a], that features special atomic propositions representing integer constraints (equalities and inequalities) on the length of the intervals over which they are predicated.

mimicked by its only moving endpoint; it follows that $\mathsf{B}\overline{\mathsf{B}}$ and $\mathsf{E}\overline{\mathsf{E}}$ are decidable over the class of all linear orders and, in particular, **NP**-complete [GMS04]. On the other hand, another fragment, $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}$, is decidable over finite linear orders, $\mathbb{Q}$, as well as over the class of all linear orders, but undecidable over Dedekind-complete linear orders (in particular, $\mathbb{N}$ and $\mathbb{R}$) [MPS14]. These snapshots on HS SAT make us conclude, with the authors of [Del+11], that

> [...] the trade-off between expressiveness and computational affordability in the family of HS fragments is rather subtle and sometimes unpredictable, with the border between decidability and undecidability cutting right across the core of that family.

1.4 The problem of MC for HS

Whereas, as we have seen, darkness reigns in the realm of SAT with few exceptions, fortunately a dim light brightens the whole land of MC for HS: in this thesis, we will consider this *decidable* problem, for which a little work has been done, if compared to SAT of HS and especially to MC of point-based temporal logics.

For quite a long time, no studies on MC for HS were undertaken: the problem was finding suitable system models over which to interpret such an interval-based property specification language (clearly, this was a novel issue to face, as the SAT problem is defined more simply on linear orders). In 2011, Dario Della Monica concluded his PhD thesis about decidability and undecidability results on SAT for HS and its fragments in this way [Del11]:

> Even if this topic [MC] has been extensively studied and successfully applied to real-world domains in the context of point-based logics, it is still quite unexplored in the interval setting. The major difficulty concerns the problem of finding a convenient way to (finitely) represent the models to be checked.

The first idea we explore in this thesis is evaluating HS formulas on (very standard) system models, consisting essentially in *labeled graphs*, called finite *Kripke structures* (which will be formally defined later), making it possible to check the correctness of the behavior of systems with respect to meaningful interval properties. In order to verify interval properties of computations, one needs to collect information about the states of a system into computation stretches. To this end, each finite path (i.e., a *trace*) in a Kripke structure is interpreted as an interval, and its labeling is defined on the basis of the labeling of the states composing it, initially according to the *homogeneity assumption* [Roe80], which enforces a proposition letter p to hold on an interval I if and only if p holds on all the subintervals of I. Such an assumption is fairly natural for many applications;

moreover, it easily enables us to interpret an *interval-based* property specification language on a model which is *point-based* (as a matter of fact, Kripke structures make explicit how a system evolves *state-by-state*, and describe which are the atomic properties that hold true at every single state).

Then we devote our efforts to relaxing homogeneity. For the purpose, we investigate the possibility of using regular expressions to define the labeling of proposition letters over intervals in terms of the component states (as we will see, homogeneity can be trivially encoded by regular expressions, as a particular case). This impacts on the complexity of MC for HS and its fragments, and on the expressiveness of the logic as well. Another possibility is completely abandoning Kripke structures in favour of different, *inherently interval-based* system models: *timelines*, structures that have been fruitfully exploited in temporal planning for quite a long time, are employed in this thesis in a different way: more expressive than Kripke structures, they allow us to directly describe systems in terms of their *interval-based behaviour and properties*, paving the way for a more general interval-based MC.

1.5 Organization of the thesis

Let us now take a deeper look at the contents of the thesis, while presenting its organization.

1. In the next chapter—which starts Part II—we show that finite Kripke structures can be suitably mapped into abstract interval models, whose worlds are all the possible traces of the former. However, since Kripke structures may have loops and thus infinitely many traces, abstract interval models feature, in general, an infinite domain. In order to devise a MC procedure for HS over finite Kripke structures—hence showing the decidability of MC for HS—we prove a small model theorem stating that, given an HS formula ψ and a finite Kripke structure $\mathcal{K}$, there exists a *finite* interval model which is equivalent to the one induced by $\mathcal{K}$ with respect to the satisfiability of ψ. The main technical ingredients are (*i*) the definition of a suitable equivalence relation over traces of $\mathcal{K}$, which is parametric in the nesting depth of modalities $\langle \mathrm{B} \rangle$ and $\langle \mathrm{E} \rangle$ in ψ, and (*ii*) the proof that the resulting quotient structure is finite and equivalent to the one induced by $\mathcal{K}$ with respect to the satisfiability of ψ.
   The chapter is concluded showing that the problem does not provably admit any polynomial-time decision procedure (as it is **EXPSPACE**-hard). For this reason,
2. in the subsequent chapters of Part II we focus on fragments of HS, which feature considerably better complexities—from **EXPSPACE**, down to **PSPACE** and to low levels of the polynomial hierarchy. Nevertheless, they have the

ability to capture meaningful interval properties of state transition systems. We describe several MC algorithms, tailored to the specific fragments being considered, which feature as building blocks concepts and techniques different from one another (e.g., small model properties, trace contractions, Boolean circuits with oracles, finite state automata,...). Most of the times, we also prove lower bounds matching the complexity of MC algorithms.
Moreover, we study the expressive power of HS in MC, in comparison with that of the standard point-based logics LTL, CTL and CTL*, still under homogeneity. Additionally, we introduce some *semantic variants* of HS and evaluate their expressiveness and succinctness with respect to the aforementioned point-based logics.

3. The homogeneity assumption is relaxed in Part III: as we have anticipated, at first we still consider Kripke structures as system models, but we use regular expressions to define the labeling of proposition letters over intervals in terms of the component states. In particular, MC for HS extended with regular expressions remains still decidable: this result is proved via a (non completely standard) automata-theoretic approach.
As for the fragments of HS, the MC problem for "small" ones features an increased (**PSPACE**) complexity, whereas formulas of other, more expressive fragments, can be checked at no extra computational cost (w.r.t. the homogeneous case). Such complexity results are achieved by heavily exploiting *non-deterministic* and *alternating* algorithms.
4. Finally, in Part IV we consider the *timeline-based MC problem*, where system models are given by the aforementioned *timelines*, and the property specification language is the logic MITL: the choice of replacing HS with the latter is justified by the need for a *timed* logic, appropriate to be interpreted over timelines, where the dimension of time is essential.
Most of Part IV will be actually devoted to the study of the so-called *timeline-based planning problem* over *dense temporal domains*: first, such problem can be considered as a necessary condition for MC, the former playing the role of a "feasibility check" of the system description (as a matter of fact, whereas Kripke structures are "extensive" system descriptions, timelines are not, given the presence of constraint rules which must be checked for satisfiability). Moreover, dense time domains (as opposed to discrete ones, which are standard in planning) are necessary to avoid discreteness in system descriptions, that can be abstracted at a higher level, enabling us to express really interval-based properties of systems. Since, in its full generality, timeline-based planning over dense domains is, unfortunately, undecidable, suitable restrictions on timelines are identified at the beginning of Part IV in order to recover its decidability, and finally deal with timeline-based MC.

Part II

MC for HS under homogeneity

2 MC for full HS under homogeneity

The references for this chapter are [Mol+16a; Boz+16b; Boz+18g].

Contents

2.1 **Preliminaries: Kripke structures and induced abstract interval models** . . . 22
2.2 **The fundamental notion of BE_k-descriptor** . . . 28
2.3 **The decidability proof** . . . 37
2.4 **EXPSPACE-hardness of MC for BE** . . . 39
2.5 **An overview on MC for HS and its fragments under homogeneity** 44
2.6 **Related work** . . . 48

In this chapter we prove two fundamental results. On one hand we show that MC for HS under homogeneity over finite Kripke structures is decidable. On the other we prove a complexity lower bound for the problem, namely, its **EXPSPACE**-hardness. In the following description of the contents of the next sections, we detail how these results are achieved.

Organization of the chapter.

- In Section 2.1, after introducing Kripke structures, we show how these system models can be mapped into abstract interval models, where each world corresponds to a trace (finite path) of the structure: this allows us to interpret HS formulas over Kripke structures, and then to define interval-based MC. Since a Kripke structure may feature loops and thus infinitely many traces, the domain of the induced abstract interval model is infinite. Thus, to prove decidability of MC,

- in Section 2.2 we introduce BE-descriptors, that are trees built starting from the traces of a Kripke structure $\mathcal{K}$, whose nodes have labels over states of $\mathcal{K}$, which allow us to define an equivalence relation of finite index over traces of $\mathcal{K}$, parametric in the nesting depth of modalities $\langle B \rangle$ and $\langle E \rangle$ in the formula ψ to check.
- We prove in Section 2.3 that such a relation represents a sufficient condition for two (related) traces to be indistinguishable w.r.t. the truth value of any HS formula with nesting depth of $\langle B \rangle$ and $\langle E \rangle$ bounded by a value equal to the depth of the considered BE-descriptors. Decidability of HS MC (under homogeneity) follows as a result of such a relation (and by minor technical ingredients). We can finally devise a *nonelementary complexity* decision procedure for the problem.
- In Section 2.4, we prove the **EXPSPACE**-hardness of MC for the HS fragment BE. Since MC for full HS is clearly at least as hard as MC for BE, such a lower bound immediately propagates to full HS. The section concludes with some remarks on the complexity gap (nonelementary–**EXPSPACE**) deriving from the results proved for the problem.
- In Section 2.5 we give an overview of the complexity of MC for several HS fragments studied under homogeneity, which will be considered in the next chapters.
- We conclude with Section 2.6, in which we shortly review some research papers (not of the present authors) dealing with HS MC (under different semantic assumptions).

2.1 Preliminaries: Kripke structures and induced abstract interval models

In the context of MC, finite state systems are usually modelled as *finite Kripke structures*, namely, graphs where nodes represent the states of the system, edges represent transitions between states, and there is a labelling function that associates every node with a set of (atomic) properties that hold true in the corresponding system state. Systems described in terms of many modelling languages (e.g., Promela) are translated (usually, at the cost of an exponential blow-up) into Kripke structures, before the MC running phase begins.

Definition 2.1.1 (Kripke structure)**.** A Kripke structure is a tuple

$$\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0),$$

where $\mathcal{AP}$ is a finite set of proposition letters, S is a set of states (*worlds*),

$R \subseteq S \times S$ is a left-total relation[a] between pairs of states (*accessibility/transition* relation), $\mu : S \to 2^{\mathcal{AP}}$ is a total labelling function, and $s_0 \in S$ is called the *initial* state. We say that $\mathcal{K}$ is finite if S is finite.

[a] A relation $R \subseteq S \times S$ is left-total if, for all $s \in S$, there exists at least one $s' \in S$ such that $(s, s') \in R$.

For all $s \in S$, $\mu(s)$ captures the set of proposition letters that hold at that state; R constrains the evolution of the system over time, namely, it specifies how the system can evolve from one state to another; it is left-total because the paths of $\mathcal{K}$ are meant to represent system computations. If $(s, s') \in R$, we say that s' is a *successor* of s, and s a *predecessor* of s'. By $R(s)$ we denote the set of successors of s.

A simple Kripke structure, consisting of two states only, is reported in the following example. We will use it as a running example throughout the thesis.

Example 2.1.2. Figure 2.1 depicts a two-state Kripke structure $\mathcal{K}_2$ (the initial state is identified by a double circle). Despite its simplicity, it features an *infinite number* of different (finite) paths. $\mathcal{K}_2$ is defined by the quintuple:

$$(\{p, q\}, \{s_0, s_1\}, \{(s_0, s_0), (s_0, s_1), (s_1, s_0), (s_1, s_1)\}, \mu, s_0),$$

where $\mu(s_0) = \{p\}$ and $\mu(s_1) = \{q\}$.



Figure 2.1: The finite Kripke structure $\mathcal{K}_2$.

The next definition formalizes the notion of path in a Kripke structure.

Definition 2.1.3 (Trace over $\mathcal{K}$)**.** A trace ρ over a Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ is a finite sequence of states $s_1 \cdots s_n$, with $n \geq 1$, such that $(s_i, s_{i+1}) \in R$ for all $i \in \{1, \ldots, n-1\}$.

Let $\mathrm{Trc}_{\mathcal{K}}$ be the (infinite) set of all traces over a finite/infinite Kripke structure $\mathcal{K}$. For any trace $\rho = s_1 \cdots s_n \in \mathrm{Trc}_{\mathcal{K}}$, we define:

- $|\rho| = n$ and, for $1 \leq i \leq |\rho|$, $\rho(i) = s_i$ (we also say that i is a "ρ-position");
- $\mathrm{states}(\rho) = \{s_1, \ldots, s_n\} \subseteq S$;
- $\mathrm{intstates}(\rho) = \{s_2, \ldots, s_{n-1}\} \subseteq S$;
- $\mathrm{fst}(\rho) = s_1$ and $\mathrm{lst}(\rho) = s_n$;
- $\rho(i, j) = s_i \cdots s_j$, for $1 \leq i \leq j \leq |\rho|$, is a subtrace of ρ;

- $\mathrm{Pref}(\rho) = \{\rho(1,i) \mid 1 \le i \le |\rho| - 1\}$ is the set of all proper prefixes of ρ;
- $\mathrm{Suff}(\rho) = \{\rho(i,|\rho|) \mid 2 \le i \le |\rho|\}$ is the set of all proper suffixes of ρ.

If $\mathrm{fst}(\rho) = s_0$—where s_0 is the initial state of $\mathcal{K}$—ρ is said to be an *initial trace*.

Given $\rho, \rho' \in \mathrm{Trc}_{\mathcal{K}}$, we denote by $\rho \cdot \rho'$ the concatenation of the traces ρ and ρ'. Moreover, if $\mathrm{lst}(\rho) = \mathrm{fst}(\rho')$, $\rho \star \rho'$ denotes $\rho(1, |\rho| - 1) \cdot \rho'$ ($\star$-concatenation). In the following, when we write $\rho \star \rho'$, we implicitly assume that $\mathrm{lst}(\rho) = \mathrm{fst}(\rho')$.

An abstract interval model (over $\mathrm{Trc}_{\mathcal{K}}$)—recall Definition 1.3.1—can be naturally associated with a finite Kripke structure by interpreting every trace as an interval bounded by its first and last states.

Definition 2.1.4 (Abstract interval model induced by $\mathcal{K}$)**.** The abstract interval model induced by a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ is the abstract interval model $\mathcal{A}_{\mathcal{K}} = (\mathcal{AP}, \mathbb{I}, A_{\mathbb{I}}, B_{\mathbb{I}}, E_{\mathbb{I}}, \sigma)$, where:

- $\mathbb{I} = \mathrm{Trc}_{\mathcal{K}}$,
- $A_{\mathbb{I}} = \{(\rho, \rho') \in \mathbb{I} \times \mathbb{I} \mid \mathrm{lst}(\rho) = \mathrm{fst}(\rho')\}$,
- $B_{\mathbb{I}} = \{(\rho, \rho') \in \mathbb{I} \times \mathbb{I} \mid \rho' \in \mathrm{Pref}(\rho)\}$,
- $E_{\mathbb{I}} = \{(\rho, \rho') \in \mathbb{I} \times \mathbb{I} \mid \rho' \in \mathrm{Suff}(\rho)\}$,
- $\sigma : \mathbb{I} \to 2^{\mathcal{AP}}$ is such that, for all $\rho \in \mathbb{I}$,

$$\sigma(\rho) = \bigcap_{s \in \mathrm{states}(\rho)} \mu(s).$$

In Definition 2.1.4, relations $A_{\mathbb{I}}$, $B_{\mathbb{I}}$, and $E_{\mathbb{I}}$ are interpreted as Allen's interval relations A, B and E, respectively. Moreover, according to the definition of σ, a proposition letter $p \in \mathcal{AP}$ holds over $\rho = s_1 \cdots s_n$ if and only if it holds over all the states $s_1, \ldots, s_n$ of ρ. This conforms to the *homogeneity principle* [Roe80], according to which a proposition letter holds over an interval if and only if it holds over all of its subintervals.

Satisfiability of an HS formula over a finite Kripke structure can now be given in terms of induced abstract interval models.

Definition 2.1.5 (Satisfiability of HS formulas over finite Kripke structures)**.** Let $\mathcal{K}$ be a finite Kripke structure, ρ be a trace in $\mathrm{Trc}_{\mathcal{K}}$, ψ be an HS formula. We say that the pair $(\mathcal{K}, \rho)$ satisfies ψ, denoted by $\mathcal{K}, \rho \models \psi$, if and only if it holds that $\mathcal{A}_{\mathcal{K}}, \rho \models \psi$.

We are now ready to formally state the *MC problem* for HS over finite Kripke structures: it is the problem of deciding whether or not $\mathcal{K} \models \psi$.

Definition 2.1.6 (Model checking)**.** Let $\mathcal{K}$ be a finite Kripke structure and ψ be an HS formula. We say that $\mathcal{K}$ models ψ, denoted by $\mathcal{K} \models \psi$, if and only if, for all *initial* traces $\rho \in \mathrm{Trc}_{\mathcal{K}}$, it holds that $\mathcal{K}, \rho \models \psi$.

It is worth pointing out that every finite Kripke structure $\mathcal{K}$ induces an abstract interval model, and that only interval models arising from finite Kripke structures are considered in the MC problem.

Given a finite $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$, being R left-total and S finite, $\mathcal{K}$ has to feature some loops, and thus an infinite number of traces; as a consequence, the *MC problem is not trivially decidable.*

We conclude this section by giving some examples of meaningful properties of Kripke structures and traces that can be expressed in HS.

Example 2.1.7. The formula $[\mathrm{B}]\,\bot$ can be used to select all and only the traces of length 1. Indeed, given any ρ with $|\rho| = 1$, independently of $\mathcal{K}$, it holds that $\mathcal{K}, \rho \models [\mathrm{B}]\,\bot$, because ρ has no proper prefixes. On the other hand, it holds that $\mathcal{K}, \rho \models \langle \mathrm{B} \rangle\, \top$ if (and only if) $|\rho| > 1$.
Modality $\langle \mathrm{B} \rangle$ can actually be used to constrain the length of an interval to be greater than, less than, or equal to any value k. Let us denote k nested applications of $\langle \mathrm{B} \rangle$ by $\langle \mathrm{B} \rangle^k$. It holds that $\mathcal{K}, \rho \models \langle \mathrm{B} \rangle^k\, \top$ if and only if $|\rho| \geq k + 1$. Analogously, $\mathcal{K}, \rho \models [\mathrm{B}]^k\, \bot$ if and only if $|\rho| \leq k$. Let length_k be a shorthand for $[\mathrm{B}]^k\, \bot \wedge \langle \mathrm{B} \rangle^{k-1}\, \top$. It holds that $\mathcal{K}, \rho \models \mathit{length}_k$ if and only if $|\rho| = k$ (this formula will be used several times in the next chapters).

Example 2.1.8. Let us consider again the finite Kripke structure $\mathcal{K}_2$ of Example 2.1.2, depicted in Figure 2.1. For the sake of brevity, for any trace ρ, we denote by ρ^n the trace obtained by concatenating n copies of ρ. The truth of the following statements can be easily checked: (*i*) $\mathcal{K}_2, (s_0 s_1)^2 \models \langle \mathrm{A} \rangle\, q$; (*ii*) $\mathcal{K}_2, s_0 s_1 s_0 \not\models \langle \mathrm{A} \rangle\, q$; (*iii*) $\mathcal{K}_2, (s_0 s_1)^2 \models \langle \overline{\mathrm{A}} \rangle\, p$; (*iv*) $\mathcal{K}_2, s_1 s_0 s_1 \not\models \langle \overline{\mathrm{A}} \rangle\, p$. The above statements show that modalities $\langle \mathrm{A} \rangle$ and $\langle \overline{\mathrm{A}} \rangle$ can be used to distinguish between traces that *start or end at different states.*
We would like to draw attention to the *branching* semantics of modalities $\langle \mathrm{A} \rangle$ and $\langle \overline{\mathrm{A}} \rangle$ (in our experience, at the beginning, this causes confusion for the reader…): $\langle \mathrm{A} \rangle$ (resp., $\langle \overline{\mathrm{A}} \rangle$) allows one to "move" to *any* trace branching on the right/future (resp., left/past) of the considered one, e.g., if $\rho = s_1 s_0$, then $\rho\, A_{\mathbb{I}}\, s_0$, $\rho\, A_{\mathbb{I}}\, s_0 s_0$, $\rho\, A_{\mathbb{I}}\, s_0 s_1$, $\rho\, A_{\mathbb{I}}\, s_0 s_0 s_0$, $\rho\, A_{\mathbb{I}}\, s_0 s_0 s_1$, $\rho\, A_{\mathbb{I}}\, s_0 s_1 s_0 s_1$, and so on. Analogously, $s_1\, A_{\mathbb{I}}\, \rho$, $s_0 s_1\, A_{\mathbb{I}}\, \rho$, $s_1 s_1\, A_{\mathbb{I}}\, \rho$,…
Figure 2.2 illustrates the "behaviour" of $\langle \mathrm{A} \rangle$ and $\langle \overline{\mathrm{A}} \rangle$.

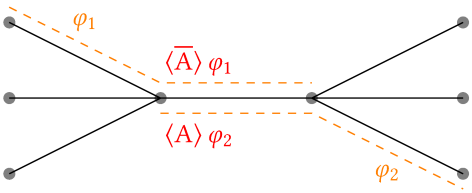


Figure 2.2: The branching semantics of modalities $\langle\mathrm{A}\rangle$ and $\langle\overline{\mathrm{A}}\rangle$

Modalities $\langle\mathrm{B}\rangle$ and $\langle\mathrm{E}\rangle$ can be exploited to distinguish between traces encompassing a different number of iterations of a given loop. This is the case, for instance, with the following statements:

- $\mathcal{K}_2, (s_1 s_0)^3 s_1 \models \langle\mathrm{B}\rangle\big(\langle\mathrm{A}\rangle p \wedge \langle\mathrm{B}\rangle(\langle\mathrm{A}\rangle p \wedge \langle\mathrm{B}\rangle\langle\mathrm{A}\rangle p)\big)$;
- $\mathcal{K}_2, (s_1 s_0)^2 s_1 \not\models \langle\mathrm{B}\rangle\big(\langle\mathrm{A}\rangle p \wedge \langle\mathrm{B}\rangle(\langle\mathrm{A}\rangle p \wedge \langle\mathrm{B}\rangle\langle\mathrm{A}\rangle p)\big)$.

HS makes it possible to distinguish between traces $\rho_1 = s_0^3 s_1 s_0$ and $\rho_2 = s_0 s_1 s_0^3$, which involve the same number of iterations of the same loops, but differ in the order of loop occurrences: $\mathcal{K}_2, \rho_1 \models \langle\mathrm{B}\rangle\big(\langle\mathrm{A}\rangle q \wedge \langle\mathrm{B}\rangle(\langle\mathrm{A}\rangle p \wedge \langle\mathrm{B}\rangle\top)\big)$, but $\mathcal{K}_2, \rho_2 \not\models \langle\mathrm{B}\rangle\big(\langle\mathrm{A}\rangle q \wedge \langle\mathrm{B}\rangle(\langle\mathrm{A}\rangle p \wedge \langle\mathrm{B}\rangle\top)\big)$.
Also $\langle\overline{\mathrm{B}}\rangle$ and $\langle\overline{\mathrm{E}}\rangle$, the inverses of $\langle\mathrm{B}\rangle$ and $\langle\mathrm{E}\rangle$, are branching—in the future and in the past, respectively—just like $\langle\mathrm{A}\rangle$ and $\langle\overline{\mathrm{A}}\rangle$. See Figure 2.3.

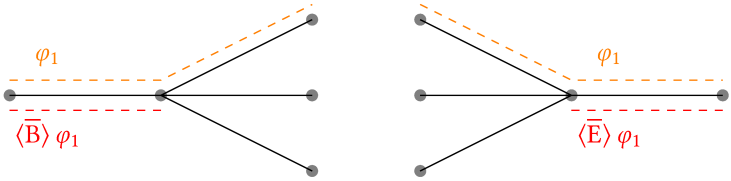


Figure 2.3: The branching semantics of modalities $\langle\overline{\mathrm{B}}\rangle$ and $\langle\overline{\mathrm{E}}\rangle$

Example 2.1.9. In Figure 2.4, we give an example of a finite Kripke structure $\mathcal{K}_{Sched}$ that models the behaviour of a scheduler serving three processes which are continuously requesting the use of a common resource. The initial state (denoted by a double circle) is s_0: no process is served in that state. In any other state s_i and $\bar{s}_i$, with $i \in \{1, 2, 3\}$, the i-th process is served (this is denoted by the fact that p_i holds in those states). For the sake of readability, edges are marked either by r_i, for request(i), or by u_i, for unlock(i). However, edge labels do not have a semantic value, i.e., they are neither part of the structure definition, nor proposition letters; they

are simply used to ease reference to edges.
Process i is served in state s_i, then, after "some time", a transition u_i from s_i to $\overline{s}_i$ is taken; subsequently, process i cannot be served again immediately, as s_i is not directly reachable from $\overline{s}_i$ (the scheduler cannot serve the same process twice in two successive rounds). A transition r_j, with $j \neq i$, from $\overline{s}_i$ to s_j is then taken and process j is served. This structure can be easily generalised to a higher number of processes.

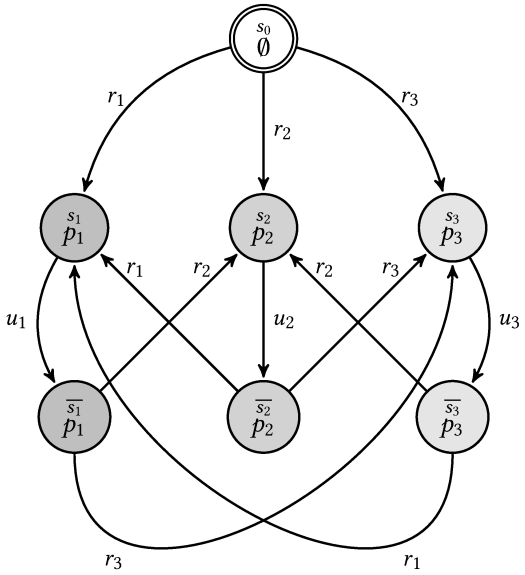


Figure 2.4: The finite Kripke structure $\mathcal{K}_{Sched}$.

We now show how some meaningful properties to be checked over $\mathcal{K}_{Sched}$ can be expressed in HS. In all formulas, we force the validity of the considered property over all legal computation sub-intervals by using modality [E] (all computation sub-intervals are suffixes of at least one initial trace). Moreover, we will use the shorthand $wit_{\geq 2}(\{p_1, p_2, p_3\})$ for the formula

$$(\langle \mathrm{D} \rangle p_1 \wedge \langle \mathrm{D} \rangle p_2) \vee (\langle \mathrm{D} \rangle p_1 \wedge \langle \mathrm{D} \rangle p_3) \vee (\langle \mathrm{D} \rangle p_2 \wedge \langle \mathrm{D} \rangle p_3),$$

which states that there exist at least two sub-intervals such that p_i holds over the former and p_j over the latter, with $i, j \in \{1, 2, 3\}$ and $j \neq i$ (such a formula can be easily generalised to an arbitrary set of proposition letters and to any natural number k).
The truth of the following statements can be easily checked:

- $\mathcal{K}_{Sched} \models [\mathrm{E}]\left(\langle \mathrm{B}\rangle^{4} \top \rightarrow wit_{\geq 2}(\{p_1, p_2, p_3\})\right)$;
- $\mathcal{K}_{Sched} \not\models [\mathrm{E}]\left(\langle \mathrm{B}\rangle^{10} \top \rightarrow \langle \mathrm{D}\rangle\, p_3\right)$;
- $\mathcal{K}_{Sched} \not\models [\mathrm{E}]\left(\langle \mathrm{B}\rangle^{6} \top \rightarrow \langle \mathrm{D}\rangle\, p_1 \wedge \langle \mathrm{D}\rangle\, p_2 \wedge \langle \mathrm{D}\rangle\, p_3\right)$.

The first formula states that in any suffix of an initial trace of length greater than or equal to 5 at least 2 proposition letters are witnessed. $\mathcal{K}_{Sched}$ satisfies the formula since a process cannot be executed twice in a row.
The second formula states that in any suffix of an initial trace of length at least 11, process 3 is executed at least once in some internal states (*non starvation*). $\mathcal{K}_{Sched}$ does not satisfy the formula since the scheduler can avoid executing a process ad libitum.
The third formula states that in any suffix of an initial trace of length greater than or equal to 7, p_1, p_2, p_3 are all witnessed. The only way to satisfy this property is to constrain the scheduler to execute the three processes in a strictly periodic manner (*strict alternation*), that is, $p_i p_j p_k p_i p_j p_k p_i p_j p_k \cdots$, for $i, j, k \in \{1, 2, 3\}$ and $i \neq j \neq k \neq i$, but $\mathcal{K}_{Sched}$ does not meet such a requirement.
This example will be referred to also in the next chapters.

2.2 The fundamental notion of BE_k-descriptor

In the previous section we have shown that, for any given finite Kripke structure $\mathcal{K}$, one can find a corresponding induced abstract interval model $\mathcal{A}_{\mathcal{K}}$, featuring one interval for each trace of $\mathcal{K}$. Since $\mathcal{K}$ has loops (each state must have at least one successor, as the transition relation R is left-total), the number of its traces, and thus the number of intervals of $\mathcal{A}_{\mathcal{K}}$, is infinite.

In this section we prove that, given a finite Kripke structure $\mathcal{K}$ and an HS formula φ, there exists a *finite* representation for $\mathcal{A}_{\mathcal{K}}$, equivalent to $\mathcal{A}_{\mathcal{K}}$ with respect to the satisfiability of φ (in fact, of a class of formulas including φ).

We start with the definition of some basic notions. The first one is the BE-nesting depth of an HS formula.

Definition 2.2.1 (BE-nesting depth of an HS formula)**.** Let ψ be an HS formula. The BE-nesting depth of ψ, denoted by $\mathrm{Nest}_{\mathrm{BE}}(\psi)$, is defined by induction on the structure of the formula as follows:

- $\mathrm{Nest}_{\mathrm{BE}}(p) = 0$ for any proposition letter $p \in \mathcal{AP}$;
- $\mathrm{Nest}_{\mathrm{BE}}(\neg\psi) = \mathrm{Nest}_{\mathrm{BE}}(\psi)$;
- $\mathrm{Nest}_{\mathrm{BE}}(\psi \wedge \varphi) = \max\{\mathrm{Nest}_{\mathrm{BE}}(\psi), \mathrm{Nest}_{\mathrm{BE}}(\varphi)\}$;
- $\mathrm{Nest}_{\mathrm{BE}}(\langle \mathrm{B}\rangle\, \psi) = \mathrm{Nest}_{\mathrm{BE}}(\langle \mathrm{E}\rangle\, \psi) = 1 + \mathrm{Nest}_{\mathrm{BE}}(\psi)$;
- $\mathrm{Nest}_{\mathrm{BE}}(\langle \mathrm{X}\rangle\, \psi) = \mathrm{Nest}_{\mathrm{BE}}(\psi)$, for $X \in \{A, \overline{A}, \overline{B}, \overline{E}\}$.

In the following, we denote by $\text{Nest}_{\text{B}}(\psi)$ the "restriction" of $\text{Nest}_{\text{BE}}(\psi)$ to $\langle \text{B} \rangle$ modality only (i.e., $\text{Nest}_{\text{B}}(\psi)$ accounts only for the nesting depth of $\langle \text{B} \rangle$ and disregards $\langle \text{E} \rangle$). Clearly $\text{Nest}_{\text{B}}(\psi) = \text{Nest}_{\text{BE}}(\psi)$ if ψ is devoid of occurrences of $\langle \text{E} \rangle$. The analogous for $\text{Nest}_{\text{E}}(\psi)$.

Making use of the notion of BE-nesting depth of a formula, we can define a relation of k-equivalence over traces.

Definition 2.2.2 (k-equivalence)**.** Let $\mathcal{K}$ be a finite Kripke structure and ρ and ρ' be two traces in $\text{Trc}_{\mathcal{K}}$. We say that ρ and ρ' are k-equivalent if and only if, for every HS formula ψ with $\text{Nest}_{\text{BE}}(\psi) = k$, we have $\mathcal{K}, \rho \models \psi$ if and only if $\mathcal{K}, \rho' \models \psi$.

It can be easily proved that k-equivalence "propagates downwards".

Proposition 2.2.3. *Let $\mathcal{K}$ be a finite Kripke structure and ρ and ρ' be two traces in* $\text{Trc}_{\mathcal{K}}$*. If ρ and ρ' are k-equivalent, then they are h-equivalent, for all $0 \leq h \leq k$.*

Proof. Let us assume that $\mathcal{K}, \rho \models \psi$, with $0 \leq \text{Nest}_{\text{BE}}(\psi) \leq k$. Consider the formula $\langle \text{B} \rangle^k \top$, whose BE-nesting depth is equal to k. It trivially holds that either $\mathcal{K}, \rho \models \langle \text{B} \rangle^k \top$ or $\mathcal{K}, \rho \models \neg \langle \text{B} \rangle^k \top$. In the first case, we have that $\mathcal{K}, \rho \models \langle \text{B} \rangle^k \top \wedge \psi$. Since $\text{Nest}_{\text{BE}}\left(\langle \text{B} \rangle^k \top \wedge \psi\right) = k$, from the hypothesis, it follows that $\mathcal{K}, \rho' \models \langle \text{B} \rangle^k \top \wedge \psi$, and thus $\mathcal{K}, \rho' \models \psi$. The other case is symmetric. □

We are now ready to introduce the notion of *descriptor*, which will play a fundamental role in the definition of finite abstract interval models.

Definition 2.2.4 (B-descriptor and E-descriptor)**.** Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure. A B-descriptor (resp., E-descriptor) is a labelled tree $\mathcal{D} = (\mathcal{V}, \mathcal{E}, \lambda)$, where $\mathcal{V}$ is a finite set of vertices, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, and $\lambda : \mathcal{V} \to S \times 2^S \times S$ is a node labelling function, that satisfies the following conditions:

1. for all $(v, v') \in \mathcal{E}$, with $\lambda(v) = (s_{in}, A, s_{fin})$ and $\lambda(v') = (s'_{in}, A', s'_{fin})$, it holds that $A' \subseteq A$, $s_{in} = s'_{in}$, and $s'_{fin} \in A$ (resp., $A' \subseteq A$, $s_{fin} = s'_{fin}$, and $s'_{in} \in A$);
2. for all pairs of edges $(v, v'), (v, v'') \in \mathcal{E}$, if the subtree rooted in v' is isomorphic to the subtree rooted in v'', then $v' = v''$ (here and in the following, we write subtree for maximal subtree).

(2.) of Definition 2.2.4 simply states that no two subtrees, whose roots are siblings, can be isomorphic (note that λ is taken into account).

For $X \in \{B, E\}$, the *depth* of an X-descriptor $(\mathcal{V}, \mathcal{E}, \lambda)$ is the depth of the tree $(\mathcal{V}, \mathcal{E})$. We call an X-descriptor of depth $k \in \mathbb{N}$ an X_k-descriptor. An X_0-descriptor $\mathcal{D}$ consists of its root only, which is denoted by root($\mathcal{D}$). A label of a node will be referred to as a *descriptor element*. Hereafter, two descriptors will be considered *equal up to isomorphism*.

The following proposition holds.

Proposition 2.2.5. *Given a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$, for all $k \in \mathbb{N}$ there exists a* finite number *of possible B_k-descriptors (resp., E_k-descriptors).*

Proof. We consider the case of B_k-descriptors (the case of E_k-descriptors is analogous). For $k = 0$, there are at most $|S| \cdot 2^{|S|} \cdot |S|$ pairwise distinct B_0-descriptors. As for the inductive step, let us assume h to be the number of pairwise distinct B-descriptors of depth at most k. The number of B_{k+1}-descriptors is at most $|S| \cdot 2^{|S|} \cdot |S| \cdot 2^h$ (there are at most $|S| \cdot 2^{|S|} \cdot |S|$ possible choices for the root, which can have any subset of the h B-descriptors of depth at most k as subtrees). By König's lemma, they are all finite as their depth is $k + 1$ and the root has a finite number of children (no two subtrees of the root can be isomorphic). □

Proposition 2.2.5 gives an upper bound to the number of distinct B_k-descriptors (resp., E_k-descriptors), and thus to the number of nodes of each B_{k+1}-descriptor (resp., E_{k+1}-descriptors), for $k \in \mathbb{N}$, which is *not elementary* with respect to $|S|$ and k, being $|S|$ the exponent and k the height of the exponential tower. As a matter of fact, this is a very rough upper bound, since some descriptors may not have depth $k + 1$ and some of the "generated" trees might not even fulfill the definition of descriptor.

We show now how B-descriptors and E-descriptors can be used to extract relevant information from the traces of a finite Kripke structure to use in MC.

Let $\mathcal{K}$ be a finite Kripke structure and ρ be a trace in $\mathrm{Trc}_{\mathcal{K}}$, with $|\rho| \geq 2$. For any $k \geq 0$, the label of the root of both the B_k-descriptor and E_k-descriptor for ρ is the triple $(\mathrm{fst}(\rho), \mathrm{intstates}(\rho), \mathrm{lst}(\rho))$. The root of the B_k-descriptor has a child for each prefix ρ', with $|\rho'| \geq 2$, of ρ, labelled with $(\mathrm{fst}(\rho'), \mathrm{intstates}(\rho'), \mathrm{lst}(\rho'))$. Such a construction is then iteratively applied to the children of the root until either depth k is reached or a trace of length 2 is being considered on a node. The length-1 prefix of ρ can be recovered as a special case, being just $\mathrm{fst}(\rho)$. We should associate with it a descriptor element $(\mathrm{fst}(\rho), \emptyset, \perp)$ as a child of the root, where $\perp$ is just a marker for "no state". However, in the rest of this section, for a more uniform and elegant characterization of descriptors, we assume *traces, suffixes and prefixes to have length at least 2*, as those having length 1 can always be treated as special (trivial) cases. The E_k-descriptor is built in a similar way by considering the suffixes of ρ.

In general B- and E-descriptors do not convey enough information to determine which trace they were built from (this will be clear shortly). However, they can be exploited to determine which HS formulas are satisfied by the trace from which they have been built:

- to check the satisfiability of proposition letters, they keep information about initial, final, and internal states of the trace;
- for $\langle \mathrm{A} \rangle \psi$ and $\langle \overline{\mathrm{A}} \rangle \psi$ formulas, they store the final and initial states of the trace;
- for $\langle \mathrm{B} \rangle \psi$ formulas, the B-descriptor keeps information on all of its prefixes;
- for $\langle \mathrm{E} \rangle \psi$ formulas, the E-descriptor keeps information on all of its suffixes;
- no additional information is needed for $\langle \overline{\mathrm{B}} \rangle \psi$ and $\langle \overline{\mathrm{E}} \rangle \psi$ formulas.

Let $\mathcal{K}$ be a finite Kripke structure. The B_k-descriptor (resp., E_k-descriptor) for a trace ρ in $\mathrm{Trc}_{\mathcal{K}}$ is formally defined as follows.

Definition 2.2.6 (B-/E-descriptor for a trace)**.** Let $\mathcal{K}$ be a finite Kripke structure, ρ be a trace in $\mathrm{Trc}_{\mathcal{K}}$, and $k \in \mathbb{N}$. The B_k-descriptor (resp., E_k-descriptor) for ρ is inductively defined as follows:

- for $k = 0$, the B_k-descriptor (resp., E_k-descriptor) for ρ is the tree $\mathcal{D} = (\mathrm{root}(\mathcal{D}), \emptyset, \lambda)$, where

$$\lambda(\mathrm{root}(\mathcal{D})) = (\mathrm{fst}(\rho), \mathrm{intstates}(\rho), \mathrm{lst}(\rho));$$

- for $k > 0$, the B_k-descriptor (resp., E_k-descriptor) for ρ is the tree $\mathcal{D} = (\mathcal{V}, \mathcal{E}, \lambda)$, where

$$\lambda(\mathrm{root}(\mathcal{D})) = (\mathrm{fst}(\rho), \mathrm{intstates}(\rho), \mathrm{lst}(\rho)),$$

  which satisfies the following conditions:

  1. for each prefix (resp., suffix) ρ' of ρ, there exists $v \in \mathcal{V}$ such that $(\mathrm{root}(\mathcal{D}), v) \in \mathcal{E}$ and the subtree rooted in v is the B_{k-1}-descriptor (resp., E_{k-1}-descriptor) for ρ';
  2. for each vertex $v \in \mathcal{V}$ such that $(\mathrm{root}(\mathcal{D}), v) \in \mathcal{E}$, there exists a prefix (resp., suffix) ρ' of ρ such that the subtree rooted in v is the B_{k-1}-descriptor (resp., E_{k-1}-descriptor) for ρ';
  3. for all pairs of edges $(\mathrm{root}(\mathcal{D}), v'), (\mathrm{root}(\mathcal{D}), v'') \in \mathcal{E}$, if the subtree rooted in v' is isomorphic to the subtree rooted in v'', then $v' = v''$.

Any B_k-descriptor (resp., E_k-descriptor) for some trace satisfies the conditions of Definition 2.2.4 (in particular (1.)), but not vice versa. See the next example.

Example 2.2.7. Consider, for instance, the B_1-descriptor reported in Figure 2.5. It is built on a set of states S including at least states s_0, s_1, s_2 and s_3, and it satisfies both conditions of Definition 2.2.4. However, no trace of a finite Kripke structure can be described by it, as no trace may feature two prefixes to be associated with the first two children of the root.

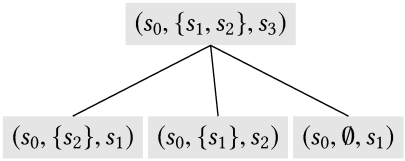


Figure 2.5: B_1-descriptor devoid of a corresponding trace (in any Kripke structure).

Example 2.2.8. In Figure 2.6 and 2.7, we depict the B_2- and E_2-descriptors for the trace $s_0s_1s_0s_0s_1$ of the Kripke structure $\mathcal{K}_2$ of Figure 2.1.

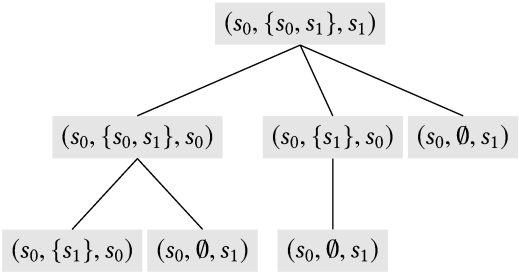


Figure 2.6: B_2-descriptor for the trace $s_0s_1s_0s_0s_1$ of $\mathcal{K}_2$.

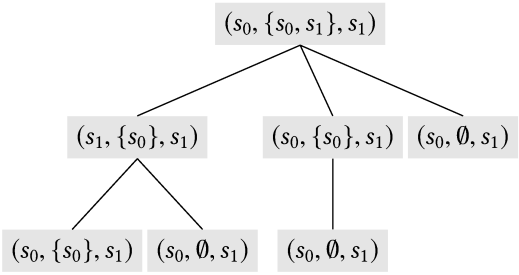


Figure 2.7: E_2-descriptor for the trace $s_0s_1s_0s_0s_1$ of $\mathcal{K}_2$.

Example 2.2.9. In Figure 2.8 we show the B_2-descriptor for the trace $\rho = s_0s_1s_0s_0s_0s_0s_1$ of $\mathcal{K}_2$. It is worth noticing that there exist two distinct prefixes of the trace ρ, that is, the traces $\rho' = s_0s_1s_0s_0s_0s_0$ and $\rho'' = s_0s_1s_0s_0s_0$, which have the same B_1-descriptor. Since, according to Definition 2.2.6, no tree can occur more than once as a subtree of the same node (in this example, the root), in the B_2-descriptor for ρ, the prefixes ρ' and ρ'' are represented by the same tree (the first subtree of the root on the left). This shows that, in general, the root of a descriptor for a trace with h proper prefixes may have less than h children.

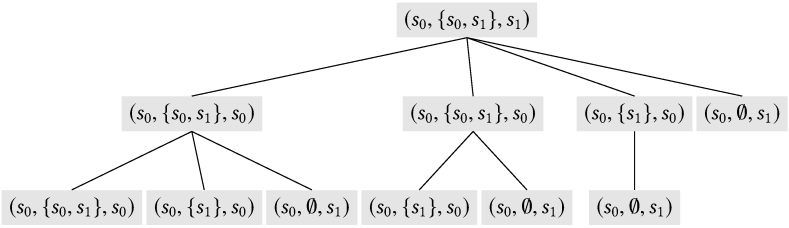


Figure 2.8: The B_2-descriptor for the trace $s_0s_1s_0s_0s_0s_0s_1$ of $\mathcal{K}_2$.

Example 2.2.10. This example shows that not all of the B_k-descriptors that can be generated from the set of states of a given finite Kripke structure are B_k-descriptors for some trace of that structure (the same fact is true for E_k-descriptors).
Let us consider the finite Kripke structure $\mathcal{K}$ in Figure 2.9 and the B_1-descriptor $\mathcal{D}_{B_1}$ in Figure 2.10. By inspecting $\mathcal{D}_{B_1}$, it can be easily checked that it can be the B_1-descriptor for traces of the form $s_0s_1^hs_3^2$, with $h \geq 2$, only. However, no trace of this form can be obtained by unravelling $\mathcal{K}$.

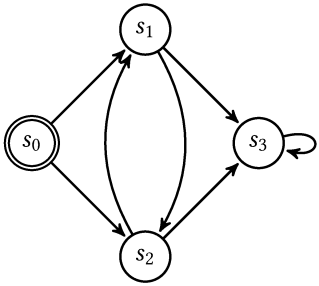


Figure 2.9: A finite Kripke structure $\mathcal{K}$.

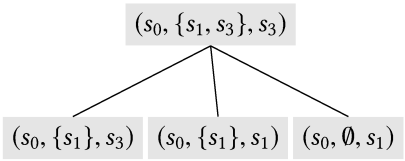


Figure 2.10: $\mathcal{D}_{B_1}$: a B_1-descriptor not corresponding to any of the traces of $\mathcal{K}$ in Figure 2.9.

To check an HS formula against a given finite Kripke structure we actually need to account for both the *started-by* (B) and *finished-by* (E) relations at the same time. To this end, we introduce BE_k-descriptors for traces. Given a finite Kripke structure $\mathcal{K}$ and a trace ρ in $\mathrm{Trc}_{\mathcal{K}}$, the BE_k-descriptor for ρ can be obtained from a suitable merging of its B_k-descriptor and E_k-descriptor. It can be viewed as a sort of "product" of the B_k-descriptor and the E_k-descriptor for ρ, and it is formally defined as follows:

Definition 2.2.11 (BE-descriptor for a trace)**.** Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure, ρ be a trace in $\mathrm{Trc}_{\mathcal{K}}$, and $k \in \mathbb{N}$. The BE_k-descriptor for ρ is a labelled tree $\mathcal{D} = (\mathcal{V}, \mathcal{E}, \lambda)$, where $\mathcal{V}$ is a finite set of vertices, $\mathcal{E} = \mathcal{E}_B \cup \mathcal{E}_E$, with $\mathcal{E}_B \subseteq \mathcal{V} \times \mathcal{V}$ the set of "B-edges", $\mathcal{E}_E \subseteq \mathcal{V} \times \mathcal{V}$ the set of "E-edges", and $\mathcal{E}_B \cap \mathcal{E}_E = \emptyset$, and $\lambda : \mathcal{V} \to S \times 2^S \times S$, which is inductively defined on $k \in \mathbb{N}$ as follows:

- for $k = 0$, the BE_k-descriptor for ρ is $\mathcal{D} = (\mathrm{root}(\mathcal{D}), \emptyset, \lambda)$, where
$$\lambda(\mathrm{root}(\mathcal{D})) = (\mathrm{fst}(\rho), \mathrm{intstates}(\rho), \mathrm{lst}(\rho)).$$
- for $k > 0$, the BE_k-descriptor for ρ is $\mathcal{D} = (\mathcal{V}, \mathcal{E}, \lambda)$ with
$$\lambda(\mathrm{root}(\mathcal{D})) = (\mathrm{fst}(\rho), \mathrm{intstates}(\rho), \mathrm{lst}(\rho))$$
which satisfies the following conditions:
  - **1a.** for each prefix ρ' of ρ, there exists $v \in \mathcal{V}$ such that $(\mathrm{root}(\mathcal{D}), v) \in \mathcal{E}_B$ and the subtree rooted in v is the BE_{k-1}-descriptor for ρ';
  - **1b.** for each vertex $v \in \mathcal{V}$ such that $(\mathrm{root}(\mathcal{D}), v) \in \mathcal{E}_B$, there exists a prefix ρ' of ρ such that the subtree rooted in v is the BE_{k-1}-descriptor for ρ';
  - **1c.** for all pairs of edges $(\mathrm{root}(\mathcal{D}), v'), (\mathrm{root}(\mathcal{D}), v'') \in \mathcal{E}_B$, if the subtree rooted in v' is isomorphic to the subtree rooted in v'', then $v' = v''$;

2a. for each suffix ρ'' of ρ, there exists $v \in \mathcal{V}$ such that $(\text{root}(\mathcal{D}), v) \in \mathcal{E}_E$ and the subtree rooted in v is the BE_{k-1}-descriptor for ρ'';

2b. for each vertex $v \in \mathcal{V}$ such that $(\text{root}(\mathcal{D}), v) \in \mathcal{E}_E$, there exists a suffix ρ'' of ρ such that the subtree rooted in v is the BE_{k-1}-descriptor for ρ'';

2c. for all pairs of edges $(\text{root}(\mathcal{D}), v'), (\text{root}(\mathcal{D}), v'') \in \mathcal{E}_E$, if the subtree rooted in v' is isomorphic to the subtree rooted in v'', then $v' = v''$.

From Definition 2.2.11, it easily follows that for all $(v, v') \in \mathcal{E}_B$, with $\lambda(v) = (s_{in}, A, s_{fin})$ and $\lambda(v') = (s'_{in}, A', s'_{fin})$, we have $A' \subseteq A$, $s_{in} = s'_{in}$, and $s'_{fin} \in A$, and for all $(v, v') \in \mathcal{E}_E$, with $\lambda(v) = (s_{in}, A, s_{fin})$ and $\lambda(v') = (s'_{in}, A', s'_{fin})$, we have $A' \subseteq A$, $s_{fin} = s'_{fin}$ and $s'_{in} \in A$.

Example 2.2.12. In Figure 2.11 on page 36, with reference to the finite Kripke structure $\mathcal{K}_2$ of Figure 2.1, we give an example of a BE_2-descriptor. B-edges are represented by solid lines, while E-edges are represented by dashed lines. It is worth pointing out that the BE_2-descriptor of Figure 2.11 turns out to be the BE_2-descriptor for both the trace $\rho = s_0 s_1 s_0^3 s_1$ and the trace $\rho' = s_0 s_1 s_0^4 s_1$ (and many others). As we will see very soon, this is not an exception, but the rule: different traces of a finite Kripke structure are described by the same BE-descriptor. Notice also that it features two isomorphic subtrees for the same node (the root). They both consist of a single node, labelled with $(s_0, \emptyset, s_1)$. However, this does not violate Definition 2.2.11 since one of them is connected to the parent via a B-edge and the other via an E-edge.

It can be easily checked that the BE_{k-1}-descriptor $\mathcal{D}_{BE_{k-1}}$ for a trace ρ can be obtained from the BE_k-descriptor $\mathcal{D}_{BE_k}$ for ρ by removing the nodes at depth k (if any) and the isomorphic subtrees possibly resulting from such a removal (see (1c.) of Definition 2.2.11).

B_k and E_k-descriptors can be recovered from BE_k ones. The B_k-descriptor $\mathcal{D}_{B_k}$ for a trace ρ can be obtained from the BE_k-descriptor $\mathcal{D}_{BE_k}$ for ρ by pruning it in such a way that only those vertices of $\mathcal{D}_{BE_k}$ which are connected to the root via paths consisting of B-edges only are maintained (the set of edges of $\mathcal{D}_{B_k}$ and its labelling function can be obtained by restricting those of $\mathcal{D}_{BE_k}$ to the nodes of $\mathcal{D}_{B_k}$). The E_k-descriptor $\mathcal{D}_{E_k}$ of ρ can be obtained in a similar way.

We focus now our attention on the relationships between the traces obtained from the unravelling of a finite Kripke structure and their BE_k-descriptors. A key

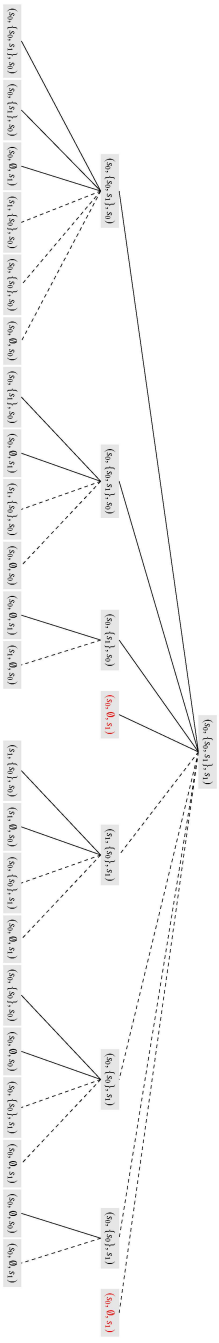


Figure 2.11: An example of BE_2-descriptor.

observation is that, even though the number of traces of a finite Kripke structure $\mathcal{K}$ is infinite, for any $k \in \mathbb{N}$ the set of BE_k-descriptors for its traces is finite. This is an immediate consequence of Definition 2.2.11 and Proposition 2.2.5. Thus, *at least one BE_k-descriptor must be the BE_k-descriptor for infinitely many traces.*

BE_k-descriptors naturally induce an *equivalence relation of finite index* over the set of traces of a finite Kripke structure, called k-descriptor equivalence relation.

Definition 2.2.13 (k-descriptor equivalence)**.** Let $\mathcal{K}$ be a finite Kripke structure, ρ, ρ' be two traces in $\mathrm{Trc}_{\mathcal{K}}$, and $k \in \mathbb{N}$. We say that ρ and ρ' are *k-descriptor equivalent*, denoted by $\rho \sim_k \rho'$, if and only if the BE_k-descriptors for ρ and ρ' coincide.

In the next section we will see that, for any given pair of traces $\rho, \rho' \in \mathrm{Trc}_{\mathcal{K}}$, if $\rho \sim_k \rho'$, then ρ and ρ' are k-equivalent (see Definition 2.2.2).

2.3 The decidability proof

In this section we will see the main results behind the decidability of the MC problem for HS formulas over finite Kripke structures. We refer to [Mol+16a] for further details and missing proofs.

As a preliminary step, we state a right extension property. Let $\mathcal{K}$ be a finite Kripke structure, $k \in \mathbb{N}$, and ρ and ρ' be two traces in $\mathrm{Trc}_{\mathcal{K}}$ with the same BE_k-descriptor (and thus, in particular, $\mathrm{lst}(\rho) = \mathrm{lst}(\rho')$). The property states that if we extend ρ and ρ' "to the right" with the same trace $\overline{\rho}$ in $\mathrm{Trc}_{\mathcal{K}}$, with $(\mathrm{lst}(\rho), \mathrm{fst}(\overline{\rho})) \in R$, then the resulting traces $\rho \cdot \overline{\rho}$ and $\rho' \cdot \overline{\rho}$ (both belonging to $\mathrm{Trc}_{\mathcal{K}}$) have the same BE_k-descriptor as well. An analogous property holds for the extension of the two traces ρ and ρ' "to the left", which guarantees that $\overline{\rho} \cdot \rho$ and $\overline{\rho} \cdot \rho'$ have the same BE_k-descriptor (left extension property).

Proposition 2.3.1 (Right extension property)**.** *Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure and let ρ and ρ' be two traces in $\mathrm{Trc}_{\mathcal{K}}$ with $\rho \sim_k \rho'$. For any trace $\overline{\rho}$ in $\mathrm{Trc}_{\mathcal{K}}$, with $(\mathrm{lst}(\rho), \mathrm{fst}(\overline{\rho})) \in R$, the two traces $\rho \cdot \overline{\rho}$ and $\rho' \cdot \overline{\rho}$ belong to $\mathrm{Trc}_{\mathcal{K}}$ and $\rho \cdot \overline{\rho} \sim_k \rho' \cdot \overline{\rho}$.*

The next theorem proves that, for any pair of traces $\rho, \rho' \in \mathrm{Trc}_{\mathcal{K}}$, if $\rho \sim_k \rho'$, then ρ and ρ' are k-equivalent (see Definition 2.2.2).

Theorem 2.3.2. *Let $\mathcal{K}$ be a finite Kripke structure, ρ and ρ' be two traces in $\mathrm{Trc}_{\mathcal{K}}$, and ψ be a HS formula with $\mathrm{Nest}_{\mathrm{BE}}(\psi) = k$. If $\rho \sim_k \rho'$ then $\mathcal{K}, \rho \models \psi \iff \mathcal{K}, \rho' \models \psi$.*

Since the set of BE_k-descriptors for the traces of a finite Kripke structure $\mathcal{K}$ is finite, i.e., the equivalence relation $\sim_k$ has a finite index, there always exists a finite number of BE_k-descriptors that "satisfy" a HS formula ψ with $\mathsf{Nest}_{\mathrm{BE}}(\psi) = k$ (this can be formally proved by a quotient construction [Mol+16a]).

Thus we can reduce the MC problem for HS over finite Kripke structures to MC for multi-modal finite Kripke structures, whose nodes are all possible *witnessed* descriptors with depth up to k, and there is a distinct accessibility relation for each one of the HS modalities A, B, E, $\overline{A}$, $\overline{B}$ and $\overline{E}$. Since the MC problem for multi-modal finite Kripke structures is decidable (in polynomial time with respect to the size of the multi-modal Kripke structure and to the length of the formula [Gab87; Lan06]), decidability of the MC problem for HS against finite Kripke structures follows, provided that we can effectively determine which are the descriptors witnessed (by traces) in the structure, as shown in the proof of the next theorem.

Theorem 2.3.3. *The MC problem for HS formulas over finite Kripke structures is decidable (with* nonelementary *complexity).*

Proof. Let $\mathcal{K}$ be a finite Kripke structure and let φ be the HS formula to check, with $\mathsf{Nest}_{\mathrm{BE}}(\varphi) = k$.

We first prove that, in order to select the BE_h-descriptors, with $0 \leq h \leq k$, witnessed by some trace in $\mathcal{K}$, we can restrict ourselves to traces devoid of prefixes associated with the same BE_k-descriptor. Let $\rho \in \mathrm{Trc}_{\mathcal{K}}$ and let ρ', ρ'' be two prefixes of ρ, with $|\rho''| < |\rho'| \leq |\rho|$ (notice that we allow ρ' to coincide with ρ). Moreover, let $\rho = \rho' \cdot \tilde{\rho}$, for some $\tilde{\rho}$ with $|\tilde{\rho}| \geq 1$ (in case $|\rho| = |\rho'|$, $\rho = \rho'$). If the BE_k-descriptors for ρ' and ρ'' are the same then, by Proposition 2.3.1, it holds that the BE_k-descriptor for $\rho'' \cdot \tilde{\rho}$ is equal to the one for $\rho' \cdot \tilde{\rho} = \rho$. Hence, we can safely replace ρ by the k-descriptor equivalent shorter trace $\rho'' \cdot \tilde{\rho}$. We can iterate such a contraction process until there are no more pairs of prefixes associated with the same BE_k-descriptor.[1]

Now, Proposition 2.2.5 provides a nonelementary upper bound to the number α of distinct BE_h-descriptors, with $0 \leq h \leq k$ (as well as to their size), with respect to the size of $\mathcal{K}$ and k. A bound on the length of the traces in $\mathrm{Trc}_{\mathcal{K}}$ that we need to consider in order to determine the witnessed BE_h-descriptors in an effective way immediately follows: it is $1 + \alpha$.

Hence, in order to generate all the witnessed BE_h-descriptors, with $0 \leq h \leq k$, it suffices to list, for all states s of $\mathcal{K}$, all the traces starting from s, ordered by length, until the above bound is reached, and then to build the corresponding BE_h-descriptors, with $0 \leq h \leq k$.

[1] As a matter of fact, the same argument can be given by referring to suffixes instead of prefixes. Anyway, as one can easily see, making use of both the right extension and the left extension properties does not allow us to improve the claimed bound.

We conclude that the derived MC problem for multi-modal finite Kripke structures has to be solved over a model whose size has a nonelementary upper bound. □

2.4 EXPSPACE-hardness of MC for BE

In the previous section we have seen that MC for HS can be decided in nonelementary time. Obviously, this does *not* imply that such problem is *provably* nonelementary, meaning that no algorithm with elementary complexity may exist for it. Currently, *the existence of such an algorithm is still an open issue.*

Conversely, the best complexity lower bound known to date is **EXPSPACE**-hardness. This derives from the HS fragment BE, whose modalities can express properties of both interval prefixes and suffixes simultaneously: we now prove that the MC problem for BE is **EXPSPACE**-hard; since MC for full HS is clearly at least as hard as MC for BE, such a lower bound immediately propagates to full HS. The result is obtained by a polynomial-time reduction from a *domino-tiling problem for grids with rows of single exponential length* [Har12] to MC for BE.

We start with the definition of the domino-tiling problem. An instance $\mathcal{I}$ of a domino-tiling problem for grids with rows of single exponential length is a tuple $\mathcal{I} = (C, \Delta, n, d_{init}, d_{final})$, where C is a finite set of colors, $\Delta \subseteq C^4$ is a set of tuples $(c_{down}, c_{left}, c_{up}, c_{right})$ of four colors, called *domino-types*, $n > 0$ is a natural number encoded in *unary*, and $d_{init}, d_{final} \in \Delta$ are two distinguished domino-types (respectively, the initial and final domino-types). The *size* of $\mathcal{I}$ is defined as $|C| + |\Delta| + n$.

Intuitively, a tiling of a grid is a color labelling of the edges of each cell (see Figure 2.12). Formally, a *tiling of* $\mathcal{I}$ is a mapping $f : [0, k] \times [0, 2^n - 1] \to \Delta$, for some $k \geq 0$, that satisfies the following constraints:

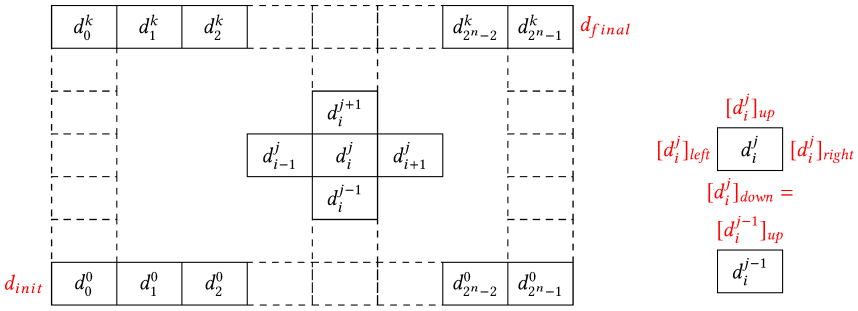


Figure 2.12: A (generic) instance of the domino-tiling problem, where d_j^i denotes $f(i, j)$.

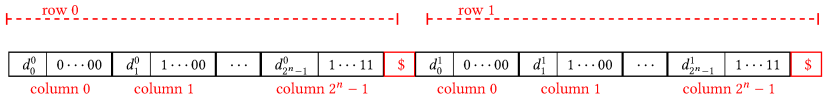


Figure 2.13: Encoding of a tiling as a word, where d_j^i denotes $f(i, j)$.

- two adjacent cells in a row have the same color on the shared edge, namely, for all $(i, j) \in [0, k] \times [0, 2^n - 2]$, $[f(i, j)]_{right} = [f(i, j + 1)]_{left}$ (*horizontal requirement*);
- two adjacent cells in a column have the same color on the shared edge, namely, for all $(i, j) \in [0, k - 1] \times [0, 2^n - 1]$, $[f(i, j)]_{up} = [f(i + 1, j)]_{down}$ (*vertical requirement*);
- $f(0, 0) = d_{init}$ (*initialization requirement*) and $f(k, 2^n - 1) = d_{final}$ (*acceptance requirement*).

Checking the existence (respectively, non-existence) of a tiling of $\mathcal{I}$ is an **EXPSPACE**-complete problem [Har12].

We now show how the domino-tiling problem can be reduced in polynomial time to the MC problem for BE. In particular, we show how to build in polynomial time a finite Kripke structure $\mathcal{K}_\mathcal{I}$ and a BE formula $\varphi_\mathcal{I}$ such that there exists an initial trace of $\mathcal{K}_\mathcal{I}$ satisfying $\varphi_\mathcal{I}$ if and only if there exists a tiling of $\mathcal{I}$. Hence, $\mathcal{K}_\mathcal{I} \models \neg\varphi_\mathcal{I}$ if and only if there is no tiling of $\mathcal{I}$.

The encoding of tilings exploits the set of proposition letters $\mathcal{AP} = \Delta \cup \{\$, 0, 1\}$. Proposition letters in $\{0, 1\}$ are used for the binary encoding of the value of an n-bit counter numbering the cells of a row of a tiling, while the proposition letter \$ is used as a separator. In particular, a cell with content $d \in \Delta$ and column number $j \in [0, 2^n - 1]$ is encoded by the word of length $n + 1$ over $\mathcal{AP}$ given by $d\, b_1 \cdots b_n$, where $b_1 \cdots b_n$ is the binary encoding of the column number j (b_n being the most significant bit). A row is then represented by the word listing the encodings of cells from left to right, and a tiling f consisting of $k + 1$ rows is encoded by the finite word $r_0\$r_1 \cdots \$r_k$, where r_i is the encoding of the i-th row of f, for all $i \in [0, k]$. See Figure 2.13 for a graphical account of a word encoding of a tiling.

The Kripke structure $\mathcal{K}_\mathcal{I}$ is trivially defined as

$$\mathcal{K}_\mathcal{I} = (\mathcal{AP}, \mathcal{AP}, \mathcal{AP} \times \mathcal{AP}, \mu, d_{init}),$$

where $\mu(p) = \{p\}$, for each $p \in \mathcal{AP}$. Thus, the initial traces of $\mathcal{K}_\mathcal{I}$ correspond to the finite words over $\mathcal{AP}$ which start with the initial domino type d_{init}.

In order to build the BE formula $\varphi_\mathcal{I}$, we use some auxiliary formulas, namely, $length_i$, $beg(p)$, $end(p)$, ϕ_{cell}, and $\theta_j(b, b')$, where $i \in [1, 2n + 2]$, $j \in [2, n + 1]$, $p \in \mathcal{AP}$, and $b, b' \in \{0, 1\}$.

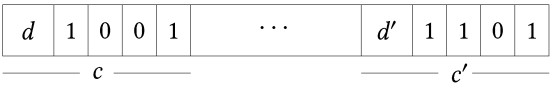


Figure 2.14: Encoding of a trace ρ starting with a cell $c = (d\,1001)$ and ending with a cell $c' = (d'\,1101)$ (here, $n = 4$). The formula $\theta_2(1,1)$ is satisfied by ρ, while $\theta_3(1,0)$ is not.

○

The formula $length_i$, already presented in Example 2.1.7, has size linear in i and characterizes the traces having length i:

$$length_i = [\mathrm{B}]^i \bot \wedge \langle \mathrm{B} \rangle^{i-1} \top.$$

The formula $beg(p)$ (resp., $end(p)$) captures the traces of $\mathcal{K}$ which start (resp., end) in the state p:

$$beg(p) = (p \wedge length_1) \vee \langle \mathrm{B} \rangle (p \wedge length_1), \qquad end(p) = (p \wedge length_1) \vee \langle \mathrm{E} \rangle (p \wedge length_1).$$

The formula ϕ_{cell} captures the traces of $\mathcal{K}_I$ which encode cells:

$$\phi_{cell} = length_{n+1} \wedge \Big(\bigvee_{d \in \Delta} beg(d) \Big) \wedge [\mathrm{E}](beg(0) \vee beg(1)).$$

Finally, for all $j \in [2, n+1]$ and $b, b' \in \{0,1\}$, the formula $\theta_j(b,b')$ is defined as:

$$\theta_j(b,b') = \langle \mathrm{B} \rangle (length_j \wedge end(b)) \wedge \langle \mathrm{E} \rangle (length_{n-j+2} \wedge beg(b')).$$

It is satisfied by a trace ρ if $|\rho| \geq j+1$, $|\rho| \geq n-j+3$, $\rho(j) = b$, and $\rho(|\rho|-n+j-1) = b'$. In particular, for a trace ρ starting with a cell c and ending with a cell c', $\theta_j(b,b')$ is satisfied by ρ if the $(j-1)$-th bit of c is b and the $(j-1)$-th bit of c' is b'. See Figure 2.14 for an example.

Additionally, we use the derived operator $\langle \mathrm{G} \rangle$ and its dual $[\mathrm{G}]$, which allow us to select arbitrary subtraces of the given trace, including the trace itself:

$$\langle \mathrm{G} \rangle \psi = \psi \vee \langle \mathrm{B} \rangle \psi \vee \langle \mathrm{E} \rangle \psi \vee \langle \mathrm{B} \rangle \langle \mathrm{E} \rangle \psi.$$

The formula $\varphi_{\mathcal{I}}$ is defined as follows:

$$\varphi_{\mathcal{I}} = \varphi_b \wedge \varphi_{req} \wedge \varphi_{inc} \wedge \varphi_{rr} \wedge \varphi_{rc}.$$

The conjunct φ_b checks that the given trace starts with a cell with content d_{init} and column number 0, and ends with a cell with content d_{final} and column number $2^n - 1$:

$$\varphi_b = \langle \mathrm{B} \rangle \phi_{cell} \wedge beg(d_{init}) \wedge \langle \mathrm{E} \rangle (\phi_{cell} \wedge beg(d_{final})) \wedge \bigwedge_{j=2}^{n+1} \theta_j(0,1).$$

The conjunct φ_{req} ensures the following two requirements: (i) each occurrence of \$ in the given trace is followed by a cell with column number 0 and (ii) each cell c in the given trace is followed either by another cell, or by the separator \$, and in the latter case c has column number $2^n - 1$. The first requirement is encoded by the formula:

$$[\mathrm{G}]\big((\mathit{length}_{n+2} \wedge \mathit{beg}(\$)) \longrightarrow \langle \mathrm{E}\rangle(\phi_{cell} \wedge [\mathrm{E}]\,\mathit{beg}(0))\big);$$

the second one by the formula:

$$[\mathrm{G}]\Big\{(\mathit{length}_{n+2} \wedge \bigvee_{d\in\Delta} \mathit{beg}(d)) \longrightarrow \Big(\langle \mathrm{B}\rangle\, \phi_{cell} \;\wedge\; (\mathit{end}(\$) \vee \bigvee_{d\in\Delta} \mathit{end}(d)) \;\wedge\; (\mathit{end}(\$) \longrightarrow [\mathrm{E}](\mathit{beg}(\$) \vee \mathit{beg}(1)))\Big)\Big\}.$$

The conjunct φ_{inc} checks that adjacent cells along the given trace have consecutive columns numbers:

$$\varphi_{inc} = [\mathrm{G}]\Big(\phi_{two_cells} \longrightarrow \bigvee_{j=2}^{n+1}\Big[\theta_j(0,1) \wedge \bigwedge_{h=2}^{j-1} \theta_h(1,0) \wedge \bigwedge_{h=j+1}^{n+1} \bigvee_{b\in\{0,1\}} \theta_h(b,b)\Big]\Big),$$

where ϕ_{two_cells} is given by $\mathit{length}_{2n+2} \wedge \langle \mathrm{B}\rangle\, \phi_{cell} \wedge \langle \mathrm{E}\rangle\, \phi_{cell}$. Note that φ_{req} and φ_{inc} ensure that column numbers are correctly encoded.

The conjunct φ_{rr} checks that adjacent cells in a row have the same color on the shared edge:

$$\varphi_{rr} = [\mathrm{G}]\Big(\phi_{two_cells} \longrightarrow \bigvee_{(d,d')\in\Delta\times\Delta \mid d_{right}=d'_{left}} (\mathit{beg}(d) \wedge \langle \mathrm{E}\rangle(\mathit{length}_{n+1} \wedge \mathit{beg}(d')))\Big).$$

Finally, the conjunct φ_{rc} checks that adjacent cells in a column have the same color on the shared edge. For this, it suffices to require that the following condition holds:

- for each subtrace of the given one containing exactly one occurrence of \$, starting with a cell c, and ending with a cell c', if c and c' have the same column number, then $d_{up} = d'_{down}$, where d (respectively, d') is the content of c (respectively, c').

Accordingly, the formula φ_{rc} is defined as follows, where we use the formulas $\theta_j(b,b)$, with $j \in [2, n+1]$ and $b \in \{0,1\}$, to express that c and c' have the same column number:

$$\varphi_{rc} = [\mathrm{G}]\Big\{\;\Big(\phi_{one}(\$) \;\wedge\; \langle \mathrm{B}\rangle\, \phi_{cell} \;\wedge\; \langle \mathrm{E}\rangle\, \phi_{cell} \;\wedge\; \bigwedge_{j=2}^{n+1} \bigvee_{b\in\{0,1\}} \theta_j(b,b)\Big)$$

$$\longrightarrow \bigvee_{(d,d')\in\Delta\times\Delta \mid d_{up}=d'_{down}} (beg(d) \wedge \langle \mathrm{E}\rangle(length_{n+1} \wedge beg(d'))) \Big\},$$

where $\phi_{one}(\$)$ is defined as

$$(\langle \mathrm{B}\rangle\, end(\$)) \;\wedge\; \neg(\langle \mathrm{B}\rangle(end(\$) \wedge \langle \mathrm{B}\rangle\, end(\$))).$$

The formula $\varphi_{\mathcal{I}}$ has length polynomial in the size of $\mathcal{I}$. By construction, a trace ρ of $\mathcal{K}_{\mathcal{I}}$ satisfies $\varphi_{\mathcal{I}}$ if and only if ρ encodes a tiling. Since the initial traces of $\mathcal{K}_{\mathcal{I}}$ are the finite words over $\mathcal{AP}$ starting with d_{init}, it follows that there exists a tiling of $\mathcal{I}$ if and only if there exists an initial trace of $\mathcal{K}_{\mathcal{I}}$ which satisfies $\varphi_{\mathcal{I}}$.

The given reduction proves the following theorem.

Theorem 2.4.1. *The MC problem for* BE *formulas over finite Kripke structures is* ***EXPSPACE****-hard (under polynomial-time reductions).*

Before concluding the section, we would like to comment on the complexity gap deriving from the upper and lower bounds proved for HS (and BE) MC.

Let us start by introducing *star-free regular expressions* over a finite alphabet Σ, with $|\Sigma| \geq 2$, that are defined by the grammar

$$r ::= \emptyset \mid a \mid r \cdot r \mid r \cup r \mid \neg r,$$

where $a \in \Sigma$. Every regular expression defines a language $\mathcal{L}(r)$ of finite words over Σ by induction on its structural complexity as follows: $\mathcal{L}(\emptyset) = \emptyset$, $\mathcal{L}(a) = \{a\}$, $\mathcal{L}(r_1 \cdot r_2) = \mathcal{L}(r_1) \cdot \mathcal{L}(r_2)$, $\mathcal{L}(r_1 \cup r_2) = \mathcal{L}(r_1) \cup \mathcal{L}(r_2)$ and $\mathcal{L}(\neg r) = \Sigma^* \setminus \mathcal{L}(r)$.[2] It is well-known that the *language-emptiness problem for star-free regular expressions is (provably) nonelementary* [SM73; Sto74] (more properly, **TOWER**-complete in the notation of [Sch16]), being negation the "difficult case".

Let us now slightly modify the definition of these expressions. We consider:

$$r ::= \emptyset \mid a \mid r \cdot \Sigma^+ \mid \Sigma^+ \cdot r \mid r \cup r \mid \neg r.$$

Again, negation is present and there is no Kleene star (as Σ^+ can be "rewritten" as $\neg\emptyset$); *concatenation is now weakened*: in fact, $r \cdot \Sigma^+$ and $\Sigma^+ \cdot r$ represent right-/left-extensions of a word in $\mathcal{L}(r)$ with any (non-empty) word. To the best of our knowledge, nothing is known about the precise complexity of this variant of star-free regular expressions.

The reader may now be wondering why we care so much about these regular expressions. The answer is given by the following fact: *there is a trivial reduction from language emptiness for this variant of expressions to* BE *MC, and vice versa.*

[2] As a standard notation, the dot $\cdot$ represents the concatenation of words, and Σ^* (resp., Σ^+) the set of all possible *finite* (resp., finite and non-empty) words over Σ.

Intuitively, $r \cdot \Sigma^+$ (resp., $\Sigma^+ \cdot r$) can be "simulated" by $\langle B \rangle$ (resp., $\langle E \rangle$).[3] As an immediate consequence, an improvement on the known (upper or lower) bounds for any of the two problems would immediately propagate to the other.

Unfortunately, the nonelementary lower bound proved for (standard) star-free regular expressions by Stockmeyer in his PhD thesis [Sto74] cannot be adapted to the variant, failing to encode the so-called "*movable rulers*", namely, regular expressions that are "satisfied" only by (all) words of a precise length. Conversely, it is not clear how the (nonelementary) complexity of the standard automata-based algorithm for deciding the emptiness of regular expressions could possibly be lowered by exploiting the weakened concatenation; analogously, the nonelementary algorithm for full HS MC represents, at the moment, the best algorithm also for BE MC.

2.5 An overview on MC for HS and its fragments under homogeneity

As the **EXPSPACE**-hardness of the fragment BE clearly propagates to full HS, HS MC represents a *provably intractable problem* (here and in the following, we say "intractable"—by borrowing the terminology from Meyer and Stockmeyer [SM73]—when a problem can not be provably solved in polynomial time). The reasons which originated a systematic, in-depth investigation on complexity and expressiveness of HS fragments should then appear obvious: lowering the computational complexity of MC has crucial importance; at the same time, identifying a good *trade-off between complexity and expressiveness* of a logic is fundamental as well. In the next chapters we start considering the issue of complexity which, for HS fragments where properties of prefixes (namely, $\langle B \rangle$) and suffixes ($\langle E \rangle$) of intervals are dealt with separately, is markedly lower than that of full HS (see Figure 2.15 for a graphical overview). Expressiveness will be addressed in Chapter 6: there we will make a comparison of different semantic variants of HS; in addition we will compare such variants with the standard point-based temporal logics LTL, CTL, and CTL*.

Since the combined use of modalities for prefixes $\langle B \rangle$ and suffixes $\langle E \rangle$ is critical, the first fragments taken into consideration were $A\overline{A}B\overline{B}\overline{E}$ and $A\overline{A}E\overline{B}\overline{E}$: these (syntactically maximal) fragments are obtained from full HS (i.e., $A\overline{A}BE\overline{B}\overline{E}$) in an obvious way, that is, by removing either B or E. In [MMP17] we devised, for both of them, an **EXPSPACE** MC algorithm which finds, for each trace of the input Kripke structure, a satisfaction-preserving trace of bounded exponential length, i.e., a *trace representative*. In this way, the algorithm needs to check only trace representatives instead of traces of unbounded length. In the same paper, it

[3] It is also worth noting that the "general" concatenation $r \cdot r$ cannot be simulated by BE: the *chop* operator would be needed, which bisects an interval into two consecutive parts/subintervals: $\psi_1 \langle C \rangle \psi_2$ predicates ψ_1 over the first subinterval and ψ_2 over the second.

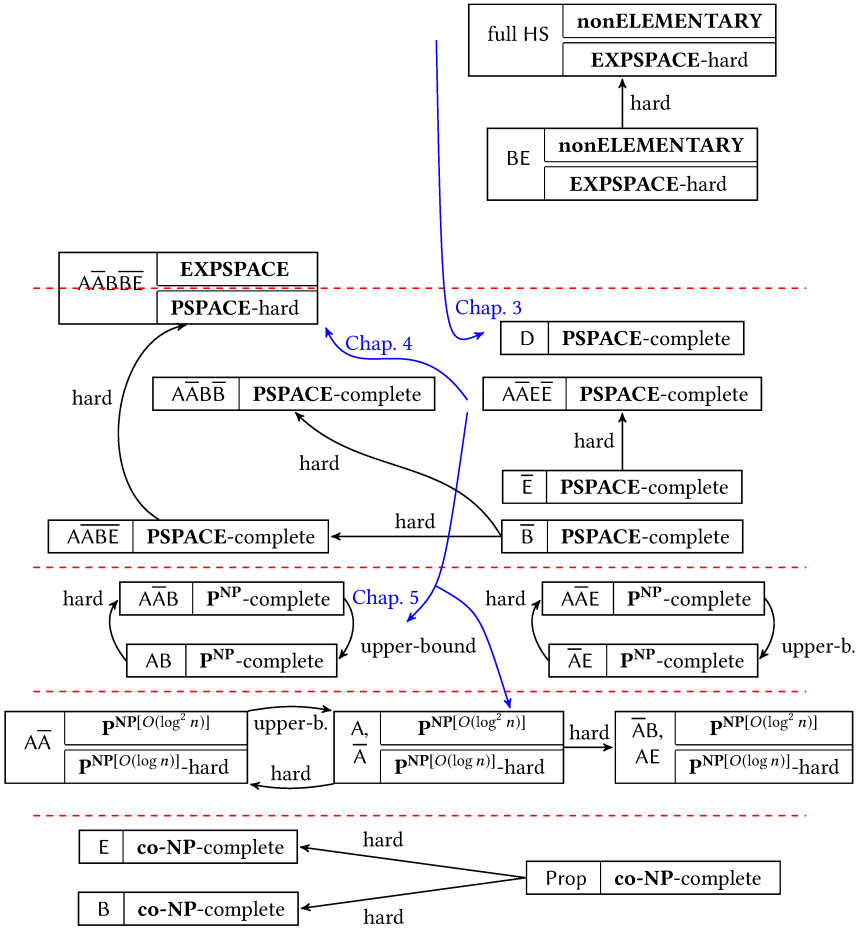


Figure 2.15: Complexity of the MC problem for HS and its fragments. Red lines separate different complexity classes. Blue arrows depict the contents of the next three chapters.

was proved that formulas satisfying a constant bound on the B-nesting depth (resp., E-nesting depth) can be checked in polynomial working space. As a consequence MC for $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{E}}$ is in **PSPACE** (its formulas do not feature $\langle \mathsf{B} \rangle / \langle \mathsf{E} \rangle$, hence $\mathrm{Nest}_{\mathsf{B}}(\psi) = \mathrm{Nest}_{\mathsf{E}}(\psi) = 0$). The techniques employed in [MMP17] will be summarized at the beginning of Section 4.2. However, the proof of existence of trace representatives is rather involved and it exploits very technical arguments. In this thesis, we will prove—in a (hopefully!) much more understandable and compact way—membership to **EXPSPACE** of MC for $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}\overline{\mathsf{E}}$ and $\mathsf{A}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{B}}\overline{\mathsf{E}}$ (Section 4.2) by having recourse to completely different notions.

Still in Chapter 4 (Section 4.1), we prove that MC for the HS fragment $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}$ (resp., $\mathsf{A}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{E}}$) is in **PSPACE**. Since MC for the HS fragment featuring only one modality for right (resp., left) interval extensions $\overline{\mathsf{B}}$ (resp., $\overline{\mathsf{E}}$) is **PSPACE**-hard (see Appendix B.3),[4] **PSPACE**-completeness immediately follows: the complexity turns out be the same as the one of LTL MC (known to be **PSPACE**-complete [SC85]). As a "byproduct", we show that MC for the one-modality fragments B and E (B and E dealt with separately!) turns out to be **co-NP**-complete.[5] These results are achieved by means of a *small-model property*: intuitively, given a trace ρ in a finite Kripke structure and a formula φ of $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}$/$\mathsf{A}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{E}}$, we prove that, by iteratively contracting ρ it is always possible to build another trace whose length is *polynomially bounded* in the size of the formula and of the Kripke structure, which preserves the satisfiability of φ with respect to ρ.

In Chapter 5, we analyze the sub-fragments of $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}$ (respectively, $\mathsf{A}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{E}}$), which are still expressive enough to capture meaningful interval properties of state transition systems and whose MC problem has a computational complexity markedly lower than that of full HS, namely, A, $\overline{\mathsf{A}}$, $\mathsf{A}\overline{\mathsf{A}}$, AB, $\overline{\mathsf{A}}\mathsf{B}$, AE, $\overline{\mathsf{A}}\mathsf{E}$, $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}$, and $\mathsf{A}\overline{\mathsf{A}}\mathsf{E}$. All these have a similar computational complexity, as their MC problem settles in one of the lowest levels of the *polynomial-time hierarchy*, $\mathbf{P}^{\mathbf{NP}}$, or below. Such a class consists of the set of problems decided by a deterministic polynomial-time bounded Turing machine, with the "support" of an oracle for the class **NP**, that is, a tool which decides, in one computation step, whether an instance of a problem belonging to **NP** is positive or not. $\mathbf{P}^{\mathbf{NP}}$ is also referred to as **P** *relative to* **NP** (relativization). However, though the fragments in the considered set are similar, some differences can be marked. In particular, the fragments A, $\overline{\mathsf{A}}$, $\mathsf{A}\overline{\mathsf{A}}$, $\overline{\mathsf{A}}\mathsf{B}$, and AE are actually "easier" than the other ones, since they require the **P** Turing machine to perform just $O(\log^2 n)$ queries to the **NP** oracle, for an input size n, instead of $O(n^k)$ queries, for some constant $k \geq 0$, as it is allowed in the general case for a polynomial running time machine. The MC problem for these fragments witnesses a "non-standard" complexity class in the polynomial-time hierarchy,

[4] The **PSPACE**-hardness of MC for $\overline{\mathsf{B}}$/$\overline{\mathsf{E}}$ gives the best (unmatching) complexity lower bound also for $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}\overline{\mathsf{E}}$ and $\mathsf{A}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{B}}\overline{\mathsf{E}}$ MC.

[5] The **co-NP**-hardness derives from that of the fragment Prop (the purely propositional fragment of HS), as proved in [MMP15a].

called *bounded-query* class, that will be presented in detail in Section 5.1. More precisely, we devise a $\mathbf{P}^{\mathbf{NP}}$ MC algorithm for $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}$ and $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{E}$ in Section 5.2, and then prove a matching complexity lower bound (Section 5.4); then we show that MC for A, $\overline{\mathsf{A}}$, $\mathsf{A}\overline{\mathsf{A}}$, $\overline{\mathsf{A}}\mathsf{B}$ and AE is still in $\mathbf{P}^{\mathbf{NP}}$, but only $O(\log^2 n)$ queries to the **NP** oracle are necessary (Section 5.3). This result is achieved by a reduction *to* the problem TB(SAT) [Sch03], whose instances are complex circuits where some of the gates are endowed with **NP** oracles. Finally, we identify a lower bound, which shows that *at least* $\log n$ *queries* are needed to solve the problem (Section 5.5). Unfortunately, such bound does not exactly match the upper bound, leaving open the question whether the problem can be solved by $o(\log^2 n)$ (i.e., strictly less than $O(\log^2 n)$) queries to an **NP** oracle, or a tighter lower bound can be proved (or both).

In the next Chapter 3, we focus on another HS fragment that has been studied, namely, D—also known as "the logic of sub-intervals"—whose MC on finite Kripke structures turns out to be **PSPACE**-complete (the same complexity result holds also for its SAT over finite linear orders, assuming homogeneity). Modality $\langle \mathsf{D} \rangle$ can be easily defined by $\langle \mathsf{B} \rangle$ and $\langle \mathsf{E} \rangle$, as $\langle \mathsf{D} \rangle \varphi = \langle \mathsf{B} \rangle \langle \mathsf{E} \rangle \varphi = \langle \mathsf{E} \rangle \langle \mathsf{B} \rangle \varphi$, hence D is a fragment of BE. However **PSPACE**-completeness of D MC and SAT strongly contrasts with the case of BE (whose MC and SAT are **EXPSPACE**-hard under homogeneity whereas, without homogeneity, SAT is undecidable over the class of finite and discrete linear orders [MM14]).[6]

We now summarize the contents of the next three chapters, by a kind of "journey" among complexity classes, which is depicted by the blue arrows of Figure 2.15.

Organization of the next chapters.

- We start with Chapter 3, presenting D MC and SAT. Whereas MC and SAT for BE are nonelementarily decidable and **EXPSPACE**-hard, the situation is much better if we restrict to BE's fragment D: we will prove **PSPACE** membership of both problems for it under homogeneity, and comment on their **PSPACE**-hardness.
- In Chapter 4, at first we remain in **PSPACE**: we prove that MC for $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}$ (resp., $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{E}}$) belongs to that class. Since MC for $\overline{\mathsf{B}}$ (resp., $\overline{\mathsf{E}}$) is **PSPACE**-hard (this is proved in Appendix B.3), **PSPACE**-completeness of all fragments "in between" $\overline{\mathsf{B}}$ and $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}$ (resp., $\overline{\mathsf{E}}$ and $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{E}}$) follows. It is surprising that $\overline{\mathsf{B}}$ and $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}$ have exactly the same complexity, being the latter much more expressive than the former, as it can also express properties of prefixes, and of intervals in the future as well as in the past. If we then *add* $\overline{\mathsf{E}}$ to

[6] We point out that homogeneity changes the status of the SAT problem for HS and its fragments. We will show in Chapter 6 that, when interpreted over the (infinite) fullpaths of a finite Kripke structure (which is not the way we interpret HS here), LTL and HS have the same expressive power under homogeneity, but the latter is provably exponentially more succinct. As a byproduct, the SAT problem for full HS, with such a semantics, turns out to be decidable. Therefore, *under homogeneity*, the relevant issue for SAT of HS becomes its complexity, rather than its decidability.

$A\overline{A}B\overline{B}$ getting $A\overline{A}B\overline{B}\overline{E}$ (resp., $\overline{B}$ to $A\overline{A}E\overline{E}$ getting $A\overline{A}E\overline{B}\overline{E}$) we show that the complexity "raises" to **EXPSPACE** (Section 4.2);[7] conversely,

- in Chapter 5, we show the results achieved by *removing* $\overline{B}$ from $A\overline{A}B\overline{B}$ (resp., $\overline{E}$ from $A\overline{A}E\overline{E}$). As we said, this leads to a "fall" towards a low level of the polynomial hierarchy, namely $\mathbf{P}^{\mathbf{NP}}$, where MC for A, $\overline{A}$, $A\overline{A}$, AB, $\overline{A}B$, AE, $\overline{A}E$, $A\overline{A}B$ and $A\overline{A}E$ is.

2.6 Related work

We would like to conclude the chapter by summarizing the content of other research papers that deal with HS MC. As already mentioned, the MC problem for interval temporal logics has not been extensively studied in literature. Indeed the only three papers (apart from the ones by these authors) that study HS MC are all by A. Lomuscio and J. Michaliszyn [LM13; LM14; LM16].

In [LM13; LM14; LM16], they address the MC problem for some fragments of HS extended with epistemic modalities. Their semantic assumptions are different from the ones we make here, thus a systematic comparison of the two research lines is quite difficult. In both cases, formulas of HS are evaluated over traces/intervals of a Kripke structure; however, in [LM13; LM14] truth of proposition letters over an interval depends only on its endpoints.

In [LM13], the authors focus on the HS fragment BED of Allen's relations *started-by*, *finished-by*, and *contains* (since modality $\langle D \rangle$ is definable in terms of modalities $\langle B \rangle$ and $\langle E \rangle$, BED is actually as expressive as BE), extended with epistemic modalities. They consider a *restricted* form of MC ("local" MC), which verifies a given specification against a single (finite) initial computation interval. Their goal is indeed to reason about a given computation of a multi-agent system, rather than on all its admissible computations. They prove that the considered MC problem is **PSPACE**-complete; furthermore, they show that the same problem restricted to the pure temporal fragment BED, that is, the one obtained by removing epistemic modalities, is in **P** as, basically, modalities $\langle B \rangle$ and $\langle E \rangle$ allow one to access only sub-intervals of the initial one, whose number is quadratic in the length (number of states) of the initial interval.

In [LM14], they show that the picture drastically changes with other HS fragments that allow one to access infinitely many intervals. In particular, they prove that the MC problem for the fragment $A\overline{B}L$ of Allen's relations *meets*, *starts*, and *before* (since modality $\langle L \rangle$ is definable in terms of modality $\langle A \rangle$, $A\overline{B}L$ is actually as expressive as $A\overline{B}$) extended with epistemic modalities, is decidable in nonelementary time. Note that, thanks to modalities $\langle A \rangle$ and $\langle \overline{B} \rangle$, formulas of $A\overline{B}L$ can possibly refer to infinitely many (future) intervals.

[7] The fragments $A\overline{A}B\overline{B}\overline{E}/A\overline{A}E\overline{B}\overline{E}$ are presented after $A\overline{A}B\overline{B}/A\overline{A}E\overline{E}$ for technical reasons.

Finally, in [LM16], Lomuscio and Michaliszyn show how to use regular expressions in order to specify the way in which intervals of a Kripke structure get labelled. Such an extension leads to a significant increase in expressiveness, as the labelling of an interval is no more determined by that of its endpoints only, but it depends on the ordered sequence of states the interval consists of. They also prove that there is no corresponding increase in computational complexity, as the bounds given in [LM13; LM14] still hold with the new semantic variant: MC for BED is in **PSPACE**, and it is nonelementarily decidable for A$\overline{\text{B}}$L. We will come back to this idea of using regular expressions to define interval labelling in Chapter 7.

3

The logic of sub-intervals D

The reference for this chapter is [Boz+17c].

Contents

3.1 **Preliminaries** 52
3.2 **A spatial representation of interval models** 54
3.3 **SAT of D over finite linear orders** 57
3.4 **MC for D over Kripke structures** 68
3.5 **An outline of PSPACE-completeness of SAT and MC for D** 71
3.6 **Conclusions** 72

In this chapter we focus on the logic D (also known as "the logic of sub-intervals") which features one modality only, corresponding to the Allen interval relation *during*. Since any sub-interval is just an initial sub-interval of an ending one, or, equivalently, an ending sub-interval of an initial one, D is a (proper) fragment of BE.

The logic of sub-intervals comes into play in the study of *temporal prepositions in natural language* [Pra05]; the connections between the temporal logic of (strict) sub-intervals and *the logic of Minkowski space-time* have been explored by Shapirovsky and Shehtman [SS05]. Finally, the temporal logic of reflexive sub-intervals has been studied for the first time by van Benthem, who proved that, when interpreted over dense linear orderings, it is equivalent to the standard modal logic S4 [van91].

From a computational point of view, D is a real character! Its SAT problem is **PSPACE**-complete over the class of dense linear orders [Bre+10b; Sha04] (whereas the problem is undecidable for BE [Lod00]), it becomes undecidable when the

logic is interpreted over the classes of finite and discrete linear orders [MM14], and the situation is still unknown over the class of all linear orders. As for its expressiveness, unlike A$\overline{\text{A}}$—which is expressively complete with respect to the two-variable fragment of first-order logic for binary relational structures over various linearly-ordered domains [Bre+09b; Ott01]—*three* variables are needed to encode D in first-order logic (the two-variable property is a sufficient condition for decidability, but it is not a necessary one).

In this chapter we show that decidability of SAT for D over the class of finite linear orders can be recovered *under the homogeneity assumption*. This will allow us to show that also MC under homogeneity is decidable. We first prove that D SAT is in **PSPACE** by exploiting a suitable contraction method. Then we show that the proposed SAT algorithm can be transformed into a **PSPACE** MC procedure for D formulas over finite Kripke structures; **PSPACE**-hardness of both problems follows via a reduction from the language universality problem of nondeterministic finite-state automata.

PSPACE-completeness of D MC strongly contrasts with the case of BE, which we showed to be nonelementarily decidable and hard for **EXPSPACE**.

Organization of the chapter.

- In Section 3.1 and 3.2, we provide some background knowledge; in particular, we introduce the logic D, interval models, and a spatial representation of interval models called "*compass structure*".
- In Section 3.3, we show the **PSPACE** membership of the SAT problem for D over finite linear orders (under homogeneity). This complexity result is proved via a contraction technique, applied to the mentioned compass structures, relying on a suitable finite-index equivalence relation.
- In Section 3.4, we show that the MC problem for D over finite Kripke structures (again, under homogeneity) is in **PSPACE** as well. The proposed MC algorithm is basically a SAT procedure driven by the computation traces of the system model.
- Finally, in Section 3.5, we comment on the **PSPACE**-hardness of both problems, referring to Appendix A.5 and Appendix A.6 for further details.

3.1 Preliminaries

To start with, we introduce some preliminary notions. Let $\mathbb{S} = (S, <)$ be a linear order. An *interval* over $\mathbb{S}$ is an ordered pair $[x, y]$, where $x, y \in S$, $x \leq y$, representing the set $\{z \in S \mid x \leq z \leq y\}$. We denote the set of all intervals over $\mathbb{S}$ by $\mathbb{I}(\mathbb{S})$. We consider three possible *sub-interval relations*:

1. the *reflexive* sub-interval relation (denoted as $\sqsubseteq$), defined by $[x, y] \sqsubseteq [x', y']$ if and only if $x' \leq x$ and $y \leq y'$,
2. the *proper (or irreflexive)* sub-interval relation (denoted as $\sqsubset$), defined by $[x, y] \sqsubset [x', y']$ if and only if $[x, y] \sqsubseteq [x', y']$ and $[x, y] \neq [x', y']$, and
3. the *strict* sub-interval relation (denoted as $\dot{\sqsubset}$), defined by $[x, y] \dot{\sqsubset} [x', y']$ if and only if $x' < x$ and $y < y'$.

The three modal logics $\mathrm{D}_{\sqsubseteq}$, $\mathrm{D}_{\sqsubset}$, and $\mathrm{D}_{\dot{\sqsubset}}$ feature the same language, consisting of a set $\mathcal{AP}$ of proposition letters, the logical connectives $\neg$ and $\vee$, and the modal operator $\langle \mathrm{D} \rangle$. Formally, formulas are defined by the grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle \mathrm{D} \rangle \varphi,$$

with $p \in \mathcal{AP}$. The other connectives, as well as the logical constants $\top$ (true) and $\bot$ (false), are defined as usual; moreover, the dual universal modal operator $[\mathrm{D}]\varphi$ is defined as $\neg \langle \mathrm{D} \rangle \neg\varphi$. The length of a formula φ, denoted as $|\varphi|$, is the number of subformulas of φ.

The semantics of $\mathrm{D}_{\dot{\sqsubset}}$, $\mathrm{D}_{\sqsubset}$, and $\mathrm{D}_{\sqsubseteq}$ only differ in the interpretation of the $\langle \mathrm{D} \rangle$ operator. For the sake of brevity, we use $\circ \in \{\dot{\sqsubset}, \sqsubset, \sqsubseteq\}$ as a shorthand for any of the three sub-interval relations. The semantics of a sub-interval logic $\mathrm{D}_\circ$ is defined in terms of *interval models*[1] $\mathbf{M} = (\mathbb{I}(\mathbb{S}), \circ, \mathcal{V})$. The *valuation function* $\mathcal{V} : \mathcal{AP} \to 2^{\mathbb{I}(\mathbb{S})}$ assigns to every proposition letter p the set of intervals $\mathcal{V}(p)$ over which p holds. The *satisfiability relation* $\models$ is defined as:

- for every proposition letter $p \in \mathcal{AP}$, $\mathbf{M}, [x, y] \models p$ if and only if $[x, y] \in \mathcal{V}(p)$;
- $\mathbf{M}, [x, y] \models \neg\psi$ if and only if $\mathbf{M}, [x, y] \not\models \psi$ (i.e. it is not true that $\mathbf{M}, [x, y] \models \psi$);
- $\mathbf{M}, [x, y] \models \psi_1 \vee \psi_2$ if and only if $\mathbf{M}, [x, y] \models \psi_1$ or $\mathbf{M}, [x, y] \models \psi_2$;
- $\mathbf{M}, [x, y] \models \langle \mathrm{D} \rangle \psi$ if and only if there exists an interval $[x', y'] \in \mathbb{I}(\mathbb{S})$ such that $[x', y'] \circ [x, y]$ and $\mathbf{M}, [x', y'] \models \psi$.

A $\mathrm{D}_\circ$ formula is *satisfiable* if it holds over some interval of an interval model, and *valid* if it holds over every interval of every interval model.

As we mentioned earlier on, it can be shown that the logic $\mathrm{D}_{\sqsubseteq}$ turns out to be equivalent to the standard modal logic S4 [van91]. Here we restrict our attention to the finite SAT problem, that is, satisfiability over the class of finite linear orders. The problem has been shown to be *undecidable* for $\mathrm{D}_{\sqsubset}$ and $\mathrm{D}_{\dot{\sqsubset}}$ [MM14] and *decidable* for $\mathrm{D}_{\sqsubseteq}$ [MPS10a]. In the following, we prove that decidability can be recovered for $\mathrm{D}_{\sqsubset}$ and $\mathrm{D}_{\dot{\sqsubset}}$ by restricting to the class of *homogeneous* interval models. We fully work out the case of $\mathrm{D}_{\sqsubset}$ (for the sake of simplicity, we will write D for $\mathrm{D}_{\sqsubset}$), and then we briefly explain how to adapt the proofs to $\mathrm{D}_{\dot{\sqsubset}}$.

[1] Not to be confused with the previously introduced *abstract interval models*.

Definition 3.1.1. An interval model $\mathbf{M} = (\mathbb{I}(\mathbb{S}), \circ, \mathcal{V})$ is *homogeneous* if, for every interval $[x, y] \in \mathbb{I}(\mathbb{S})$ and every proposition letter $p \in \mathcal{AP}$, it holds that $[x, y] \in \mathcal{V}(p)$ if and only if $[x', x'] \in \mathcal{V}(p)$ for every $x \leq x' \leq y$.

Hereafter we will assume the logic D to be interpreted over homogeneous interval models.

3.2 A spatial representation of interval models

Let us now introduce some basic definitions and notation which will be extensively used in the following, concluding the section with the notion of compass structure. Given a D formula φ, we define the *closure of* φ, denoted by $\mathsf{CL}(\varphi)$, as the set of all subformulas ψ of φ and of their negations $\neg\psi$ (we identify $\neg\neg\psi$ with ψ).

Definition 3.2.1. Given a D-formula φ, a φ-atom A is a subset of $\mathsf{CL}(\varphi)$ such that:

- for every $\psi \in \mathsf{CL}(\varphi)$, $\psi \in A$ if and only if $\neg\psi \notin A$, and
- for every $\psi_1 \vee \psi_2 \in \mathsf{CL}(\varphi)$, $\psi_1 \vee \psi_2 \in A$ if and only if $\psi_1 \in A$ or $\psi_2 \in A$.

The idea underlying atoms is to enforce a "local" (or Boolean) form of consistency among the formulas it contains, that is, a φ-atom A is a *maximal, locally consistent subset* of $\mathsf{CL}(\varphi)$. As an example, $\neg(\psi_1 \vee \psi_2) \in A$ if and only if $\neg\psi_1 \in A$ and $\neg\psi_2 \in A$. However, note that the definition does not set any constraint on $\langle \mathrm{D} \rangle \psi$ formulas, hence the word "local". We denote the set of all φ-atoms as $\mathcal{A}_\varphi$; its cardinality is clearly bounded by $2^{|\varphi|}$ (by the first point of Definition 3.2.1). Atoms are connected by the following binary relation D_φ.

Definition 3.2.2. Let D_φ be a binary relation over $\mathcal{A}_\varphi$ such that, for each pair of atoms $A, A' \in \mathcal{A}_\varphi$, $A \; D_\varphi \; A'$ holds if and only if both $\psi \in A'$ and $[\mathrm{D}]\,\psi \in A'$ for each formula $[\mathrm{D}]\,\psi \in A$.

Let A be a φ-atom. We denote by $\mathcal{R}eq_D(A)$ the set $\{\psi \in \mathsf{CL}(\varphi) \mid \langle \mathrm{D} \rangle \psi \in A\}$ of "*temporal requests*" of A. In particular, if $\psi \notin \mathcal{R}eq_D(A)$, then $[\mathrm{D}]\,\neg\psi \in A$ (by the definition of φ-atom). Moreover, we denote by REQ_φ the set of all arguments of D formulas in $\mathsf{CL}(\varphi)$, namely, $\mathsf{REQ}_\varphi = \{\psi \mid \langle \mathrm{D} \rangle \psi \in \mathsf{CL}(\varphi)\}$. Finally, we denote by $\mathcal{O}bs_D(A)$ the set $\{\psi \in A \mid \psi \in \mathsf{REQ}_\varphi\}$ of "*observables*" of A.

It is easy to prove by induction the next proposition stating that, once the proposition letters of A and its temporal requests are fixed, A is determined.

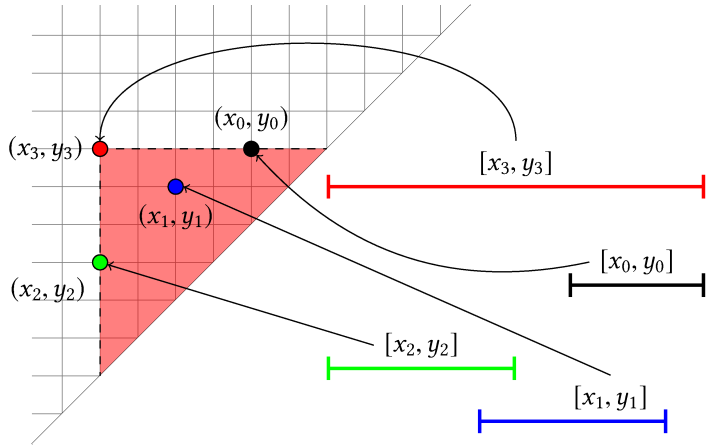


Figure 3.1: Correspondence between intervals and points of the compass structure.

Proposition 3.2.3. *For any* D *formula* φ*, given a set* $R \subseteq \mathsf{REQ}_\varphi$ *and a set* $P \subseteq \mathsf{CL}(\varphi) \cap \mathcal{AP}$*, there exists a unique* φ*-atom* A *that satisfies* $\mathcal{R}eq_D(A) = R$ *and* $A \cap \mathcal{AP} = P$*.*

We now provide a natural interpretation of D over grid-like structures, called *compass structures*, by exploiting the existence of a natural bijection between intervals $[x, y]$ and points (x, y), with $x \leq y$, of an $\mathcal{S} \times \mathcal{S}$ grid, where $\mathbb{S} = (\mathcal{S}, <)$ is a finite linear order. Such an interpretation was originally proposed by Venema in [Ven90], and it can also be given for HS and all its (other) fragments.

As an example, Figure 3.1 shows four intervals $[x_0, y_0], \ldots, [x_3, y_3]$, respectively represented by the points in the grid $(x_0, y_0), \ldots, (x_3, y_3)$, such that:

- $[x_0, y_0], [x_1, y_1], [x_2, y_2] \sqsubset [x_3, y_3]$,
- $[x_1, y_1] \boxdot [x_3, y_3]$, and
- $[x_0, y_0], [x_2, y_2] \not\sqsubset [x_3, y_3]$.

The red region highlighted in Figure 3.1 contains all and only the points (x, y) such that $[x, y] \sqsubset [x_3, y_3]$. Allen interval relation *contains* can thus be represented as a spatial relation between pairs of points. In the following, we make use of $\sqsubset$ also for relating points, i.e., given two points $(x, y), (x', y')$ of the grid, $(x', y') \sqsubset (x, y)$ if and only if $(x', y') \neq (x, y)$ and $x \leq x' \leq y' \leq y$. Compass structures, repeatedly exploited in the following to establish the next complexity results, can be formally defined as follows.

Definition 3.2.4. Given a finite linear order $\mathbb{S} = (S, <)$ and a D formula φ, a *compass φ-structure* is a pair $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$, where $\mathbb{P}_\mathbb{S}$ is the set of points of the form (x, y), with $x, y \in S$ and $x \leq y$, and $\mathcal{L}$ is a function that maps any point $(x, y) \in \mathbb{P}_\mathbb{S}$ to a φ-atom $\mathcal{L}(x, y)$ in such a way that for every pair of points $(x, y) \neq (x', y') \in \mathbb{P}_\mathbb{S}$, if $x \leq x' \leq y' \leq y$ then $\mathcal{L}(x, y)\ D_\varphi\ \mathcal{L}(x', y')$ (*temporal consistency*).

Due to temporal consistency, the following important property holds in compass structures.

Lemma 3.2.5. *Given a compass φ-structure $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$, for all pairs of points $(x', y'), (x, y) \in \mathbb{P}_\mathbb{S}$, if $(x', y') \sqsubset (x, y)$, then $\mathcal{R}eq_D(\mathcal{L}(x', y')) \subseteq \mathcal{R}eq_D(\mathcal{L}(x, y))$ and $\mathcal{O}bs_D(\mathcal{L}(x', y')) \subseteq \mathcal{R}eq_D(\mathcal{L}(x, y))$.*

Proof. By Definition 3.2.4 we have $\mathcal{L}(x, y)\ D_\varphi\ \mathcal{L}(x', y')$. Let us assume by contradiction that there exists $\psi \in \mathcal{R}eq_D(\mathcal{L}(x', y')) \setminus \mathcal{R}eq_D(\mathcal{L}(x, y))$. By definition of $\mathcal{R}eq_D$ and by Definition 3.2.1, we have that $\psi \in \mathcal{R}eq_D(\mathcal{L}(x', y'))$ implies $\langle \mathrm{D} \rangle\, \psi \in \mathcal{L}(x', y')$, and $\psi \notin \mathcal{R}eq_D(\mathcal{L}(x, y))$ implies $\neg \langle \mathrm{D} \rangle\, \psi = [\mathrm{D}]\, \neg\psi \in \mathcal{L}(x, y)$. Since $\mathcal{L}(x, y)\ D_\varphi\ \mathcal{L}(x', y')$, then $[\mathrm{D}]\, \neg\psi \in \mathcal{L}(x', y')$ and thus we can conclude that both $[\mathrm{D}]\, \neg\psi$ and $\langle \mathrm{D} \rangle\, \psi$ belong to $\mathcal{L}(x', y')$ (contradiction).

$\mathcal{O}bs_D(\mathcal{L}(x', y')) \subseteq \mathcal{R}eq_D(\mathcal{L}(x, y))$ can analogously be proved by contradiction. □

Fulfilling compass structures are defined as follows.

Definition 3.2.6. A compass φ-structure $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$ is said to be *fulfilling* if, for every point $(x, y) \in \mathbb{P}_\mathbb{S}$ and each formula $\psi \in \mathcal{R}eq_D(\mathcal{L}(x, y))$, there exists a point $(x', y') \sqsubset (x, y)$ in $\mathbb{P}_\mathbb{S}$ such that $\psi \in \mathcal{L}(x', y')$.

Note that if $\mathcal{G}$ is fulfilling, then $\mathcal{R}eq_D(\mathcal{L}(x, x)) = \emptyset$ for all points "on the diagonal" $(x, x) \in \mathbb{P}_\mathbb{S}$. We say that a compass φ-structure $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$ *features* a formula ψ if there exists a point $(x, y) \in \mathbb{P}_\mathbb{S}$ such that $\psi \in \mathcal{L}(x, y)$. The following result holds.

Proposition 3.2.7. *A* D *formula φ is satisfiable if and only if there exists a fulfilling compass φ-structure that features it.*

In a fulfilling compass φ-structure $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$, where $S = \{0, \ldots, t\}$, w.l.o.g., we will sometimes assume φ to be satisfied by the maximal interval $[0, t]$, that is, $\varphi \in \mathcal{L}(0, t)$.

The notion of homogeneous models directly transfers to compass structures.

Definition 3.2.8. A compass φ-structure $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$ is *homogeneous* if, for every point $(x, y) \in \mathbb{P}_\mathbb{S}$ and each proposition letter $p \in \mathcal{AP}$, we have that $p \in \mathcal{L}(x, y)$ if and only if $p \in \mathcal{L}(x', x')$ for all $x \leq x' \leq y$.

Proposition 3.2.7 can be tailored to homogeneous compass structures.

Proposition 3.2.9. *A* D *formula φ is satisfiable if and only if there exists a fulfilling homogeneous compass φ-structure that features it.*

3.3 SAT of D over finite linear orders

In this section, we devise a SAT checking procedure for D formulas over finite linear orders, which will also allow us to easily derive a MC algorithm for D over finite Kripke structures.

To start with, we show that there is a ternary relation between φ-atoms, that we denote by $=^{D_\varphi}\!\!\rightarrow$, such that if it holds among all atoms in consecutive positions of a compass φ-structure, then the structure is fulfilling. Hence, we may say that $=^{D_\varphi}\!\!\rightarrow$ is the rule for labeling fulfilling compasses.

Next, we introduce an equivalence relation $\sim$ between *rows* of a compass φ-structure. Since it has finite index—exponentially bounded by $|\varphi|$—and it preserves fulfillment of compasses, it is intuitively possible to "contract" the structures when we can find two related rows. Moreover, any contraction done according to $\sim$ keeps the same atoms (only the number of their occurrences may vary), and thus if a compass features φ before the contraction, then φ is still featured after it. This fact is exploited to build a SAT algorithm for D formulas which makes use of *polynomial working space*, because (*i*) it only needs to keep track of two rows of a compass at a time, (*ii*) all rows satisfy some nice properties that make it possible to succinctly encode them, and (*iii*) compass contractions are implicitly performed by means of a reachability check in a suitable graph, whose nodes are the equivalence classes of $\sim$.

Let us now introduce the aforementioned ternary relation $=^{D_\varphi}\!\!\rightarrow$ among atoms.

Definition 3.3.1. Given three φ-atoms A_1, A_2 and A_3, we say that A_3 is D_φ-generated by A_1, A_2, written $A_1 A_2 =^{D_\varphi}\!\!\rightarrow A_3$, if:

- $A_3 \cap \mathcal{AP} = A_1 \cap A_2 \cap \mathcal{AP}$ and
- $\mathcal{R}eq_D(A_3) = \mathcal{R}eq_D(A_1) \cup \mathcal{R}eq_D(A_2) \cup \mathcal{O}bs_D(A_1) \cup \mathcal{O}bs_D(A_2)$.

It is immediate to show that $A_1 A_2 =^{D_\varphi}\!\!\rightarrow A_3$ if and only if $A_2 A_1 =^{D_\varphi}\!\!\rightarrow A_3$ (i.e., the order of the first two components in the ternary relation is irrelevant). Notice that the first point of the definition enforces the homogeneity assumption.

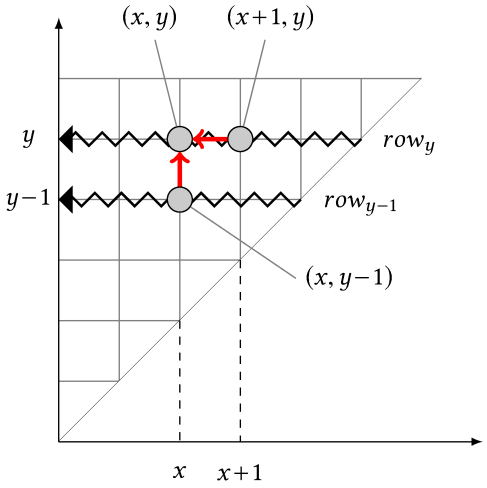


Figure 3.2: Rule for labeling homogeneous fulfilling compass φ-structures

The next result, immediately following from Proposition 3.2.3, proves that $=^{D_\varphi}\!\!\rightarrow$ expresses a *functional dependency* on φ-atoms.

Lemma 3.3.2. *Given two φ-atoms $A_1, A_2 \in \mathcal{A}_\varphi$, there exists* exactly one *φ-atom $A_3 \in \mathcal{A}_\varphi$ such that $A_1A_2 =^{D_\varphi}\!\!\rightarrow A_3$.*

Definition 3.3.1 and Lemma 3.3.2 can be exploited to label a homogeneous compass φ-structure $\mathcal{G}$, namely, to determine the φ-atoms labeling all the points (x, y) of $\mathcal{G}$, starting from the ones on the diagonal. The idea is the following: if two φ-atoms A_1 and A_2 label respectively the greatest proper prefix $[x, y-1]$, that is, the point $(x, y-1)$, and the greatest proper suffix $[x+1, y]$, that is, $(x+1, y)$, of the same interval $[x, y]$, then the atom A_3 labeling $[x, y]$ is unique, and it is precisely the one satisfying $A_1A_2 =^{D_\varphi}\!\!\rightarrow A_3$ (see Figure 3.2). The next lemma, proved in Appendix A.1, claims that this is the general rule for labeling fulfilling homogeneous compasses.

Lemma 3.3.3. *Let $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$. $\mathcal{G}$ is a fulfilling homogeneous compass φ-structure if and only if, for every pair $x, y \in \mathcal{S}$, we have:*

- $\mathcal{L}(x, y-1)\mathcal{L}(x+1, y) =^{D_\varphi}\!\!\rightarrow \mathcal{L}(x, y)$ *if* $x < y$, *and*
- $Req_D(\mathcal{L}(x, y)) = \emptyset$ *if* $x = y$.

Now we introduce the concept of φ-row, which can be viewed as the ordered sequence of (the occurrences of) atoms labelling a row of a compass φ-structure.

Given an atom $A \in \mathcal{A}_\varphi$, we call it *reflexive* if $A\ D_\varphi\ A$, and *irreflexive* otherwise.

Definition 3.3.4. A φ-row is a finite sequence of φ-atoms

$$row = A_0^{m_0} \cdots A_n^{m_n},$$

where A^m stands for m repetitions of A, such that for each $0 \leq i \leq n$, we have that $m_i > 0$—if $m_i > 1$, then A_i is reflexive—and for each $0 \leq j < i$, it holds that $A_i\ D_\varphi\ A_j$, $A_i \neq A_j$, and $(A_j \cap \mathcal{AP}) \supseteq (A_i \cap \mathcal{AP})$. Moreover, $\mathcal{R}eq_D(A_0) = \emptyset$.

The length of a φ-row $row = A_0^{m_0} \cdots A_n^{m_n}$ is defined as $|row| = \sum_{0 \leq i \leq n} m_i$, and for each $0 \leq j < |row|$, the j-th element, denoted by $row(j)$, is the j-th symbol in the word $A_0^{m_0} \cdots A_n^{m_n}$, e.g., $row(0) = A_0$, $row(m_0) = A_1$, …; finally $row(i, j)$, for $0 \leq i \leq j < |row|$, represents the sub-word $row(i)row(i+1) \cdots row(j)$. We denote by $\mathcal{R}ows_\varphi$ the set of all possible φ-rows. This set may be infinite.

The number of distinct atoms in any φ-row is bounded. Since for each $0 \leq i \leq n$ and each $0 \leq j < i$, $A_i\ D_\varphi\ A_j$, it holds that $\mathcal{R}eq_D(A_j) \subseteq \mathcal{R}eq_D(A_i)$. Therefore, two monotonic sequences for every φ-row can be considered, one increasing, i.e., $\emptyset = \mathcal{R}eq_D(A_0) \subseteq \mathcal{R}eq_D(A_1) \subseteq \ldots \subseteq \mathcal{R}eq_D(A_n)$, and one decreasing, i.e., $(A_0 \cap \mathcal{AP}) \supseteq (A_1 \cap \mathcal{AP}) \supseteq \ldots \supseteq (A_n \cap \mathcal{AP})$. The number of distinct elements is bounded by $|\varphi|$ in the former sequence and by $|\varphi| + 1$ in the latter (as $|\mathsf{REQ}_\varphi| \leq |\varphi| - 1$ and $|\mathcal{AP}| \leq |\varphi|$–w.l.o.g., we can consider only the letters actually occurring in φ). Since, as already shown (Proposition 3.2.3), a set of requests and a set of proposition letters uniquely determine a φ-atom, any φ-row may feature at most $2|\varphi|$ distinct atoms, namely, $n < 2|\varphi|$.

Given a homogeneous compass φ-structure $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$, for every $y \in \mathcal{S}$, we define row_y as the word of φ-atoms $row_y = \mathcal{L}(y, y) \cdots \mathcal{L}(0, y)$, i.e., the sequence of atoms labeling points of $\mathcal{G}$ with the same y-coordinate, starting from the one on the diagonal *inwards*. See Figure 3.2.

The next result holds; its proof can be found in Appendix A.2.

Lemma 3.3.5. *Let $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$ be a fulfilling homogeneous compass φ-structure. For every $y \in \mathcal{S}$, row_y is a φ-row.*

We now define the *successor* relation between pairs of φ-rows, denoted as $=^{row_\varphi}\!\!\rightarrow$, which is basically a component-wise application of $=^{D_\varphi}\!\!\rightarrow$ over the elements of two φ-rows (remember that atoms on rows are collected right-to-left).

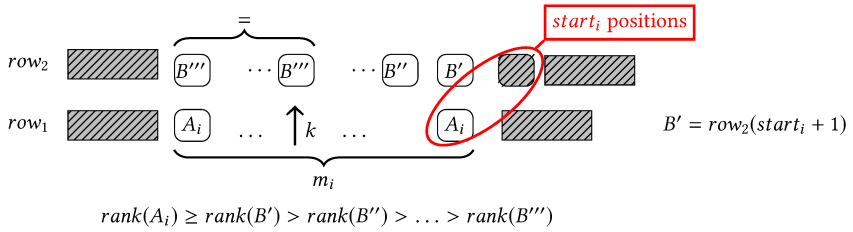


Figure 3.3: A graphical account of the proof of Lemma 3.3.9

Definition 3.3.6. Given two φ-rows row and row', we say that row' is a *successor* of row, written as $row {=^{row_\varphi}\!\!\Rightarrow} row'$, if $|row'| = |row| + 1$, and for all $0 \leq i < |row|$, $row(i) row'(i) {=^{D_\varphi}\!\!\Rightarrow} row'(i+1)$.

The next lemma, whose proof is in Appendix A.3, states that consecutive rows in homogeneous fulfilling compass φ-structures respect the successor relation.

Lemma 3.3.7. *Let $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$, with $\mathcal{R}eq_D(\mathcal{L}(x,x)) = \emptyset$ for all $(x,x) \in \mathbb{P}_\mathbb{S}$. $\mathcal{G}$ is a fulfilling homogeneous compass φ-structure if and only if, for each $0 \leq y < |\mathcal{S}| - 1$, $row_y {=^{row_\varphi}\!\!\Rightarrow} row_{y+1}$.*

Given an atom $A \in \mathcal{A}_\varphi$, we define the *rank of* A, written $rank(A)$, as $|\mathrm{REQ}_\varphi| - |\mathcal{R}eq_D(A)|$. Clearly, $rank(A) < |\varphi|$. Whenever $A \; D_\varphi \; A'$, for some $A' \in \mathcal{A}_\varphi$, $\mathcal{R}eq_D(A') \subseteq \mathcal{R}eq_D(A)$, and thus $rank(A) \leq rank(A')$ and $|\mathcal{R}eq_D(A) \setminus \mathcal{R}eq_D(A')| \leq rank(A')$. We can see the *rank* of an atom as the "number of degrees of freedom" that it gives to the atoms that stay "above it". In particular, by definition, for every φ-row $row = A_0^{m_0} \cdots A_n^{m_n}$, we have $rank(A_0) \geq \ldots \geq rank(A_n)$. The next lemmas use the notion of rank to provide an insight on how consecutive φ-rows are connected (see Figure 3.3).

Lemma 3.3.8. *Let row_1 and row_2 be two φ-rows, with $row_1 = A_0^{m_0} \cdots A_n^{m_n}$ and $row_1 {=^{row_\varphi}\!\!\Rightarrow} row_2$. For each $0 \leq i \leq n$, let $st_i = \sum_{0 \leq j < i} m_j$. If, for some $st_i < j \leq st_i + m_i$, the atom $row_2(j)$ is reflexive, then for each $j \leq j' \leq st_i + m_i$, $row_2(j') = row_2(j)$.*

Proof. If $j = st_i + m_i$ there is nothing to prove. Thus we consider $j < st_i + m_i$. Since $row_2(j)$ is reflexive, $\mathcal{O}bs_D(row_2(j)) \subseteq \mathcal{R}eq_D(row_2(j))$. Since $row_1 {=^{row_\varphi}\!\!\Rightarrow} row_2$, we have that $\mathcal{R}eq_D(A_i), \mathcal{O}bs_D(A_i) \subseteq \mathcal{R}eq_D(row_2(j))$ and $\mathcal{R}eq_D(row_2(j+1)) = \mathcal{R}eq_D(row_2(j)) \cup \mathcal{O}bs_D(row_2(j)) \cup \mathcal{R}eq_D(A_i) \cup \mathcal{O}bs_D(A_i) = \mathcal{R}eq_D(row_2(j))$. Moreover, from $row_1 {=^{row_\varphi}\!\!\Rightarrow} row_2$, we have $row_2(j) \cap \mathcal{AP} = row_2(j-1) \cap A_i \cap \mathcal{AP}$

and $row_2(j+1) \cap \mathcal{AP} = row_2(j) \cap A_i \cap \mathcal{AP} = row_2(j-1) \cap A_i \cap \mathcal{AP}$. Thus, $row_2(j+1) = row_2(j)$, because the two atoms feature exactly the same requests and proposition letters (Proposition 3.2.3). Then, since $A_i\ row_2(j) =D_\varphi\Rightarrow row_2(j+1)$, by iterating the reasoning and exploiting Lemma 3.3.2, we can conclude that $row_2(j) = row_2(j')$ for each $j \leq j' \leq st_i + m_i$. □

Lemma 3.3.9. *Let row_1 and row_2 be two φ-rows, with $row_1 = A_0^{m_0} \cdots A_n^{m_n}$ and $row_1 =^{row_\varphi}\Rightarrow row_2$. For each $0 \leq i \leq n$, let $st_i = \sum_{0 \leq j < i} m_j$. If $m_i > rank(A_i)$, then there exists $st_i < k \leq st_i + m_i$ such that:*

- *$k \leq st_i + 1 + rank(A_i)$;*
- *$row_2(k)$ is reflexive;*
- *$rank(row_2(j)) > rank(row_2(j+1))$ for each $st_i < j < k$;*
- *$row_2(j) = row_2(j+1)$ for each $k \leq j < st_i + m_i$;*
- *if m' is the exponent of the atom $row_2(k)$ in $row_2(st_i + 1, st_i + m_i)$ then $m' > rank(row_2(k))$.*

Proof. If $m_i = 1$, by hypothesis we have $rank(A_i) = 0$. Hence, $rank(row_2(st_i + 1)) = 0$, because $row_1 =^{row_\varphi}\Rightarrow row_2$, and thus $row_2(st_i + 1)$ is (trivially) reflexive. All claims hold by choosing $k = st_i + 1$.

Let us then assume $m_i > 1$. It can be easily shown that if we have two atoms A and A' such that $A\ D_\varphi\ A'$ and A' is irreflexive, then $rank(A) < rank(A')$. Moreover, Lemma 3.3.8 proves that we cannot interleave reflexive atoms with irreflexive ones "above" the A_i's (i.e., all irreflexive atoms must "come before" reflexive ones in the part of row_2 "above" the A_i's). Thus, in the worst case, the atoms $row_2(st_i+1), \ldots, row_2(st_i+rank(A_i))$ may be irreflexive (as $rank(row_2(st_i+1)) > \ldots > rank(row_2(st_i + rank(A_i)))$ and $rank(A_i) \geq rank(row_2(st_i + 1))$). Note that these irreflexive atoms may be the "first" $rank(A_i)$ atoms above the A_i's only, and not the "first" $rank(A_i) + 1$, since any atom with rank equal to 0 is reflexive. We conclude that $row_2(st_i + rank(A_i) + 1)$ must be reflexive. Thus, we can choose $k = st_i + rank(A_i) + 1$. Since by hypothesis $m_i \geq rank(A_i) + 1$, we get that $st_i < k \leq st_i + m_i$. Finally, by Lemma 3.3.8, $row_2(j) = row_2(j+1)$ for each $k \leq j < st_i + m_i$.

As for the last claim, $rank(row_2(k)) \leq rank(row_2(st_i + 1)) - (k - st_i - 1) \leq rank(A_i) - (k - st_i - 1)$. Then, the exponent m' of $row_2(k)$ in $row_2(st_i+1, st_i+m_i)$ is such that $m' \geq m_i - (rank(A_i) - rank(row_2(k)))$, that is, at least $m_i - (rank(A_i) - rank(row_2(k)))$ atoms labelled by $row_2(k)$ occur in the block $st_i+1, \ldots, st_i+m_i$ of row_2 (see Figure 3.3). Since by hypothesis $m_i > rank(A_i)$, then $m_i - rank(A_i) > 0$ and $rank(row_2(k)) < m'$. □

Now we introduce an equivalence relation $\sim$ over $\mathcal{R}ows_\varphi$ which is the key ingredient of the proofs showing that both SAT and MC for D formulas are decidable.

Definition 3.3.10. Given two φ-rows $row_1 = A_0^{m_0} \cdots A_n^{m_n}$ and $row_2 = \hat{A}_0^{\hat{m}_0} \cdots \hat{A}_{\hat{n}}^{\hat{m}_{\hat{n}}}$, we say that they are *equivalent*, written $row_1 \sim row_2$, if

- $n = \hat{n}$, and
- for each $0 \leq i \leq n$, $A_i = \hat{A}_i$, and $m_i = \hat{m}_i$ or both m_i and $\hat{m}_i$ are (strictly) greater than $rank(A_i)$.

Note that if two rows feature the same set of atoms, the lower the rank of an atom A_i, the lower the number of occurrences of A_i both the rows have to feature in order to belong to the same equivalence class. As an example, let row_1 and row_2 be two rows with $row_1 = A_0^{m_0} A_1^{m_1}$, $row_2 = A_0^{\overline{m}_0} A_1^{\overline{m}_1}$, $rank(A_0) = 4$, and $rank(A_1) = 3$. If $m_1 = 4$ and $\overline{m}_1 = 5$, they are both greater than $rank(A_1)$, and thus they do not violate the condition for $row_1 \sim row_2$. On the other hand, if $m_0 = 4$ and $\overline{m}_0 = 5$, we have that m_0 is less than or equal to $rank(A_0)$. Thus, in this case, $row_1 \nsim row_2$ due to the indexes of A_0. This happens because $rank(A_0)$ is greater than $rank(A_1)$. Two cases in which $row_1 \sim row_2$ are $m_0 = \overline{m}_0$ and $m_0, \overline{m}_0 \geq 5$.

The relation $\sim$ has finite index, which is roughly bounded by the number of all the possible φ-rows $row = A_0^{m_0} \cdots A_n^{m_n}$, with exponents m_i ranging from 1 to $|\varphi|$. Since (i) the number of possible atoms is $2^{|\varphi|}$, (ii) the number of *distinct* atoms in any φ-row is at most $2|\varphi|$, and (iii) the number of possible functions $f : \{1, \ldots, \ell\} \to \{1, \ldots, |\varphi|\}$ is $|\varphi|^\ell$, we have that the number of distinct equivalence classes of $\sim$ is bounded by

$$\sum_{j=1}^{2|\varphi|} (2^{|\varphi|})^j \cdot |\varphi|^j \leq 2^{3|\varphi|^2},$$

which is exponential in the length of the input formula φ. We denote the set of equivalence classes of $\sim$ over all the possible φ-rows by $\mathcal{R}ows_\varphi^\sim$.

Now we extend $=^{row_\varphi}\!\!\rightarrow$ to equivalence classes of $\sim$ in the following way.

Definition 3.3.11. Given two φ-row classes $[row_1]_\sim$ and $[row_2]_\sim$, we say that $[row_2]_\sim$ is a successor of $[row_1]_\sim$, written $[row_1]_\sim =^{row_\varphi}\!\!\rightarrow [row_2]_\sim$, if there exist $row_1' \in [row_1]_\sim$ and $row_2' \in [row_2]_\sim$ such that $row_1' =^{row_\varphi}\!\!\rightarrow row_2'$.

The following fundamental result proves that if some $row_1' \in [row_1]_\sim$ has a successor in $[row_2]_\sim$, then *every* φ-row of $[row_1]_\sim$ has a successor in $[row_2]_\sim$ (hence the definition of $=^{row_\varphi}\!\!\rightarrow$ over the equivalence classes of $\sim$ is well-given).

Lemma 3.3.12. *Given two φ-row classes $[row_1]_\sim$ and $[row_2]_\sim$ such that $[row_1]_\sim =^{row_\varphi}\!\!\rightarrow [row_2]_\sim$, for every $row \in [row_1]_\sim$ there exists $row' \in [row_2]_\sim$ such that $row =^{row_\varphi}\!\!\rightarrow row'$.*

Before starting the proof we observe that, since $[row_1]_\sim \Rightarrow^{=row_\varphi} [row_2]_\sim$, there exists $row \in [row_1]_\sim$ and $\overline{row} \in [row_2]_\sim$ such that $row \Rightarrow^{=row_\varphi} \overline{row}$. Thus, if $|[row_1]_\sim| = 1$, the lemma trivially holds.

In the following we suppose that $|[row_1]_\sim| \geq 2$: then there is $row' \neq row$ such that $row' \sim row$. We let $row = A_0^{m_0} \cdots A_n^{m_n}$; by definition we have $row' = A_0^{m'_0} \cdots A_n^{m'_n}$, where for every $0 \leq i \leq n$, $m'_i = m_i$ if $m_i \leq rank(A_i)$, and $m'_i > rank(A_i)$ if $m_i > rank(A_i)$.

Hereafter, we will say that "A_i exceeds its rank in row" whenever $m_i > rank(A_i)$. Moreover, for the sake of the proof it is worth observing that any subword $row(0, i)$ is a φ-row for every $0 \leq i < |row|$, and, since $row \Rightarrow^{=row_\varphi} \overline{row}$, then $row(0, i) \Rightarrow^{=row_\varphi} \overline{row}(0, i+1)$.

We now want to determine $\overline{row}'$ such that $row' \Rightarrow^{=row_\varphi} \overline{row}'$ and $\overline{row}' \sim \overline{row}$, proving the lemma. To this aim, we define a list of φ-rows $\overline{row}'_{-1}, \overline{row}'_0, \ldots, \overline{row}'_n$ such that $\overline{row}'_{-1} = \overline{row}(0)$ and, for all $0 \leq i \leq n$, $\overline{row}'_i$ extends $\overline{row}'_{i-1}$; we will show that $\overline{row}'_n = \overline{row}'$ (i.e., the row we want to determine).

The φ-rows $\overline{row}'_i$, for $i \geq 0$, are formally defined as:

$$\overline{row}'_i = \begin{cases} \overline{row}'_{i-1} \cdot \overline{row}(st_i + 1, st_i + m_i) & \text{if } m_i = m'_i \\ \overline{row}'_{i-1} \cdot \overline{row}(st_i + 1, st_i + m'_i) & \text{if } rank(A_i) < m'_i < m_i \\ \overline{row}'_{i-1} \cdot \overline{row}(st_i + 1, st_i + m_i) \cdot (\overline{row}(st_i + m_i))^{m'_i - m_i} & \text{if } rank(A_i) < m_i < m'_i \end{cases}$$

Intuitively, if $m_i = m'_i$, $\overline{row}'_i$ is obtained by appending the block $\overline{row}(st_i+1, st_i+m_i)$ to $\overline{row}'_{i-1}$. If $rank(A_i) < m'_i < m_i$, we do almost the same thing, but we "truncate" the block after m'_i positions, instead of m_i. Finally, if $rank(A_i) < m_i < m'_i$, then $\overline{row}'_i$ is obtained by appending $\overline{row}(st_i + 1, st_i + m_i)$ to $\overline{row}'_{i-1}$, with $m'_i - m_i$ additional repetitions of $\overline{row}(st_i + m_i)$ at the end, as we need to "pump" $\overline{row}'_i$ in order for it to have a suitable length with respect to row', thus respecting $\Rightarrow^{=row_\varphi}$.

We are now ready to prove Lemma 3.3.12

Proof. We show that the following properties hold by induction on $0 \leq i \leq n$, where $st_i = \sum_{0 \leq j < i} m_j$ and $st'_i = \sum_{0 \leq j < i} m'_j$ are respectively the starting positions of A_i in row and row':

0. $\overline{row}'_i$ is a φ-row featuring atoms of $\overline{row}(0, st_i + m_i)$;

1. $\overline{row}'_i \sim \overline{row}(0, st_i + m_i)$;

2. $row'(0, st'_i + m'_i - 1) \Rightarrow^{=row_\varphi} \overline{row}'_i$.

In particular, for $i = n$, the properties allow us to show that $\overline{row}'_n \sim \overline{row}$ and $row' \Rightarrow^{=row_\varphi} \overline{row}'_n$, proving the lemma. Figure 3.4 illustrates such properties.

The base case for $i = 0$ is omitted, as it is just a simplification of the inductive step. Let us consider $i \geq 1$. By the inductive hypothesis, we have $\overline{row}'_{i-1} \sim \overline{row}(0, st_{i-1} + m_{i-1})$ (hence in particular $\mathrm{lst}(\overline{row}'_{i-1}) = \overline{row}(st_{i-1} + m_{i-1})$, where $\mathrm{lst}(row)$ denotes the last atom of row) and $row'(0, st'_{i-1} + m'_{i-1} - 1) \Rightarrow^{=row_\varphi} \overline{row}'_{i-1}$. We consider three cases, corresponding to the cases of the definition of $\overline{row}'_i$.

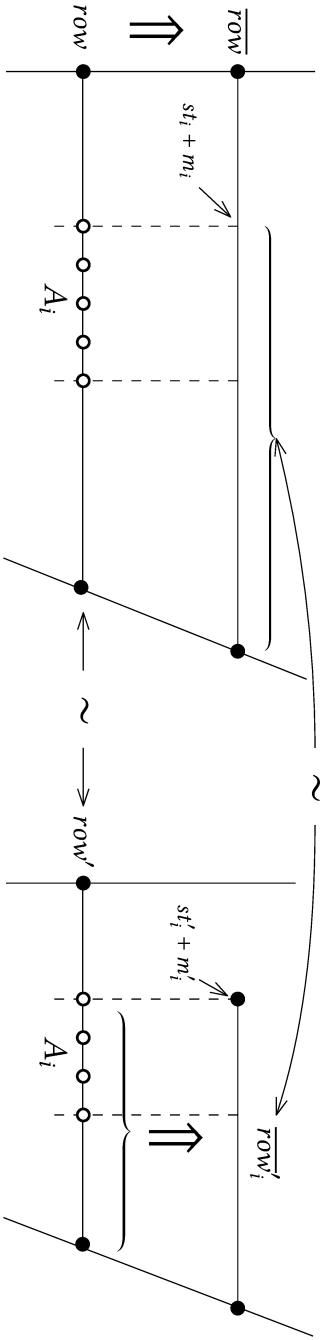


Figure 3.4: Illustration of the properties of $\overline{row}'_i$.

- $m_i = m'_i$. Then $\overline{row}'_i = \overline{row}'_{i-1} \cdot \overline{row}(st_i{+}1, st_i{+}m_i)$ is a φ-row. If $\mathrm{lst}(\overline{row}'_{i-1}) \neq (\overline{row}(st_i + 1))$, then it immediately follows that $\overline{row}'_i \sim \overline{row}(0, st_i + m_i)$. Conversely, if $\mathrm{lst}(\overline{row}'_{i-1}) = \overline{row}(st_i + 1) = \overline{row}(st_{i-1} + m_{i-1})$, then $\overline{row}(st_i + 1)$ is reflexive. By Lemma 3.3.8, $\overline{row}(st_i + 1) = \ldots = \overline{row}(st_i + m_i)$. It follows that $\overline{row}'_i \sim \overline{row}(0, st_i + m_i)$, because we append to $\overline{row}'_{i-1}$ exactly $\overline{row}(st_i + 1, st_i + m_i)$ (note that either $\overline{row}(st_{i-1} + m_{i-1})$ already exceeded its rank in $\overline{row}(0, st_{i-1} + m_{i-1})$ and so did $\mathrm{lst}(\overline{row}'_{i-1})$ in $\overline{row}'_{i-1}$, or the ranks of $\overline{row}(st_{i-1} + m_{i-1})$ in $\overline{row}(0, st_{i-1} + m_{i-1})$ and of $\mathrm{lst}(\overline{row}'_{i-1})$ in $\overline{row}'_{i-1}$ were equal). Moreover, in both cases, $row'(0, st'_i + m'_i - 1) {=^{row_\varphi}\!\!\rightarrow} \overline{row}'_i$ by definition of ${=^{D_\varphi}\!\!\rightarrow}$, as $row'(st'_i) = row(st_i) = A_i$ (recall that $row \sim row'$).

- $rank(A_i) < m'_i < m_i$. Then $\overline{row}'_i = \overline{row}'_{i-1} \cdot \overline{row}(st_i + 1, st_i + m'_i)$ is a φ-row. First, we have $row'(0, st'_i + m'_i - 1) {=^{row_\varphi}\!\!\rightarrow} \overline{row}'_i$ by definition of ${=^{D_\varphi}\!\!\rightarrow}$, as $row'(st'_i) = row(st_i) = A_i$. By Lemma 3.3.9, being $rank(A_i) < m_i$, there is $st_i{+}1 \leq k \leq st_i{+}m_i$ such that $\overline{row}(k)$ exceeds its rank in $\overline{row}(st_i{+}1, st_i{+}m_i)$, $\overline{row}(k) = \ldots = \overline{row}(st_i + m_i)$, and $k \leq st_i + 1 + rank(A_i)$. We first note that, being $1 + rank(A_i) \leq m'_i$, we have $k \leq st_i + m'_i$, thus the atom $\overline{row}(k)$ is present in the block $\overline{row}(st_i + 1, st_i + m'_i)$ of $\overline{row}'_i$. By Lemma 3.3.9, being also $rank(A_i) < m'_i$, we have that the atom $\overline{row}(k)$ exceeds its rank in the block $\overline{row}(st_i + 1, st_i + m'_i)$ of $\overline{row}'_i$ as well. Thus $\overline{row}'_i \sim \overline{row}(0, st_i + m_i)$.

- $rank(A_i) < m_i < m'_i$. Then $\overline{row}'_i = \overline{row}'_{i-1} \cdot \overline{row}(st_i + 1, st_i + m_i) \cdot (\overline{row}(st_i + m_i))^{m'_i - m_i}$. By Lemma 3.3.9, being $rank(A_i) < m_i$, there is $st_i{+}1 \leq k \leq st_i{+}m_i$ such that $\overline{row}(k)$ is reflexive, it exceeds its rank in $\overline{row}(st_i + 1, st_i + m_i)$, and $\overline{row}(k) = \ldots = \overline{row}(st_i + m_i)$. First we observe that $\overline{row}'_i$ is a φ-row, being $\overline{row}(st_i{+}m_i)$ reflexive. Then the atom $\overline{row}(k)$ is (trivially) also present in the block $\overline{row}(st_i{+}1, st_i{+}m_i)$ of $\overline{row}'_i$. As a consequence $\overline{row}'_i \sim \overline{row}(0, st_i{+}m_i)$. Moreover, $A_i\ \overline{row}(st_i + m_i) {=^{D_\varphi}\!\!\rightarrow} \overline{row}(st_i + m_i)$ (as $\overline{row}(st_i + m_i)$ is reflexive), therefore $row'(0, st'_i + m'_i - 1) {=^{row_\varphi}\!\!\rightarrow} \overline{row}'_i$. □

The following result arranges the equivalence classes $\mathcal{R}ows^{\sim}_{\varphi}$ in a graph $G_{\varphi\sim}$.

Definition 3.3.13. Let φ be a D formula. The *$\varphi\sim$graph* of φ is defined as $G_{\varphi\sim} = (\mathcal{R}ows^{\sim}_{\varphi}, {=^{row_\varphi}\!\!\rightarrow})$.

The next theorem reduces the problem of SAT checking for a D formula φ over finite linear orders (equivalent, by Proposition 3.2.9, to deciding if there is a homogeneous fulfilling compass φ-structure that features φ) to a reachability problem in the $\varphi\sim$graph, allowing us to determine the computational complexity of the former problem.

Theorem 3.3.14. *Given a* D *formula φ, there exists a homogeneous fulfilling compass φ-structure $\mathcal{G} = (\mathbb{P}_{\mathbb{S}}, \mathcal{L})$ that features φ if and only if there exists a path in $G_{\varphi^\sim} = (\mathcal{R}ows^\sim_\varphi, {=^{row_\varphi}\!\!\Rightarrow})$ from some class $[row]_\sim \in \mathcal{R}ows^\sim_\varphi$ to some class $[row']_\sim \in \mathcal{R}ows^\sim_\varphi$ such that*

1. *there exists $row_1 \in [row]_\sim$ with $|row_1| = 1$, and*
2. *there exist $row_2 \in [row']_\sim$ and $0 \leq i < |row_2|$ such that $\varphi \in row_2(i)$.*

Proof. Preliminarily we observe that, in (1.), if $|row_1| = 1$, then $\{row_1\} = [row]_\sim$; moreover, in (2.), if for $row_2 \in [row']_\sim$ and $0 \leq i < |row_2|$ we have that $\varphi \in row_2(i)$, then for any $row'_2 \in [row']_\sim$, there is $0 \leq i' < |row'_2|$ such that $\varphi \in row'_2(i')$.

($\Rightarrow$) Let us consider a homogeneous fulfilling compass φ-structure $\mathcal{G} = (\mathbb{P}_{\mathbb{S}}, \mathcal{L})$ that features φ. By Lemma 3.3.5 and 3.3.7, $\mathcal{L}(0,0) {=^{row_\varphi}\!\!\Rightarrow} row_1 {=^{row_\varphi}\!\!\Rightarrow} \cdots {=^{row_\varphi}\!\!\Rightarrow} row_{\max(\mathcal{S})}$. Thus there exist two indexes $0 \leq j \leq \max(\mathcal{S})$ and $0 \leq i < |row_j|$ for which $\varphi \in row_j(i)$. By Definition 3.3.11, we get that $[\mathcal{L}(0,0)]_\sim {=^{row_\varphi}\!\!\Rightarrow} [row_1]_\sim {=^{row_\varphi}\!\!\Rightarrow} \cdots {=^{row_\varphi}\!\!\Rightarrow} [row_j]_\sim$ is a path in $G_{\varphi^\sim}$; it is immediate to check that it fulfills requirements (1.) and (2.).

($\Leftarrow$) Let us assume that there exists a path $[row_0]_\sim {=^{row_\varphi}\!\!\Rightarrow} \cdots {=^{row_\varphi}\!\!\Rightarrow} [row_m]_\sim$ in $G_{\varphi^\sim} = (\mathcal{R}ows^\sim_\varphi, {=^{row_\varphi}\!\!\Rightarrow})$ for which $|row_0| = 1$ and there exists i such that $\varphi \in row_m(i)$. By applying repeatedly Lemma 3.3.12 we get that there exists a sequence $row'_0 {=^{row_\varphi}\!\!\Rightarrow} \cdots {=^{row_\varphi}\!\!\Rightarrow} row'_m$ of φ-rows where $row'_0 = row_0$, for every $0 \leq j \leq m$, $row'_j \in [row_j]_\sim$, and there exists i' such that $\varphi \in row'_m(i')$. We observe that, by Definition 3.3.6, $|row'_j| = |row'_{j-1}| + 1$ for $1 \leq j \leq m$ and, since $|row'_0| = 1$, we have $|row'_j| = j + 1$. Let us now define $\mathcal{G} = (\mathbb{P}_{\mathbb{S}}, \mathcal{L})$ where $\mathcal{S} = \{0, \ldots, m\}$ and $\mathcal{L}(x, y) = row'_y(y - x)$ for every $0 \leq x \leq y \leq m$. By Lemma 3.3.7, $\mathcal{G}$ is a fulfilling homogeneous compass φ-structure. Finally, since $\varphi \in row'_m(i')$, $\mathcal{G}$ features φ. □

The size of $G_{\varphi^\sim} = (\mathcal{R}ows^\sim_\varphi, {=^{row_\varphi}\!\!\Rightarrow})$ is bounded by $|\mathcal{R}ows^\sim_\varphi|^2$, which is exponential in $|\varphi|$. However, it is possible to (non-deterministically) perform a reachability in $G_{\varphi^\sim}$ by using space *logarithmic* in $|\mathcal{R}ows^\sim_\varphi|^2$.

○

Algorithm 3.1 SAT(φ)

1: $M \leftarrow 2^{3|\varphi|^2}$, $step \leftarrow 0$ and $row \leftarrow A$ for some atom $A \in \mathcal{A}_\varphi$ with $\mathcal{R}eq_D(A) = \emptyset$
2: **if** there exists $0 \leq i < |row|$ such that $\varphi \in row(i)$ **then**
3: **return** "satisfiable"
4: **if** $step = M - 1$ **then**
5: **return** "unsatisfiable"
6: Non-deterministically generate a φ-row row' and check that $row {=^{row_\varphi}\!\!\Rightarrow} row'$
7: $step \leftarrow step + 1$ and $row \leftarrow row'$
8: Go back to line 2

The *non-deterministic* procedure $\texttt{SAT}(\varphi)$ in Algorithm 3.1 exploits this fact in order to decide the satisfiability of a D formula φ, by using only a working space *polynomial* in $|\varphi|$: it searches for a suitable path in $G_{\varphi\sim}$, namely $[row_0]_\sim =^{row_\varphi}\!\!\Rightarrow \cdots =^{row_\varphi}\!\!\Rightarrow [row_m]_\sim$, where $row_0 = A$ for $A \in \mathcal{A}_\varphi$ with $\mathcal{R}eq_D(A) = \emptyset$, $m < M$, and $\varphi \in row_m(i)$ for $0 \leq i < |row_m|$. At the j-th iteration of line 6., row_j is non-deterministically generated, and it is checked whether $row_{j-1} =^{row_\varphi}\!\!\Rightarrow row_j$. The procedure terminates after at most M iterations, where M is the maximum possible length of a simple path in $G_{\varphi\sim}$.

The working space used by the procedure is polynomial: M and *step* (which ranges in $[0, M-1]$) can be encoded in binary with $\lceil \log_2 M \rceil + 1 = O(|\varphi|^2)$ bits. At each step, we need to keep track of two φ-rows at a time, the current one, *row*, and its successor, *row′*: each φ-row can be represented as a sequence of at most $2|\varphi|$ (distinct) atoms, each one with an exponent that, by construction, cannot exceed M. Moreover, each φ-atom A can be represented using exactly $|\varphi|$ bits (for each $\psi \in \mathsf{CL}(\varphi)$, we set a bit to 1 if $\psi \in A$, and to 0 if $\neg\psi \in A$). Hence a φ-row can be encoded using $2|\varphi| \cdot (|\varphi| + \lceil \log_2 M \rceil + 1) = O(|\varphi|^3)$ bits. Finally, the condition $row =^{row_\varphi}\!\!\Rightarrow row'$ can be checked by $O(|\varphi|^2)$ bits of space once we have guessed *row′*. This analysis entails the following result (we recall that **NPSPACE** = **PSPACE**).

Theorem 3.3.15. *The SAT problem for* D *formulas over finite linear orders is in* ***PSPACE****.*

We now outline which are the modifications to the previous concepts needed for proving the decidability of SAT for D with the strict relation $\boxdot$, in place of $\sqsubset$. It is sufficient to replace the definitions of $=^{D_\varphi}\!\!\Rightarrow$, *φ-row* and $=^{row_\varphi}\!\!\Rightarrow$ with the following ones. For the sake of simplicity, we introduce a dummy atom $\boxdot$, for which we assume $\mathcal{R}eq_D(\boxdot) = \mathcal{O}bs_D(\boxdot) = \emptyset$.

Definition 3.3.16. Given four φ-atoms A_1, A_3, $A_4 \in \mathcal{A}_\varphi$ and $A_2 \in \mathcal{A}_\varphi \cup \{\boxdot\}$, we say that A_4 is D_φ $\boxdot$-generated by A_1, A_2, A_3, written $A_1, A_2, A_3 =^{D_\varphi \boxdot}\!\!\Rightarrow A_4$, if:

- $A_4 \cap \mathcal{AP} = A_1 \cap A_3 \cap \mathcal{AP}$ and
- $\mathcal{R}eq_D(A_4) = \mathcal{R}eq_D(A_1) \cup \mathcal{R}eq_D(A_3) \cup \mathcal{O}bs_D(A_2)$.

The idea of this definition is that, if an interval $[x, y]$, with $x < y$, is labeled by A_4, and the three subintervals $[x, y-1]$, $[x+1, y-1]$, and $[x+1, y]$ by A_1, A_2, A_3, respectively, we want $A_1, A_2, A_3 =^{D_\varphi \boxdot}\!\!\Rightarrow A_4$. In particular, if $x = y - 1$, then $A_2 = \boxdot$ (because $[x+1, y-1]$ is not a valid interval). Note that only $[x+1, y-1] \boxdot [x, y]$, hence we want $\mathcal{O}bs_D(A_2) \subseteq \mathcal{R}eq_D(A_4)$; moreover, since

the requests of A_1 and A_3 are fulfilled by a strict subinterval of $[x, y]$, it must be $\mathcal{R}eq_D(A_1) \subseteq \mathcal{R}eq_D(A_4)$ and $\mathcal{R}eq_D(A_3) \subseteq \mathcal{R}eq_D(A_4)$.

Definition 3.3.17. A φ-$\boxdot$-row is a finite sequence of φ-atoms

$$row = A_0^{m_0} \cdots A_n^{m_n},$$

such that for each $0 \leq i \leq n$, we have $m_i > 0$, and for each $0 \leq j < i$, it holds $\mathcal{R}eq_D(A_j) \subseteq \mathcal{R}eq_D(A_i)$, $A_i \neq A_j$, and $(A_j \cap \mathcal{AP}) \supseteq (A_i \cap \mathcal{AP})$. Moreover, $\mathcal{R}eq_D(A_0) = \emptyset$.

Definition 3.3.18. Given two φ-rows row and row', we say that row' is a successor of row, written as $row =^{row_\varphi \boxdot}\!\Rightarrow row'$, if $|row'| = |row| + 1$, and for all $0 \leq i < |row|$, $row(i)row(i-1)row'(i) =^{D_\varphi \boxdot}\!\Rightarrow row'(i+1)$, where we assume $row(i-1) = \boxdot$ if $i = 0$.

We will come back to SAT later, showing the **PSPACE**-completeness of such problem (under both the *strict* and the *proper* semantic variants). In the next section, we will consider MC for D.

3.4 MC for D over Kripke structures

In this section we focus our attention on the MC problem for D formulas over Kripke structures, under homogeneity. Let us start with the following (technical) definition, basically mapping traces of a Kripke structure into interval models.

Definition 3.4.1. The interval model $\mathbf{M}_\rho = (\mathbb{I}(\mathbb{S}), \circ, \mathcal{V})$ *induced by a trace* ρ of a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ is the homogeneous interval model such that:

1. $\mathcal{S} = \{0, \ldots, |\rho| - 1\}$, and
2. for all $x \in \mathcal{S}$ and $p \in \mathcal{AP}$, $[x, x] \in \mathcal{V}(p)$ if and only if $p \in \mu(\rho(x+1))$.[a]

[a] We add 1 to the index x of ρ just because traces are 1-based (whereas linear orders and interval models are 0-based).

Clearly we have $\mathcal{K}, \rho(i+1, j+1) \models \psi$ if and only if $\mathbf{M}_\rho, [i, j] \models \psi$.

We now describe how, with a slight modification of the previous SAT procedure, it is possible to derive a MC algorithm for D formulas φ over finite Kripke structures $\mathcal{K}$. The idea is to consider some finite linear orders—not all the possible ones, unlike the case of SAT—precisely those corresponding to (some) initial traces of $\mathcal{K}$, checking whether $\neg\varphi$ holds over them: in such a case we have found a

counterexample, and we can conclude that $\mathcal{K} \not\models \varphi$. To ensure this kind of "SAT driven by the traces of $\mathcal{K}$", we make a product between $\mathcal{K}$ and the previous graph $G_{\varphi\sim}$, getting what we call a "$(\varphi \sim \mathcal{K})$-*graph*". In the following, we will also exploit the notion of "compass structure *induced by a trace* ρ of $\mathcal{K}$", which is a fulfilling homogeneous compass φ-structure built from ρ and completely determined by it.

Given a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ and a D formula φ, we consider the $(\varphi \sim \mathcal{K})$-graph $G_{\varphi\sim\mathcal{K}}$, which is basically the product of $\mathcal{K}$ and $G_{\varphi\sim} = (\mathcal{R}ows^{\sim}_{\varphi}, {}^{=row_\varphi}\!\!\Rightarrow)$, and is formally defined as

$$G_{\varphi\sim\mathcal{K}} = (\Gamma, \Xi),$$

where:

- Γ is the maximal subset of $S \times \mathcal{R}ows^{\sim}_{\varphi}$ such that: if $(s, [row]_\sim) \in \Gamma$ then $\mu(s) = row(0) \cap \mathcal{AP}$;
- $\big((s_1, [row_1]_\sim), (s_2, [row_2]_\sim)\big) \in \Xi$ iff (i) $\big((s_1, [row_1]_\sim), (s_2, [row_2]_\sim)\big) \in \Gamma^2$, (ii) $(s_1, s_2) \in R$, and (iii) $[row_1]_\sim \,{}^{=row_\varphi}\!\!\Rightarrow [row_2]_\sim$.

Note that the definition of Γ is well-given, since for all $row' \in [row]_\sim$, $row'(0) = row(0)$. The size of $G_{\varphi\sim\mathcal{K}}$ is bounded by $(|S| \cdot |\mathcal{R}ows^{\sim}_{\varphi}|)^2$.

Given a generic trace ρ of $\mathcal{K}$, we define the compass φ-structure *induced by* ρ as the fulfilling homogeneous compass φ-structure $\mathcal{G}_{(\mathcal{K},\rho)} = (\mathbb{P}_{\mathbb{S}}, \mathcal{L})$, where $\mathcal{S} = \{0, \ldots, |\rho| - 1\}$, and for $0 \le x < |\rho|$, $\mathcal{L}(x, x) \cap \mathcal{AP} = \mu(\rho(x + 1))$ and $\mathcal{R}eq_D(\mathcal{L}(x, x)) = \emptyset$. Note that, given ρ, $\mathcal{G}_{(\mathcal{K},\rho)}$ always exists and is unique: all φ-atoms $\mathcal{L}(x, x)$ "on the diagonal" are determined by the labeling of $\rho(x + 1)$ (and by the absence of requests). Moreover, by Lemma 3.3.7, all the other atoms $\mathcal{L}(x, y)$, for $0 \le x < y < |\rho|$, are determined by the ${}^{=row_\varphi}\!\!\Rightarrow$ relation between φ-rows.

The following property can easily be proved by induction.

Proposition 3.4.2. *Given a finite Kripke structure $\mathcal{K}$, a trace ρ of $\mathcal{K}$, and a* D *formula φ, for all $0 \le x \le y < |\rho|$ and for all subformulas ψ of φ, we have $\mathcal{K}, \rho(x + 1, y + 1) \models \psi$ if and only if $\psi \in \mathcal{L}(x, y)$ in $\mathcal{G}_{(\mathcal{K},\rho)}$.*

We can now introduce Theorem 3.4.3, that can be regarded as a version of Theorem 3.3.14 for MC. Its proof is in Appendix A.4.

Theorem 3.4.3. *Given a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ and a* D *formula φ, there exists an initial trace ρ of $\mathcal{K}$ such that $\mathcal{K}, \rho \models \varphi$ if and only if there exists a path in $G_{\varphi\sim\mathcal{K}} = (\Gamma, \Xi)$ from some node $(s_0, [row]_\sim) \in \Gamma$ to some node $(s, [row']_\sim) \in \Gamma$ such that:*

1. *there exists $row_1 \in [row]_\sim$ with $|row_1| = 1$, and*
2. *there exists $row_2 \in [row']_\sim$ with $\varphi \in row_2(|row_2| - 1)$.*

Algorithm 3.2 `Counterexample`$(\mathcal{K}, \varphi)$

1: $M \leftarrow |S| \cdot 2^{3|\varphi|^2}$, $step \leftarrow 0$ and $(s, row) \leftarrow (s_0, A)$ for some atom $A \in \mathcal{A}_\varphi$ with $\mathcal{R}eq_D(A) = \emptyset$ and $A \cap \mathcal{AP} = \mu(s_0)$
2: **if** $\varphi \notin row(|row| - 1)$ **then**
3: **return** "yes"
4: **if** $step = M - 1$ **then**
5: **return** "no"
6: Non-deterministically choose s' such that $(s, s') \in R$
7: Non-deterministically generate a φ-row row' and check that $row'(0) \cap \mathcal{AP} = \mu(s')$ and $row =^{row_\varphi}\!\!\Rightarrow row'$
8: $step \leftarrow step + 1$ and $(s, row) \leftarrow (s', row')$
9: Go back to line 2

Now, analogously to the case of SAT, we can perform a reachability in $G_{\varphi\sim\mathcal{K}}$, exploiting the previous theorem to decide whether there is an initial trace ρ of $\mathcal{K}$ such that $\mathcal{K}, \rho \models \neg\varphi$, for a D-formula φ (i.e., the complementary problem of MC $\mathcal{K} \models \varphi$); ρ is called a "counterexample" to φ. The *non-deterministic* procedure `Counterexample`$(\mathcal{K}, \varphi)$ in Algorithm 3.2 searches for a suitable path in $G_{\varphi\sim\mathcal{K}}$, $(s_0, [row_0]_\sim) \xrightarrow{\Xi} \cdots \xrightarrow{\Xi} (s_m, [row_m]_\sim)$, where $row_0 = A \in \mathcal{A}_\varphi$ with $\mathcal{R}eq_D(A) = \emptyset$, $A \cap \mathcal{AP} = \mu(s_0)$, $m < M$, and $\neg\varphi \in row_m(|row_m| - 1)$ (i.e., $\varphi \notin row_m(|row_m| - 1)$). At the j-th iteration of lines 6./7., $(s_{j-1}, s_j) \in R$ is selected, and row_j is non-deterministically generated checking that $row_j(0) \cap \mathcal{AP} = \mu(s_j)$ and $row_{j-1} =^{row_\varphi}\!\!\Rightarrow row_j$.

Basically, the same observations about the working space of the procedure in Algorithm 3.1 can be done also for this algorithm, except for the space used to encode in binary $M \leq |S| \cdot 2^{3|\varphi|^2}$ and $step$, ranging in $[0, M - 1]$, which is $O(\log|S| + |\varphi|^2)$ bits. Moreover we need to store two states, s and s' of $\mathcal{K}$, that need $O(\log|S|)$ bits to be represented.

We conclude the section by stating the main result.

Theorem 3.4.4. *The MC problem for* D *formulas over finite Kripke structures is in* ***PSPACE****. Moreover, for constant-length formulas, it is in* ***NLOGSPACE****.*

Proof. Membership is immediate by the previous space analysis, and the fact that the complexity classes **NPSPACE** = **PSPACE** and **NLOGSPACE** are closed under complement. □

Finally, it is possible to adapt the procedure also for *strict* D (using Definitions 3.3.16–3.3.18).

In the next section we will comment on **PSPACE**-hardness, and thus **PSPACE**-completeness, of SAT and MC for D formulas.

3.5 An outline of PSPACE-completeness of SAT and MC for D

In Appendix A.5 we prove in detail that MC for D formulas is **PSPACE**-hard: we provide a reduction from the **PSPACE**-complete *problem of (non-)universality* of the language of a non-deterministic finite state automaton (NFA) $\mathcal{A}$ [HK11]. Here we only give an account of the main ideas behind the reduction.

We build a Kripke structure $\mathcal{K}_\mathcal{A}$ and a D formula $\Phi_\mathcal{A}$ which, together, allow us to consider *the* deterministic computation of a DFA $\mathcal{D}$, equivalent to the original NFA $\mathcal{A}$ (i.e., accepting the same language), over some word w *not* accepted by $\mathcal{A}$ (if such w exists). The computation is built "on-the-fly" (i.e., we do not construct the—possibly exponentially larger—equivalent DFA $\mathcal{D}$), and it is "captured" by an initial trace of $\mathcal{K}_\mathcal{A}$, which is a concatenation of suitable subtraces, each one encoding a state $\tilde{q}$ of $\mathcal{D}$, where $\tilde{q}$ is a subset of $\mathcal{A}$-states, by listing the $\mathcal{A}$-states that belong to $\tilde{q}$. Each $\mathcal{D}$-state (subtrace) is copied two times along the initial trace: this is necessary to enforce a suitable "orientation", something that D is unaware of, as such logic is completely "symmetric".

By this construction and Theorem 3.4.4, the following holds.

Theorem 3.5.1. *The MC problem for* D *formulas over finite Kripke structures is **PSPACE**-complete.*

Moreover,

Theorem 3.5.2. *The MC problem for* constant-length D *formulas over finite Kripke structures is **NLOGSPACE**-complete.*

Proof. Membership is stated by Theorem 3.4.4. To prove the **NLOGSPACE**-hardness, there exists a trivial reduction from the (**NLOGSPACE**-complete) *problem of (non-)reachability* of two nodes in a directed graph. □

We now briefly turn to D SAT, outlining its **PSPACE**-hardness over finite linear orders. Our construction (thoroughly worked out in Appendix A.6) mimics that of Section 3.2 and 3.3 of [MM14], in which the authors show that it is possible to build a D formula $\Psi_\mathcal{K}$ that encodes a finite Kripke structure $\mathcal{K}$. We instantiate it over $\mathcal{K}_\mathcal{A}$, getting $\Psi_{\mathcal{K}_\mathcal{A}}$, with the result that any finite linear order satisfying $\Psi_{\mathcal{K}_\mathcal{A}}$ represents an initial trace of $\mathcal{K}_\mathcal{A}$. In this way we get a reduction from the

problem of non-universality of the language of $\mathcal{A}$ to SAT for D: the language of $\mathcal{A}$ is non-universal if and only if the formula $\Psi_{\mathcal{K}_\mathcal{A}} \wedge \Phi_\mathcal{A}$ is satisfiable.

By also recalling Theorem 3.3.15, the next result is proved.

Theorem 3.5.3. *The SAT problem for* D *formulas over finite linear orders is* ***PSPACE****-complete.*

3.6 Conclusions

In this chapter we have shown **PSPACE**-completeness of both SAT and MC for D formulas φ under homogeneity. This result is proved thanks to a finite-index equivalence relation $\sim$ between rows of a compass φ-structure, that preserves fulfillment of compasses, and allows us to "contract" these structures whenever we find two related rows. Moreover, if a compass features φ before the contraction, then φ is still featured after it. These facts are exploited to build a SAT and an MC algorithm for D, which implicitly make compass contractions by a reachability check in suitable graphs, whose nodes are basically the equivalence classes of $\sim$. Since $\sim$ has finite index, exponentially bounded by $|\varphi|$, and the algorithms only need to keep track of two rows of a compass at a time (encoding them succinctly), **PSPACE**-membership of the problems follows.

In the next chapter, we will see that also the fragments $\mathsf{A\overline{A}B\overline{B}}$ and $\mathsf{A\overline{A}E\overline{E}}$ feature an MC problem complete for **PSPACE**. However, as the reader will see, completely different techniques and notions are used to prove the result.

4 HS fragments in PSPACE and EXPSPACE

The references for this chapter are [Boz+18g; Boz+16b; MMP15b; MMP17; Boz+16a].

Contents

4.1 The fragments $A\overline{A}B\overline{B}$ and $A\overline{A}E\overline{E}$ 74
4.1.1 Polynomial small-model property for $A\overline{A}B\overline{B}$ and $A\overline{A}E\overline{E}$ 74
4.1.2 **PSPACE** MC algorithm for $A\overline{A}E\overline{E}$ 80
4.1.3 **co-NP** MC algorithm for B and E 82
4.2 The fragments $A\overline{A}B\overline{B}E$ and $A\overline{A}E\overline{B}E$: exponential small-model property 84
4.3 Conclusions 90

In this chapter, we prove that MC for the HS fragment $A\overline{A}B\overline{B}$ (resp., $A\overline{A}E\overline{E}$), that allows one to express properties of future and past intervals, interval prefixes (resp., suffixes), and right (resp., left) interval extensions, is in **PSPACE** (under homogeneity). Since MC for the HS fragment featuring only one modality for right (resp., left) interval extensions $\overline{B}$ (resp., $\overline{E}$) can be shown to be **PSPACE**-hard (Appendix B.3), **PSPACE**-completeness immediately follows. Moreover, if we restrict HS to modalities either for interval prefixes B or for interval suffixes E only, MC turns out to be **co-NP**-complete.

These results are proved by a small-model property based on the notion of *induced trace*: given a trace ρ in a finite Kripke structure and a formula φ of $A\overline{A}B\overline{B}$/$A\overline{A}E\overline{E}$, it is always possible to build, by iteratively contracting ρ, another ("induced") trace, whose length is *polynomially bounded* in the size of φ and of the Kripke structure, which preserves the satisfiability of φ with respect to ρ.

The lower bound for BE MC (recall Section 2.4) shows that there is no way to provide an MC algorithm for the extension of $A\overline{A}B\overline{B}$ with E (resp., of $A\overline{A}E\overline{E}$ with B) with a "good" computational complexity. The picture is not so clear for the extension of $A\overline{A}B\overline{B}$ with $\overline{E}$ (resp., $A\overline{A}E\overline{E}$ with $\overline{B}$): membership of $A\overline{A}B\overline{B}\overline{E}$ (resp., $A\overline{A}E\overline{B}\overline{E}$) to **EXPSPACE** was already shown in [MMP17] and the **PSPACE**-hardness of MC for $\overline{B}$ (resp., $\overline{E}$) currently gives the best (unmatching) complexity lower bound. In this chapter we provide a much more understandable proof of membership to **EXPSPACE** of the MC problem for $A\overline{A}B\overline{B}\overline{E}$ (w.r.t. the one given in [MMP17]), which makes use of (a suitable extension of) the notion of induced trace, which we exploit in the proof of the small model property for $A\overline{A}B\overline{B}$.

Organization of the chapter.

- In Section 4.1, we first introduce the notion of induced trace and then we prove, via a contraction technique, a polynomial small-model property for $A\overline{A}B\overline{B}$ and $A\overline{A}E\overline{E}$ (Section 4.1.1), which allows us to devise a **PSPACE** MC algorithm for them (Section 4.1.2). In addition we consider the one-modality fragments B and E, and prove their **co-NP**-completeness (Section 4.1.3).
- In Section 4.2, we focus on the fragment $A\overline{A}B\overline{B}\overline{E}$ and the symmetric fragment $A\overline{A}E\overline{B}\overline{E}$. We first define the equivalence relation of *trace bisimilarity*, and then we introduce the notion of *prefix sampling*. With these tools, we prove an exponential small-model property for $A\overline{A}B\overline{B}\overline{E}$ (and $A\overline{A}E\overline{B}\overline{E}$), resulting into an easier proof of membership of the two fragments to **EXPSPACE**.

4.1 The fragments $A\overline{A}B\overline{B}$ and $A\overline{A}E\overline{E}$

In this section, we show that the MC problem for the fragments $A\overline{A}B\overline{B}$ and $A\overline{A}E\overline{E}$ is **PSPACE**-complete. Moreover, we prove that MC for the smaller fragments B and E is **co-NP**-complete.

4.1.1 Polynomial small-model property for $A\overline{A}B\overline{B}$ and $A\overline{A}E\overline{E}$

We first prove membership to **PSPACE** of the MC problem for the fragments $A\overline{A}B\overline{B}$ and $A\overline{A}E\overline{E}$ by showing that they feature a *polynomial small-model property*, that is, we prove that if a trace ρ of a finite Kripke structure $\mathcal{K}$ satisfies a given formula φ of $A\overline{A}E\overline{E}$ or $A\overline{A}B\overline{B}$, then there exists a trace π, whose length is polynomial in the sizes of φ and $\mathcal{K}$, starting from, and leading to, the same states as ρ, that satisfies φ. In the following, we focus on $A\overline{A}E\overline{E}$, being the case of $A\overline{A}B\overline{B}$ completely symmetric.

Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure. We start by introducing the basic notions of *induced trace* and *well-formed trace*, that will be extensively used to prove the polynomial small-model property. Intuitively, we say that a trace

$\pi \in \mathrm{Trc}_{\mathcal{K}}$ is induced by a trace $\rho \in \mathrm{Trc}_{\mathcal{K}}$ if it can be obtained from ρ by suitably contracting it, that is, by concatenating some subtraces of ρ. Well-formedness adds a condition on the suffixes of an induced trace. For the sake of readability, hereafter we denote by ρ^i the suffix $\rho(i, |\rho|)$ of ρ (hence, ρ^1 is just ρ).

Definition 4.1.1 (Induced and well-formed trace)**.** Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure and let $\rho \in \mathrm{Trc}_{\mathcal{K}}$, with $|\rho| = n$. A trace $\pi \in \mathrm{Trc}_{\mathcal{K}}$ is *induced* by ρ if there exists an increasing sequence of ρ-positions $i_1 < \ldots < i_k$, with $i_1 = 1$ and $i_k = n$, such that $\pi = \rho(i_1) \cdots \rho(i_k)$. For $j = 1, \ldots, k$, the π-position j and the ρ-position i_j are said to be *corresponding positions*.
An induced trace π is *well-formed* (with respect to ρ) if, for all π-positions j, with corresponding ρ-positions i_j, and all proposition letters $p \in \mathcal{AP}$, it holds that

$$\mathcal{K}, \pi^j \models p \iff \mathcal{K}, \rho^{i_j} \models p.$$

Example 4.1.2. As an example, let us consider Figure 4.1. The trace $\pi = \rho(1)\rho(4)\rho(5)\rho(7)\rho(10)$ is induced by ρ, provided that both ρ and π are traces of a Kripke structure $\mathcal{K}$, and the positions 1, 2, 3, 4, and 5 of π correspond to the positions $i_1 = 1$, $i_2 = 4$, $i_3 = 5$, $i_4 = 7$ and $i_5 = 10$ of ρ.

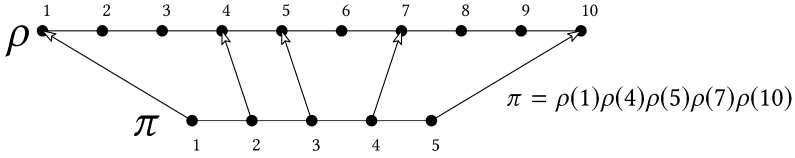


Figure 4.1: A trace π induced by ρ.

Note that if π is induced by ρ, then $\mathrm{fst}(\pi) = \mathrm{fst}(\rho)$, $\mathrm{lst}(\pi) = \mathrm{lst}(\rho)$, and $|\pi| \leq |\rho|$ ($|\pi| = |\rho|$ if and only if $\pi = \rho$).

Well-formedness implies that the suffix of π starting from position j and that of ρ starting from the corresponding position i_j agree over all the proposition letters in $\mathcal{AP}$, that is, they have the same "satisfaction pattern" of proposition letters. In particular, for all $p \in \mathcal{AP}$, $\mathcal{K}, \pi \models p$ if and only if $\mathcal{K}, \rho \models p$. It can be easily checked that the *well-formedness relation is transitive*.

The following proposition shows how it is possible to contract a trace while preserving the same satisfaction pattern of proposition letters with respect to suffixes. Such a criterion represents a "basic step" in a contraction process that will allow us to prove the polynomial small-model property.

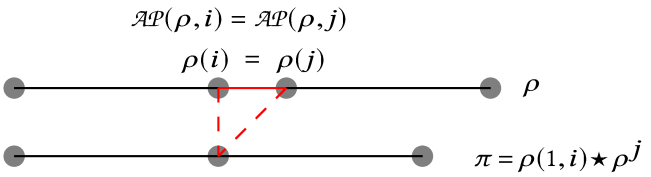


Figure 4.2: The contraction step of Proposition 4.1.3 ($\mathcal{AP}(\rho, k) = \{p \in \mathcal{AP} \mid \mathcal{K}, \rho^k \models p\}$).

Proposition 4.1.3. *Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure. For any trace $\rho \in \mathrm{Trc}_{\mathcal{K}}$, there exists a well-formed (with respect to ρ) trace $\pi \in \mathrm{Trc}_{\mathcal{K}}$ such that $|\pi| \leq |S| \cdot (|\mathcal{AP}| + 1)$.*

Proof. Let $\rho \in \mathrm{Trc}_{\mathcal{K}}$, with $|\rho| = n$. If $n \leq |S| \cdot (|\mathcal{AP}| + 1)$, the thesis trivially holds. Let us assume $n > |S| \cdot (|\mathcal{AP}| + 1)$. We show that there exists $\pi \in \mathrm{Trc}_{\mathcal{K}}$, with $|\pi| < n$, which is well-formed with respect to ρ.

Since $n > |S| \cdot (|\mathcal{AP}| + 1)$, there is some state $s \in S$ occurring in ρ at least $|\mathcal{AP}| + 2$ times. Assume that for all ρ-positions i and j, with $j > i$, if $\rho(i) = \rho(j) = s$, then there exists some $p \in \mathcal{AP}$ such that $\mathcal{K}, \rho^j \models p$ and $\mathcal{K}, \rho^i \not\models p$. This assumption leads to a contradiction, as the suffixes of ρ may feature at most $|\mathcal{AP}| + 1$ distinct satisfaction patterns of proposition letters (due to the homogeneity assumption in Definition 2.1.4), while there are at least $|\mathcal{AP}| + 2$ occurrences of s. As a consequence, there are two ρ-positions i and j, with $j > i$, such that $\rho(i) = \rho(j) = s$ and, for all $p \in \mathcal{AP}$, $\mathcal{K}, \rho^j \models p$ if and only if $\mathcal{K}, \rho^i \models p$ (see Figure 4.2 for a graphical account). It is easy to see that $\pi = \rho(1, i) \star \rho(j, n) \in \mathrm{Trc}_{\mathcal{K}}$ is well-formed with respect to ρ and $|\pi| < n$. If $|\pi| \leq |S| \cdot (|\mathcal{AP}| + 1)$, the thesis is proved; otherwise, the same basic step can be iterated a finite number of times, and the thesis follows by transitivity of the well-formedness relation. □

The next definition identifies some distinguished positions in a trace, called *witness positions*. As we will see in the proof of Theorem 4.1.5, if we perform a contraction (see the proof of Proposition 4.1.3, and its graphical account in Figure 4.2) between a pair of such positions, we get a trace which is equivalent to the original one with respect to satisfiability of the considered $\mathrm{A\overline{A}E\overline{E}}$ formula. In the following, we restrict ourselves to formulas in *negation normal form* (NNF), namely, formulas where negation is applied only to proposition letters. By using De Morgan's laws and the dual modalities [E], [$\overline{\mathrm{E}}$], [A], and [$\overline{\mathrm{A}}$] of ⟨E⟩, ⟨$\overline{\mathrm{E}}$⟩, ⟨A⟩, and ⟨$\overline{\mathrm{A}}$⟩, respectively, we can trivially convert (in linear time) a formula into an equivalent one in NNF, having at most double length.

Definition 4.1.4 (Witness position). Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure, $\rho \in \text{Trc}_{\mathcal{K}}$, and φ be a formula of $\overline{\text{A}}\overline{\text{A}}\text{E}\overline{\text{E}}$. Let us denote by $E(\varphi, \rho)$ the set of subformulas of the form $\langle \text{E} \rangle \psi$ of φ such that $\mathcal{K}, \rho \models \langle \text{E} \rangle \psi$. The *set $Wt(\varphi, \rho)$ of witness positions of ρ for φ* is the *minimal* set of ρ-positions satisfying the following constraint:

- for each $\langle \text{E} \rangle \psi \in E(\varphi, \rho)$, the greatest ρ-position $i > 1$ such that $\mathcal{K}, \rho^i \models \psi$ belongs to $Wt(\varphi, \rho)$.[a]

[a] Note that such a ρ-position exists by definition of $E(\varphi, \rho)$.

It is easy to see that the cardinalities of $E(\varphi, \rho)$ and of $Wt(\varphi, \rho)$ are at most $|\varphi| - 1$. We are now ready to prove the polynomial small-model property.

Theorem 4.1.5 (Polynomial small-model property for $\overline{\text{A}}\overline{\text{A}}\text{E}\overline{\text{E}}$). *Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure, $\rho, \sigma \in \text{Trc}_{\mathcal{K}}$, and φ be an $\overline{\text{A}}\overline{\text{A}}\text{E}\overline{\text{E}}$ formula in NNF such that $\mathcal{K}, \rho \star \sigma \models \varphi$. Then there exists π, induced by ρ, such that $\mathcal{K}, \pi \star \sigma \models \varphi$, and $|\pi| \leq |S| \cdot (|\varphi| + 1)^2$.*

As a preliminary remark, we note that the theorem holds in particular if $|\sigma| = 1$, and thus $\rho \star \sigma = \rho$ and $\pi \star \sigma = \pi$. In such a case, if $\mathcal{K}, \rho \models \varphi$, then $\mathcal{K}, \pi \models \varphi$, where π is induced by ρ and $|\pi| \leq |S| \cdot (|\varphi| + 1)^2$. The more general statement of Theorem 4.1.5 is needed for technical reasons in the soundness/completeness proofs of the algorithms for MC given in the following.

Proof. W.l.o.g., we restrict ourselves to the proposition letters occurring in φ, thus having $|\mathcal{AP}| \leq |\varphi|$. Let $Wt(\varphi, \rho \star \sigma)$ be the set of witness positions of $\rho \star \sigma$ for φ, let $\{i_1, \ldots, i_k\}$ be the ordering of $Wt(\varphi, \rho \star \sigma)$ such that $i_1 < \ldots < i_k$, and let $i_0 = 1$ and $i_{k+1} = |\rho \star \sigma|$. Hence, $1 = i_0 < i_1 < \ldots < i_k \leq i_{k+1} = |\rho \star \sigma|$.

If $|\rho| \leq |S| \cdot (|\varphi| + 1)^2$, the thesis trivially holds. Let us assume that $|\rho| > |S| \cdot (|\varphi| + 1)^2$. We show that there exists a trace π induced by ρ, with $|\pi| < |\rho|$, such that $\mathcal{K}, \pi \star \sigma \models \varphi$.

W.l.o.g., we can assume that, for some $j \geq 0$, $i_0 < i_1 < \ldots < i_j$ are ρ-positions, while $i_{j+1} < \ldots < i_{k+1}$ are $(\rho \star \sigma)$-positions not in ρ. Then, either (i) there exists $t \in [0, j-1]$ such that $i_{t+1} - i_t > |S| \cdot (|\varphi| + 1)$ or (ii) $|\rho(i_j, |\rho|)| > |S| \cdot (|\varphi| + 1)$. By way of contradiction, suppose that neither (i) nor (ii) holds. We need to distinguish two cases. If $\rho \star \sigma = \rho$, then $|\rho| = (i_{k+1} - i_0) + 1 \leq (k+1) \cdot |S| \cdot (|\varphi| + 1) + 1$; otherwise ($|\rho| < |\rho \star \sigma|$), $|\rho| = (i_j - i_0) + |\rho(i_j, |\rho|)| \leq j \cdot |S| \cdot (|\varphi| + 1) + |S| \cdot (|\varphi| + 1) \leq (k+1) \cdot |S| \cdot (|\varphi| + 1)$. The contradiction follows since $(k+1) \cdot |S| \cdot (|\varphi| + 1) + 1 \leq |\varphi| \cdot |S| \cdot (|\varphi| + 1) + 1 \leq |S| \cdot (|\varphi| + 1)^2$.

Let $(\alpha, \beta) = (i_t, i_{t+1})$ in case (i), and $(\alpha, \beta) = (i_j, |\rho|)$ in case (ii) and let $\rho' = \rho(\alpha, \beta)$. In both cases, $|\rho'| > |S| \cdot (|\varphi| + 1) \geq |S| \cdot (|\mathcal{AP}| + 1)$, being $|\mathcal{AP}| \leq |\varphi|$. By Proposition 4.1.3, there exists a trace π' of $\mathcal{K}$, well-formed with respect to ρ',

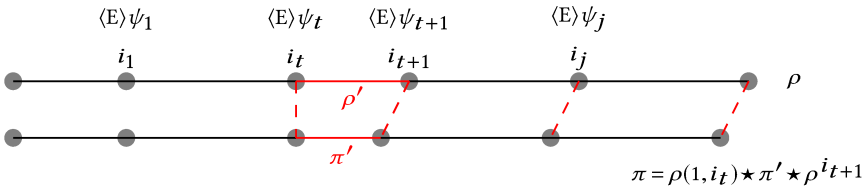


Figure 4.3: Representation of the contraction step of Theorem 4.1.5—case (*i*)

such that $|\pi'| \leq |S| \cdot (|\mathcal{AP}| + 1) < |\rho'|$. Let π be the trace induced by ρ obtained by replacing the subtrace ρ' of ρ by π' (see Figure 4.3 for a graphical account). Since $|\pi| < |\rho|$, it remains to prove that $\mathcal{K}, \pi \star \sigma \models \varphi$.

Let us denote $\pi \star \sigma$ by $\overline{\pi}$ and $\rho \star \sigma$ by $\overline{\rho}$. Let $H : [1, |\overline{\pi}|] \to [1, |\overline{\rho}|]$ be the function mapping positions of $\overline{\pi}$ into positions of $\overline{\rho}$ in such a way that positions "outside" π', i.e., outside the interval $[\alpha, \alpha + |\pi'| - 1]$, are mapped into their original positions in $\overline{\rho}$ while those "inside" π', i.e., in $[\alpha, \alpha + |\pi'| - 1]$, are mapped into the corresponding positions in ρ' (exploiting well-formedness of π' w.r.t. ρ'):

$$H(m) = \begin{cases} m & \text{if} \quad m < \alpha; \\ \alpha + \ell_{m-\alpha+1} - 1 & \text{if} \quad \alpha \leq m < \alpha + |\pi'|; \\ m + (|\rho'| - |\pi'|) & \text{if} \quad m \geq \alpha + |\pi'|, \end{cases}$$

where ℓ_s is the ρ'-position corresponding to the π'-position s, with $\ell_s \in [1, |\rho'|]$ and $s \in [1, |\pi'|]$.

It is easy to check that H satisfies the following properties:

1. H is strictly monotonic, i.e., for all $j, j' \in [1, |\overline{\pi}|]$, $j < j' \iff H(j) < H(j')$;
2. for all $j \in [1, |\overline{\pi}|]$, $\overline{\pi}(j) = \overline{\rho}(H(j))$;
3. $H(1) = 1$ and $H(|\overline{\pi}|) = |\overline{\rho}|$;
4. $Wt(\varphi, \overline{\rho}) \subseteq \{H(j) \mid j \in [1, |\overline{\pi}|]\}$, i.e., all witness positions are preserved;
5. for each $j \in [1, |\overline{\pi}|]$ and $p \in \mathcal{AP}$, $\mathcal{K}, \overline{\pi}^j \models p \iff \mathcal{K}, \overline{\rho}^{H(j)} \models p$.

The fact that $\mathcal{K}, \overline{\pi} \models \varphi$ is an immediate consequence of the following claim, considering that $H(1) = 1$, $\mathcal{K}, \overline{\rho} \models \varphi$, $\overline{\rho}^1 = \overline{\rho}$, and $\overline{\pi}^1 = \overline{\pi}$.

Claim 4.1.6. For all $j \in [1, |\overline{\pi}|]$, all subformulas ψ of φ, and all $u \in \mathrm{Trc}_{\mathcal{K}}$, it holds that

$$\mathcal{K}, u \star \overline{\rho}^{H(j)} \models \psi \implies \mathcal{K}, u \star \overline{\pi}^j \models \psi.$$

Proof. Assume that $\mathcal{K}, u \star \overline{\rho}^{H(j)} \models \psi$. Note that $u \star \overline{\rho}^{H(j)}$ is defined if and only if $u \star \overline{\pi}^j$ is defined. We prove by induction on the structure of ψ that $\mathcal{K}, u \star \overline{\pi}^j \models \psi$. Since φ is in NNF, only the following cases occur:

- $\psi = p$ or $\psi = \neg p$ for some $p \in \mathcal{AP}$. By Property 5 of H, $\mathcal{K}, \overline{\pi}^j \models p$ if and only if $\mathcal{K}, \overline{\rho}^{H(j)} \models p$. Hence, $\mathcal{K}, u \star \overline{\pi}^j \models p$ if and only if $\mathcal{K}, u \star \overline{\rho}^{H(j)} \models p$, and the result holds.
- $\psi = \theta_1 \wedge \theta_2$ or $\psi = \theta_1 \vee \theta_2$, for some $\mathrm{A}\overline{\mathrm{A}}\mathrm{E}\overline{\mathrm{E}}$ formulas θ_1 and θ_2: the result directly follows from the inductive hypothesis.
- $\psi = [\mathrm{E}]\,\theta$. We need to show that for each proper suffix η of $u \star \overline{\pi}^j$, $\mathcal{K}, \eta \models \theta$. We distinguish two cases:
  - η is *not* a proper suffix of $\overline{\pi}^j$. Hence, η is of the form $u^h \star \overline{\pi}^j$ for some $h \in [2, |u|]$. Since $\mathcal{K}, u \star \overline{\rho}^{H(j)} \models [\mathrm{E}]\,\theta$, then $\mathcal{K}, u^h \star \overline{\rho}^{H(j)} \models \theta$. By the inductive hypothesis, $\mathcal{K}, u^h \star \overline{\pi}^j \models \theta$.
  - η is a proper suffix of $\overline{\pi}^j$. Hence, $\eta = \overline{\pi}^h$ for some $h \in [j+1, |\overline{\pi}|]$. By Property 1 of H, $H(h) > H(j)$, and since $\mathcal{K}, u \star \overline{\rho}^{H(j)} \models [\mathrm{E}]\,\theta$, we have that $\mathcal{K}, \overline{\rho}^{H(h)} \models \theta$. By the inductive hypothesis, $\mathcal{K}, \overline{\pi}^h \models \theta$.

  Therefore, $\mathcal{K}, u \star \overline{\pi}^j \models [\mathrm{E}]\,\theta$.
- $\psi = \langle \mathrm{E} \rangle\,\theta$. We need to show that there exists a proper suffix of $u \star \overline{\pi}^j$ satisfying θ. Since $\mathcal{K}, u \star \overline{\rho}^{H(j)} \models \psi$, there exists a proper suffix η' of $u \star \overline{\rho}^{H(j)}$ such that $\mathcal{K}, \eta' \models \theta$. We distinguish two cases:
  - η' is *not* a proper suffix of $\overline{\rho}^{H(j)}$. Hence, η' is of the form $u^h \star \overline{\rho}^{H(j)}$ for some $h \in [2, |u|]$. By the inductive hypothesis, $\mathcal{K}, u^h \star \overline{\pi}^j \models \theta$. Hence, $\mathcal{K}, u \star \overline{\pi}^j \models \langle \mathrm{E} \rangle\,\theta$.
  - η' is a proper suffix of $\overline{\rho}^{H(j)}$. Hence, $\eta' = \overline{\rho}^i$ for some $i \in [H(j)+1, |\overline{\rho}|]$, and $\mathcal{K}, \overline{\rho}^i \models \theta$. Let i' be the greatest position of $\overline{\rho}$ such that $\mathcal{K}, \overline{\rho}^{i'} \models \theta$. Hence $i' \geq i$ and, by Definition 4.1.4, $i' \in Wt(\varphi, \overline{\rho})$. By Property 4 of H, $i' = H(h)$ for some $\overline{\pi}$-position h. Since $H(h) > H(j)$, it holds that $h > j$ (Property 1). By the inductive hypothesis, $\mathcal{K}, \overline{\pi}^h \models \theta$, and we obtain that $\mathcal{K}, u \star \overline{\pi}^j \models \langle \mathrm{E} \rangle\,\theta$.

  Therefore, in both the cases, $\mathcal{K}, u \star \overline{\pi}^j \models \langle \mathrm{E} \rangle\,\theta$.
- $\psi = [\overline{\mathrm{E}}]\,\theta$ or $\psi = \langle \overline{\mathrm{E}} \rangle\,\theta$: the thesis holds as a direct consequence of the inductive hypothesis.
- $\psi = [\mathrm{A}]\,\theta$, $\psi = \langle \mathrm{A} \rangle\,\theta$, $\psi = [\overline{\mathrm{A}}]\,\theta$, or $\psi = \langle \overline{\mathrm{A}} \rangle\,\theta$. Since $u \star \overline{\pi}^j$ and $u \star \overline{\rho}^{H(j)}$ start at the same state and lead to the same state (by Properties 2 and 3 of H), the thesis trivially follows. This concludes the proof of the claim. □

We have shown that $\mathcal{K}, \overline{\pi} \models \varphi$, with $|\pi| < |\rho|$. Now, if $|\pi| \leq |S| \cdot (|\varphi| + 1)^2$, the thesis is proved; otherwise, the above contraction step can be iterated a finite number of times, until the bound is reached, proving the thesis of Theorem 4.1.5. □

4.1.2 PSPACE MC algorithm for $\mathsf{A\overline{A}E\overline{E}}$

By exploiting the polynomial small-model property stated by Theorem 4.1.5, it is easy to define a **PSPACE** MC algorithm for $\mathsf{A\overline{A}E\overline{E}}$. The main MC procedure for $\mathsf{A\overline{A}E\overline{E}}$ formulas is $\texttt{ModCheck}(\mathcal{K}, \psi)$ (Algorithm 4.1). All the initial traces σ, obtained by visiting the unravelling of $\mathcal{K}$ from its initial state s_0 up to depth $|S| \cdot (2|\psi| + 3)^2$, are checked with respect to ψ by the function $\texttt{Check}(\mathcal{K}, \psi, \sigma)$ (Algorithm 4.2) which decides whether $\mathcal{K}, \sigma \models \psi$. The `Check` function is iteratively called until either some initial trace is found that does not satisfy ψ or all bounded initial traces satisfy ψ (and thus $\mathcal{K} \models \psi$). The invocation of $\texttt{Check}(\mathcal{K}, \psi, \sigma)$ (Algorithm 4.2) decides whether $\mathcal{K}, \sigma \models \psi$ or not by recursively calling itself on the subformulas of ψ either over σ or over (bounded) traces obtained by unraveling $\mathcal{K}$ forward (starting from $\mathrm{lst}(\sigma)$) for occurrences of the modality $\langle \mathrm{A} \rangle$, and backward (starting from $\mathrm{fst}(\sigma)$) for occurrences of $\langle \overline{\mathrm{A}} \rangle$ and $\langle \overline{\mathrm{E}} \rangle$.

Note that the considered bound on the length of initial traces is actually $|S| \cdot (2|\psi| + 3)^2 \geq |S| \cdot (|\mathsf{NNF}(\neg\psi)| + 1)^2$ (line 1 of the `ModCheck` procedure). The reason is that the correctness proof of the algorithm exploits the polynomial bound of Theorem 4.1.5 over the formula $\neg\psi$, that has to be converted into NNF.

The next results state soundness and completeness of Algorithm 4.2 and Algorithm 4.1, respectively. Their proofs can be found in Appendix B.1 and B.2.

Lemma 4.1.7. *Let ψ be an $\mathsf{A\overline{A}E\overline{E}}$ formula, $\mathcal{K}$ be a finite Kripke structure, and $\sigma \in \mathrm{Trc}_{\mathcal{K}}$. Then, $\texttt{Check}(\mathcal{K}, \psi, \sigma) = 1$ if and only if $\mathcal{K}, \sigma \models \psi$.*

Theorem 4.1.8. *Let ψ be an $\mathsf{A\overline{A}E\overline{E}}$ formula and $\mathcal{K}$ be a finite Kripke structure. Then, $\texttt{ModCheck}(\mathcal{K}, \psi) = 1$ if and only if $\mathcal{K} \models \psi$.*

The MC procedures require *polynomial working space*, since:

- `ModCheck` needs to store only a trace no longer than $|S| \cdot (2|\psi|+3)^2$ (obviously, many traces are generated while visiting the unravelling of $\mathcal{K}$, but only one at a time needs to be stored);
- every recursive call to `Check` (possibly) needs space for a trace no longer than $|S| \cdot (2|\varphi| + 1)^2$, where φ is a subformula of ψ such that $|\varphi| \leq |\psi| - 1$;
- at most one call to `ModCheck` and $|\psi|$ calls to `Check` can be simultaneously active.

Thus the maximum space needed by the given algorithms is $(|\psi| + 1) \cdot O(\log |S|) \cdot (|S| \cdot (2|\psi| + 3)^2)$ bits, where $O(\log |S|)$ bits are needed to represent a state of $\mathcal{K}$.

Theorem 4.1.8, along with the above space analysis and the fact that MC for the fragment $\overline{\mathsf{E}}$ is **PSPACE**-hard (see Appendix B.3), entail the following corollary.

Algorithm 4.1 `ModCheck`$(\mathcal{K}, \psi)$

for all initial traces $\sigma \in \mathrm{Trc}_{\mathcal{K}}$ such that $|\sigma| \leq |S| \cdot (2|\psi| + 3)^2$ **do**
  if `Check`$(\mathcal{K}, \psi, \sigma) = 0$ **then**
    return 0: "$\mathcal{K}, \sigma \not\models \psi$" ◃ Counterexample found
return 1: "$\mathcal{K} \models \psi$"

Algorithm 4.2 `Check`$(\mathcal{K}, \psi, \sigma)$

if $\psi = p$, for $p \in \mathcal{AP}$ **then**
  if $p \in \bigcap_{s \in \mathrm{states}(\sigma)} \mu(s)$ **then**
    return 1 **else return** 0
else if $\psi = \neg\varphi$ **then**
  return $1 -$ `Check`$(\mathcal{K}, \varphi, \sigma)$
else if $\psi = \varphi_1 \wedge \varphi_2$ **then**
  if `Check`$(\mathcal{K}, \varphi_1, \sigma) = 0$ **then**
    return 0
  else
    return `Check`$(\mathcal{K}, \varphi_2, \sigma)$
else if $\psi = \langle \mathrm{A} \rangle \varphi$ **then**
  for all $\pi \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{fst}(\pi) = \mathrm{lst}(\sigma)$, and $|\pi| \leq |S| \cdot (2|\varphi| + 1)^2$ **do**
    if `Check`$(\mathcal{K}, \varphi, \pi) = 1$ **then**
      return 1
  return 0
else if $\psi = \langle \mathrm{E} \rangle \varphi$ **then**
  for each proper suffix π of σ **do**
    if `Check`$(\mathcal{K}, \varphi, \pi) = 1$ **then**
      return 1
  return 0
else if $\psi = \langle \overline{\mathrm{E}} \rangle \varphi$ **then**
  for all $\pi \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{lst}(\pi) = \mathrm{fst}(\sigma)$, and $2 \leq |\pi| \leq |S| \cdot (2|\varphi| + 1)^2$ **do**
    if `Check`$(\mathcal{K}, \varphi, \pi \star \sigma) = 1$ **then**
      return 1
  return 0
... ◃ $\psi = \langle \overline{\mathrm{A}} \rangle \varphi$ is analogous to $\psi = \langle \mathrm{A} \rangle \varphi$

Corollary 4.1.9. *The MC problem for* $\mathsf{A\overline{A}E\overline{E}}$ *formulas over finite Kripke structures is* ***PSPACE****-complete.*

The same result, that is, **PSPACE**-completeness, clearly holds also for any sub-fragment of $\mathsf{A\overline{A}E\overline{E}}$ which features modality $\overline{\mathsf{E}}$.

4.1.3 co-NP MC algorithm for B and E

We now conclude the section by showing that the MC problem for the fragments B and E is in **co-NP**, that is, they have the same complexity as the purely propositional fragment Prop (proved to be hard for **co-NP** in [MMP15a]). We focus on E, as the case of B is completely symmetric. As we will see, the MC algorithm heavily rests again on the polynomial small-model property.

The algorithm is based on the *non-deterministic* procedure `CounterExE`$(\mathcal{K}, \psi)$ (Algorithm 4.3) which searches for counterexamples to the input E formula ψ (that is, initial traces satisfying $\neg\psi$). If such a counterexample is found, clearly $\mathcal{K} \not\models \psi$. First, the procedure generates in a *non-deterministic way* an initial trace ρ, whose length is at most $|S| \cdot (2|\psi| + 3)^2$, by means of `A_trace`$(\mathcal{K}, s_0, |\psi|)$. Then, the *deterministic* function `CheckE`$(\mathcal{K}, \psi, \rho)$, reported in Algorithm 4.4, evaluates ψ over ρ. If `CheckE` returns $\bot$, a counterexample has been found and `CounterExE` returns **Yes** (thus the non-deterministic computation of the algorithm is successful). Otherwise, it returns **No** (the computation fails).

As for the function `CheckE`, the following result holds.

Proposition 4.1.10. *Let* ψ *be an* E *formula,* $\mathcal{K}$ *be a finite Kripke structure, and* ρ *be a trace of* $\mathcal{K}$*. Then,* `CheckE`$(\mathcal{K}, \psi, \rho) = \top$ *if and only if* $\mathcal{K}, \rho \models \psi$*.*

`CheckE` exploits a Boolean table T with an entry for each pair consisting of a subformula of ψ and the starting position of a suffix of ρ (the size of T is then $|\psi| \cdot |\rho|$). The function scans all the subformulas φ of the input ψ by increasing length, and it stores in the Boolean entry $T[\varphi, i]$, for $1 \leq i \leq |\rho|$, whether $\mathcal{K}, \rho^i \models \varphi$ or not. Note that the result of the evaluation of ψ over ρ is stored in $T[\psi, 1]$, as

Algorithm 4.3 `CounterExE`$(\mathcal{K}, \psi)$

```
1: ρ ← A_trace(K, s_0, |ψ|)        ◃ a trace of K from s_0 of length ≤ |S| · (2|ψ| + 3)^2
2: if CheckE(K, ψ, ρ) = ⊥ then
3:     return Yes: "K, ρ ⊭ ψ"       ◃ Counterexample found
4: else
5:     return No: "K, ρ ⊨ ψ"        ◃ Counterexample not found
```

Algorithm 4.4 `CheckE`$(\mathcal{K}, \psi, \rho)$

1: $T \leftarrow$ `New_Bool_Table`$(|\psi|, |\rho|)$ ◃ creates new table of $|\psi| \cdot |\rho|$ Boolean entries
2: **for** all subformulas φ of ψ by increasing length **do**
3: **if** $\varphi = p$, for $p \in \mathcal{AP}$ **then**
4: $T[p, |\rho|] \leftarrow p \in \mu(\mathrm{lst}(\rho))$
5: **for** $i = |\rho| - 1, \ldots, 1$ **do**
6: $T[p, i] \leftarrow T[p, i+1]$ and $p \in \mu(\rho(i))$
7: **else if** $\varphi = \neg\varphi_1$ **then**
8: **for** $i = |\rho|, \ldots, 1$ **do**
9: $T[\varphi, i] \leftarrow$ not $T[\varphi_1, i]$
10: **else if** $\varphi = \varphi_1 \wedge \varphi_2$ **then**
11: **for** $i = |\rho|, \ldots, 1$ **do**
12: $T[\varphi, i] \leftarrow T[\varphi_1, i]$ and $T[\varphi_2, i]$
13: **else if** $\varphi = \langle \mathrm{E} \rangle \varphi_1$ **then**
14: $T[\varphi, |\rho|] \leftarrow \bot$
15: **for** $i = |\rho| - 1, \ldots, 1$ **do**
16: $T[\varphi, i] \leftarrow T[\varphi, i+1]$ or $T[\varphi_1, i+1]$
17: **return** $T[\psi, 1]$

$\rho^1 = \rho$. Since subformulas of ψ are considered by increasing length order, during an iteration starting at line 2, when a subformula φ of ψ is being processed, it holds that $T[\xi, i]$ is defined for all other subformulas ξ processed in some previous iteration, and $T[\xi, i] = \top$ iff $\mathcal{K}, \rho^i \models \xi$. Hence, at the end, $T[\psi, 1] = \top$ iff $\mathcal{K}, \rho \models \psi$.

We can now prove that the procedure `CounterExE` is sound and complete. If `CounterExE`$(\mathcal{K}, \psi)$ has a successful computation, then there exists an initial trace ρ such that `CheckE`$(\mathcal{K}, \psi, \rho) = \bot$. This means that $\mathcal{K}, \rho \not\models \psi$, and thus $\mathcal{K} \not\models \psi$. Conversely, if $\mathcal{K} \not\models \psi$ then there exists an initial trace ρ such that $\mathcal{K}, \rho \not\models \psi$. By Theorem 4.1.5, there exists an initial trace π, whose length is bounded by $|S| \cdot (|\psi'| + 1)^2 \leq |S| \cdot (2|\psi| + 3)^2$, such that $\mathcal{K}, \pi \models \psi'$, where ψ' is the NNF of $\neg\psi$. Now, some non-deterministic instance of `A_trace`$(\mathcal{K}, s_0, |\psi|)$ generates exactly such π, being $|\pi| \leq |S| \cdot (2|\psi| + 3)^2$. Moreover, `CheckE`$(\mathcal{K}, \psi, \pi) = \bot$, and thus `CounterExE`$(\mathcal{K}, \psi)$ has a successful computation.

`CounterExE`$(\mathcal{K}, \psi)$ is in **NP**, as the generated trace(s) ρ has (have) a length polynomial in $|S|$ and $|\psi|$, and can thus be computed in polynomial time. Moreover, `CheckE` performs a polynomial number of steps, since all it has to do is filling in the table T, which features $|\psi| \cdot |\rho|$ entries.

Corollary 4.1.11. *The MC problem for* E *formulas over finite Kripke structures is* ***co-NP****-complete.*

Proof. Since CounterExE($\mathcal{K}, \psi$) has a successful computation if and only if $\mathcal{K} \not\models \psi$ and such a procedure runs in (non-deterministic) polynomial time, the MC problem belongs to **co-NP**. The **co-NP**-hardness derives immediately from that of the purely propositional HS fragment Prop, as proved in [MMP15a]. □

In the next section we will see what happens if we add $\overline{\mathsf{B}}$ to $\mathsf{A\overline{A}E\overline{E}}$, and $\overline{\mathsf{E}}$ to $\mathsf{A\overline{A}B\overline{B}}$: we shall prove a different (this time, exponential) small-model property for $\mathsf{A\overline{A}E\overline{B}\overline{E}}$ and $\mathsf{A\overline{A}B\overline{B}\overline{E}}$, which will allow us to devise an **EXPSPACE** MC algorithm for them.

4.2 The fragments $\mathsf{A\overline{A}B\overline{B}\overline{E}}$ and $\mathsf{A\overline{A}E\overline{B}\overline{E}}$: exponential small-model property

Before going into the details of the section, we would like to summarize the main ideas of [MMP17], in which the authors gave the first exponential small-model for $\mathsf{A\overline{A}B\overline{B}\overline{E}}$ and $\mathsf{A\overline{A}E\overline{B}\overline{E}}$, different from the one we shall see in the following.

We recall from Chapter 2 that being associated with the same B-descriptor is a sufficient condition for two traces to be indistinguishable with respect to the fulfillment of $\mathsf{A\overline{A}B\overline{B}\overline{E}}$ formulas, provided that the B-nesting depth of the considered formula is less than or equal to the depth of the descriptor itself. Thus we may say that a B_k-descriptor provides a finite encoding for a possibly infinite set of traces (the traces associated with that descriptor). Unfortunately, the representation of B_k-descriptors as trees labelled over descriptor elements is highly redundant. For instance, given any pair of subtrees rooted in some children of the root of a descriptor, it is always the case that one of them is a subtree of the other: the two subtrees are associated with two (different) prefixes of a trace and one of them is necessarily a prefix of the other. In practice, redundancy of the tree representation of B_k-descriptors prevents their direct use in MC algorithms, and makes it difficult to determine their intrinsic complexity.

In [MMP17], the authors devise a more compact representation of B_k-descriptors. Their idea is selecting, for each B_k-descriptor witnessed in $\mathcal{K}$, a trace called *trace representative*, which is associated with such a descriptor, and whose length is (exponentially) bounded in both the size of S (the set of states of $\mathcal{K}$) and k. Clearly, they have to prove that such a finite bound exists. To the aim they consider suitable ordered sequences (possibly with repetitions) of descriptor elements of a B_k-descriptor. Let the *descriptor sequence* for a trace be the ordered sequence of descriptor elements associated with its prefixes. In general, in a descriptor sequence, descriptor elements can be repeated. The authors introduce an equivalence relation that allows them to put together (contracting them) indistinguishable occurrences of the same descriptor element in a descriptor sequence, that is, to detect those occurrences which are associated with prefixes of the trace with the

same B_k-descriptor. A trace representative for a B_k-descriptor should not feature indistinguishable occurrences of any descriptor elements: it can be obtained by iteratively applying such a *contraction method*, which always leads to an exponential (in $|S|$ and k) bounded-length trace indistinguishable from the original one. All this is done avoiding the expensive operation of explicitly constructing B_k-descriptors.

Thanks to trace representatives, they finally devise an **EXPSPACE** "representative-based" MC algorithm for $\mathsf{A\overline{A}B\overline{BE}}$, which basically can restrict the verification of $\mathsf{A\overline{A}B\overline{BE}}$ formulas only over trace representatives, while retaining completeness.

In [MMP17], the bound on the length of representatives is calculated in a very technical (and tricky) way. We now give a much more understandable proof of **EXPSPACE** membership of MC for $\mathsf{A\overline{A}B\overline{BE}}$, which makes use of the presented notion of induced trace (Definition 4.1.1).

More precisely, we prove that $\mathsf{A\overline{A}B\overline{BE}}$ (and $\mathsf{A\overline{A}E\overline{BE}}$) features an *exponential small-model property* guaranteeing that, for each $h \geq 0$ and trace ρ of a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$, it is always possible to find another trace of $\mathcal{K}$ *induced by* ρ, whose length is at most $(|S| + 2)^{h+2}$, which is indistinguishable from ρ with respect to the satisfiability of any $\mathsf{A\overline{A}B\overline{BE}}$ (resp., $\mathsf{A\overline{A}E\overline{BE}}$) formula φ with B-nesting depth $\mathrm{Nest}_{\mathrm{B}}(\varphi)$ (resp., E-nesting depth $\mathrm{Nest}_{\mathrm{E}}(\varphi)$) at most h.

To prove such a property, we first introduce the notion of *h-prefix bisimilarity* (resp., *h-suffix bisimilarity*), which defines an equivalence relation over $\mathrm{Trc}_{\mathcal{K}}$ ensuring that equivalent traces satisfy the same $\mathsf{A\overline{A}B\overline{BE}}$ (resp., $\mathsf{A\overline{A}E\overline{BE}}$) formulas with B-nesting (resp., E-nesting) depth at most h.

Then, we show how to determine, for a given trace ρ, a subset of positions of ρ that allow us to build another trace ρ', with length at most $(|S| + 2)^{h+2}$, such that ρ and ρ' are h-prefix bisimilar (resp., h-suffix bisimilar). We call such a set of ρ-positions *prefix* (resp., *suffix*) *sampling* of ρ. Intuitively, they play a role which is analogous to that of witness positions (Definition 4.1.4) of Section 4.1.

Let $h \geq 0$; *h-prefix bisimilarity* and *h-suffix bisimilarity* between a pair of traces ρ and ρ' of a Kripke structure are defined as follows.

Definition 4.2.1 (Prefix-bisimilarity and Suffix-bisimilarity)**.** Let $h \geq 0$. Two traces ρ and ρ' of a finite Kripke structure $\mathcal{K}$ are *h-prefix bisimilar* if the following conditions inductively hold:

- for $h = 0$: $\mathrm{fst}(\rho) = \mathrm{fst}(\rho')$, $\mathrm{lst}(\rho) = \mathrm{lst}(\rho')$, and $\mathrm{states}(\rho) = \mathrm{states}(\rho')$;
- for $h > 0$: ρ and ρ' are 0-prefix bisimilar and for each proper prefix ν of ρ (resp., proper prefix ν' of ρ'), there exists a proper prefix ν' of ρ' (resp., proper prefix ν of ρ) such that ν and ν' are $(h-1)$-prefix bisimilar.

The notion of *h-suffix bisimilarity* is defined in a symmetric way by considering suffixes of traces, instead of prefixes.

As it will be established by Proposition 4.2.5 below, h-prefix (resp., h-suffix) bisimilarity is a sufficient condition for two traces ρ and ρ' to be indistinguishable with respect to the satisfiability of $\mathsf{A\overline{A}B\overline{B}E}$ (resp., $\mathsf{A\overline{A}E\overline{B}E}$) formulas with B-nesting (resp., E-nesting) depth at most h.

The following property can easily be shown.

Property 4.2.2. Given a finite Kripke structure $\mathcal{K}$, for all $h \geq 0$, h-prefix (resp., h-suffix) bisimilarity is an equivalence relation over $\mathrm{Trc}_{\mathcal{K}}$.

The following property states that h-suffix bisimilarity and h-prefix bisimilarity "propagate downwards".

Property 4.2.3. Let $h > 0$, and ρ and ρ' be two h-prefix (resp., h-suffix) bisimilar traces of a finite Kripke structure $\mathcal{K}$. Then ρ and ρ' are also $(h-1)$-prefix (resp., $(h-1)$-suffix) bisimilar.

h-prefix and h-suffix bisimilarity are preserved by left and right $\star$-concatenation with the same trace. The property can be easily proved by induction on $h \geq 0$.

Proposition 4.2.4. *Let $h \geq 0$, and let ρ and ρ' be two h-prefix (resp., h-suffix) bisimilar traces of a finite Kripke structure $\mathcal{K}$. Then, for each trace ρ'' of $\mathcal{K}$, it holds that:*

1. *$\rho'' \star \rho$ and $\rho'' \star \rho'$ are h-prefix (resp., h-suffix) bisimilar;*
2. *$\rho \star \rho''$ and $\rho' \star \rho''$ are h-prefix (resp., h-suffix) bisimilar.*

By Proposition 4.2.4 and a straightforward induction on the structural complexity of formulas, we obtain that h-prefix (resp., h-suffix) bisimilarity preserves the satisfiability of $\mathsf{A\overline{A}B\overline{B}E}$ (resp., $\mathsf{A\overline{A}E\overline{B}E}$) formulas with B-nesting (resp., E-nesting) depth at most h. In other words, h-prefix (resp., h-suffix) bisimilarity is a sufficient condition for two traces to be indistinguishable with respect to any $\mathsf{A\overline{A}B\overline{B}E}$ (resp., $\mathsf{A\overline{A}E\overline{B}E}$) formula ψ with $\mathsf{Nest_B}(\psi) \leq h$ (resp., $\mathsf{Nest_E}(\psi) \leq h$).

Proposition 4.2.5. *Let $h \geq 0$, and let ρ and ρ' be two h-prefix (resp., h-suffix) bisimilar traces of a finite Kripke structure $\mathcal{K}$. For each $\mathsf{A\overline{A}B\overline{B}E}$ (resp., $\mathsf{A\overline{A}E\overline{B}E}$) formula ψ with $\mathsf{Nest_B}(\psi) \leq h$ (resp., $\mathsf{Nest_E}(\psi) \leq h$), it holds*

$$\mathcal{K}, \rho \models \psi \iff \mathcal{K}, \rho' \models \psi.$$

In the remaining part of the section, we will focus on the fragment $\mathsf{A\overline{A}B\overline{B}E}$ (the case of $\mathsf{A\overline{A}E\overline{B}E}$ is completely symmetric). We show how to determine a subset of positions of a trace ρ (a *prefix sampling* of ρ), starting from which it

is possible to build another trace ρ', of bounded exponential length, which is indistinguishable from ρ with respect to the satisfiability of $A\overline{A}B\overline{BE}$ formulas up to a given B-nesting depth.

We start by introducing the notions of *prefix-skeleton sampling* and *h-prefix sampling*, and prove some related properties. In the following, we fix a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$, and, given a set I of natural numbers, by "two consecutive elements of I" we mean a pair of elements $i, j \in I$ such that $i < j$ and $I \cap [i, j] = \{i, j\}$.

Definition 4.2.6 (Prefix-skeleton sampling)**.** Let ρ be a trace of $\mathcal{K}$. Given two ρ-positions i and j, with $i \leq j$, the *prefix-skeleton sampling of ρ in the interval* $[i, j]$ is the *minimal* set P of ρ-positions in the interval $[i, j]$ satisfying the next conditions:

- $i, j \in P$;
- for each state $s \in S$ occurring in $\rho(i+1, j-1)$, the minimum position $k \in [i+1, j-1]$ such that $\rho(k) = s$ belongs to P.

Example 4.2.7. Figure 4.4 gives a graphical account of the prefix-skeleton sampling of a trace ρ in $[i, j]$, where $\rho(i, j) = s_1^4 s_2 s_1 s_3 s_1 s_2 s_3 s_1 s_3$.

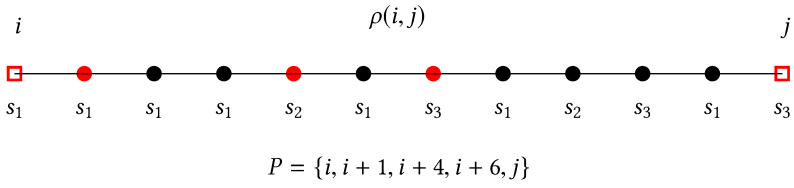


Figure 4.4: The set P is the prefix-skeleton sampling of $\rho(i, j) = s_1^4 s_2 s_1 s_3 s_1 s_2 s_3 s_1 s_3$.

From Definition 4.2.6, it immediately follows that the prefix-skeleton sampling P of (any) trace ρ in $[i, j]$ is such that $|P| \leq |S| + 2$, and if $i < j$, then $i + 1 \in P$.

Definition 4.2.8 (h-prefix sampling)**.** Let ρ be a trace of $\mathcal{K}$. For each $h \geq 1$, the *h-prefix sampling of ρ* is the *minimal* set P_h of ρ-positions inductively satisfying the following conditions:

- for $h = 1$: P_1 is the prefix-skeleton sampling of ρ in $[1, |\rho|]$;
- for $h > 1$: (i) $P_h \supseteq P_{h-1}$ and (ii) for all pairs of consecutive positions $i, j \in P_{h-1}$, the prefix-skeleton sampling of ρ in $[i, j]$ belongs to P_h.

The following upper bound to the cardinality of prefix samplings easily follows from Definition 4.2.8.

Property 4.2.9. Let $h \geq 1$ and ρ be a trace of $\mathcal{K}$. The h-prefix sampling P_h of ρ is such that $|P_h| \leq (|S| + 2)^h$.

Lemma 4.2.11 and Theorem 4.2.12 below show how to derive, from any trace ρ of the given finite Kripke structure $\mathcal{K}$, another trace ρ', induced by ρ and h-prefix bisimilar to ρ, such that $|\rho'| \leq (|S| + 2)^{h+2}$. By Proposition 4.2.5, ρ' is indistinguishable from ρ with respect to the satisfiability of any $\overline{\text{A}}\overline{\text{A}}\text{B}\overline{\text{B}}\overline{\text{E}}$ formula ψ with $\text{Nest}_{\text{B}}(\psi) \leq h$.

In order to build ρ', we first compute the $(h+1)$-prefix sampling P_{h+1} of ρ. Then, for all the pairs of consecutive ρ-positions $i, j \in P_{h+1}$, we consider a trace induced by $\rho(i, j)$, with no repeated occurrences of any state, except at most the first and last ones (hence, it is not longer than $|S| + 2$). The trace ρ' is just the ordered concatenation (by means of the $\star$-concatenation operator) of all these traces. The aforementioned bound on $|\rho'|$ holds as, by Property 4.2.9, $|P_{h+1}| \leq (|S| + 2)^{h+1}$. Lemma 4.2.11 states that ρ and ρ' are indeed h-prefix bisimilar. The proof of such lemma exploits the following technical result (proved in Appendix B.4).

Lemma 4.2.10. *Let $h \geq 1$, ρ be a trace of $\mathcal{K}$, and let i, j be two consecutive ρ-positions in the h-prefix sampling of ρ. For all ρ-positions n, $n' \in [i + 1, j]$ such that $\rho(n) = \rho(n')$, it holds that $\rho(1, n)$ and $\rho(1, n')$ are $(h - 1)$-prefix bisimilar.*

Lemma 4.2.11. *Let $h \geq 1$, ρ be a trace of $\mathcal{K}$, and $\rho' = \rho(i_1)\rho(i_2)\cdots\rho(i_k)$ be a trace induced by ρ, where $1 = i_1 < i_2 < \ldots < i_k = |\rho|$ and $P_{h+1} \subseteq \{i_1, \ldots, i_k\}$, with P_{h+1} the $(h + 1)$-prefix sampling of ρ. Then for all $j \in [1, k]$, $\rho'(1, j)$ and $\rho(1, i_j)$ are h-prefix bisimilar.*

Note that, as a straightforward consequence, ρ and ρ' are h-prefix bisimilar.

Proof. Let $Q = \{i_1, \ldots, i_k\}$ (hence $P_{h+1} \subseteq Q$) and let $j \in [1, k]$. We prove by induction on j that $\rho'(1, j)$ and $\rho(1, i_j)$ are h-prefix bisimilar. As for the base case ($j = 1$), the result holds, since $i_1 = 1$.

Let us assume that $j > 1$. We first show that $\rho(1, i_j)$ and $\rho'(1, j)$ are 0-prefix bisimilar. Clearly, $\rho(1) = \rho(i_1) = \rho'(1)$, $\rho(i_j) = \rho'(j)$, and $\text{states}(\rho'(1, j)) \subseteq \text{states}(\rho(1, i_j))$. Now, if, by contradiction, there was a state s such that $s \in \text{states}(\rho(1, i_j)) \setminus \text{states}(\rho'(1, j))$, then for all $l \in Q$, with $l \leq i_j$, $\rho(l) \neq s$. However, the prefix-skeleton sampling P_1 of ρ in $[1, |\rho|]$ is contained in Q, and the minimal ρ-position l' such that $\rho(l') = s$ belongs to P_1. Since $s \in \text{states}(\rho(1, i_j))$, we have $l' \leq i_j$ and we get a contradiction, implying that $\text{states}(\rho'(1, j)) = \text{states}(\rho(1, i_j))$.

It remains to prove that:

1. for each proper prefix ν' of $\rho'(1,j)$, there exists a proper prefix ν of $\rho(1,i_j)$ such that ν and ν' are $(h-1)$-prefix bisimilar, and
2. for each proper prefix ν of $\rho(1,i_j)$, there exists a proper prefix ν' of $\rho'(1,j)$ such that ν and ν' are $(h-1)$-prefix bisimilar.

As for (1.), let ν' be a proper prefix of $\rho'(1,j)$. Hence, there exists $m \in [1, j-1]$ such that $\nu' = \rho'(1,m)$. By the inductive hypothesis, $\rho'(1,m)$ and $\rho(1,i_m)$ are h-prefix bisimilar, and thus $(h-1)$-prefix bisimilar as well by Property 4.2.3. Since $\rho(1,i_m)$ is a proper prefix of $\rho(1,i_j)$, by choosing $\nu = \rho(1,i_m)$, (1.) follows.

As for (2.), assume that ν is a proper prefix of $\rho(1,i_j)$. Therefore, there exists $n \in [1, i_j - 1]$ such that $\nu = \rho(1,n)$. We distinguish two cases:

- $n \in P_{h+1}$. Since $n < i_j$, there exists $m \in [1, j-1]$ such that $n = i_m$. By the inductive hypothesis, $\rho(1,n)$ and $\rho'(1,m)$ are h-prefix bisimilar, and thus $(h-1)$-prefix bisimilar as well by Property 4.2.3. Since $\rho'(1,m)$ is a proper prefix of $\rho'(1,j)$, by choosing $\nu' = \rho'(1,m)$, (2.) follows.
- $n \notin P_{h+1}$. It follows that there exist two consecutive positions i' and j' in P_{h+1}, with $i' < j'$, such that $n \in [i'+1, j'-1]$. By definition of $(h+1)$-prefix sampling, there exist two consecutive positions i'' and j'' in the h-prefix sampling of ρ, with $i'' < j''$, such that i' and j' are two consecutive positions in the prefix-skeleton sampling of ρ in the interval $[i'', j'']$.
  First, we observe that $i' \neq i''$ (otherwise, $j' = i' + 1$, which contradicts the fact that $[i'+1, j'-1] \neq \emptyset$, as $n \in [i'+1, j'-1]$). Thus, by definition of prefix-skeleton sampling applied to ρ in $[i'', j'']$, and since $n \in [i'+1, j'-1]$, there must be $\ell \in [i''+1, i']$ such that $\rho(\ell) = \rho(n)$ and ℓ is in the prefix-skeleton sampling of ρ in $[i'', j'']$. Hence, $\ell \in P_{h+1}$ by definition of $(h+1)$-prefix sampling. As a consequence, since $\ell < n < i_j$, there exists $m \in [1, j-1]$ such that $\ell = i_m$. By applying Lemma 4.2.10, we deduce that $\rho(1,n)$ and $\rho(1,i_m)$ are $(h-1)$-prefix bisimilar. Moreover, by the inductive hypothesis, $\rho(1,i_m)$ and $\rho'(1,m)$ are $(h-1)$-prefix bisimilar. Thus, by choosing $\nu' = \rho'(1,m)$, ν' is a proper prefix of $\rho'(1,j)$ which is $(h-1)$-prefix bisimilar to $\nu = \rho(1,n)$.

This concludes the proof of Lemma 4.2.11. □

We are now ready to prove the exponential small-model property.

Theorem 4.2.12 (Exponential small-model property for $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}\overline{\mathsf{E}}$)**.** *Let ρ be a trace of a finite Kripke structure $\mathcal{K}$ and let $h \geq 0$. Then there exists a trace ρ' induced by ρ, whose length is at most $(|S|+2)^{h+2}$, such that for every $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}\overline{\mathsf{E}}$ formula ψ with $\mathsf{Nest}_{\mathrm{B}}(\psi) \leq h$, it holds*

$$\mathcal{K}, \rho \models \psi \iff \mathcal{K}, \rho' \models \psi.$$

Proof. Let P_{h+1} be the $(h+1)$-prefix sampling of ρ. For all pairs of consecutive ρ-positions i and j in P_{h+1}, there exists a trace induced by $\rho(i,j)$, whose length is at most $|S|+2$, featuring no repeated occurrences of any internal state. We now define ρ' as the trace of $\mathcal{K}$ obtained by an ordered concatenation of all these induced traces by means of the $\star$-concatenation operator. It is immediate to see that $\rho' = \rho(i_1)\rho(i_2)\cdots\rho(i_k)$, for some indexes $1 = i_1 < i_2 < \ldots < i_k = |\rho|$, where $\{i_1, \ldots, i_k\}$ contains the $(h+1)$-prefix sampling P_{h+1} of ρ. It holds that $|\rho'| \leq |P_{h+1}| \cdot (|S|+2)$ and since, by Property 4.2.9, $|P_{h+1}| \leq (|S|+2)^{h+1}$, we obtain that $|\rho'| \leq (|S|+2)^{h+2}$. Moreover, by Lemma 4.2.11, ρ and ρ' are h-prefix bisimilar. By Proposition 4.2.5 the thesis follows. □

Theorem 4.2.12 allows us to easily devise an **EXPSPACE** MC algorithm for $\overline{\mathsf{A}}\mathsf{A}\mathsf{B}\overline{\mathsf{B}}\mathsf{E}$ formulas (and symmetrically for $\overline{\mathsf{A}}\mathsf{A}\mathsf{E}\overline{\mathsf{B}}\mathsf{E}$ formulas), which can be obtained from Algorithms 4.1 and 4.2 by adapting the bounds on the length of considered traces.

We observe that the polynomial small-model property for $\overline{\mathsf{A}}\mathsf{A}\mathsf{B}\overline{\mathsf{B}}/\overline{\mathsf{A}}\mathsf{A}\mathsf{E}\overline{\mathsf{E}}$ of the previous section depends *on the specific formula* φ we are considering (as the input of the MC problem), whereas the exponential small-model property for $\overline{\mathsf{A}}\mathsf{A}\mathsf{B}\overline{\mathsf{B}}\mathsf{E}/\overline{\mathsf{A}}\mathsf{A}\mathsf{E}\overline{\mathsf{B}}\mathsf{E}$ states the existence of a shorter trace ρ' equivalent to (a generic) ρ with respect to *all* formulas up to a given B/E-nesting depth h. As a matter of fact, the former relies on the witness positions, which are defined on φ; the latter on h-prefix/suffix bisimilarity and h-prefix/suffix samplings, that are independent of any formula (they are only based on h, i.e., the maximum B/E-nesting depth of formulas we want to consider). Therefore, we can say that the latter small-model states a stronger property; however, this *may* lead to a bound on the length of equivalent traces higher than necessary: we have proved the MC problem for $\overline{\mathsf{A}}\mathsf{A}\mathsf{B}\overline{\mathsf{B}}\mathsf{E}/\overline{\mathsf{A}}\mathsf{A}\mathsf{E}\overline{\mathsf{B}}\mathsf{E}$ to be in **EXPSPACE**, but it is only known to be **PSPACE**-hard (since $\overline{\mathsf{E}}$ or $\overline{\mathsf{B}}$ are enough for **PSPACE**-hardness, as shown in Appendix B.3). We do not know whether this complexity gap is due to the small-model proving a loose bound (that might be strengthened by finding another characterization depending on the input formula as well), or to a weak complexity lower-bound (here, exploiting the other modalities A, $\overline{\mathsf{A}}$ and B/E, along with $\overline{\mathsf{B}}$ and $\overline{\mathsf{E}}$ jointly, may enable us to prove a stronger one), or to both at the same time. We will come back to $\overline{\mathsf{A}}\mathsf{A}\mathsf{B}\overline{\mathsf{B}}\mathsf{E}/\overline{\mathsf{A}}\mathsf{A}\mathsf{E}\overline{\mathsf{B}}\mathsf{E}$ in Chapter 7, where we will relax homogeneity and will be able to decrease the complexity upper bound (yet still not matching the lower) also for the homogeneous case.

4.3 Conclusions

In this chapter we have studied two well-behaved **PSPACE**-complete fragments, $\overline{\mathsf{A}}\mathsf{A}\mathsf{B}\overline{\mathsf{B}}$ and $\overline{\mathsf{A}}\mathsf{A}\mathsf{E}\overline{\mathsf{E}}$, which are quite promising from the point of view of applications. MC for $\overline{\mathsf{A}}\mathsf{A}\mathsf{B}\overline{\mathsf{B}}\mathsf{E}$ and $\overline{\mathsf{A}}\mathsf{A}\mathsf{E}\overline{\mathsf{B}}\mathsf{E}$ turns out to be in **EXPSPACE** and **PSPACE**-hard.

Membership to **PSPACE** for the former two fragments and to **EXPSPACE** for the latter have been proved by means of small-models which, in turn, rely on suitable (depending on the specific fragment) contraction techniques applied to the traces of a finite Kripke structure. While the first result is novel, the second substantially simplifies the constructions and the complexity of the proofs given for the same problem in [MMP17].

5 HS fragments at the bottom of the polynomial hierarchy

The references for this chapter are [Boz+18f; Boz+16d; Mol+16b].

Contents

5.1 Some complexity classes in the polynomial hierarchy 96
5.2 $\mathbf{P}^{\mathbf{NP}}$ MC algorithm for $A\overline{A}B$ and $A\overline{A}E$ 98
5.3 $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$ MC algorithm for $A\overline{A}$, $\overline{A}B$, and AE 103
5.3.1 The problem $TB(SAT)_{1\times M}$ 103
5.3.2 Reduction of MC for $A\overline{A}$ to $TB(SAT)_{1\times M}$ 105
5.3.3 Reduction of MC for $\overline{A}B$ (resp., AE) to $TB(SAT)_{1\times M}$ 109
5.4 $\mathbf{P}^{\mathbf{NP}}$-hardness of MC for AB and $\overline{A}E$ 110
5.5 $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$-hardness of MC for A and $\overline{A}$ 115
5.6 Conclusions 119

In this chapter, we study the MC problem for some of the sub-fragments of $A\overline{A}B\overline{B}$ and of $A\overline{A}E\overline{E}$—removing $\overline{B}$ and $\overline{E}$ which lead to **PSPACE**-hardness—namely, A, $\overline{A}$, $A\overline{A}$, AB, $\overline{A}B$, AE, $\overline{A}E$, $A\overline{A}B$, and $A\overline{A}E$. All these have a similar computational complexity, as their MC settles in one of the lowest levels of the *polynomial-time hierarchy*, $\mathbf{P}^{\mathbf{NP}}$, or even below. We recall that $\mathbf{P}^{\mathbf{NP}}$ is the set of problems decided by a deterministic polynomial-time Turing machine, endowed with an oracle for the class **NP** which decides, in one computation step, whether an instance of a problem belonging to **NP** is positive or not. $\mathbf{P}^{\mathbf{NP}}$ is also referred to as **P** *relative to* **NP**.

In order to solve their MC, A$\overline{\text{A}}$B, A$\overline{\text{A}}$E, AB, and $\overline{\text{A}}$E require the **P** Turing machine to perform $O(n^k)$ queries to the **NP** oracle, for an input of size n and for some constant $k \geq 0$. Conversely, A, $\overline{\text{A}}$, A$\overline{\text{A}}$, $\overline{\text{A}}$B, and AE are "easier" since, for them, $O(\log^2 n)$ queries are always enough: the MC problem for the latter fragments witnesses a "non-standard" complexity class in the polynomial-time hierarchy, called *bounded-query* class and denoted as $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$, that will be presented in more detail below.

More formally, we first prove that MC for AB, $\overline{\text{A}}$E, A$\overline{\text{A}}$B, and A$\overline{\text{A}}$E is a $\mathbf{P}^{\mathbf{NP}}$-complete problem. To this end, we design a $\mathbf{P}^{\mathbf{NP}}$ algorithm exploiting the polynomial small-model property of Section 4.1 and we prove a matching complexity lower bound.Next, we devise another MC algorithm for all the remaining fragments, A, $\overline{\text{A}}$, A$\overline{\text{A}}$, $\overline{\text{A}}$B, and AE, via a reduction *to* the problem TB(SAT)$_{1\times M}$ [Sch03] (a problem complete for the aforementioned bounded-query class), whose instances are complex circuits in which some of the gates feature **NP** oracles. Finally, we prove a lower bound showing that at least $\log n$ queries are needed to solve MC; unfortunately, it does not match the upper bound, leaving open the question whether the problem can be solved by $o(\log^2 n)$ (i.e., strictly less than $O(\log^2 n)$) queries to an **NP** oracle, or a tighter lower bound can be proved (or both). Figure 5.1 depicts the scenario of complexity of MC for A$\overline{\text{A}}$B$\overline{\text{B}}$ and all its sub-fragments.

Before moving on, we want to intuitively explain why, for instance, $\overline{\text{A}}$B is "easier" than AB (the same holds for the symmetric fragments AE and $\overline{\text{A}}$E); this is justified by the different expressiveness of the two fragments. Let us consider an AB formula of the form $\langle \text{B} \rangle \langle \text{A} \rangle \theta$. A trace ρ satisfies $\langle \text{B} \rangle \langle \text{A} \rangle \theta$ if there exists a prefix $\tilde{\rho}$ of ρ from which a branch, i.e., a trace starting from $\text{lst}(\tilde{\rho})$, satisfying θ departs. Hence, AB allows one to state specific properties of the branches departing from a state occurring in a given path. Such an ability will be exploited in Section 5.4 to prove the $\mathbf{P}^{\mathbf{NP}}$-hardness of AB. This kind of properties cannot be expressed in the fragment $\overline{\text{A}}$B. Indeed, for any given trace ρ, modality $\langle \overline{\text{A}} \rangle$ only allows one to "select" traces leading to the first state of ρ, and modality $\langle \text{B} \rangle$ is of no help: if we consider any prefix $\tilde{\rho}$ of ρ, the set of traces leading to its first state is exactly the same as the set of those leading to the first state of ρ, as $\text{fst}(\tilde{\rho}) = \text{fst}(\rho)$. Therefore, pairing $\langle \overline{\text{A}} \rangle$ and $\langle \text{B} \rangle$ does not give any advantage in terms of expressiveness. Finally, since A, $\overline{\text{A}}$, and A$\overline{\text{A}}$ are devoid of modalities for prefixes (and suffixes), they analogously belong to $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$.

Organization of the chapter.

- In the next section, we start by recalling some complexity classes that come into play.
- In Section 5.2, we describe a $\mathbf{P}^{\mathbf{NP}}$ MC algorithm for A$\overline{\text{A}}$B (and A$\overline{\text{A}}$E).
- In Section 5.3, we provide a different MC algorithm for the fragments A$\overline{\text{A}}$, $\overline{\text{A}}$B, and AE, whose computational complexity is lower, as it requires only $O(\log^2 n)$ queries to a **NP** oracle.

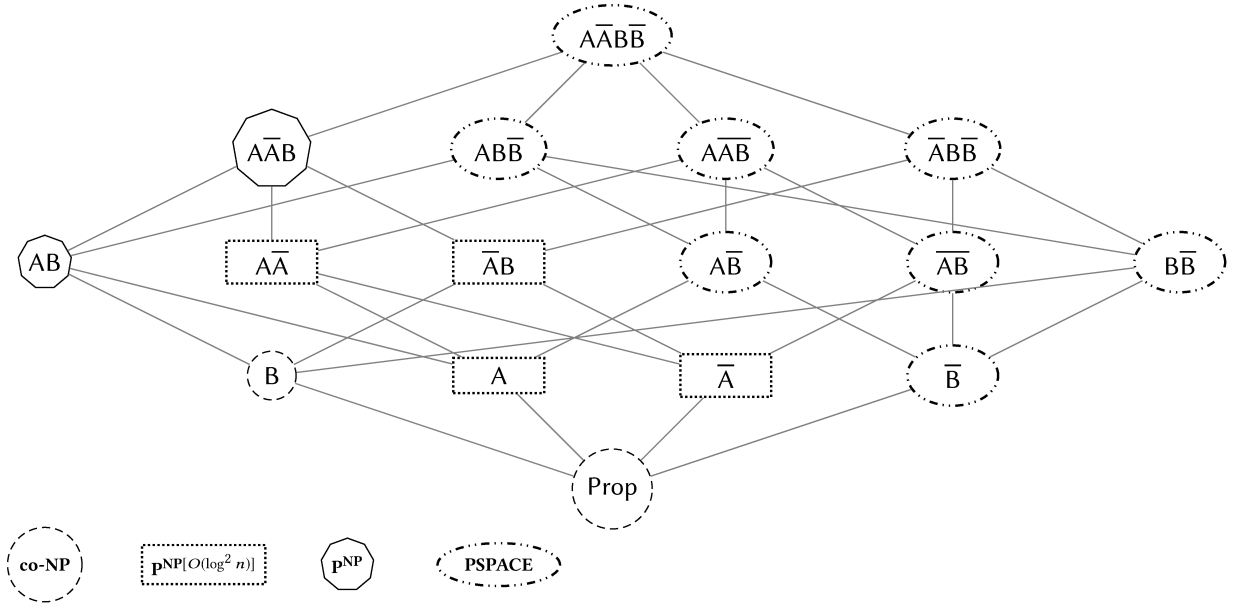


Figure 5.1: The computational complexity of MC for the sub-fragments of AĀBB̄.

- In Section 5.4, we prove a $\mathbf{P}^{\mathbf{NP}}$ lower bound for AB (resp., $\overline{\mathsf{A}}\mathsf{E}$) MC. The bound immediately propagates to $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}$ (resp., $\mathsf{A}\overline{\mathsf{A}}\mathsf{E}$), closing the complexity gap and proving that the MC problem for AB, $\overline{\mathsf{A}}\mathsf{E}$, $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}$, and $\mathsf{A}\overline{\mathsf{A}}\mathsf{E}$ is $\mathbf{P}^{\mathbf{NP}}$-complete.
- In Section 5.5, we prove that MC for A and $\overline{\mathsf{A}}$ formulas requires at least $\log n$ queries to a **NP** oracle: this bound propagates to $\mathsf{A}\overline{\mathsf{A}}$, $\overline{\mathsf{A}}\mathsf{B}$, and AE.

Before going into the results, let us make the following example.

Example 5.0.1. We show how some meaningful properties, to be checked against the Kripke structure $\mathcal{K}_{Sched}$ of Figure 2.4 at page 27, can be expressed by formulas of $\overline{\mathsf{A}}\mathsf{E}$ (a fragment that will be studied in detail next). As in Example 2.1.9, in all formulas we force the validity of the considered property over all legal computation sub-intervals by using modality [E] (all computation sub-intervals are suffixes of at least one initial trace).
Truth of the next statements can easily be checked (formulas express the same properties as those in Example 2.1.9, but are here adapted to $\overline{\mathsf{A}}\mathsf{E}$):

- $\mathcal{K}_{Sched} \models [\mathrm{E}]\left(\langle \mathrm{E}\rangle^{3} \top \rightarrow (\chi(p_1,p_2) \vee \chi(p_1,p_3) \vee \chi(p_2,p_3))\right)$, where $\chi(p,q)$ stands for $\langle \mathrm{E}\rangle \langle \overline{\mathrm{A}}\rangle p \wedge \langle \mathrm{E}\rangle \langle \overline{\mathrm{A}}\rangle q$;
- $\mathcal{K}_{Sched} \not\models [\mathrm{E}](\langle \mathrm{E}\rangle^{10} \top \rightarrow \langle \mathrm{E}\rangle \langle \overline{\mathrm{A}}\rangle p_3)$;
- $\mathcal{K}_{Sched} \not\models [\mathrm{E}](\langle \mathrm{E}\rangle^{5} \rightarrow (\langle \mathrm{E}\rangle \langle \overline{\mathrm{A}}\rangle p_1 \wedge \langle \mathrm{E}\rangle \langle \overline{\mathrm{A}}\rangle p_2 \wedge \langle \mathrm{E}\rangle \langle \overline{\mathrm{A}}\rangle p_3))$.

The first formula states that in any suffix having length at least 4 of an initial trace, at least 2 proposition letters are witnessed. $\mathcal{K}_{Sched}$ satisfies the formula since a process cannot be executed twice in a row.
The second formula states that in any suffix of an initial trace having length at least 11, process 3 is executed at least once in some internal states (*non starvation*). $\mathcal{K}_{Sched}$ does not satisfy the formula since the scheduler can avoid executing a process ad libitum.
The third formula requires that in any suffix of an initial trace having length at least 6, p_1, p_2 and p_3 are all witnessed. The only way to satisfy this property is to constrain the scheduler to execute the three processes in a strictly periodic manner (*strict alternation*), that is, $p_i p_j p_k p_i p_j p_k p_i p_j p_k \ldots$, $i,j,k \in \{1,2,3\}$, $i \neq j \neq k \neq i$, but this is not the case.

5.1 Some complexity classes in the polynomial hierarchy

The *polynomial-time hierarchy*, denoted by **PH**, was introduced by Stockmeyer in [Sto76], and is defined as

$$\mathbf{PH} = \bigcup_{k \in \mathbb{N}} \Delta_k^p,$$

where $\Delta_0^p = \Sigma_0^p = \Pi_0^p = \mathbf{P}$ and, for all $k \geq 1$,

$$\Delta_k^p = \mathbf{P}^{\Sigma_{k-1}^p}, \qquad \Sigma_k^p = \mathbf{NP}^{\Sigma_{k-1}^p}, \qquad \Pi_k^p = \mathbf{co}\text{-}\Sigma_k^p.$$

In particular, we have that $\Delta_1^p = \mathbf{P}$, $\Sigma_1^p = \mathbf{NP}$, and $\Delta_2^p = \mathbf{P}^{\mathbf{NP}}$. A well-known example of complete problem for Σ_k^p (resp., Π_k^p) is to decide the truth of fully-quantified formulas of the form

$$Q_1 x_1 Q_2 x_2 \cdots Q_n x_n \phi(x_1, x_2, \ldots, x_n),$$

where $\phi(x_1, x_2, \ldots, x_n)$ is a quantifier-free Boolean formula whose variables range in the set $\{x_1, x_2, \ldots, x_n\}$, $Q_i \in \{\exists, \forall\}$, for all $2 \leq i \leq n$, $Q_1 = \exists$ (resp., $Q_1 = \forall$), and there are $k-1$ quantifier alternations, that is, $k-1$ different indexes $j > 1$ such that $Q_j \neq Q_{j-1}$. On the contrary, Δ_k^p does not feature very popular complete problems. As an example, for each $k \geq 1$, a Δ_{k+1}^p-complete problem is to decide whether, given a true quantified Boolean formula of the form

$$\exists x_1 \cdots \exists x_r \forall x_{r+1} Q_{r+2} x_{r+2} \cdots Q_n x_n \phi(x_1, \ldots, x_n),$$

with $k-1$ quantifier alternations, the lexicographically maximum truth assignment v to the variables $(x_1, \ldots, x_r)$ such that

$$\forall x_{r+1} Q_{r+2} x_{r+2} \cdots Q_n x_n \phi(v(x_1), \ldots, v(x_r), x_{r+1}, \ldots, x_n)$$

is true assigns 1 to x_r [Got95].

As a particular case, given a satisfiable Boolean formula $\phi(x_1, \ldots, x_n)$, the problem of deciding whether the lexicographically maximum truth assignment to $(x_1, \ldots, x_n)$ satisfying ϕ assigns 1 to x_n is complete for $\Delta_2^p = \mathbf{P}^{\mathbf{NP}}$. For other examples of $\mathbf{P}^{\mathbf{NP}}$-complete problems (many of them are related to MC) we refer the reader to [BMP09; LMS01; LMS02; LMP10].

Above **NP** and **co-NP**, but below $\mathbf{P}^{\mathbf{NP}}$, is the class $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$, introduced by Papadimitriou and Zachos in [PZ82], which is the set of problems decided by a deterministic **P** algorithm (Turing machine) which requires only $O(\log n)$ queries to an **NP** oracle (being n the input size). Analogously, $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$ is the set of problems decided by a **P** algorithm requiring $O(\log^2 n)$ queries to an **NP** oracle.[1] These complexity classes (and all others which set a bound on the number of allowed queries) are called *bounded query classes*. Note that $\mathbf{P}^{\mathbf{NP}}$, $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$ and $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$ are closed under complementation, as well as under **LOGSPACE** (many-one) reductions.

[1] Here and in the following, we assume that the polynomial hierarchy **PH** is not collapsing, and that $\mathbf{P}^{\mathbf{NP}}$, $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$, and $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$ are *distinct*, as it is widely conjectured.

As for $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$, it has been proved (see [BH91; Wag90]) that $\mathbf{P}^{\mathbf{NP}[O(\log n)]} = \mathbf{LOGSPACE}^{\mathbf{NP}} = \mathbf{P}^{\mathbf{NP}}_{\|}$, where $\mathbf{P}^{\mathbf{NP}}_{\|}$ is the class of problems decided by a deterministic **P** algorithm which performs a single round (or a *constant* number of rounds) of parallel queries to an **NP** oracle. By *parallel queries*, we mean that each query is independent of the outcome of any other or, equivalently, that all queries have to be formulated before the oracle is consulted. Obviously, the constraint of parallelism is not necessarily fulfilled in the class $\mathbf{P}^{\mathbf{NP}}$, where a query to the oracle may be *adaptive*, that is, it may depend on the results of previously performed queries. An example of complete problem for $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$ is PARITY(SAT): given a set of Boolean formulas $\Gamma = \{\phi_1, \ldots, \phi_n\}$, the problem is to decide if the number of *satisfiable* formulas in Γ is odd or even [Wag87].

As for $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$, it has been proved in [CS92] that $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]} = \mathbf{P}^{\mathbf{NP}}_{\|O(\log n)}$ (in $\mathbf{P}^{\mathbf{NP}}_{\|O(\log n)}$ a succession of $O(\log n)$ parallel query rounds are allowed). To the best of our knowledge, the first complete problems for this class were introduced in [Sch03]. Among these, a detailed account of the problem $\mathrm{TB(SAT)}_{1\times M}$ will be given in Section 5.3.

5.2 $\mathbf{P}^{\mathbf{NP}}$ MC algorithm for $\mathrm{A\overline{A}B}$ and $\mathrm{A\overline{A}E}$

In this section, we present an MC algorithm for $\mathrm{A\overline{A}B}$ formulas (see the procedure `MC` reported in Algorithm 5.1) with complexity in the class $\mathbf{P}^{\mathbf{NP}}$. W.l.o.g., we restrict our attention to $\mathrm{A\overline{A}B}$ formulas devoid of occurrences of conjunctions and universal modalities (definable, as usual, by means of disjunctions, negations, and existential modalities).

The MC procedure `MC` for a formula ψ against a Kripke structure $\mathcal{K}$ exploits two global vectors, V_{A} and $V_{\overline{\mathsf{A}}}$, which can be seen as the tabular representations of two Boolean functions taking as arguments a subformula ϕ of ψ and a state s

Algorithm 5.1 `MC`$(\mathcal{K}, \psi,$ DIRECTION$)$

1: **for** all $\langle \mathsf{A} \rangle \phi \in \mathrm{ModSubf}_{\mathrm{A\overline{A}}}(\psi)$ **do**
2: `MC`$(\mathcal{K}, \phi,$ FORWARD$)$
3: **for** all $\langle \overline{\mathsf{A}} \rangle \phi \in \mathrm{ModSubf}_{\mathrm{A\overline{A}}}(\psi)$ **do**
4: `MC`$(\mathcal{K}, \phi,$ BACKWARD$)$
5: **for** all $s \in S$ **do**
6: **if** DIRECTION is FORWARD **then**
7: $V_{\mathsf{A}}(\psi, s) \leftarrow \mathit{Success}(\texttt{Oracle}(\mathcal{K}, \psi, s,$ FORWARD$, V_{\mathsf{A}} \cup V_{\overline{\mathsf{A}}}))$
8: **else if** DIRECTION is BACKWARD **then**
9: $V_{\overline{\mathsf{A}}}(\psi, s) \leftarrow \mathit{Success}(\texttt{Oracle}(\mathcal{K}, \psi, s,$ BACKWARD$, V_{\mathsf{A}} \cup V_{\overline{\mathsf{A}}}))$

of $\mathcal{K}$. The function $V_{\mathrm{A}}(\phi, s)$ (resp., $V_{\overline{\mathrm{A}}}(\phi, s)$) returns $\top$ if and only if there exists a trace $\rho \in \mathrm{Trc}_{\mathcal{K}}$ starting from the state s (resp., leading to the state s) such that $\mathcal{K}, \rho \models \phi$. MC is initially invoked with parameters ($\mathcal{K}$, $\neg\psi$, FORWARD). During the execution, it instantiates the entries of V_{A} and $V_{\overline{\mathrm{A}}}$, which are exploited in order to answer the MC problem $\mathcal{K} \models \psi$. In the end, this is equivalent to checking whether $V_{\mathrm{A}}(\neg\psi, s_0) = \bot$, where s_0 is the initial state of $\mathcal{K}$.

Let us consider MC in more detail. Along with the Kripke structure $\mathcal{K}$ and the formula ψ, MC features a third parameter, DIRECTION, which can be assigned the value FORWARD (resp., BACKWARD), that is used in combination with the modality $\langle \mathrm{A} \rangle$ (resp., $\langle \overline{\mathrm{A}} \rangle$) for a forward (resp., backward) unravelling of $\mathcal{K}$. MC is applied recursively (lines 1–4) on the nesting of modalities $\langle \mathrm{A} \rangle$ and $\langle \overline{\mathrm{A}} \rangle$ in the formula ψ (in the base case, ψ features no occurrences of $\langle \mathrm{A} \rangle$ or $\langle \overline{\mathrm{A}} \rangle$). In order to instantiate the Boolean vectors V_{A} and $V_{\overline{\mathrm{A}}}$, an oracle is invoked (lines 5–9) for each state s of the Kripke structure. Such an invocation is syntactically represented by $\mathit{Success}(\mathtt{Oracle}(\mathcal{K}, \psi, s, \textsc{direction}, V_{\mathrm{A}} \cup V_{\overline{\mathrm{A}}}))$, and it returns $\top$ whenever there exists a computation of the non-deterministic algorithm $\mathtt{Oracle}(\mathcal{K}, \psi, s, \textsc{direction}, V_{\mathrm{A}} \cup V_{\overline{\mathrm{A}}})$ returning $\top$, namely, whenever there is a suitable trace starting from, or leading to s (depending on the value of the parameter DIRECTION), satisfying ψ.

We define now the set of $\mathrm{A}\overline{\mathrm{A}}$-*modal subformulas* of ψ ($\mathsf{ModSubf}_{\mathrm{A}\overline{\mathrm{A}}}(\psi)$) used to "direct" the recursive calls of MC (lines 1–4).

Definition 5.2.1. The set $\mathsf{ModSubf}_{\mathrm{A}\overline{\mathrm{A}}}(\psi)$ of $\mathrm{A}\overline{\mathrm{A}}$-*modal subformulas* of an $\mathrm{A}\overline{\mathrm{A}}\mathrm{B}$ formula ψ is the set of subformulas of ψ either of the form $\langle \mathrm{A} \rangle \psi'$ or of the form $\langle \overline{\mathrm{A}} \rangle \psi'$, for some ψ', which are *not in the scope of any* $\langle \mathrm{A} \rangle$ *or* $\langle \overline{\mathrm{A}} \rangle$ *modality*.

As an example, it holds that

- $\mathsf{ModSubf}_{\mathrm{A}\overline{\mathrm{A}}}(\langle \mathrm{A} \rangle \langle \overline{\mathrm{A}} \rangle q) = \{\langle \mathrm{A} \rangle \langle \overline{\mathrm{A}} \rangle q\}$, and
- $\mathsf{ModSubf}_{\mathrm{A}\overline{\mathrm{A}}}\big(\big(\langle \mathrm{A} \rangle p \wedge \langle \mathrm{A} \rangle \langle \overline{\mathrm{A}} \rangle q\big) \rightarrow \langle \mathrm{A} \rangle p\big) = \{\langle \mathrm{A} \rangle p, \langle \mathrm{A} \rangle \langle \overline{\mathrm{A}} \rangle q\}$.

MC is recursively called on each formula ϕ such that $\langle \mathrm{A} \rangle \phi$ or $\langle \overline{\mathrm{A}} \rangle \phi$ belongs to the set $\mathsf{ModSubf}_{\mathrm{A}\overline{\mathrm{A}}}(\psi)$ (lines 1–4). In this way, we can recursively gather in the Boolean vectors V_{A} and $V_{\overline{\mathrm{A}}}$, by increasing nesting depth of the modalities $\langle \mathrm{A} \rangle$ and $\langle \overline{\mathrm{A}} \rangle$, the oracle answers for all the formulas ψ' such that $\langle \mathrm{A} \rangle \psi'$, or $\langle \overline{\mathrm{A}} \rangle \psi'$, is a subformula (be it maximal or not) of ψ.

We now consider the *non-deterministic polynomial time* procedure $\mathtt{Oracle}(\mathcal{K}, \psi, s, \textsc{direction}, V_{\mathrm{A}} \cup V_{\overline{\mathrm{A}}})$ (see Algorithm 5.2), which is used as the "basic engine" by the oracle in the aforementioned MC Algorithm 5.1. The idea underlying Algorithm 5.2 is to first non-deterministically generate a trace $\tilde{\rho}$ by unravelling the Kripke structure $\mathcal{K}$ according to the parameter DIRECTION, and then to verify ψ over $\tilde{\rho}$. Such a procedure exploits the proved *polynomial small-model property*

Algorithm 5.2 `Oracle`$(\mathcal{K}, \psi, s,$ DIRECTION$, V_{\mathsf{A}} \cup V_{\overline{\mathsf{A}}})$

$\tilde{\rho} \leftarrow$ `A_trace`$(\mathcal{K}, s, |S| \cdot (2|\psi| + 1)^2,$ DIRECTION$)$ ◁ a trace of $\mathcal{K}$ from/to s having length $\leq |S| \cdot (2|\psi| + 1)^2$
for all $\langle \mathsf{A} \rangle \phi \in \mathsf{ModSubf}_{\mathsf{A}\overline{\mathsf{A}}}(\psi)$ **do**
  for $i = 1, \ldots, |\tilde{\rho}|$ **do**
    $T[\langle \mathsf{A} \rangle \phi, i] \leftarrow V_{\mathsf{A}}(\phi, \tilde{\rho}(i))$
for all $\langle \overline{\mathsf{A}} \rangle \phi \in \mathsf{ModSubf}_{\mathsf{A}\overline{\mathsf{A}}}(\psi)$ **do**
  for $i = 1, \ldots, |\tilde{\rho}|$ **do**
    $T[\langle \overline{\mathsf{A}} \rangle \phi, i] \leftarrow V_{\overline{\mathsf{A}}}(\phi, \mathrm{fst}(\tilde{\rho}))$
for all subformulas φ of ψ, not contained in (or equal to) $\mathsf{A}\overline{\mathsf{A}}$-modal subformulas of ψ, by increasing length **do**
  if $\varphi = p$, for $p \in \mathcal{AP}$ **then**
    $T[p, 1] \leftarrow p \in \mu(\mathrm{fst}(\tilde{\rho}))$
    for $i = 2, \ldots, |\tilde{\rho}|$ **do**
      $T[p, i] \leftarrow T[p, i-1]$ and $p \in \mu(\tilde{\rho}(i))$
  else if $\varphi = \neg\varphi_1$ **then**
    for $i = 1, \ldots, |\tilde{\rho}|$ **do**
      $T[\varphi, i] \leftarrow$ not $T[\varphi_1, i]$
  else if $\varphi = \varphi_1 \vee \varphi_2$ **then**
    for $i = 1, \ldots, |\tilde{\rho}|$ **do**
      $T[\varphi, i] \leftarrow T[\varphi_1, i]$ or $T[\varphi_2, i]$
  else if $\varphi = \langle \mathsf{B} \rangle \varphi_1$ **then**
    $T[\varphi, 1] \leftarrow \bot$
    for $i = 2, \ldots, |\tilde{\rho}|$ **do**
      $T[\varphi, i] \leftarrow T[\varphi, i-1]$ or $T[\varphi_1, i-1]$
return $T[\psi, |\tilde{\rho}|]$

for formulas of $\mathsf{A\overline{A}B\overline{B}}$ (Theorem 4.1.5), guaranteeing that, in order to check the satisfiability of an $\mathsf{A\overline{A}B\overline{B}}$ formula ϕ, it is enough to consider traces whose length is bounded by $|S| \cdot (2|\phi| + 1)^2$.

An execution of $\texttt{Oracle}(\mathcal{K}, \psi, s, \textsc{direction}, V_{\mathsf{A}} \cup V_{\overline{\mathsf{A}}})$ starts (line 1) by *non-deterministically* generating a trace $\tilde{\rho}$ (having length at most $|S| \cdot (2|\psi| + 1)^2$), with s as its first (resp., last) state if the DIRECTION parameter is FORWARD (resp., BACKWARD). The trace is generated by visiting the unravelling of $\mathcal{K}$ (resp., of $\mathcal{K}$ with transposed edges). The remaining part of the algorithm *deterministically* checks whether $\mathcal{K}, \tilde{\rho} \models \psi$ or not. Such a verification is performed in a bottom-up way: for all the subformulas ϕ of ψ (starting from the minimal ones) and for all the prefixes $\tilde{\rho}(1, i)$ of $\tilde{\rho}$, with $1 \leq i \leq |\tilde{\rho}|$ (starting from the shorter ones), the procedure establishes if $\mathcal{K}, \tilde{\rho}(1, i) \models \phi$, and this result is stored in the entry $T[\phi, i]$ of a Boolean table T. Note that if the considered subformula of ψ is an element of $\mathsf{ModSubf}_{\mathsf{A\overline{A}}}(\psi)$, the algorithm does not need to perform any verification, since the result is already available in the Boolean vectors V_{A} and $V_{\overline{\mathsf{A}}}$ (as a consequence of the previously completed calls to the procedure `Oracle`), and the table T is updated accordingly (lines 2–7). For the remaining subformulas, the entries of T are computed, as we already said, in a bottom-up fashion (lines 8–22). The result of the overall verification is stored in $T[\psi, |\tilde{\rho}|]$ and returned (line 23).

Such an algorithm for checking formulas of $\mathsf{A\overline{A}B}$ can trivially be adapted to check formulas of the symmetric fragment $\mathsf{A\overline{A}E}$.

The next lemma states soundness and completeness of the procedure `Oracle` (its proof is in the Appendix C.1).

Lemma 5.2.2. *Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure, ψ be an $\mathsf{A\overline{A}B}$ formula, and $V_{\mathsf{A}}(\cdot,\cdot)$, $V_{\overline{\mathsf{A}}}(\cdot,\cdot)$ be two Boolean arrays. We assume that*

1. *for each $\langle \mathsf{A} \rangle \phi \in \mathsf{ModSubf}_{\mathsf{A\overline{A}}}(\psi)$ and $s' \in S$, $V_{\mathsf{A}}(\phi, s') = \top$ if and only if there exists $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{fst}(\rho) = s'$ and $\mathcal{K}, \rho \models \phi$, and*
2. *for each $\langle \overline{\mathsf{A}} \rangle \phi \in \mathsf{ModSubf}_{\mathsf{A\overline{A}}}(\psi)$ and $s' \in S$, $V_{\overline{\mathsf{A}}}(\phi, s') = \top$ if and only if there exists $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{lst}(\rho) = s'$ and $\mathcal{K}, \rho \models \phi$.*

Then $\texttt{Oracle}(\mathcal{K}, \psi, s, \textsc{direction}, V_{\mathsf{A}} \cup V_{\overline{\mathsf{A}}})$ features a successful computation (returning $\top$) if and only if:

- *there exists $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{fst}(\rho) = s$ and $\mathcal{K}, \rho \models \psi$, when DIRECTION is FORWARD;*
- *there exists $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{lst}(\rho) = s$ and $\mathcal{K}, \rho \models \psi$, when DIRECTION is BACKWARD.*

The following theorem states soundness and completeness of the MC procedure `MC` (Algorithm 5.1).

Theorem 5.2.3. *Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure, ψ be an A$\overline{\text{A}}$B formula, and $V_{\text{A}}(\cdot,\cdot)$ and $V_{\overline{\text{A}}}(\cdot,\cdot)$ be two Boolean arrays. If $\texttt{MC}(\mathcal{K}, \psi,$ DIRECTION$)$ is executed, then, for all $s \in S$:*

- *if* DIRECTION *is* FORWARD, $V_{\text{A}}(\psi, s) = \top$ *if and only if there exists* $\rho \in \text{Trc}_{\mathcal{K}}$ *such that* $\text{fst}(\rho) = s$ *and* $\mathcal{K}, \rho \models \psi$;
- *if* DIRECTION *is* BACKWARD, $V_{\overline{\text{A}}}(\psi, s) = \top$ *if and only if there exists* $\rho \in \text{Trc}_{\mathcal{K}}$ *such that* $\text{lst}(\rho) = s$ *and* $\mathcal{K}, \rho \models \psi$.

Proof. The proof is by induction on the number n of occurrences of $\langle \text{A} \rangle$ and $\langle \overline{\text{A}} \rangle$ modalities in ψ.

If $n = 0$, since $\text{ModSubf}_{\text{A}\overline{\text{A}}}(\psi) = \emptyset$, (1.) and (2.) of Lemma 5.2.2 are satisfied and the thesis trivially holds.

Otherwise, $n > 0$ and the formula ψ contains at least an $\langle \text{A} \rangle$ or an $\langle \overline{\text{A}} \rangle$ modality, and thus $\text{ModSubf}_{\text{A}\overline{\text{A}}}(\psi) \neq \emptyset$. Since each recursive call to MC (either at line 2 or 4) is performed on a formula ϕ featuring a number of occurrences of $\langle \text{A} \rangle$ and $\langle \overline{\text{A}} \rangle$ which is strictly less than the number of their occurrences in ψ, we can apply the inductive hypothesis. As a consequence, when the control flow reaches line 5, it holds that:

1. for each $\langle \text{A} \rangle \phi \in \text{ModSubf}_{\text{A}\overline{\text{A}}}(\psi)$ and $s' \in S$, $V_{\text{A}}(\phi, s') = \top$ if and only if there exists $\rho \in \text{Trc}_{\mathcal{K}}$ such that $\text{fst}(\rho) = s'$ and $\mathcal{K}, \rho \models \phi$;
2. for each $\langle \overline{\text{A}} \rangle \phi \in \text{ModSubf}_{\text{A}\overline{\text{A}}}(\psi)$ and $s' \in S$, $V_{\overline{\text{A}}}(\phi, s') = \top$ if and only if there exists $\rho \in \text{Trc}_{\mathcal{K}}$ such that $\text{lst}(\rho) = s'$ and $\mathcal{K}, \rho \models \phi$.

This implies that (1.) and (2.) of Lemma 5.2.2 are fulfilled. Hence (assuming that DIRECTION is FORWARD), it holds that, for $s \in S$, $V_{\text{A}}(\psi, s) = \top$ if and only if there exists $\rho \in \text{Trc}_{\mathcal{K}}$ such that $\text{fst}(\rho) = s$ and $\mathcal{K}, \rho \models \psi$. The case for DIRECTION = BACKWARD is symmetric, and thus omitted. □

As an immediate consequence, we have that the procedure MC solves the MC problem for A$\overline{\text{A}}$B formulas with an algorithm belonging to the class $\mathbf{P}^{\mathbf{NP}}$.

Corollary 5.2.4. *Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure and ψ be an A$\overline{\text{A}}$B formula. If $\texttt{MC}(\mathcal{K}, \neg\psi,$ FORWARD$)$ is executed, then $V_{\text{A}}(\neg\psi, s_0) = \bot$ if and only if $\mathcal{K} \models \psi$.*

Corollary 5.2.5. *The MC problem for A$\overline{\text{A}}$B formulas over finite Kripke structures is in $\boldsymbol{P}^{\boldsymbol{NP}}$.*

Proof. Given a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ and an A$\overline{\text{A}}$B formula ψ, the number of recursive calls performed by $\texttt{MC}(\mathcal{K}, \neg\psi,$ FORWARD$)$ is at most $|\psi|$. Each one costs $O(|\psi| + |S| \cdot (|\mathcal{K}| + |\psi| + |\psi| \cdot |S|))$, where the first addend it

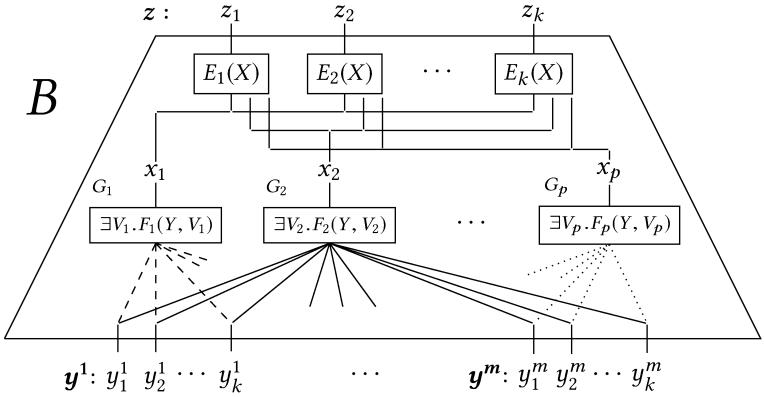


Figure 5.2: General form of a block.

is due to the search of ψ for its modal subformulas (lines 1–4), and the second one to the preparation of the input for the oracle call, for each $s \in S$ (lines 5–9). Therefore, its (deterministic) complexity is $O(|\psi|^2 \cdot |\mathcal{K}|^2)$.

As for `Oracle`$(\mathcal{K}, \psi, s,$ DIRECTION$, V_{\text{A}} \cup V_{\overline{\text{A}}})$, its (non-deterministic) complexity is $O(|\psi|^3 \cdot |\mathcal{K}|)$, where $|\psi|$ is a bound to the number of subformulas and $O(|\psi|^2 \cdot |\mathcal{K}|)$ is the number of steps necessary to generate and check $\tilde{\rho}$. □

By a straightforward adaptation of the procedure `Oracle`, it is easy to prove that also the MC problem for the symmetric fragment A$\overline{\text{A}}$E is in $\mathbf{P}^{\mathbf{NP}}$. As we will show in Section 5.4, both problems are actually *complete for* $\mathbf{P}^{\mathbf{NP}}$.

5.3 $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$ MC algorithm for A$\overline{\text{A}}$, $\overline{\text{A}}$B, and AE

In this section, we first propose an MC algorithm for the fragment A$\overline{\text{A}}$ with complexity $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$, thus lower than the one described in the previous section for A$\overline{\text{A}}$B. In fact, we do not directly devise an MC algorithm; we proceed instead *via a reduction to* a $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$-complete problem, TB(SAT)$_{1\times M}$ (a restriction of TB(SAT), see [Sch03]), whose instances are complex circuits where some of the gates are endowed with **NP** oracles.

5.3.1 The problem TB(SAT)$_{1\times M}$

In order to introduce TB(SAT), we need to preliminarily describe its basic component, which is the *block*. A block B (see Figure 5.2) is a circuit whose input lines are organized in m bit vectors $\boldsymbol{y}^1, \ldots, \boldsymbol{y}^m$, each of which has k entries, namely $\boldsymbol{y}^i = (y^i_1, \ldots, y^i_k)$. The values m and k are respectively called the

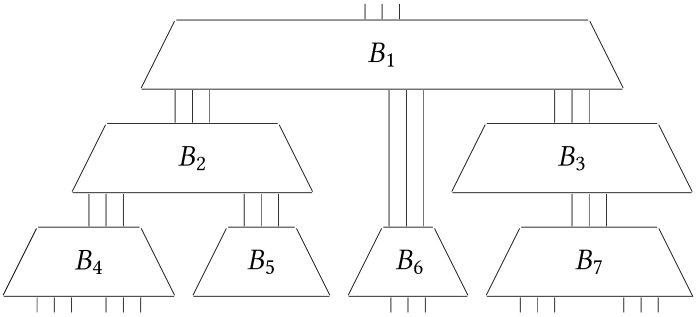


Figure 5.3: A tree of blocks (B_5 has degree $m = 0$).

degree and the *width* of B. The input lines are connected to p internal gates $G_1, \ldots, G_p$. Each gate G_i features a Boolean formula $F_i(Y, V_i)$ associated with it, where $Y = \{y_s^j \mid j = 1, \ldots, m,\ s = 1, \ldots, k\}$ and V_i is a set of private variables of F_i, not occurring in any other F_j, with $j \neq i$, that is, $V_i \cap V_j = \emptyset$ for $j \neq i$. The gate G_i queries a SAT oracle in order to decide whether the associated Boolean formula is satisfiable. The output of G_i is denoted by x_i, and it evaluates to $\top$ if and only if $F_i(Y, V_i)$ is satisfiable. Finally, k classic circuits (without oracles) $E_1, \ldots, E_k$ compute, from $X = \{x_1, \ldots, x_p\}$, their outputs $z_1, \ldots, z_k$, which are also the final k outputs of the block B. The size of B is defined as the total number of gates, plus the lengths of all the associated Boolean formulas. In the following, to make clear that a gate G_i (respectively, input y_i, block output z_i, gate output x_i) is an element of a block B, we write $B(G_i)$ (respectively, $B(y_i)$, $B(z_i)$, $B(x_i)$).

Given the $k \cdot m$ input bits, determining the output value of any z_i is a $\mathbf{P}_{\|}^{\mathbf{NP}}$ problem: the p queries associated with the oracle gates, which determine the outputs x_j's, can be performed in parallel (they are independent of each other) and then the value of the block output z_i can be calculated in deterministic polynomial time.

Blocks of *the same width* can be combined together to form a tree-structured complex circuit, called a *tree of blocks*. See Figure 5.3 for an example. Every block in the tree-structure has a level: blocks which are leaves of the tree are at level 1; a block B_i whose inputs depend on (at least) a block B_j at level $d - 1$ and possibly on other blocks at levels less than d, is at level d. In Figure 5.3, B_4, B_5, B_6, B_7 are at level 1, B_2 and B_3 at level 2, and B_1 at level 3. If the root of the tree-structure T is at level d, the k outputs of T can be determined by d rounds of parallel queries: all the queries relative to blocks placed at the same level d' can be answered in parallel once all those at level $d' - 1$ have been answered.

TB(SAT) is the problem of deciding whether a specific output z_i of (the root of) a tree-structure of blocks T is $\top$ or $\bot$, given the values for the inputs (of the leaf blocks) of T. As proved in [Sch03], the problem TB(SAT) is $\mathbf{P}^{\mathbf{NP}}$-complete.

The problem TB(SAT)$_{1\times M}$ is a constrained version of TB(SAT): any Boolean formula (SAT query) associated with a block B of the tree-structure must have the following form:

$$\exists \ell_1, \ldots, \ell_m \in \{1, \ldots, k\}\, \exists V_i'.F_i(y^1_{\ell_1}, \ldots, y^m_{\ell_m}, \ell_1, \ldots, \ell_m, V_i'),$$

where m and k are respectively the degree and the width of B. This amounts to say that F_i can use *only one bit from each input vector* of B (no matter which), hence "$1 \times M$". The existential quantification over the indexes $\ell_1, \ldots, \ell_m$ is an abuse of notation borrowed from [Sch03]: $\exists \ell_j \in \{1, \ldots, k\}$ is just a shorthand for k bits (belonging to the set of private variables) "$\ell_j = 1$", ..., "$\ell_j = k$", among which exactly one is $\top$. In the formula above, V_i' is V_i deprived of such bits.

In [Sch03], it is proved that TB(SAT)$_{1\times M}$ is a $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$-complete problem. In particular, the proof of membership to $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$ exploits the *squeeze* technique of [Got95] applied to TREE(SAT) instances. The particular form "$1 \times M$" of the queries allows us to "reshape" the tree-structure of blocks, in such a way that the height becomes logarithmic in the number of blocks. Therefore, only $O(\log n)$ rounds of parallel queries are needed, allowing us to prove membership of the problem to $\mathbf{P}^{\mathbf{NP}}_{\|O(\log n)} = \mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$.

5.3.2 Reduction of MC for A$\overline{\text{A}}$ to TB(SAT)$_{1\times M}$

Let us show now how to reduce the MC problem for A$\overline{\text{A}}$ formulas to TB(SAT)$_{1\times M}$. As in the previous section, w.l.o.g., we assume that only *existential* modalities occur in the A$\overline{\text{A}}$ formula ψ to be checked over a Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_1)$, with $S = \{s_1, \ldots, s_{|S|}\}$.[2] We consider its negation $\neg\psi$ and build from it a tree-structure of blocks $T_{\mathcal{K},\neg\psi}$. Each block of $T_{\mathcal{K},\neg\psi}$ has a type, FORWARD or BACKWARD, and it is associated with a subformula of $\neg\psi$. The root block, B_{root}, is always of type FORWARD and it is associated with $\neg\psi$. Each block B has an output line z_i for each state $s_i \in S$, thus the *width* of all blocks is $k = |S|$.

Starting from B_{root}, $T_{\mathcal{K},\neg\psi}$ is built by recursive applications of the following *basic step*, which are guided by the A$\overline{\text{A}}$-modal subformulas (recall Definition 5.2.1): if some (generic) block B is associated with a formula φ, then

- for every $\phi \in \text{ModSubf}_{\text{A}\overline{\text{A}}}(\varphi)$, where $\phi = \langle \text{A} \rangle\, \xi$, we create a FORWARD child B' of B associated with ξ, and
- for every $\phi' \in \text{ModSubf}_{\text{A}\overline{\text{A}}}(\varphi)$, where $\phi' = \langle \overline{\text{A}} \rangle\, \xi'$, we create a BACKWARD child B'' of B associated with ξ'.

[2] Here, for technical reasons, we assume an arbitrary order of the states of the Kripke structure, $s_1, \ldots, s_{|S|}$, where s_1 is the initial state.

Then, the basic step is recursively applied to all the generated children of B, terminating when $\mathsf{ModSubf}_{\mathsf{A}\overline{\mathsf{A}}}(\varphi) = \emptyset$. Note that a block B associated with a formula φ has *degree* $m = |\mathsf{ModSubf}_{\mathsf{A}\overline{\mathsf{A}}}(\varphi)|$.

In such a way, we determine the tree-structure of blocks $T_{\mathcal{K},\neg\psi}$. We now describe the internal structure of blocks.

As a preliminary step, we suitably transform $\mathsf{A}\overline{\mathsf{A}}$ formulas φ into Boolean ones by replacing all the occurrences of proposition letters and modal subformulas in φ by Boolean variables, as described by the next definition.

Definition 5.3.1. Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_1)$ be a finite Kripke structure and let χ be an $\mathsf{A}\overline{\mathsf{A}}$ formula. We define $\overline{\chi}(V_{\mathcal{AP}}, V_{modSubf})$—where $V_{\mathcal{AP}} = \{v_p \mid p \in \mathcal{AP}\}$ and $V_{modSubf} = \{v_{\chi'} \mid \chi' \in \mathsf{ModSubf}_{\mathsf{A}\overline{\mathsf{A}}}(\chi)\}$ are sets of Boolean variables—as the *Boolean* formula obtained from χ by replacing

- each (occurrence of a) $\mathsf{A}\overline{\mathsf{A}}$-modal subformula $\chi' \in \mathsf{ModSubf}_{\mathsf{A}\overline{\mathsf{A}}}(\chi)$ by the variable $v_{\chi'}$,
- and *subsequently* each (occurrence of a) proposition letter $p \in \mathcal{AP}$ by the variable v_p.

Given a trace $\rho \in \mathrm{Trc}_{\mathcal{K}}$ and an $\mathsf{A}\overline{\mathsf{A}}$ formula χ, it is easy to prove (by induction on the complexity of $\overline{\chi}(V_{\mathcal{AP}}, V_{modSubf})$) that if ω is an interpretation (truth assignment) of the variables of $V_{\mathcal{AP}} \cup V_{modSubf}$ such that $\omega(v_p) = \top \iff \mathcal{K}, \rho \models p$, for all $p \in \mathcal{AP}$, and $\omega(v_{\chi'}) = \top \iff \mathcal{K}, \rho \models \chi'$, for all $\chi' \in \mathsf{ModSubf}_{\mathsf{A}\overline{\mathsf{A}}}(\chi)$, then it holds that $\mathcal{K}, \rho \models \chi \iff \omega(\overline{\chi}(V_{\mathcal{AP}}, V_{modSubf})) = \top$.

We are now ready to describe the internal structure of a block B for a formula φ in $T_{\mathcal{K},\neg\psi}$. Let us assume that B has type FORWARD and let us refer to the block depicted in Figure 5.2 for the description. The block features a gate G_i, with $1 \leq i \leq |S|$, for each state of $\mathcal{K}$. Each output line z_i of B is directly linked to the output x_i of the oracle gate G_i, avoiding the use of circuits $E_1, \ldots, E_k$.

Now, let $F_i(Y, V)$ be the Boolean formula for the gate G_i, with $1 \leq i \leq |S|$ (for the sake of simplicity, we write V instead of V_i). The basic idea is that $F_i(Y, V)$ is satisfiable if and only if there is a trace having length at most $|S|^2 + 2$, starting from the i-th state of S, which satisfies $\overline{\varphi}(V_{\mathcal{AP}}, V_{modSubf})$, where φ is the formula associated with the block B. To check the existence of such a witness trace, we need a set of private variables $V_{trace} = \{v_1^1, \ldots, v_{|S|}^1, v_1^2, \ldots, v_{|S|}^2, \ldots, v_1^{|S|^2+2}, \ldots, v_{|S|}^{|S|^2+2}\}$. In particular, the subset of variables $v_1^j, \ldots, v_{|S|}^j$, with $1 \leq j \leq |S|^2 + 2$, is used to "encode" the state in the j-th position of the trace. The encoding requires that exactly one variable v_k^j of the subset is assigned to $\top$, for $1 \leq k \leq |S|$, meaning that the k-th state of $\mathcal{K}$ occurs in the j-th position of the sequence. Moreover, we use a set of private variables $V_{last} = \{v_1, v_2, \ldots, v_{|S|}\}$ which are used to encode

the last state of the witness trace (note that the length of the witness trace can be actually less than the bound $|S|^2 + 2$).

In detail, the Boolean formula $trace(V_{trace}, V_{last}, V_{\mathcal{AP}})$, which ensures that a truth assignment of the private variables V_{trace} properly encodes a trace ρ of $\mathcal{K}$ of length ℓ, for $1 \leq \ell \leq |S|^2 + 2$, is as follows.

$$trace(V_{trace}, V_{last}, V_{\mathcal{AP}}) = \bigvee_{\ell=1}^{|S|^2+2} \Big[\bigwedge_{t=1}^{\ell} one_t(v_1^t, v_2^t, \ldots, v_{|S|}^t) \wedge \bigwedge_{t=1}^{\ell-1} edge_t(v_1^t, \ldots, v_{|S|}^t, v_1^{t+1}, \ldots, v_{|S|}^{t+1}) \wedge \underbrace{\bigwedge_{t=1}^{|S|} (v_t^\ell \leftrightarrow v_t)}_{(1)} \wedge \underbrace{\bigwedge_{p \in \mathcal{AP}} \Big((v_p \rightarrow \bigwedge_{t=1}^{\ell} \bigwedge_{j=1}^{|S|} (v_j^t \rightarrow VAL(s_j, p))) \wedge (\neg v_p \rightarrow \bigvee_{t=1}^{\ell} \bigwedge_{j=1}^{|S|} (v_j^t \rightarrow \neg VAL(s_j, p))) \Big)}_{(2)} \Big].$$

For any $1 \leq t \leq \ell$, being ℓ the length of the witness trace, the Boolean formula $one_t(v_1^t, v_2^t, \ldots, v_{|S|}^t)$ "checks" that the variables $v_1^t, v_2^t, \ldots, v_{|S|}^t$ encode (exactly) one state for the t-th element of the trace:

$$one_t(v_1^t, v_2^t, \ldots, v_{|S|}^t) = \Big(\bigvee_{j=1}^{|S|} v_j^t \Big) \wedge \Big(\bigwedge_{j=1}^{|S|} \bigwedge_{k=j+1}^{|S|} \neg(v_j^t \wedge v_k^t) \Big).$$

Then, for any $1 \leq t \leq \ell - 1$, the formula $edge_t(v_1^t, \ldots, v_{|S|}^t, v_1^{t+1}, \ldots, v_{|S|}^{t+1})$ checks that if s_k and s_j are states which occur consecutively in the encoded trace (v_k^t and v_j^{t+1} are set to $\top$), then $(s_k, s_j) \in R$:

$$edge_t(v_1^t, \ldots, v_{|S|}^t, v_1^{t+1}, \ldots, v_{|S|}^{t+1}) = \bigvee_{(s_k, s_j) \in R} (v_k^t \wedge v_j^{t+1}).$$

Then, conjunct (1) ensures that the private variables in V_{last} encode the last state of the witness trace, that is, the ℓ-th state. Finally, conjunct (2) enforces the homogeneity assumption, ensuring that a variable $v_p \in V_{\mathcal{AP}}$ evaluates to $\top$ if and only if p holds over all the states of the witness trace ($VAL(s_j, p)$ is just a shorthand for $\top$ if $p \in \mu(s_j)$, and $\bot$ otherwise).

Now, taking the set of private variables $V = V_{last} \cup V_{trace} \cup V_{\mathcal{AP}} \cup V_{modSubf}$, the Boolean formula $F_i(Y, V)$ for the gate G_i is formally defined as:

$$F_i(Y,V) = v_i^1 \wedge \overline{\varphi}(V_{\mathcal{AP}}, V_{modSubf}) \wedge trace(V_{trace}, V_{last}, V_{\mathcal{AP}}) \wedge \bigwedge_{\langle \mathrm{A} \rangle\, \xi \in \mathrm{ModSubf}_{\mathrm{A}\overline{\mathrm{A}}}(\varphi)} \left(v_{\langle \mathrm{A} \rangle\, \xi} \leftrightarrow \bigvee_{j=1}^{|S|} (v_j \wedge y_j^{\xi})\right) \wedge \bigwedge_{\langle \overline{\mathrm{A}} \rangle\, \xi' \in \mathrm{ModSubf}_{\mathrm{A}\overline{\mathrm{A}}}(\varphi)} \left(v_{\langle \overline{\mathrm{A}} \rangle\, \xi'} \leftrightarrow \bigvee_{j=1}^{|S|} (v_j^1 \wedge y_j^{\xi'})\right)$$

The first conjunct of $F_i(Y,V)$ requires that the witness trace starts with the i-th state of $\mathcal{K}$. The fourth one requires that each private variable $v_{\langle \mathrm{A} \rangle\, \xi}$, for $\langle \mathrm{A} \rangle\, \xi \in \mathrm{ModSubf}_{\mathrm{A}\overline{\mathrm{A}}}(\varphi)$ has exactly the same truth assignment as the j-th output, y_j^{ξ}, of the block for ξ (which is a child of B)—provided that the final state of the trace is the j-th state of $\mathcal{K}$. Since exactly one among the variables of $V_{last} = \{v_1, \ldots, v_{|S|}\}$ is set to $\top$, it is guaranteed that at most one bit for every child-block is considered by B, thus fulfilling the restriction of $\mathrm{TB(SAT)}_{1\times M}$. The last conjunct of $F_i(Y,V)$ forces the symmetric constraint for modal subformulas of the form $\langle \overline{\mathrm{A}} \rangle\, \xi'$.

The formula $F_i(Y,V)$ for a gate G_i in a BACKWARD block is very similar: we just need to replace the first conjunct v_i^1 by v_i.

The following proposition states the correctness of the encoding for traces.

Proposition 5.3.2. *Given a trace $\rho \in \mathrm{Trc}_{\mathcal{K}}$, with $|\rho| \leq |S|^2 + 2$, there exists a truth assignment ω to the variables in V which satisfies the formula $trace(V_{trace}, V_{last}, V_{\mathcal{AP}})$, and*

- *for any $1 \leq t \leq |\rho|$ and $1 \leq j \leq |S|$, $\rho(t) = s_j \iff \omega(v_j^t) = \top$ and $\omega(v_j^{|\rho|}) = \omega(v_j)$;*
- *for any $p \in \mathcal{AP}$, $\omega(v_p) = \top \iff \mathcal{K}, \rho \models p$.*

Conversely, if a truth assignment ω to the variables in V satisfies the r-th disjunct of $trace(V_{trace}, V_{last}, V_{\mathcal{AP}})$, then there exists a trace $\rho \in \mathrm{Trc}_{\mathcal{K}}$, with $|\rho| = r$, such that

- *for any $1 \leq t \leq |\rho|$ and $1 \leq j \leq |S|$, $\rho(t) = s_j \iff \omega(v_j^t) = \top$;*
- *for any $p \in \mathcal{AP}$, $\mathcal{K}, \rho \models p \iff \omega(v_p) = \top$.*

The following theorem states the correctness of the construction of $T_{\mathcal{K},\neg\psi}$ (the proof is given in Appendix C.2).

Theorem 5.3.3. *Let ψ be an $\mathrm{A}\overline{\mathrm{A}}$ formula and $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_1)$ be a finite Kripke structure. For every block B of $T_{\mathcal{K},\neg\psi}$, if B is associated with an $\mathrm{A}\overline{\mathrm{A}}$ formula φ, then*

- *if B is a FORWARD block, for all $i \in \{1, \ldots, |S|\}$, $B(z_i) = \top$ if and only if there exists a trace $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{fst}(\rho) = s_i$ and $\mathcal{K}, \rho \models \varphi$;*

- *if B is a* BACKWARD *block, for all $i \in \{1, \ldots, |S|\}$, $B(z_i) = \top$ if and only if there exists a trace $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{lst}(\rho) = s_i$ and $\mathcal{K}, \rho \models \varphi$.*

The two next corollaries immediately follow.

Corollary 5.3.4. *Let ψ be an A$\overline{\text{A}}$ formula, $\mathcal{K}$ be a finite Kripke structure, and B_{root} be the root block of $T_{\mathcal{K},\neg\psi}$. Then, it holds that $B_{root}(z_1) = \bot$ if and only if $\mathcal{K} \models \psi$.*

Corollary 5.3.5. *The MC problem for A$\overline{\text{A}}$ formulas over finite Kripke structures is in $\boldsymbol{P}^{\boldsymbol{NP}[O(\log^2 n)]}$.*

Proof. The result follows from Corollary 5.3.4 and the fact that the instance of TB(SAT)$_{1\times M}$ generated from an A$\overline{\text{A}}$ formula ψ and a Kripke structure $\mathcal{K}$ is polynomial in $|\psi|$ and $|\mathcal{K}|$. □

5.3.3 Reduction of MC for $\overline{\text{A}}$B (resp., AE) to TB(SAT)$_{1\times M}$

We conclude the section by showing that it is possible to adapt the above-described reduction to the fragment $\overline{\text{A}}$B and the symmetric fragment AE. Let us focus on $\overline{\text{A}}$B (the case for AE can be dealt with in an analogous way).

Having in mind that $\overline{\text{A}}$B is a fragment of A$\overline{\text{A}}$B, by removing the case for modality $\langle \text{A} \rangle$ in Algorithm 5.2, we get a procedure for which Lemma 5.2.2 still holds. Since Algorithm 5.2 is in **NP**, there must exist a reduction to SAT, that is, given an instance $(\mathcal{K}, \psi, s, \text{DIRECTION}, V_{\overline{\text{A}}})$ for `Oracle`, there exists a Boolean formula $\Psi_{(\mathcal{K},\psi,s,\text{DIRECTION},V_{\overline{\text{A}}})}$, which depends on $(\mathcal{K}, \psi, s, \text{DIRECTION}, V_{\overline{\text{A}}})$, that is satisfiable if and only if `Oracle`$(\mathcal{K}, \psi, s, \text{DIRECTION}, V_{\overline{\text{A}}})$ admits a successful computation on the given input. By Lemma 5.2.2, this is the case if and only if:

- there exists $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{fst}(\rho) = s$ and $\mathcal{K}, \rho \models \psi$, in the case DIRECTION is FORWARD;
- there exists $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{lst}(\rho) = s$ and $\mathcal{K}, \rho \models \psi$, in the case DIRECTION is BACKWARD,

provided that for each $\langle \overline{\text{A}} \rangle \phi \in \mathsf{ModSubf}_{\text{A}\overline{\text{A}}}(\psi)$ and $s' \in S$, $V_{\overline{\text{A}}}(\phi, s') = \top$ if and only if there exists $\rho' \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{lst}(\rho') = s'$ and $\mathcal{K}, \rho' \models \phi$.

The idea is that such a formula $\Psi_{(\mathcal{K},\psi,s,\text{DIRECTION},V_{\overline{\text{A}}})}$ can be used as a SAT query for an oracle gate G_i in a generic block associated with the formula ψ. The role of the global Boolean vector $V_{\overline{\text{A}}}$ is instead played by the local communication among blocks in the tree-structure; $\Psi_{(\mathcal{K},\psi,s,\text{DIRECTION},V_{\overline{\text{A}}})}$ basically has the same structure as the Boolean formula $F_i(Y, V)$, with some minor differences outlined below.

First of all, we observe that Algorithm 5.2 works with traces whose length is bounded by $|S| \cdot (2|\psi| + 1)^2$ (instead of $|S|^2 + 2$ as in $F_i(Y, V)$). A trace is then encoded exactly as in $F_i(Y, V)$ by using a set of private variables $V_{trace} = \{v_j^t \mid j = 1, \ldots, |S|, t = 1, \ldots, |S| \cdot (2|\psi| + 1)^2\}$; $\Psi_{(\mathcal{K},\psi,s,\text{DIRECTION},V_{\overline{\text{A}}})}$ has also to encode the Boolean table T of `Oracle` with entries $T[\varphi, i]$, where φ is a subformula of ψ, and $1 \leq i \leq |S| \cdot (2|\psi| + 1)^2$ is the length of a prefix of the considered trace. Therefore, there is a variable $x_{\varphi,i}$ in $\Psi_{(\mathcal{K},\psi,s,\text{DIRECTION},V_{\overline{\text{A}}})}$ for each entry $T[\varphi, i]$, with the intuitive meaning that $x_{\varphi,i}$ is assigned $\top$ if and only if $T[\varphi, i] = \top$. Actually, in this encoding we do not need any entry for a modal subformula $\langle \overline{\text{A}} \rangle \xi$, whose truth value is conveyed by y_j^{ξ}, namely, the j-th input connected to the child block for the subformula ξ (assuming that the starting state of the trace is the j-th state of $\mathcal{K}$).

It is worth noting that this construction is possible since all the prefixes of the trace ρ encoded by the assignment to V_{trace}, and ρ itself, share the same starting point, and thus agree on the truth value of any $\overline{\text{A}}$-modal subformula. The most relevant consequence of this property is that the constraint of $\text{TB(SAT)}_{1\times M}$ on the form of SAT queries is respected.

As for the construction of $T_{\mathcal{K},\neg\psi}$, it is exactly as before where, in particular, the root block B_{root} has type FORWARD, and all the other blocks have type BACKWARD.

The following result can be stated.

Theorem 5.3.6. *The MC problem for* $\overline{\text{A}}\text{B}$ *(resp.,* A$\overline{\text{E}}$*) formulas over finite Kripke structures is in* $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$.

The construction we have sketched cannot be adapted to the fragment AB. This is due to the fact that the *right* endpoints of the prefixes of a trace differ in general, and thus they do not necessarily agree on the truth value of A-modal subformulas, hence the restriction of $\text{TB(SAT)}_{1\times M}$ on the form of SAT queries cannot be respected. In the next section, we will prove that MC for AB formulas is indeed *inherently* more difficult than MC for $\overline{\text{A}}\text{B}$.

5.4 $\mathbf{P}^{\mathbf{NP}}$-hardness of MC for AB and $\overline{\text{A}}$E

In this section, we show that the $\mathbf{P}^{\mathbf{NP}}$-complete problem named *Sequentially Nested SATisfiability* (SNSAT [LMS01]) can be reduced to MC for formulas of the fragment AB (and similarly $\overline{\text{A}}$E), thus proving $\mathbf{P}^{\mathbf{NP}}$-hardness of the latter. SNSAT is a logical problem featuring a series of nested satisfiability questions.

Definition 5.4.1. An instance $\mathcal{I}$ of SNSAT consists of a set of n Boolean variables $X = \{x_1, \ldots, x_n\}$ and a set of n Boolean formulas

$$\{F_1(Z_1), F_2(x_1, Z_2), \ldots, F_n(x_1, \ldots, x_{n-1}, Z_n)\},$$

where, for $i = 1, \ldots, n$, $F_i(x_1, \ldots, x_{i-1}, Z_i)$ has variables in $\{x_1, \ldots, x_{i-1}\}$ and in the set of private variables $Z_i = \{z_i^1, \ldots, z_i^{j_i}\}$, that is, $Z_i \cap Z_t = \emptyset$, for $t \neq i$, and $X \cap Z_i = \emptyset$. We denote by $|\mathcal{I}|$ the cardinality $|X| = n$.
Let $v_{\mathcal{I}}$ be a truth-assignment of the variables in X defined as follows:

$$v_{\mathcal{I}}(x_i) = \top \iff F_i(v_{\mathcal{I}}(x_1), \ldots, v_{\mathcal{I}}(x_{i-1}), Z_i) \text{ is satisfiable}$$

(by a suitable truth-assignment to the private variables $z_i^1, \ldots, z_i^{j_i} \in Z_i$).
SNSAT is the problem of deciding, given an instance $\mathcal{I}$ with $|\mathcal{I}| = n$, whether $v_{\mathcal{I}}(x_n) = \top$ or not. In such a case, we say that $\mathcal{I}$ is a positive instance of SNSAT.

Given an SNSAT instance $\mathcal{I}$, with $|\mathcal{I}| = n$, the truth-assignment $v_{\mathcal{I}}$ is unique and it can be easily computed by a $\mathbf{P}^{\mathbf{NP}}$ algorithm as follows. A first query to a SAT oracle determines whether $v_{\mathcal{I}}(x_1)$ is $\top$ or $\bot$, since $v_{\mathcal{I}}(x_1) = \top$ if and only if $F_1(Z_1)$ is satisfiable. Then, we replace x_1 by the value $v_{\mathcal{I}}(x_1)$ in $F_2(x_1, Z_2)$ and another query to the SAT oracle is performed to determine whether $F_2(v_{\mathcal{I}}(x_1), Z_2)$ is satisfiable, yielding the value of $v_{\mathcal{I}}(x_2)$. This step is iterated other $n - 2$ times, finally obtaining the value of $v_{\mathcal{I}}(x_n)$.

Let $\mathcal{I}$ be an instance of SNSAT, with $|\mathcal{I}| = n$. We now show how to build a finite Kripke structure $\mathcal{K}_{\mathcal{I}}$ and an AB formula $\Phi_{\mathcal{I}}$ in polynomial time, such that $\mathcal{I}$ is a positive instance of SNSAT if and only if $\mathcal{K}_{\mathcal{I}} \models \Phi_{\mathcal{I}}$. Such a reduction is inspired by similar constructions from [LMS01].

Let $Z = \bigcup_{i=1}^{n} Z_i$ and let $\tilde{R} = \{r_i \mid i = 1, \ldots, n\}$ and $\tilde{R}_i = \tilde{R} \setminus \{r_i\}$ be $n + 1$ sets of auxiliary variables. The Kripke structure $\mathcal{K}_{\mathcal{I}}$ consists of a suitable composition of n instances of a *gadget* (an instance for each variable $x_1, \ldots, x_n \in X$). The structure of the gadget for x_i, with $1 \leq i \leq n$, is shown in Figure 5.4, assuming that the labeling of states (nodes) is defined as follows:

- $\mu(w_{x_i}) = X \cup Z \cup \{s, t\} \cup \tilde{R}_i$, and $\mu(\overline{w_{x_i}}) = (X \setminus \{x_i\}) \cup Z \cup \{s, t\} \cup \tilde{R}_i \cup \{p_{\overline{x_i}}\}$;
- for $u_i = 1, \ldots, j_i$, $\mu(w_{z_i^{u_i}}) = X \cup Z \cup \{s, t\} \cup \tilde{R}_i$, and $\mu(\overline{w_{z_i^{u_i}}}) = X \cup (Z \setminus \{z_i^{u_i}\}) \cup \{s, t\} \cup \tilde{R}_i$;
- $\mu(\overline{s_i}) = X \cup Z \cup \{t\} \cup \tilde{R}_i$.

The Kripke structure $\mathcal{K}_{\mathcal{I}}$ is obtained by sequentializing (adding suitable edges) the n instances of the gadget (in reverse order, from x_n to x_1), adding a collector

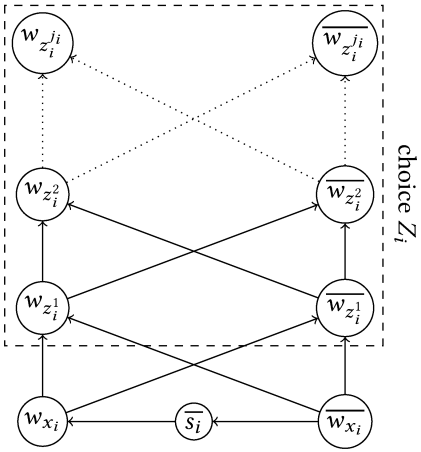


Figure 5.4: The gadget for x_i.

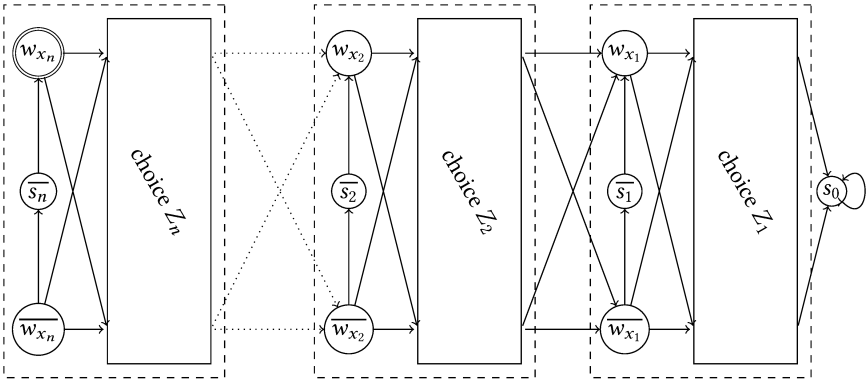


Figure 5.5: Kripke structure $\mathcal{K}_\mathcal{I}$ associated with an SNSAT instance $\mathcal{I}$, with $|\mathcal{I}| = n$. Note that the states $\overline{s_n}$ and $\overline{w_{x_n}}$ are unreachable.

terminal state s_0, with labeling $\mu(s_0) = X \cup Z \cup \{s\} \cup \tilde{R}$, and setting w_{x_n} as the initial state. The overall construction is reported in Figure 5.5. Formally, we have

$$\mathcal{K}_{\mathcal{I}} = (\mathcal{AP}, S, R, \mu, w_{x_n}),$$

where, in particular, $\mathcal{AP} = X \cup Z \cup \{s, t\} \cup \tilde{R} \cup \{p_{\overline{x_i}} \mid i = 1, \ldots, n\}$ and

$$S = \{s_0\} \cup \bigcup_{i=1}^{n} \left(\{\overline{s_i}, w_{x_i}, \overline{w_{x_i}}\} \cup \{w_{z_i^{u_i}}, \overline{w_{z_i^{u_i}}} \mid u_i = 1, \ldots, j_i\}\right).$$

The Kripke structure $\mathcal{K}_{\mathcal{I}}$ features the following properties:

- any trace satisfying s does not pass through any $\overline{s_i}$, for $1 \leq i \leq n$;
- any trace *not* satisfying t has s_0 as its last state;
- any trace *not* satisfying r_i passes through some state of the i-th gadget, for $1 \leq i \leq n$;
- the only trace satisfying $p_{\overline{x_i}}$ is $\overline{w_{x_i}}$ (note that $|\overline{w_{x_i}}| = 1$), for $1 \leq i \leq n$.

A trace $\rho \in \mathrm{Trc}_{\mathcal{K}_{\mathcal{I}}}$ *induces* a truth assignment of all the proposition letters, denoted by ω_ρ, which is defined as $\omega_\rho(y) = \top \iff \mathcal{K}_{\mathcal{I}}, \rho \models y$, for any letter y. In the following, we shall write $\omega_\rho(Z_i)$ for $\omega_\rho(z_i^1), \ldots, \omega_\rho(z_i^{j_i})$. In particular, if ρ starts from some state w_{x_i} or $\overline{w_{x_i}}$, and satisfies $s \wedge \neg t$ (that is, it reaches the collector state s_0 without visiting any node $\overline{s_j}$, for $1 \leq j \leq i$), ω_ρ fulfills the following conditions: for $1 \leq m \leq i$,

- if $w_{x_m} \in \mathrm{states}(\rho)$, then $\omega_\rho(x_m) = \top$, and if $\overline{w_{x_m}} \in \mathrm{states}(\rho)$, then $\omega_\rho(x_m) = \bot$;
- for $1 \leq u_m \leq j_m$, if $w_{z_m^{u_m}} \in \mathrm{states}(\rho)$, then $\omega_\rho(z_m^{u_m}) = \top$, and if $\overline{w_{z_m^{u_m}}} \in \mathrm{states}(\rho)$, then $\omega_\rho(z_m^{u_m}) = \bot$.

It easily follows that $\mathcal{K}_{\mathcal{I}}, \rho \models F_m(x_1, \ldots, x_{m-1}, Z_m)$ if and only if $F_m(\omega_\rho(x_1), \ldots, \omega_\rho(x_{m-1}), \omega_\rho(Z_m)) = \top$. Finally, let $\mathcal{F}_{\mathcal{I}} = \{\psi_k \mid 0 \leq k \leq n+1\}$ be the set of formulas defined as:

$$\psi_0 = \bot \text{ and, for } k \geq 1,$$

$$\psi_k = \langle \mathrm{A} \rangle \underbrace{\left[\begin{array}{c} (s \wedge \neg t) \wedge \bigwedge_{i=1}^{n} \Big((x_i \wedge \neg r_i) \rightarrow F_i(x_1, \ldots, x_{i-1}, Z_i) \Big) \\ \wedge \\ [\mathrm{B}] \Big(\Big(\bigvee_{i=1}^{n} \langle \mathrm{A} \rangle p_{\overline{x_i}} \Big) \rightarrow \langle \mathrm{A} \rangle \left(\neg s \wedge \mathit{length}_2 \wedge \langle \mathrm{A} \rangle (\mathit{length}_2 \wedge \neg \psi_{k-1}) \right) \Big) \end{array} \right]}_{\varphi_k},$$

where $\mathit{length}_2 = \langle \mathrm{B} \rangle \top \wedge [\mathrm{B}][\mathrm{B}] \bot$ (Example 2.1.7) is satisfied only by traces of length 2. The first conjunct of φ_k, i.e, $s \wedge \neg t$, forces the trace to reach the collector

state s_0, without visiting any state $\overline{s_j}$. The second conjunct checks that if the trace assigns the truth value $\top$ to x_m passing through w_{x_m}, with $1 \leq m \leq n$, then $F_m(x_1, \ldots, x_{m-1}, Z_m)$ is satisfied by ω_ρ (which amounts to say that the SAT problem connected with Z_m has a positive answer, for the selected values of $x_1, \ldots, x_{m-1}$). Conversely, the third conjunct ensures that if the trace assigns the truth value $\bot$ to some x_m by passing through $\overline{w_{x_m}}$, then, intuitively, the SAT problem connected with Z_m has no assignment satisfying $F_m(x_1, \ldots, x_{m-1}, Z_m)$. As a matter of fact, if ρ satisfies φ_k, for some $k \geq 2$, and assigns $\bot$ to x_m, then there is a prefix $\tilde{\rho}$ of ρ ending in $\overline{w_{x_m}}$. Since $\bigvee_{i=1}^{n} \langle \mathrm{A} \rangle p_{\overline{x_i}}$ is satisfied by $\tilde{\rho}$, $\langle \mathrm{A} \rangle \big(\neg s \wedge \mathit{length}_2 \wedge \langle \mathrm{A} \rangle(\mathit{length}_2 \wedge \neg\psi_{k-1})\big)$ must be satisfied as well. The only possibility is that the trace $\overline{s_m} \cdot w_{x_m}$ does not model ψ_{k-1}, as $\overline{w_{x_m}} \cdot \overline{s_m}$ has to model $\langle \mathrm{A} \rangle(\mathit{length}_2 \wedge \neg\psi_{k-1})$. However, since $\psi_{k-1} = \langle \mathrm{A} \rangle \varphi_{k-1}$, this holds if and only if $\mathcal{K}, w_{x_m} \not\models \psi_{k-1}$.

The following theorem states the correctness of the construction. Its proof can be found in Appendix C.3.

Theorem 5.4.2. *Let $\mathcal{I}$ be an instance of SNSAT, with $|\mathcal{I}| = n$, and let $\mathcal{K}_\mathcal{I}$ and $\mathcal{F}_\mathcal{I}$ be defined as above. For all $0 \leq k \leq n+1$ and all $r = 1, \ldots, n$, it holds that:*

1. *if $k \geq r$, then $v_\mathcal{I}(x_r) = \top \iff \mathcal{K}_\mathcal{I}, w_{x_r} \models \psi_k$;*
2. *if $k \geq r+1$, then $v_\mathcal{I}(x_r) = \bot \iff \mathcal{K}_\mathcal{I}, \overline{w_{x_r}} \models \psi_k$.*

The correctness of the reduction from SNSAT to MC for AB follows.

Corollary 5.4.3. *Let $\mathcal{I}$ be an instance of SNSAT, with $|\mathcal{I}| = n$, and let $\mathcal{K}_\mathcal{I}$ and $\mathcal{F}_\mathcal{I}$ be defined as above. Then, $v_\mathcal{I}(x_n) = \top \iff \mathcal{K}_\mathcal{I} \models [\mathrm{B}] \bot \rightarrow \psi_n$.*

Proof. By Theorem 5.4.2, $v_\mathcal{I}(x_n) = \top \iff \mathcal{K}_\mathcal{I}, w_{x_n} \models \psi_n$. If $v_\mathcal{I}(x_n) = \top$ then $\mathcal{K}_\mathcal{I}, w_{x_n} \models \psi_n$ and, since w_{x_n} is the only initial trace satisfying $[\mathrm{B}] \bot$ (this formula is satisfied by traces having length equal to 1 only), $\mathcal{K}_\mathcal{I} \models [\mathrm{B}] \bot \rightarrow \psi_n$. Conversely, if $\mathcal{K}_\mathcal{I} \models [\mathrm{B}] \bot \rightarrow \psi_n$, then $\mathcal{K}_\mathcal{I}, w_{x_n} \models \psi_n$, allowing us to conclude that $v_\mathcal{I}(x_n) = \top$. □

Eventually we can state the complexity of the problem.

Corollary 5.4.4. *The MC problem for* AB *formulas over finite Kripke structures is $\mathbf{P}^{\mathbf{NP}}$-hard (under polynomial-time reductions).*

Proof. The result follows from Corollary 5.4.3 considering that, for an instance of SNSAT $\mathcal{I}$, with $|\mathcal{I}| = n$, $\mathcal{K}_\mathcal{I}$ and $\psi_n \in \mathcal{F}_\mathcal{I}$ have a size polynomial in n and in the length of the formulas of $\mathcal{I}$. □

We can prove the same complexity lower bound for the symmetric fragment $\overline{\mathrm{AE}}$, just by transposing the edges of $\mathcal{K}_\mathcal{I}$, and by replacing [B] with [E] and $\langle \mathrm{A} \rangle$

with $\langle\overline{\mathsf{A}}\rangle$ in the definition of ψ_n. This hardness result immediately propagates to the bigger fragments $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}$ and $\mathsf{A}\overline{\mathsf{A}}\mathsf{E}$.

Finally, we summarize the $\mathbf{P}^{\mathbf{NP}}$-completeness results that can be obtained by putting together Corollary 5.2.5 in Section 5.2 and Corollary 5.4.4.

Corollary 5.4.5. *The MC problem for* AB, $\overline{\mathsf{A}}\mathsf{E}$, $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}$, *and* $\mathsf{A}\overline{\mathsf{A}}\mathsf{E}$ *formulas over finite Kripke structures is* $\boldsymbol{P}^{\boldsymbol{NP}}$*-complete.*

In the next section, we will prove a different complexity lower bound for the fragments A, $\overline{\mathsf{A}}$, $\mathsf{A}\overline{\mathsf{A}}$, $\overline{\mathsf{A}}\mathsf{B}$, and AE, to which the present one does not apply.

5.5 $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$-hardness of MC for A and $\overline{\mathsf{A}}$

In this section, we prove that MC for formulas of the fragment A (and of $\overline{\mathsf{A}}$, respectively) over finite Kripke structures is $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$-hard, by reducing to it *PARITY(SAT)* [Wag87], a problem complete for $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$. PARITY(SAT) is to decide, for a set of Boolean formulas Γ, if the number of *satisfiable* formulas in Γ is odd or even. Hardness of MC for A and $\overline{\mathsf{A}}$ immediately propagates to $\mathsf{A}\overline{\mathsf{A}}$, AE, $\mathsf{A}\overline{\mathsf{A}}$, $\overline{\mathsf{A}}\mathsf{B}$.

Let Γ be a set of n Boolean formulas $\{\phi_i(x_1^i, \ldots, x_{m_i}^i) \mid 1 \leq i \leq n,\ m_i \in \mathbb{N}\}$. We provide a Kripke structure $\mathcal{K}_{PAR}^{\Gamma}$ and an A-formula Φ_Γ such that $\mathcal{K}_{PAR}^{\Gamma} \models \Phi_\Gamma$ if and only if the number of satisfiable Boolean formulas in Γ is *odd*.

We start by defining a Boolean formula, parity(F, Z), over two sets of Boolean variables, $F = \{f_1, \ldots, f_n\}$ and $Z = \{z_1, \ldots, z_t\}$, with $t = 3 \cdot (n-1) + 1$. Such a formula allows one to decide the parity of the number of variables in F that evaluate to $\top$. Z is a set of auxiliary variables, whose truth values are *functionally determined* by those assigned to the variables in F. Given a truth assignment, the number of variables in F set to $\top$ is even if parity(F, Z) evaluates to $\top$, and, in particular, its last variable z_t evaluates to $\top$. The formula parity(F, Z) is defined as follows:

$$\text{parity}(f_1, \ldots, f_n, z_1, \ldots, z_t) = z_t \wedge \text{par}_n(f_1, \ldots, f_n, z_1, \ldots, z_t),$$

where $t = 3 \cdot (n-1) + 1$ and, for $i \geq 1$, $\text{par}_i(f_1, f_2, \ldots, f_i, z_1, \ldots, z_{3(i-1)+1})$ is inductively defined as:

$$\text{par}_1(f_1, z_1) = \neg f_1 \leftrightarrow z_1,$$

and, for all $i \geq 2$,

$$\begin{aligned}\text{par}_i(f_1, f_2, \ldots, f_i, z_1, \ldots, z_{\alpha+3}) = \\ \big(z_{\alpha+1} \leftrightarrow (f_i \wedge \neg z_\alpha)\big) \wedge \big(z_{\alpha+2} \leftrightarrow (\neg f_i \wedge z_\alpha)\big) \wedge \big(z_{\alpha+3} \leftrightarrow (z_{\alpha+2} \vee z_{\alpha+1})\big) \wedge \\ \text{par}_{i-1}(f_1, f_2, \ldots, f_{i-1}, z_1, \ldots, z_\alpha),\end{aligned}$$

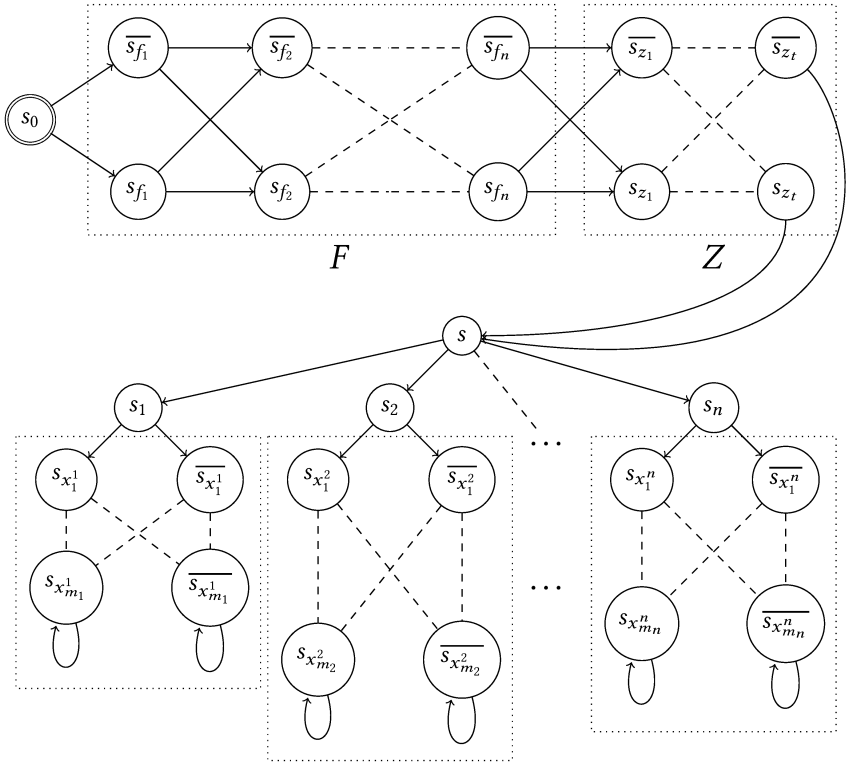


Figure 5.6: The Kripke structure $\mathcal{K}^{\Gamma}_{PAR}$.

with $\alpha = 3 \cdot (i-2) + 1$.

Each assignment satisfying par_i has to set z_α to the parity value for the set of Boolean variables $f_1, f_2, \ldots, f_{i-1}$. Such a value is then possibly changed according to the truth of f_i and "assigned" to $z_{\alpha+3}$. Note that the length of parity(F, Z) is polynomial in n.

We now show how to build the Kripke structure $\mathcal{K}^{\Gamma}_{PAR}$ depicted in Figure 5.6, such that a subset of its traces encode all the possible truth assignments to the variables of $F \cup Z$ and to all the variables occurring in formulas of Γ. $\mathcal{K}^{\Gamma}_{PAR}$ features a pair of states for each Boolean variable in $F \cup Z$ as well as for all the variables of formulas in Γ (one state for each truth value). Each path from the initial state s_0 to the state s represents a truth assignment to the variables in $F \cup Z$. Then, the structure branches into n substructures, each one modeling the possible truth assignments to the variables of a formula in Γ.

Formally, $\mathcal{K}^{\Gamma}_{PAR} = (\mathcal{AP}, S, R, \mu, s_0)$, where

- $\mathcal{AP} = \{p, q\} \cup F \cup Z \cup \{aux_i \mid 1 \leq i \leq n\} \cup \{x^i_{j_i} \mid 1 \leq i \leq n,\ 1 \leq j_i \leq m_i\}$,
- $S = \{s_0\} \cup \{s_{f_i}, \overline{s_{f_i}} \mid 1 \leq i \leq n\} \cup \{s_z, \overline{s_z} \mid z \in Z\} \cup \{s\} \cup \{s_i \mid 1 \leq i \leq n\} \cup \{s_{x^i_{j_i}}, \overline{s_{x^i_{j_i}}} \mid 1 \leq i \leq n,\ 1 \leq j_i \leq m_i\}$,
- $R = \{(s_0, s_{f_1}), (s_0, \overline{s_{f_1}})\} \cup \{(s_{f_i}, s_{f_{i+1}}), (s_{f_i}, \overline{s_{f_{i+1}}}), (\overline{s_{f_i}}, s_{f_{i+1}}), (\overline{s_{f_i}}, \overline{s_{f_{i+1}}}) \mid 1 \leq i < n\} \cup \{(s_{f_n}, s_{z_1}), (\overline{s_{f_n}}, s_{z_1}), (s_{f_n}, \overline{s_{z_1}}), (\overline{s_{f_n}}, \overline{s_{z_1}})\} \cup \{(s_{z_i}, s_{z_{i+1}}), (s_{z_i}, \overline{s_{z_{i+1}}}), (\overline{s_{z_i}}, s_{z_{i+1}}), (\overline{s_{z_i}}, \overline{s_{z_{i+1}}}) | 1 \leq i < t\} \cup \{(s_{z_t}, s), (\overline{s_{z_t}}, s)\} \cup \{(s, s_i), (s_i, s_{x^i_1}), (s_i, \overline{s_{x^i_1}}) \mid 1 \leq i \leq n\} \cup \{(s_{x^i_{j_i}}, s_{x^i_{j_i+1}}), (\overline{s_{x^i_{j_i}}}, s_{x^i_{j_i+1}}), (\overline{s_{x^i_{j_i}}}, s_{x^i_{j_i+1}}), (\overline{s_{x^i_{j_i}}}, \overline{s_{x^i_{j_i+1}}}) | 1 \leq i \leq n,\ 1 \leq j_i < m_i\} \cup \{(s_{x^i_{m_i}}, s_{x^i_{m_i}}), (\overline{s_{x^i_{m_i}}}, \overline{s_{x^i_{m_i}}}) | 1 \leq i \leq n\}$,
- and the labeling function μ is defined as follows:
  - $\mu(s_0) = \{p, q\} \cup F \cup Z$;
  - for all $1 \leq i \leq n$, $\mu(s_{f_i}) = \{p, q\} \cup F \cup Z$; $\mu(\overline{s_{f_i}}) = \{p, q\} \cup (F \setminus \{f_i\}) \cup Z$;
  - for all $z \in Z$, $\mu(s_z) = \{p, q\} \cup F \cup Z$; $\mu(\overline{s_z}) = \{p, q\} \cup F \cup (Z \setminus \{z\})$;
  - $\mu(s) = \{q\} \cup F \cup Z \cup \{aux_i \mid 1 \leq i \leq n\} \cup \{x^i_{j_i} \mid 1 \leq i \leq n,\ 1 \leq j_i \leq m_i\}$;
  - for all $1 \leq i \leq n$, $\mu(s_i) = \{aux_i\} \cup \{x^i_{j_i} \mid 1 \leq j_i \leq m_i\}$;
  - for all $1 \leq i \leq n$, $1 \leq k_i \leq m_i$, $\mu(s_{x^i_{k_i}}) = \{aux_i\} \cup \{x^i_{j_i} \mid 1 \leq j_i \leq m_i\}$; $\mu(\overline{s_{x^i_{k_i}}}) = \{aux_i\} \cup \{x^i_{j_i} \mid 1 \leq j_i \leq m_i\} \setminus \{x^i_{k_i}\}$.

According to the definition of $\mathcal{K}^{\Gamma}_{PAR}$, it holds that:

1. each trace ρ from s_0 to s encodes a truth assignment to the proposition letters in $F \cup Z$ (for all $y \in F \cup Z$, y is $\top$ in ρ if and only if $y \in \bigcap_{w \in \mathrm{states}(\rho)} \mu(w)$). Conversely, for each truth assignment to the proposition letters in $F \cup Z$, there exists an initial trace ρ, reaching the state s, encoding such an assignment. Note that, among the initial traces, the ones leading to s are exactly those satisfying $q \wedge \neg p$.
2. An initial trace leading to s satisfies parity(F, Z) if the induced assignment sets an even number of f_i's to $\top$, and every $z \in Z$ to the truth value which is functionally implied by the values of the f_i's.
3. A trace $\tilde{\rho}$ starting from s and ending in a state s, s_i, $s_{x^i_j}$ or $\overline{s_{x^i_j}}$, with $1 \leq i \leq n$ and $1 \leq j \leq m_i$, encodes a truth assignment to the proposition letters $x_1, \ldots, x_{m_i}$ (if the trace ends in s or s_i, all the variables are assigned to $\top$; if it ends in $s_{x^i_j}$ or $\overline{s_{x^i_j}}$, in particular all the variables $x^i_{j+1}, \ldots, x^i_{m_i}$ are assigned to $\top$, by homogeneity).
4. A Boolean formula $\phi_i(x^i_1, \ldots, x^i_{m_i}) \in \Gamma$ is satisfiable if and only if there exists a trace $\tilde{\rho}$ starting from s and ending in a state s, s_i, $s_{x^i_j}$ or $\overline{s_{x^i_j}}$, for some $j = 1, \ldots, m_i$, such that $\mathcal{K}^{\Gamma}_{PAR}, \tilde{\rho} \models \phi_i(x^i_1, \ldots, x^i_{m_i})$.

Finally, let us consider the A formula

$$\psi = q \wedge \neg p \wedge \text{parity}(F, Z) \wedge \bigwedge_{i=1}^{n} \big(f_i \leftrightarrow \langle \text{A} \rangle (aux_i \wedge \phi_i(x_1^i, \ldots, x_{m_i}^i))\big).$$

In view of the above observations, ψ is satisfied by an initial trace $\overline{\rho}$ if (and only if) (i) $\overline{\rho}$ leads to s, (ii) $\overline{\rho}$ induces an assignment which sets an even number of f_i's to $\top$ and all $z \in Z$ accordingly, and (iii) for all $1 \leq i \leq n$, f_i is $\top$ if and only if there exists a trace $\tilde{\rho}$ starting from s and ending in a state s, s_i, $s_{x_j^i}$ or $\overline{s_{x_j^i}}$, such that $\mathcal{K}^{\Gamma}_{PAR}, \tilde{\rho} \models \phi_i(x_1^i, \ldots, x_{m_i}^i)$. The length of ψ is polynomial in the input size.

Let us now assume we are given an instance of PARITY(SAT) Γ with an *even* number of satisfiable Boolean formulas. Then, there exists an initial trace ρ ending in s such that, for all i, $s_{f_i} \in \text{states}(\rho)$ if $\phi_i(x_1^i, \ldots, x_{m_i}^i)$ is satisfiable, and $\overline{s_{f_i}} \in \text{states}(\rho)$ otherwise. Moreover, ρ can be chosen in such a way that $\mathcal{K}^{\Gamma}_{PAR}, \rho \models \text{parity}(F, Z)$. It immediately follows that, for all i, $\mathcal{K}^{\Gamma}_{PAR}, \rho \models f_i$ if and only if $\mathcal{K}^{\Gamma}_{PAR}, \rho \models \langle \text{A} \rangle (aux_i \wedge \phi_i(x_1^i, \ldots, x_{m_i}^i))$, concluding that $\mathcal{K}^{\Gamma}_{PAR}, \rho \models \psi$.

Conversely, let ρ be an initial trace such that $\mathcal{K}^{\Gamma}_{PAR}, \rho \models \psi$. It holds that ρ ends in s and sets an even number of f_i's to $\top$. Furthermore, if $\mathcal{K}^{\Gamma}_{PAR}, \rho \models f_i$, then there exists $\tilde{\rho}$ starting from s and ending in s, s_i, $s_{x_j^i}$ or $\overline{s_{x_j^i}}$, such that $\mathcal{K}^{\Gamma}_{PAR}, \tilde{\rho} \models \phi_i(x_1^i, \ldots, x_{m_i}^i)$, hence $\phi_i(x_1^i, \ldots, x_{m_i}^i)$ is satisfiable. If $\mathcal{K}^{\Gamma}_{PAR}, \rho \models \neg f_i$, then there exists no $\tilde{\rho}$ starting from s and ending in s, s_i, $s_{x_j^i}$ or $\overline{s_{x_j^i}}$, such that $\mathcal{K}^{\Gamma}_{PAR}, \tilde{\rho} \models \phi_i(x_1^i, \ldots, x_{m_i}^i)$. Thus $\phi_i(x_1^i, \ldots, x_{m_i}^i)$ is unsatisfiable. Hence, Γ contains an even number of satisfiable formulas.

Therefore we have proved that the number of *satisfiable* Boolean formulas of Γ is *even* if and only if there exists an initial trace ρ such that $\mathcal{K}^{\Gamma}_{PAR}, \rho \models \psi$. This amounts to say that Γ contains an *odd* number of satisfiable Boolean formulas (the PARITY(SAT) problem) if and only if $\mathcal{K}^{\Gamma}_{PAR} \models \Phi_{\Gamma}$, where $\Phi_{\Gamma} = \neg\psi$ (the MC problem). The next theorem immediately follows.

Theorem 5.5.1. *The MC problem for* A *formulas over finite Kripke structures is* $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$*-hard (under polynomial-time reductions).*

A similar proof can be given for $\overline{\text{A}}$ (roughly speaking, we replace all the occurrences of $\langle \text{A} \rangle$ in Φ_{Γ} by $\langle \overline{\text{A}} \rangle$, and we stick the n substructures of $\mathcal{K}^{\Gamma}_{PAR}$—after transposing all their edges—on s_0, instead of s).

Finally, we observe that the $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$-hardness of $\overline{\text{A}}$ and A immediately propagates to A$\overline{\text{A}}$, $\overline{\text{A}}$B and AE, yielding, together with Corollary 5.3.5 and Theorem 5.3.6, the following result.

Theorem 5.5.2. *The MC problem for* A, $\overline{\mathsf{A}}$, A$\overline{\mathsf{A}}$, $\overline{\mathsf{A}}$B *and* AE *formulas over finite Kripke structures is in* $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$ *and it is hard for* $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$.

It is still an open issue if MC for the above fragments can be solved by $o(\log^2 n)$ queries to the **NP** oracle, or it is possible to prove a stronger lower bound, or both (e.g., the problem may be complete for $\mathbf{P}^{\mathbf{NP}[O(\log n \log\log n)]}$). As a matter of fact, any attempt to reduce $\mathrm{TB(SAT)}_{1\times M}$ *to* MC for A, AE, or A$\overline{\mathsf{A}}$ has failed, because in such "reduction" we need an HS formula of length $\Theta(n^{\log n})$, which clearly cannot be generated in polynomial time.

5.6 Conclusions

In this chapter, we have proved that the fragments AB, $\overline{\mathsf{A}}$E, A$\overline{\mathsf{A}}$B, and A$\overline{\mathsf{A}}$E are complete for $\mathbf{P}^{\mathbf{NP}}$, thus joining other (point-based) temporal logics—e.g., CTL$^+$, FCTL, and ECTL$^+$—whose MC problem is complete for that class [LMS01] as well. In addition, we have shown that MC for A, $\overline{\mathsf{A}}$, A$\overline{\mathsf{A}}$, $\overline{\mathsf{A}}$B, and AE has a lower complexity, placed in between $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$ and $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$. This result has been proved by reducing MC to $\mathrm{TB(SAT)}_{1\times M}$, the problem of deciding the output value of a complex circuit, where some gates feature an **NP** oracle.

Both the MC algorithms we propose can be efficiently implemented in practice by means of a polynomial-time procedure which iteratively invokes a SAT-solver, whose extreme efficiency can be "imported" in a straightforward way: the procedure just generates some suitable Boolean formulas, feeds the SAT-solver, and stores the results. The modular and repetitive structure of the required Boolean formulas allows us to efficiently generate them and also to exploit the *warm-restart* feature (or *incrementality*) of SAT-solvers to quickly solve formulas following a common structural pattern.

In the next chapter—as we anticipated in the introduction—we will (mostly) put aside complexity issues on HS MC, and, conversely, focus on expressiveness: we will compare the expressive power of three different semantic versions of HS among themselves, and to the standard point-based temporal logics CTL, LTL and CTL*.

6 Interval vs. point temporal logic MC: an expressiveness comparison

The references for this chapter are [Boz+18e; Boz+16c].

Contents

6.1 Preliminaries . . . 125
6.1.1 Standard point-based temporal logics . . . 125
6.1.2 Three semantic variants of HS for MC . . . 126
6.1.3 An example: a vending machine . . . 129
6.2 Equivalence between LTL and HS_{lin} . . . 132
6.3 A characterization of HS_{ct} . . . 135
6.3.1 From finitary CTL* to HS_{ct} . . . 135
6.3.2 From HS_{ct} to finitary CTL* . . . 139
6.4 Expressiveness comparison of HS_{lin}, HS_{st}, HS_{ct} . . . 146
6.5 Conclusions . . . 149

The MC methodology mainly considers two types of point-based temporal logics (PTLs) as the property specification language—*linear* and *branching*—which differ in the underlying model of time. In linear PTLs, such as LTL [Pnu77], each moment in time has a unique possible future: formulas are interpreted over (infinite) *paths* of a Kripke structure, and thus they refer to a single computation of a system. In branching PTLs, such as CTL and CTL* [EH86], each moment in time may evolve into several possible futures: formulas are interpreted over *states* of the Kripke structure, hence referring to all the possible system computations.

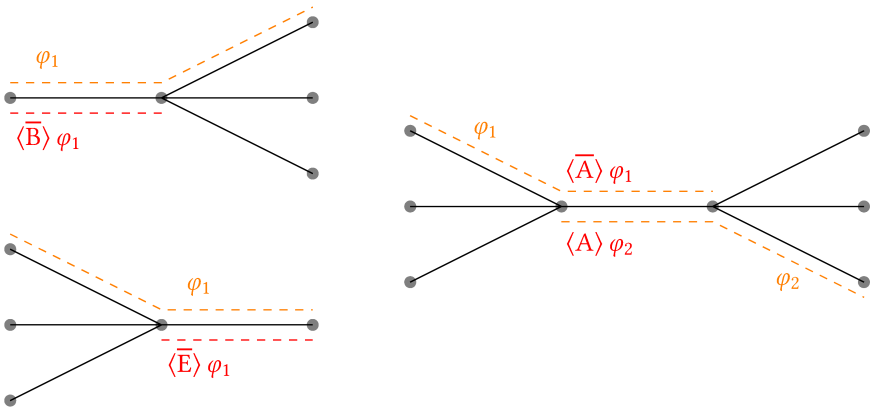


Figure 6.1: State-based semantic variant HS_{st}: past and future are branching

In the previous chapters we have assumed a *state-based* semantics for HS, which induces a branching reference both in the future and in the past: intervals/traces are "forgetful" of the history leading to their initial state, and the initial (resp., final) state of an interval may feature several predecessors (resp., successors). A graphical account of the state-based semantics can be found in Figure 6.1; a detailed explanation will be given in the following.

However, as we already said, HS MC has been simultaneously and independently studied also by Lomuscio and Michaliszyn in [LM13; LM14; LM16]: there, the considered HS fragments are interpreted over the unwinding of a Kripke structure (*computation-tree-based* semantics—see Figure 6.2). This induces a linear reference in the past (the initial state of an interval may feature only one predecessor), but branching in the future (the final state features several successors). Moreover, the computation history is never forgotten and increases with time.

In this chapter, we study the expressiveness of HS, in the context of MC, in comparison with that of the standard PTLs LTL, CTL, and CTL*. The analysis is carried on *enforcing the homogeneity assumption*. We prove that HS endowed with the state-based semantics (hereafter denoted as HS_{st}) is not comparable with LTL, CTL, and CTL*. On one hand, the result supports the intuition that HS_{st} gains some expressiveness by the ability of branching in the past. On the other hand, HS_{st} does not feature the possibility of forcing the verification of a property over an infinite path, thus implying that the formalisms are not comparable. With the aim of having a more "effective" comparison base, we consider two additional semantic variants of HS, namely, the *computation-tree-based semantic variant* (denoted as HS_{ct}) and the *trace-based* one (HS_{lin}).

The state-based (see Figure 6.1) and computation-tree-based (see Figure 6.2) approaches rely on a *branching*-time setting and differ in the nature of past. In the

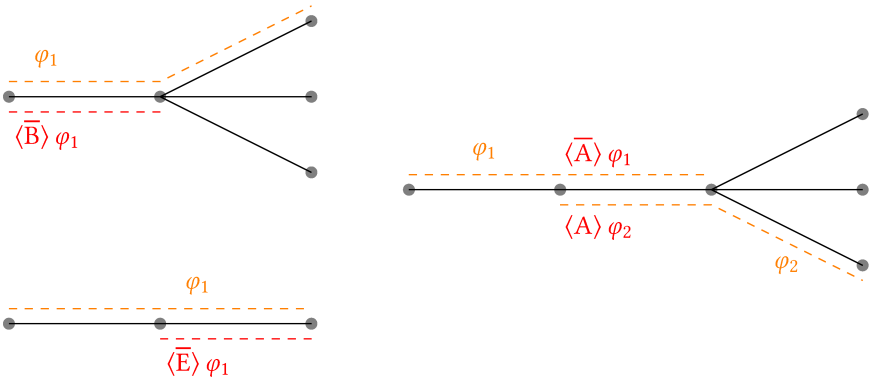


Figure 6.2: Computation-tree-based semantic variant $\mathrm{HS}_{\mathrm{ct}}$: future is branching, past is linear, finite and cumulative.

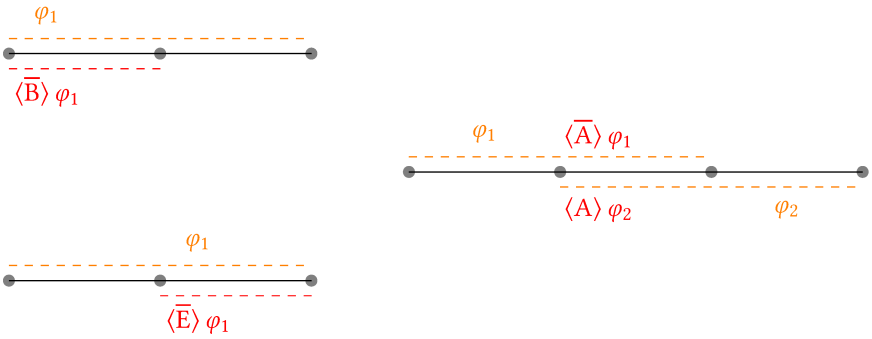


Figure 6.3: Trace-based semantic variant $\mathrm{HS}_{\mathrm{lin}}$: neither past nor future are branching

latter approach, past is *linear*: each interval may have several possible futures, but only a unique past. Moreover, past is assumed to be *finite* and *cumulative*, that is, the story of the current situation increases with time, and is never forgotten. The trace-based approach relies on a *linear*-time setting (see Figure 6.3), where the infinite paths (computations) of the given Kripke structure are the main semantic entities. Branching is neither allowed in the past nor in the future. Note that the linear-past (rather than branching) approach is more suited to the specification of dynamic behaviors, because it considers states in a computation tree, while the branching-past approach considers machine states, where past is not very meaningful for the specification of behavioral constraints [LS95].

The variant $\mathsf{HS}_{\mathsf{ct}}$ is a natural candidate for an expressiveness comparison with the branching time logics CTL and CTL*. The most interesting and technically involved result is the characterization of the expressive power of $\mathsf{HS}_{\mathsf{ct}}$: $\mathsf{HS}_{\mathsf{ct}}$ turns out to be expressively equivalent to finitary CTL*, that is, the variant of CTL* with quantification over *finite* paths. As for CTL, a non comparability result can be stated. The variant $\mathsf{HS}_{\mathsf{lin}}$ is a natural candidate for an expressiveness comparison with LTL: we prove that $\mathsf{HS}_{\mathsf{lin}}$ and LTL are equivalent (this result holds true even for a very small fragment of $\mathsf{HS}_{\mathsf{lin}}$), but the former is at least exponentially more succinct than the latter.

We complete the picture with a comparison of the three semantic variants $\mathsf{HS}_{\mathsf{st}}$, $\mathsf{HS}_{\mathsf{ct}}$, and $\mathsf{HS}_{\mathsf{lin}}$. We show that, as expected, $\mathsf{HS}_{\mathsf{lin}}$ is not comparable with either of the branching versions, $\mathsf{HS}_{\mathsf{ct}}$ and $\mathsf{HS}_{\mathsf{st}}$. The interesting result is that, on the other hand, $\mathsf{HS}_{\mathsf{ct}}$ is strictly included in $\mathsf{HS}_{\mathsf{st}}$: this supports $\mathsf{HS}_{\mathsf{st}}$ as a reasonable and adequate semantic choice.

The complete picture of the expressiveness results is reported in Figure 6.4 (the symbols $\neq$, $\equiv$, and $<$ denote incomparability, equivalence, and strict inclusion, respectively).

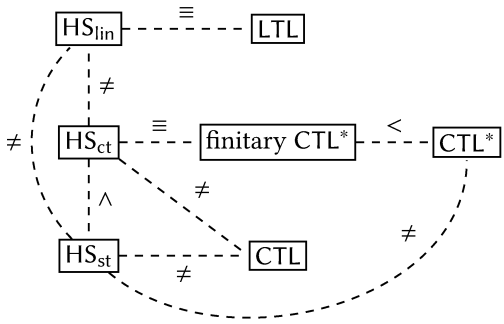


Figure 6.4: Overview of the expressiveness results.

Organization of the chapter.

- In the next section we start with some preliminaries; in particular, in Section 6.1.1 we recall the well-known PTLs LTL, CTL and CTL*; in Section 6.1.2 we define the three semantic variants of HS ($\mathsf{HS}_{\mathsf{st}}$, $\mathsf{HS}_{\mathsf{ct}}$ and $\mathsf{HS}_{\mathsf{lin}}$). In Section 6.1.3 we provide a detailed example which gives an intuitive account of the three semantic variants and highlights their differences.
- In the next three sections we analyze and compare the expressiveness of these logics. In Section 6.2 we show the expressive equivalence of LTL and $\mathsf{HS}_{\mathsf{lin}}$. Then, in Section 6.3 we prove the equivalence of $\mathsf{HS}_{\mathsf{ct}}$ and finitary CTL*. In Section 6.4 we compare the expressiveness of $\mathsf{HS}_{\mathsf{st}}$, $\mathsf{HS}_{\mathsf{ct}}$ and $\mathsf{HS}_{\mathsf{lin}}$.

6.1 Preliminaries

In the following, let Σ be an alphabet and w be a non-empty finite or infinite word over Σ. We denote by $|w|$ the length of w ($|w| = \infty$ if w is infinite). For all $i, j \in \mathbb{N}$ with $i \leq j$, $w(i)$ denotes the i-th letter of w, while $w(i, j)$ denotes the finite subword of w given by $w(i) \cdots w(j)$. The set of all the finite words over Σ is denoted by Σ^*, and Σ^+ represents $\Sigma^* \setminus \{\varepsilon\}$, where ε is the empty word.

Clearly, a trace ρ of a Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ can be considered as a finite word over S, where $(\rho(i), \rho(i+1)) \in R$ for all $0 \leq i < |\rho| - 1$. In addition to traces, we define an *infinite path* π of $\mathcal{K}$ as an infinite word over S such that $(\pi(i), \pi(i+1)) \in R$ for all $i \geq 0$. Here, unlike the previous chapters, we assume traces and paths to be 0-based (and not 1-based) for notational convenience.

6.1.1 Standard point-based temporal logics

We start now by recalling the standard propositional temporal logics CTL*, CTL and LTL [EH86; Pnu77]. Given a set of proposition letters $\mathcal{AP}$, the formulas φ of CTL* are defined as follows:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi \mid \exists\varphi,$$

where $p \in \mathcal{AP}$, X and U are the "next" and "until" temporal modalities, and $\exists$ is the existential path quantifier.[1] We also use the standard shorthands $\forall\varphi = \neg\exists\neg\varphi$ ("universal path quantifier"), $\mathsf{F}\varphi = \top \mathsf{U} \varphi$ ("eventually" or "in the future") and its dual $\mathsf{G}\varphi = \neg\mathsf{F}\neg\varphi$ ("always" or "globally"). We denote by $|\varphi|$ the size of φ, that is, the number of its subformulas. The logic CTL is the fragment of CTL* where each temporal modality is immediately preceded by a path quantifier, whereas LTL corresponds to the path-quantifier-free fragment of CTL*.

[1] Hereafter, we denote by $\exists/\forall$ the existential/universal path quantifiers (instead of by the usual E/A), in order not to confuse them with the HS modalities E/A.

Given a Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$, an infinite path π of $\mathcal{K}$, and a position $i \geq 0$ along π, the satisfaction relation $\mathcal{K}, \pi, i \models \varphi$ for CTL*, written simply $\pi, i \models \varphi$ when $\mathcal{K}$ is clear from the context, is defined as follows (Boolean connectives are treated as usual):

$$\begin{array}{ll}
\pi, i \models p & \Leftrightarrow p \in \mu(\pi(i)), \\
\pi, i \models \mathsf{X}\varphi & \Leftrightarrow \pi, i+1 \models \varphi, \\
\pi, i \models \varphi_1 \mathsf{U} \varphi_2 & \Leftrightarrow \text{for some } j \geq i : \pi, j \models \varphi_2, \text{ and } \pi, k \models \varphi_1 \text{ for all } i \leq k < j, \\
\pi, i \models \exists\varphi & \Leftrightarrow \text{for some infinite path } \pi' \text{ starting from } \pi(i), \pi', 0 \models \varphi.
\end{array}$$

The MC problem is defined as follows: $\mathcal{K}$ is a model of φ, written $\mathcal{K} \models \varphi$, if for all initial infinite paths π of $\mathcal{K}$, it holds that $\mathcal{K}, \pi, 0 \models \varphi$.

We also consider a variant of CTL*, called *finitary* CTL*, where the path quantifier $\exists$ of CTL* is replaced by the finitary path quantifier $\exists_f$. In this setting, path quantification ranges over the traces starting from the current state. The satisfaction relation $\rho, i \models \varphi$, where ρ is a trace and i is a position along ρ, is similar to that given for CTL* with the only difference of finiteness of paths, and the fact that, for a formula $\mathsf{X}\varphi$, we have $\rho, i \models \mathsf{X}\varphi$ if and only if $i + 1 < |\rho|$ and $\rho, i+1 \models \varphi$. A Kripke structure $\mathcal{K}$ is a model of a finitary CTL* formula if, for each initial trace ρ of $\mathcal{K}$, it holds that $\mathcal{K}, \rho, 0 \models \varphi$.

The MC problem for both CTL* and LTL is **PSPACE**-complete [EL85; SC85]. It is not difficult to show that, as it happens with finitary LTL [DV13], MC for finitary CTL* is **PSPACE**-complete as well.

6.1.2 Three semantic variants of HS for MC

In this section we formally define the three presented variants of HS semantics $\mathsf{HS}_{\mathsf{st}}$ (state-based), $\mathsf{HS}_{\mathsf{ct}}$ (computation-tree-based), and $\mathsf{HS}_{\mathsf{lin}}$ (trace-based) for model checking HS formulas against Kripke structures. For each variant, the related (finite) MC problem consists in deciding whether or not a finite Kripke structure is a model of an HS formula under such a semantic variant, as defined in the following.

State-based variant $\mathsf{HS}_{\mathsf{st}}$. Let us recall the state-based variant, which is basically the one we have been assuming so far, where an abstract interval model (Definition 1.3.1) is naturally associated with a given Kripke structure $\mathcal{K}$ by considering the set of intervals as the set $\mathrm{Trc}_{\mathcal{K}}$ of traces of $\mathcal{K}$.

Definition 6.1.1 (Abstract interval model induced by a Kripke structure, Definition 2.1.4)**.** The abstract interval model induced by a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ is $\mathcal{A}_{\mathcal{K}} = (\mathcal{AP}, \mathbb{I}, B_{\mathbb{I}}, E_{\mathbb{I}}, \sigma)$, where

- $\mathbb{I} = \mathrm{Trc}_{\mathcal{K}}$,
- $B_{\mathbb{I}} = \{(\rho, \rho') \in \mathbb{I} \times \mathbb{I} \mid \rho' \in \mathrm{Pref}(\rho)\}$,

- $E_{\mathbb{I}} = \{(\rho, \rho') \in \mathbb{I} \times \mathbb{I} \mid \rho' \in \mathrm{Suff}(\rho)\}$, and
- $\sigma : \mathbb{I} \longrightarrow 2^{\mathcal{AP}}$ such that, for all $\rho \in \mathbb{I}$,

$$\sigma(\rho) = \bigcap_{s \in \mathrm{states}(\rho)} \mu(s).$$

Definition 6.1.2 (State-based HS—$\mathrm{HS}_{\mathrm{st}}$, Definition 2.1.5 and 2.1.6)**.** Let $\mathcal{K}$ be a finite Kripke structure and ψ be an HS formula. A trace $\rho \in \mathrm{Trc}_{\mathcal{K}}$ satisfies ψ under the state-based semantic variant, denoted as $\mathcal{K}, \rho \models_{\mathrm{st}} \psi$, if and only if $\mathcal{A}_{\mathcal{K}}, \rho \models \psi$ (Definition 1.3.2).
Moreover, $\mathcal{K}$ is a model of ψ under the state-based semantic variant, denoted as $\mathcal{K} \models_{\mathrm{st}} \psi$, if and only if, for all *initial* traces $\rho \in \mathrm{Trc}_{\mathcal{K}}$, it holds that $\mathcal{K}, \rho \models_{\mathrm{st}} \psi$.

Computation-tree-based semantic variant $\mathrm{HS}_{\mathrm{ct}}$. We now describe the computation-tree-based semantic variant: to the aim, we consider the abstract interval model *induced by the computation tree* of a Kripke structure (this will be formally defined in the following), and basically proceed as in the previous case.

We start by introducing the notion of *D-tree structure*, namely, an infinite tree-shaped Kripke structure with branches over a set D of directions.

Definition 6.1.3 (D-tree structure)**.** Given a set D of directions, a *D-tree structure* over a set of proposition letters $\mathcal{AP}$ is a Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ such that $s_0 \in D$, S is a prefix closed subset of D^+, and R is the set of pairs $(s, s') \in S \times S$ such that there exists $d \in D$ for which $s' = s \cdot d$ (note that R is completely specified by S). The states of a D-tree structure are called *nodes*.

A Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ induces an S-tree structure, called the *computation tree of* $\mathcal{K}$, denoted by $\mathcal{C}(\mathcal{K})$, which is obtained by unwinding $\mathcal{K}$ from the initial state (note that the directions are the set of states of $\mathcal{K}$). Formally, $\mathcal{C}(\mathcal{K}) = (\mathcal{AP}, \mathrm{Trc}^0_{\mathcal{K}}, R', \mu', s_0)$, where the set of nodes is the set of initial traces of $\mathcal{K}$—hereafter denoted as $\mathrm{Trc}^0_{\mathcal{K}}$—and, for all $\rho, \rho' \in \mathrm{Trc}^0_{\mathcal{K}}$, $\mu'(\rho) = \mu(\mathrm{lst}(\rho))$ and $(\rho, \rho') \in R'$ if and only if $\rho' = \rho \cdot s$ for some $s \in S$. See Figure 6.5 for an example.

Notice that since each state in a computation tree has a unique predecessor (with the exception of the initial state), this HS variant enforces a linear reference in the past.

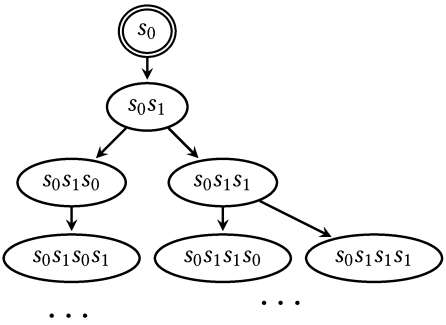


Figure 6.5: Computation tree $\mathcal{C}(\mathcal{K})$ of a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ with $S = \{s_0, s_1\}$ and $R = \{(s_0, s_1), (s_1, s_1), (s_1, s_0)\}$.

Definition 6.1.4 (Computation-tree-based HS—HS_{ct})**.** A finite Kripke structure $\mathcal{K}$ is a model of an HS formula ψ under the computation-tree-based semantic variant, denoted as $\mathcal{K} \models_{\text{ct}} \psi$, if and only if $\mathcal{C}(\mathcal{K}) \models_{\text{st}} \psi$.

Trace-based semantic variant HS_{lin}. Finally, we introduce the trace-based semantic variant, which exploits the interval structures induced by the infinite paths of a Kripke structure, as defined in the following.

We recall that, given a strict partial ordering $\mathbb{S} = (S, <)$, an *interval* in $\mathbb{S}$ is an ordered pair $[x, y]$, such that $x, y \in S$ and $x \leq y$, which represents the subset of S given by all points $z \in S$ such that $x \leq z \leq y$. We denote by $\mathbb{I}(\mathbb{S})$ the set of intervals in $\mathbb{S}$.

The following notion has already been presented in Section 3.1, but here we use (for technical convenience) a different notation.

Definition 6.1.5 (Interval structure)**.** An *interval structure (or interval model)* $\mathcal{IS}$ over $\mathcal{AP}$ is a pair $\mathcal{IS} = (\mathbb{S}, \sigma)$ such that $\mathbb{S} = (S, <)$ is a strict partial ordering and $\sigma : \mathbb{I}(\mathbb{S}) \to 2^{\mathcal{AP}}$ is a labeling function assigning a set of proposition letters to each interval in $\mathbb{S}$.

The next definition shows how to derive an abstract interval model from an interval structure, allowing us to interpret HS over interval structures.

Definition 6.1.6 (Abstract interval model induced by an interval structure)**.** An interval structure $\mathcal{IS} = (\mathbb{S}, \sigma)$ over $\mathcal{AP}$, where $\mathbb{S} = (S, <)$ is a

strict partial ordering, induces the abstract interval model

$$\mathcal{A}_{IS} = (\mathcal{AP}, \mathbb{I}(\mathbb{S}), B_{\mathbb{I}(\mathbb{S})}, E_{\mathbb{I}(\mathbb{S})}, \sigma),$$

where $[x, y]\, B_{\mathbb{I}(\mathbb{S})}\, [v, z]$ if and only if $x = v$ and $z < y$, and $[x, y]\, E_{\mathbb{I}(\mathbb{S})}\, [v, z]$ if and only if $y = z$ and $x < v$.

Given an interval I and an HS formula ψ, we write $IS, I \models \psi$ to mean that $\mathcal{A}_{IS}, I \models \psi$. We now interpret HS over the infinite paths of a Kripke structure by mapping them into interval structures.

Definition 6.1.7 (Interval structure induced by an infinite path)**.** For a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ and an infinite path $\pi = \pi(0)\pi(1)\cdots$ of $\mathcal{K}$, the *interval structure induced by* π is $IS_{\mathcal{K},\pi} = ((\mathbb{N}, <), \sigma)$, where for each interval $[i, j]$, we have $\sigma([i, j]) = \bigcap_{h=i}^{j} \mu(\pi(h))$.

Definition 6.1.8 (Trace-based HS—$\mathsf{HS}_{\mathsf{lin}}$)**.** A finite Kripke structure $\mathcal{K}$ is a model of an HS formula ψ under the trace-based semantic variant, denoted as $\mathcal{K} \models_{\mathsf{lin}} \psi$, if and only if, for each initial infinite path π and for each initial interval $[0, i]$, it holds that $IS_{\mathcal{K},\pi}, [0, i] \models \psi$.

With this definition we conclude the presentation of the three semantic variants of HS. In the next sections we will compare the expressiveness of the logics $\mathsf{HS}_{\mathsf{st}}$, $\mathsf{HS}_{\mathsf{ct}}$, $\mathsf{HS}_{\mathsf{lin}}$, LTL, CTL, and CTL* when interpreted over finite Kripke structures.

Given two logics L_1 and L_2, and two formulas $\varphi_1 \in L_1$ and $\varphi_2 \in L_2$, we say that φ_1 in L_1 is *equivalent* to φ_2 in L_2 if, for every finite Kripke structure $\mathcal{K}$, $\mathcal{K}$ is a model of φ_1 in L_1 if and only if $\mathcal{K}$ is a model of φ_2 in L_2. We say that L_2 *is subsumed by* L_1, denoted as $L_1 \geq L_2$, if for each formula $\varphi_2 \in L_2$, there exists a formula $\varphi_1 \in L_1$ such that φ_1 in L_1 is equivalent to φ_2 in L_2. Moreover, L_1 is *as expressive as* L_2 (or, L_1 and L_2 have *the same expressive power*), written $L_1 \equiv L_2$, if both $L_1 \geq L_2$ and $L_2 \geq L_1$. We say that L_1 is *(strictly) more expressive than* L_2 if $L_1 \geq L_2$ and $L_2 \not\geq L_1$. Finally, L_1 and L_2 are *expressively incomparable* if both $L_1 \not\geq L_2$ and $L_2 \not\geq L_1$.

6.1.3 An example: a vending machine

In this section, we give an example highlighting the differences among the HS semantic variants $\mathsf{HS}_{\mathsf{st}}$, $\mathsf{HS}_{\mathsf{ct}}$, and $\mathsf{HS}_{\mathsf{lin}}$.

The Kripke structure of Figure 6.6 represents a *vending machine*, which can dispense water, hot dogs, and candies. In state s_0 (the initial one), no coin has been inserted into the machine (hence, the proposition letter $p_{\$=0}$ holds there). Three

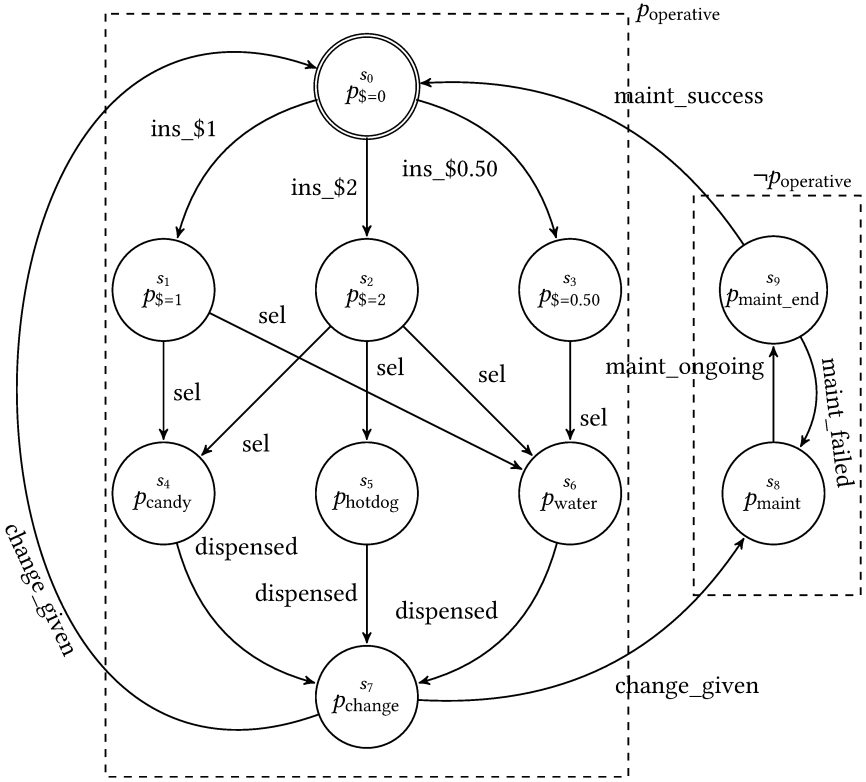


Figure 6.6: Kripke structure representing a vending machine.

edges, labelled by "ins_$1", "ins_$2", and "ins_$0.50", connect s_0 to s_1, s_2, and s_3, respectively. Edge labels do not convey semantic value (they are neither part of the structure definition nor associated with proposition letters) and are simply used for an easy reference to edges. In s_1 (resp., s_2, s_3) the proposition letter $p_{\$=1}$ (resp., $p_{\$=2}$, $p_{\$=0.50}$) holds, representing the fact that 1 Dollar (resp., 2, 0.50 Dollars) has been inserted into the machine. The cost of a bottle of water (resp., a candy, a hot dog) is \$0.50 (resp., \$1, \$2). A state s_i, for $i = 1, 2, 3$, is connected to a state s_j, for $j = 4, 5, 6$, only if the available credit allows one to buy the corresponding item. Then, edges labelled by "dispensed" connect s_4, s_5, and s_6 to s_7. In s_7, the machine gives change, and can nondeterministically move back to s_0 (ready for dispensing another item), or to s_8, where it begins an automatic maintenance activity (p_{maint} holds there). Afterwards, state s_9 is reached, where maintenance ends. From there, if the maintenance activity fails (edge "maint_failed"), s_8 is reached again (another maintenance cycle is attempted); otherwise, maintenance

concludes successfully ("maint_success") and s_0 is reached. Since the machine is operating in states $s_0, \ldots, s_7$, and under maintenance in s_8 and s_9, $p_{\text{operative}}$ holds over the former, and it does not on the latter.

In the following, we will make use of the B formulas length_n, with $n \geq 1$, presented in Example 2.1.7.

We now give some examples of properties we can formalize under all, or some, of the HS semantic variants $\mathsf{HS}_{\mathsf{st}}$, $\mathsf{HS}_{\mathsf{ct}}$, and $\mathsf{HS}_{\mathsf{lin}}$.

- In any run of length 50, during which the machine never enters maintenance mode, it dispenses at least a hotdog, a bottle of water and a candy.

$$\mathcal{K} \not\models (p_{\text{operative}} \wedge \mathit{length}_{50}) \longrightarrow \big((\langle \mathrm{B} \rangle \langle \mathrm{E} \rangle p_{\text{hotdog}}) \wedge (\langle \mathrm{B} \rangle \langle \mathrm{E} \rangle p_{\text{water}}) \wedge (\langle \mathrm{B} \rangle \langle \mathrm{E} \rangle p_{\text{candy}})\big)$$

  Clearly this property is false, as the machine can possibly dispense only one or two kinds of items. We start by observing that the above formula is equivalent in all of the three semantic variants of HS: since modalities $\langle \mathrm{B} \rangle$ and $\langle \mathrm{E} \rangle$ only allow one to "move" from an interval to its subintervals, $\mathsf{BE}_{\mathsf{lin}}$, $\mathsf{BE}_{\mathsf{st}}$, and $\mathsf{BE}_{\mathsf{ct}}$ coincide (for this reason, we have omitted the subscript from the symbol $\models$). Homogeneity plays a fundamental role here: asking $p_{\text{operative}}$ to be true implies that such a letter is true along the whole trace (thus s_8 and s_9 are always avoided).
  It is worth observing that the same property can be expressed in LTL, for instance as follows:

$$\bigwedge_{i \in \{0,\ldots,49\}} \mathsf{X}^i p_{\text{operative}} \wedge \bigvee_{i,j,k \in \{1,\ldots,48\}, i \neq j \neq k \neq i} (\mathsf{X}^i p_{\text{hotdog}}) \wedge (\mathsf{X}^j p_{\text{water}}) \wedge (\mathsf{X}^k p_{\text{candy}}).$$

  The length of this LTL formula is *exponential* in the number of items (in this case, 3), whereas the length of the above HS one is only linear. As a matter of fact, we will prove (Theorem 6.2.5) that BE is at least exponentially more succinct than LTL.

- If the credit is \$0.50, then no hot dog or candy may be provided.

$$\mathcal{K} \models (\langle \mathrm{E} \rangle p_{\$=0.50}) \longrightarrow \neg \langle \mathrm{A} \rangle (\mathit{length}_2 \wedge \langle \mathrm{E} \rangle (p_{\text{hotdog}} \vee p_{\text{candy}}))$$

  We observe that a trace satisfies $\langle \mathrm{E} \rangle p_{\$=0.50}$ if and only if it ends in s_3. This property is satisfied under all of the three semantic variants, even though the nature of future differs among them (recall Figure 6.1, 6.2, and 6.3). As we have already mentioned, a linear setting (rather than branching) is suitable for the specification of dynamic behaviors, because it considers states *of a computation*; conversely, a branching approach focuses on machine states (and thus on the structure of a system).
  In this case, only the state s_6 can be reached from s_3, regardless of the nature of future. For this reason, $\mathsf{HS}_{\mathsf{st}}$, $\mathsf{HS}_{\mathsf{ct}}$, and $\mathsf{HS}_{\mathsf{lin}}$ behave in the same way.

- Let us exemplify now a difference between $\mathsf{HS}_{\mathsf{st}}$ (and $\mathsf{HS}_{\mathsf{ct}}$) and $\mathsf{HS}_{\mathsf{lin}}$.

$$\begin{array}{l}\mathcal{K} \models_{\mathsf{st}} \\ \mathcal{K} \models_{\mathsf{ct}} \quad (\langle \mathrm{E} \rangle\, p_{\mathrm{maint_end}}) \longrightarrow \langle \mathrm{A} \rangle \langle \mathrm{E} \rangle\, p_{\mathrm{operative}} \\ \mathcal{K} \not\models_{\mathsf{lin}}\end{array}$$

  This is a structural property, requiring that when the machine enters state s_9 (where maintenance ends), it can become again operative reaching state s_0 (s_9 is not a lock state for the system). This is clearly true when future is branching and it is not when future is linear: $\mathsf{HS}_{\mathsf{lin}}$ refers to system computations, and some of these may ultimately loop between s_8 and s_9.
- Conversely, some properties make sense only if they are predicated over computations. This is the case, for instance, of fairness.

$$\begin{array}{l}\mathcal{K} \models_{\mathsf{st}} \\ \mathcal{K} \models_{\mathsf{ct}} \quad ([\mathrm{A}] \langle \mathrm{A} \rangle \langle \mathrm{E} \rangle\, p_{\mathrm{maint}}) \longrightarrow [\mathrm{A}] \langle \mathrm{A} \rangle \langle \mathrm{E} \rangle\, p_{\mathrm{operative}} \\ \mathcal{K} \not\models_{\mathsf{lin}}\end{array}$$

  Assuming the trace-based semantics, the property requires that if a system computation enters infinitely often into maintenance mode, it will infinitely often enter operation mode. Again, this is not true, as some system computations may ultimately loop between s_8 and s_9 (hence, they are not fair). On the contrary, such a property is trivially true under $\mathsf{HS}_{\mathsf{st}}$ or $\mathsf{HS}_{\mathsf{ct}}$, as, for any initial trace ρ, it holds that $\mathcal{K}, \rho \models \langle \mathrm{A} \rangle \langle \mathrm{E} \rangle\, p_{\mathrm{operative}}$.
- We conclude with a property showing the difference between linear and branching *past*, that is, between $\mathsf{HS}_{\mathsf{st}}$ and $\mathsf{HS}_{\mathsf{lin}}$ (and $\mathsf{HS}_{\mathsf{ct}}$). The requirement is the following: the machine may dispense water with any amount of (positive) credit.

$$\begin{array}{l}\mathcal{K} \models_{\mathsf{st}} \\ \mathcal{K} \not\models_{\mathsf{ct}} \quad (\langle \mathrm{E} \rangle\, p_{\mathrm{water}}) \longrightarrow \langle \mathrm{E} \rangle \left(p_{\mathrm{water}} \wedge \bigwedge_{p \in \{p_{\$=2}, p_{\$=1}, p_{\$=0.50}\}} \langle \overline{\mathrm{A}} \rangle (\mathit{length}_2 \wedge \langle \mathrm{B} \rangle\, p)\right) \\ \mathcal{K} \not\models_{\mathsf{lin}}\end{array}$$

  Again, this one is a structural property, that cannot be expressed in $\mathsf{HS}_{\mathsf{lin}}$ or $\mathsf{HS}_{\mathsf{ct}}$, as these refer to a specific computation in the past. Conversely, it is true under $\mathsf{HS}_{\mathsf{st}}$, since s_6 is backward reachable in one step by s_1, s_2, and s_3.

6.2 Equivalence between LTL and $\mathsf{HS}_{\mathsf{lin}}$

In this section, we show that $\mathsf{HS}_{\mathsf{lin}}$ is as expressive as LTL even for small syntactical fragments of $\mathsf{HS}_{\mathsf{lin}}$. To this end, we exploit the well-known equivalence between LTL and the first-order fragment of monadic second-order logic over infinite words (FO for short).

Recall that, given a countable set $\{x, y, z, \ldots\}$ of (position) variables, the FO formulas φ over a set of proposition letters $\mathcal{AP} = \{p, \ldots\}$ are defined as:

$$\varphi ::= \top \mid p(x) \mid x \leq y \mid x < y \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists x.\varphi \ .$$

We interpret FO formulas φ over infinite paths π of Kripke structures $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$. Given a variable *valuation* g, assigning to each variable a position $i \geq 0$, the satisfaction relation $(\pi, g) \models \varphi$ corresponds to the standard satisfaction relation $(\mu(\pi), g) \models \varphi$, where $\mu(\pi)$ is the infinite word over $2^{\mathcal{AP}}$ given by $\mu(\pi(0))\mu(\pi(1)) \cdots$. More precisely, $(\pi, g) \models \varphi$ is inductively defined as follows (we omit the standard rules for the Boolean connectives):

$$\begin{array}{ll} (\pi, g) \models p(x) & \Leftrightarrow p \in \mu(\pi(g(x))), \\ (\pi, g) \models x \ op \ y & \Leftrightarrow g(x) \ op \ g(y), \text{ for } op \in \{<, \leq\}, \\ (\pi, g) \models \exists x.\varphi & \Leftrightarrow (\pi, g[x \leftarrow i]) \models \varphi \text{ for some } i \geq 0, \end{array}$$

where $g[x \leftarrow i](x) = i$ and $g[x \leftarrow i](y) = g(y)$ for $y \neq x$. Note that the satisfaction relation depends only on the values assigned to the variables occurring free in the given formula φ. We write $\pi \models \varphi$ to mean that $(\pi, g_0) \models \varphi$, where $g_0(x) = 0$ for each variable x. An FO sentence is a formula with no free variables.

The following is a well-known result (Kamp's theorem [Kam68]).

Proposition 6.2.1. *Given an FO sentence φ over $\mathcal{AP}$, one can construct an LTL formula ψ such that, for all Kripke structures $\mathcal{K}$ over $\mathcal{AP}$ and infinite paths π, it holds that $\pi \models \varphi$ if and only if $\pi, 0 \models \psi$.*

Given a $\mathsf{HS}_{\mathsf{lin}}$ formula ψ, we now construct an FO sentence ψ_{FO} such that, for all Kripke structures $\mathcal{K}$, $\mathcal{K} \models_{\mathsf{lin}} \psi$ if and only if for each initial infinite path π of $\mathcal{K}$, $\pi \models \psi_{\mathsf{FO}}$.

We start by defining a mapping h assigning to each triple (φ, x, y), consisting of a HS formula φ and two distinct position variables x, y, an FO formula having as free variables x and y. The mapping h returns the FO formula defining the semantics of the HS formula φ interpreted over an interval bounded by the positions x and y.

The function h is homomorphic with respect to the Boolean connectives, and is defined for proposition letters and modal operators as follows (here z is a fresh position variable):

$$\begin{array}{ll} h(p, x, y) & = \forall z.((z \geq x \wedge z \leq y) \rightarrow p(z)), \\ h(\langle \mathrm{E} \rangle \psi, x, y) & = \exists z.(z > x \wedge z \leq y \wedge h(\psi, z, y)), \\ h(\langle \mathrm{B} \rangle \psi, x, y) & = \exists z.(z \geq x \wedge z < y \wedge h(\psi, x, z)), \\ h(\langle \overline{\mathrm{E}} \rangle \psi, x, y) & = \exists z.(z < x \wedge h(\psi, z, y)), \\ h(\langle \overline{\mathrm{B}} \rangle \psi, x, y) & = \exists z.(z > y \wedge h(\psi, x, z)). \end{array}$$

It is worth noting that homogeneity plays a crucial role in the definition of $h(p, x, y)$ (without it, a binary predicate would be necessary to encode the truth of p over the interval $[x, y]$).

Given a Kripke structure $\mathcal{K}$, an infinite path π, an interval of positions $[i, j]$, and an $\mathsf{HS}_{\mathsf{lin}}$ formula ψ, by a straightforward induction on the structure of ψ, we can show that $IS_{\mathcal{K},\pi}, [i, j] \models \psi$ if and only if $(\pi, g) \models h(\psi, x, y)$ for any valuation such that $g(x) = i$ and $g(y) = j$.

Let us consider the FO sentence $h(\psi)$ given by $\exists x((\forall z.z \geq x) \wedge \forall y.h(\psi, x, y))$. Clearly $\mathcal{K} \models_{\mathsf{lin}} \psi$ if and only if for each initial infinite path π of $\mathcal{K}$, $\pi \models h(\psi)$. By Proposition 6.2.1, it follows that one can construct an LTL formula $h'(\psi)$ such that $h'(\psi)$ in LTL is equivalent to ψ in $\mathsf{HS}_{\mathsf{lin}}$. Thus, we obtain the following expressiveness containment.

Theorem 6.2.2. *LTL* $\geq$ HS_{lin}.

Now we show that also the converse containment holds, that is, LTL can be translated into $\mathsf{HS}_{\mathsf{lin}}$. Actually, it is worth noting that for such a purpose the fragment AB of $\mathsf{HS}_{\mathsf{lin}}$, featuring only modalities for $\langle \mathrm{A} \rangle$ and $\langle \mathrm{B} \rangle$, is expressive enough.

Theorem 6.2.3. *Given an LTL formula φ, one can construct in linear time an* AB *formula ψ such that φ in LTL is equivalent to ψ in* $\mathsf{AB}_{\mathsf{lin}}$.

Proof. Let $f : \mathsf{LTL} \to \mathsf{AB}$ be the mapping, homomorphic with respect to the Boolean connectives, defined as follows:

$$\begin{array}{ll} f(p) & = p, \text{ for each } p \in \mathcal{AP}, \\ f(\mathrm{X}\psi) & = \langle \mathrm{A} \rangle (\mathit{length}_2 \wedge \langle \mathrm{A} \rangle (\mathit{length}_1 \wedge f(\psi))), \\ f(\psi_1 \mathrm{U} \psi_2) & = \langle \mathrm{A} \rangle \big(\langle \mathrm{A} \rangle (\mathit{length}_1 \wedge f(\psi_2)) \wedge [\mathrm{B}] (\langle \mathrm{A} \rangle (\mathit{length}_1 \wedge f(\psi_1))) \big). \end{array}$$

Given a Kripke structure $\mathcal{K}$, an infinite path π, a position $i \geq 0$, and an LTL formula ψ, by a straightforward induction on the structure of ψ we can show that $\pi, i \models \psi$ if and only if $IS_{\mathcal{K},\pi}, [i, i] \models f(\psi)$. Hence $\mathcal{K} \models \psi$ if and only if $\mathcal{K} \models_{\mathsf{lin}} \mathit{length}_1 \to f(\psi)$. □

The next corollary follows immediately from Theorem 6.2.2 and Theorem 6.2.3.

Corollary 6.2.4. HS_{lin} *and LTL have the same expressive power.*

While there is no difference in the expressive power between LTL and $\mathsf{HS}_{\mathsf{lin}}$, things change if we consider succinctness. Whereas Theorem 6.2.3 shows that it is possible to convert any LTL formula into an equivalent $\mathsf{HS}_{\mathsf{lin}}$ one in linear time, the following theorem holds.

Theorem 6.2.5. *HS_{lin} is at least exponentially more succinct than LTL.*

Proof. To prove the statement, it suffices to provide an $\mathrm{HS}_{\mathrm{lin}}$ formula ψ for which there exists no LTL equivalent formula whose size is polynomial in $|\psi|$.

To this end, we restrict our attention to the fragment $\mathrm{BE}_{\mathrm{lin}}$. Since modalities $\langle \mathrm{B} \rangle$ and $\langle \mathrm{E} \rangle$ only allow one to "move" from an interval to its subintervals, $\mathrm{BE}_{\mathrm{lin}}$ actually coincides with $\mathrm{BE}_{\mathrm{st}}$, whose MC is hard for **EXPSPACE** (Section 2.4). Thus, in particular, it is possible to encode by means of a $\mathrm{BE}_{\mathrm{lin}}$ formula ψ_{cpt} the (unique) computation of a deterministic Turing machine using $b(n) \in O(2^n)$ bits that, when executed on input 0^n, for some natural number $n \geq 1$, counts in binary from 0 to $2^{2^n} - 1$, by repeatedly summing 1, and finally accepts. The length of ψ_{cpt} is *polynomial* in n, and the unique trace which satisfies it (that is, that encodes such a computation) has length $\ell(n) \geq b(n) \cdot 2^{2^n}$.

Conversely, it is known that LTL features a *single-exponential small-model property* [DGL16], stating that, for every *satisfiable* LTL formula φ, there are $u, w \in S^*$ with $|u| \leq 2^{|\varphi|}$ and $|w| \leq |\varphi| \cdot 2^{|\varphi|}$, such that $u \cdot w^\omega, 0 \models \varphi$. This allows us to conclude (by a straightforward contradiction argument) that there is no polynomial-length (with respect to $|\psi_{\mathrm{cpt}}|$, and thus to n) LTL formula that can encode the aforementioned computation. An *exponential-length* LTL formula would be needed for such an encoding. □

Exactly the same argument can be used to show that $\mathrm{HS}_{\mathrm{lin}}$ is at least exponentially more succinct than the extension of LTL with past modalities (denoted in the following as LTL_p) [LP00].

6.3 A characterization of $\mathrm{HS}_{\mathrm{ct}}$

In this section, we will focus our attention on the computation-tree-based semantic variant $\mathrm{HS}_{\mathrm{ct}}$, showing that it is as expressive as *finitary* CTL^*. As a matter of fact, the result can be proved to hold already for the syntactical fragment ABE which does not feature inverse modalities. In addition, we show that $\mathrm{HS}_{\mathrm{ct}}$ is subsumed by CTL^*.

6.3.1 From finitary CTL^* to $\mathrm{HS}_{\mathrm{ct}}$

We first show that finitary CTL^* is subsumed by $\mathrm{HS}_{\mathrm{ct}}$. As a preliminary fundamental step, we prove that, when interpreted over finite words, the BE fragment of HS and LTL define the same class of finitary languages (Theorem 6.3.5).

For an LTL formula φ with proposition letters over an alphabet Σ (in our case Σ is $2^{\mathcal{AP}}$), let us denote by $\mathcal{L}_{act}(\varphi)$ the set of non-empty finite words over Σ satisfying φ under the standard action-based semantics of LTL, interpreted over finite words

(see [Var96]). A similar notion can be given for BE formulas φ with proposition letters in Σ (under the homogeneity assumption). Then, φ denotes a language, written $\mathcal{L}_{act}(\varphi)$, of non-empty finite words over Σ inductively defined as:

- $\mathcal{L}_{act}(a) = a^+$, for each $a \in \Sigma$ (this definition reflects homogeneity);
- $\mathcal{L}_{act}(\neg\varphi) = \Sigma^+ \setminus \mathcal{L}_{act}(\varphi)$;
- $\mathcal{L}_{act}(\varphi_1 \wedge \varphi_2) = \mathcal{L}_{act}(\varphi_1) \cap \mathcal{L}_{act}(\varphi_2)$;
- $\mathcal{L}_{act}(\langle \mathrm{B} \rangle \varphi) = \{w \in \Sigma^+ \mid \mathrm{Pref}(w) \cap \mathcal{L}_{act}(\varphi) \neq \emptyset\}$;
- $\mathcal{L}_{act}(\langle \mathrm{E} \rangle \varphi) = \{w \in \Sigma^+ \mid \mathrm{Suff}(w) \cap \mathcal{L}_{act}(\varphi) \neq \emptyset\}$.

We prove that, under the action-based semantics, BE formulas and LTL formulas define the same class of finitary languages.

To prove that the finitary languages defined by LTL formulas are subsumed by those defined by BE formulas, we exploit an algebraic condition introduced by Wilke in [Wil99], called *LTL-closure*, which gives, for a class of finitary languages, a sufficient condition to guarantee the inclusion of the class of LTL-definable languages. The converse inclusion, that is, the class of finitary languages defined by the fragment BE is subsumed by that defined by LTL, can be proved by a technique similar to that used in Section 6.2, and is thus omitted.

We start by considering the former inclusion recalling from [Wil99] a sufficient condition for a class of finitary languages to include the class of finitary languages which are LTL-definable.

Definition 6.3.1 (LTL-closure)**.** A class C of languages of finite words over finite alphabets is *LTL-closed* if and only if the following conditions are satisfied, where Σ and Δ are finite alphabets, $b \in \Sigma$ and $\Gamma = \Sigma \setminus \{b\}$:

1. C is closed under language complementation and intersection;
2. if $\mathcal{L} \in C$ with $\mathcal{L} \subseteq \Gamma^+$, then $\Sigma^* b \mathcal{L}$, $\Sigma^* b(\mathcal{L} + \varepsilon)$, $\mathcal{L} b \Sigma^*$, and $(\mathcal{L} + \varepsilon) b \Sigma^*$ are in C;
3. Let $U_0 = \Gamma^* b$, $h_0 : U_0 \to \Delta$, and $h : U_0^+ \to \Delta^+$ be defined by $h(u_0 u_1 \cdots u_n) = h_0(u_0) \cdots h_0(u_n)$. Assume that for each $d \in \Delta$, the language $\mathcal{L}_d = \{u \in \Gamma^+ \mid h_0(ub) = d\}$ is in C. Then for each language $\mathcal{L} \in C$ such that $\mathcal{L} \subseteq \Delta^+$, the language $\Gamma^* b h^{-1}(\mathcal{L}) \Gamma^*$ is in C.

In Figure 6.7, we graphically depict (3.) of the definition of LTL-closure. In the proposed example, we have: (i) for all i, $d_i \in \Delta$ and $\gamma_i \in \Gamma$, (ii) $w = (\gamma_1\gamma_2\gamma_3 b)(\gamma_4\gamma_5 b)(\gamma_6\gamma_7\gamma_8 b)(\gamma_9\gamma_{10}\gamma_{11} b) \in U_0^4$, (iii) $w' = h(w) = h_0(\gamma_1\gamma_2\gamma_3 b) h_0(\gamma_4\gamma_5 b)$ $h_0(\gamma_6\gamma_7\gamma_8 b) h_0(\gamma_9\gamma_{10}\gamma_{11} b) = d_1 d_2 d_3 d_1 \in \Delta^4$. For instance, $\gamma_1\gamma_2\gamma_3, \gamma_9\gamma_{10}\gamma_{11} \in L_{d_1}$ and $\gamma_4\gamma_5 \in L_{d_2}$.

The following result holds [Wil99].

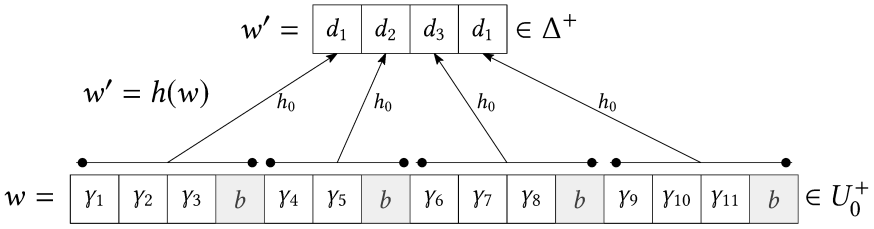


Figure 6.7: Visual description of condition 3 of Definition 6.3.1 (LTL-closure).

Theorem 6.3.2. *Any LTL-closed class C of finitary languages includes the class of LTL-definable finitary languages.*

Therefore, to prove that the finitary languages defined by BE formulas subsume those defined by LTL, as stated by Theorem 6.3.5 below, it suffices to prove that *the class of finitary languages definable by* BE *formulas is LTL-closed*, and to apply Theorem 6.3.2. We observe that, by definition, the class of BE-definable languages is obviously closed under language complementation and intersection ((1.) of Definition 6.3.1). The fulfillment of (2.) and (3.) of Definition 6.3.1 is then proved by the two following Lemma 6.3.3 and 6.3.4, respectively, whose proofs are in Appendix D.1 and D.2.

Lemma 6.3.3. *Let Σ be a finite alphabet, $b \in \Sigma$, $\Gamma = \Sigma \setminus \{b\}$, $\mathcal{L} \subseteq \Gamma^+$, and ψ be a* BE *formula over Γ such that $\mathcal{L}_{act}(\psi) = \mathcal{L}$. Then, there are* BE *formulas defining (under the action-based semantics) the languages $b\mathcal{L}$, $\Sigma^* b\mathcal{L}$, $\Sigma^* b(\mathcal{L} + \varepsilon)$, $\mathcal{L}b$, $\mathcal{L}b\Sigma^*$, $(\mathcal{L} + \varepsilon)b\Sigma^*$, and $b\mathcal{L}b$.*

Lemma 6.3.4. *Let Σ and Δ be finite alphabets, $b \in \Sigma$, $\Gamma = \Sigma \setminus \{b\}$, $U_0 = \Gamma^* b$, $h_0 : U_0 \to \Delta$ and $h : U_0^+ \to \Delta^+$ be defined by $h(u_0 u_1 \cdots u_n) = h_0(u_0) \cdots h_0(u_n)$. Assume that, for each $d \in \Delta$, there is a* BE *formula capturing the language $\mathcal{L}_d = \{u \in \Gamma^+ \mid h_0(ub) = d\}$. Then, for each* BE *formula φ over Δ, one can construct a* BE *formula over Σ capturing the language $\Gamma^* b h^{-1}(\mathcal{L}_{act}(\varphi))\Gamma^*$.*

Since, by Lemma 6.3.3 and 6.3.4, the class of finitary languages definable by BE formulas is LTL-closed, by Theorem 6.3.2 we finally get the following result.

Theorem 6.3.5. *Let φ be an LTL formula over a finite alphabet Σ. Then there exists a* BE *formula φ_{HS} over Σ such that $\mathcal{L}_{act}(\varphi_{HS}) = \mathcal{L}_{act}(\varphi)$.*

The result of Theorem 6.3.5 is used to prove that finitary CTL* is subsumed by the fragment ABE under the state-based semantics.

Theorem 6.3.6. *Let φ be a finitary CTL* formula over $\mathcal{AP}$. Then there is an* ABE *formula φ_{HS} over $\mathcal{AP}$ such that, for all Kripke structures $\mathcal{K}$ over $\mathcal{AP}$ and traces ρ, we have $\mathcal{K}, \rho, 0 \models \varphi$ if and only if $\mathcal{K}, \rho \models_{\mathrm{st}} \varphi_{HS}$.*

Proof. The proof is by induction on the nesting depth of modality $\exists_f$ in φ. In the base case, φ is a finitary LTL formula over $\mathcal{AP}$. Since what we need to deal with it is just the first part of the work we have to do for the inductive step, it is omitted and only the inductive step is detailed.

Let H be the non-empty set of subformulas of φ of the form $\exists_f \psi$ which do not occur in the scope of the path quantifier $\exists_f$, that is, the $\exists_f \psi$ formulas which are maximal with respect to the nesting depth of modality $\exists_f$. Then, φ can be seen as an LTL formula over the extended set of proposition letters $\overline{\mathcal{AP}} = \mathcal{AP} \cup H$. Let $\Sigma = 2^{\overline{\mathcal{AP}}}$ and $\overline{\varphi}$ be the LTL formula over Σ obtained from φ by replacing each occurrence of $p \in \overline{\mathcal{AP}}$ in φ with the formula $\bigvee_{P \in \Sigma \,:\, p \in P} P$, according to the LTL action-based semantics.

Given a Kripke structure $\mathcal{K}$ over $\mathcal{AP}$ with labeling μ and a trace ρ of $\mathcal{K}$, we denote by ρ_H the finite word over $2^{\overline{\mathcal{AP}}}$ of length $|\rho|$ defined as

$$\rho_H(i) = \mu(\rho(i)) \cup \{\exists_f \psi \in H \mid \mathcal{K}, \rho, i \models \exists_f \psi\},$$

for all $i \in [0, |\rho| - 1]$. One can easily prove by structural induction on $\overline{\varphi}$ that $\mathcal{K}, \rho, 0 \models \varphi$ if and only if $\rho_H \in \mathcal{L}_{act}(\overline{\varphi})$. By Theorem 6.3.5, there exists a BE formula $\overline{\varphi}_{\mathrm{HS}}$ over Σ such that $\mathcal{L}_{act}(\overline{\varphi}) = \mathcal{L}_{act}(\overline{\varphi}_{\mathrm{HS}})$.

Now, by the inductive hypothesis, for each formula $\exists_f \psi \in H$, there exists an ABE formula ψ_{HS} such that, for all Kripke structures $\mathcal{K}$ and traces ρ of $\mathcal{K}$, we have $\mathcal{K}, \rho, 0 \models \psi$ if and only if $\mathcal{K}, \rho \models_{\mathrm{st}} \psi_{\mathrm{HS}}$. Since ρ is arbitrary, $\mathcal{K}, \rho, i \models \exists_f \psi$ if and only if $\mathcal{K}, \rho(i, i), 0 \models \exists_f \psi$, if and only if $\mathcal{K}, \rho(i, i) \models_{\mathrm{st}} \langle \mathrm{A} \rangle \psi_{\mathrm{HS}}$, for each $i \geq 0$.

Let φ_{HS} be the ABE formula over $\mathcal{AP}$ obtained from the BE formula $\overline{\varphi}_{\mathrm{HS}}$ by replacing each occurrence of $P \in \Sigma$ in $\overline{\varphi}_{\mathrm{HS}}$ with the formula

$$[\mathrm{G}] \Big(\mathit{length}_1 \longrightarrow \bigwedge_{\exists_f \psi \in H \cap P} \langle \mathrm{A} \rangle \psi_{\mathrm{HS}} \;\wedge\; \bigwedge_{\exists_f \psi \in H \setminus P} \neg \langle \mathrm{A} \rangle \psi_{\mathrm{HS}} \;\wedge\; \bigwedge_{p \in \mathcal{AP} \cap P} p \;\wedge\; \bigwedge_{p \in \mathcal{AP} \setminus P} \neg p \Big).$$

Since for all $i \geq 0$ and $\exists_f \psi \in H$, we have $\mathcal{K}, \rho, i \models \exists_f \psi$ if and only if $\mathcal{K}, \rho(i, i) \models_{\mathrm{st}} \langle \mathrm{A} \rangle \psi_{\mathrm{HS}}$, it is possible to prove by a straightforward induction on the structure of $\overline{\varphi}_{\mathrm{HS}}$ that, for any Kripke structure $\mathcal{K}$ and trace ρ of $\mathcal{K}$, we have $\mathcal{K}, \rho \models_{\mathrm{st}} \varphi_{\mathrm{HS}}$ if and only if $\rho_H \in \mathcal{L}_{act}(\overline{\varphi}_{\mathrm{HS}})$.

Therefore, since $\mathcal{K}, \rho, 0 \models \varphi$ if and only if $\rho_H \in \mathcal{L}_{act}(\overline{\varphi})$ and $\mathcal{L}_{act}(\overline{\varphi}) = \mathcal{L}_{act}(\overline{\varphi}_{\mathrm{HS}})$, then $\mathcal{K}, \rho, 0 \models \varphi$ if and only if $\mathcal{K}, \rho \models_{\mathrm{st}} \varphi_{\mathrm{HS}}$, for any Kripke structure $\mathcal{K}$ and trace ρ of $\mathcal{K}$. This concludes the proof of the theorem. □

Since the fragment ABE of HS does not feature any modalities unravelling a Kripke structure backward (namely, $\langle\overline{\mathrm{A}}\rangle$ and $\langle\overline{\mathrm{E}}\rangle$), the computation-tree-based semantics coincides with the state-based semantics (recall Figure 6.1 and 6.2), and thus the following corollary immediately follows from Theorem 6.3.6.

Corollary 6.3.7. *Finitary CTL* is subsumed by both $\mathsf{HS}_{\mathrm{st}}$ and $\mathsf{HS}_{\mathrm{ct}}$.*

6.3.2 From $\mathsf{HS}_{\mathrm{ct}}$ to finitary CTL*

We show now that $\mathsf{HS}_{\mathrm{ct}}$ is subsumed by both CTL* and its finitary variant. To prove this result, we first introduce a hybrid and linear-past extension of CTL*, called *hybrid* CTL^*_{lp}, and its finitary variant, called *finitary hybrid* CTL^*_{lp}.

Besides standard modalities, hybrid logics make use of explicit variables and quantifiers that bind them [BS96]. Variables and binders allow us to easily mark points in a path, which will be considered as starting and ending points of intervals, thus permitting a natural encoding of $\mathsf{HS}_{\mathrm{ct}}$. Actually, we will show that the restricted use of variables and binders exploited in our encoding does not increase the expressive power of (finitary) CTL* (as it happens for an unrestricted use), thus proving the desired result. We start defining *hybrid* CTL^*_{lp}.

For a countable set $\{x, y, z, \ldots\}$ of (position) variables, the set of formulas φ of hybrid CTL^*_{lp} over $\mathcal{AP}$ is defined as follows:

$$\varphi ::= \top \mid p \mid x \mid \neg\varphi \mid \varphi \vee \varphi \mid \downarrow x.\varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi \mid \mathsf{X}^{-}\varphi \mid \varphi \mathsf{U}^{-}\varphi \mid \exists\varphi,$$

where X^- ("previous") and U^- ("since") are the past counterparts of the "next" and "until" modalities X and U, and $\downarrow x$ is the *downarrow binder operator* [BS96], which binds x to the current position along the given initial infinite path. We also use the standard shorthands $\mathsf{F}^-\varphi := \top\mathsf{U}^-\varphi$ ("eventually in the past") and its dual $\mathsf{G}^-\varphi := \neg\mathsf{F}^-\neg\varphi$ ("always in the past"). As usual, a sentence is a formula with no free variables.

Let $\mathcal{K}$ be a Kripke structure and φ be a hybrid CTL^*_{lp} formula. For an *initial* infinite path π of $\mathcal{K}$, a variable valuation g, that assigns to each variable x a position along π, and $i \geq 0$, the satisfaction relation $\pi, g, i \models \varphi$ is defined as follows (we omit the clauses for the Boolean connectives and for U and X):

$$\begin{aligned}
&\pi, g, i \models \mathsf{X}^{-}\varphi && \Leftrightarrow i > 0 \text{ and } \pi, g, i-1 \models \varphi,\\
&\pi, g, i \models \varphi_1 \mathsf{U}^{-} \varphi_2 && \Leftrightarrow \text{for some } j \leq i \text{ we have } \pi, g, j \models \varphi_2, \text{ and}\\
&&& \quad \pi, g, k \models \varphi_1 \text{ for all } j < k \leq i,\\
&\pi, g, i \models \exists\varphi && \Leftrightarrow \text{for some initial infinite path } \pi' \text{ such that}\\
&&& \quad \pi'(0, i) = \pi(0, i), \text{ we have } \pi', g, i \models \varphi,\\
&\pi, g, i \models x && \Leftrightarrow g(x) = i,\\
&\pi, g, i \models \downarrow x.\varphi && \Leftrightarrow \pi, g[x \leftarrow i], i \models \varphi,
\end{aligned}$$

where $g[x \leftarrow i](x) = i$ and $g[x \leftarrow i](y) = g(y)$ for $y \neq x$. A Kripke structure $\mathcal{K}$ is a model of a formula φ if $\pi, g_0, 0 \models \varphi$, for every initial infinite path π of $\mathcal{K}$, with g_0 the variable valuation assigning 0 to each variable. Note that the path quantification is "memoryful", i.e., it ranges over infinite paths that start at the root and visit the current node of the computation tree. Clearly, the semantics for the syntactical fragment CTL* coincides with the standard one. If we disallow the use of variables and binder modalities, we obtain the logic CTL$^*_{lp}$, a well-known linear-past extension of CTL* which is as expressive as CTL* [KPV12]. We also consider the finitary variant of hybrid CTL$^*_{lp}$, where the path quantifier $\exists$ is replaced with the finitary path quantifier $\exists_f$. This logic corresponds to an extension of finitary CTL* and its semantics is similar to that of hybrid CTL$^*_{lp}$ with the exception that path quantification ranges over the *finite* paths (traces) that start at the root and visit the current node of the computation tree.

In the following, we will use the fragment of hybrid CTL$^*_{lp}$ consisting of *well-formed* formulas, namely, formulas φ where:

- each subformula $\exists\psi$ of φ has at most one free variable (namely, not bound by the downarrow binder operator);
- each subformula $\exists\psi(x)$ of φ having x as free variable occurs in φ in the context $(\mathsf{F}^- x) \wedge \exists\psi(x)$.

Intuitively, the above conditions affirm that, for each state subformula $\exists\psi$, the unique free variable (if any) refers to ancestors of the current node in the computation tree.[2] The notion of well-formed formula of finitary hybrid CTL$^*_{lp}$ is similar: the path quantifier $\exists$ is replaced by its finitary version $\exists_f$.

We first show that HS$_{\mathrm{ct}}$ can be translated into the well-formed fragment of hybrid CTL$^*_{lp}$ (resp., well-formed fragment of finitary hybrid CTL$^*_{lp}$). Then we show that this fragment is subsumed by CTL* (resp., finitary CTL*).

Proposition 6.3.8. *Given a HS$_{\mathrm{ct}}$ formula φ, one can construct in linear-time an equivalent well-formed sentence of hybrid CTL$^*_{lp}$ (resp., finitary hybrid CTL$^*_{lp}$).*

Proof. We focus on the translation from HS$_{\mathrm{ct}}$ into the well-formed fragment of hybrid CTL$^*_{lp}$. The translation from HS$_{\mathrm{ct}}$ into the well-formed fragment of finitary hybrid CTL$^*_{lp}$ is similar, and thus omitted.

Let φ be a HS$_{\mathrm{ct}}$ formula. The desired hybrid CTL$^*_{lp}$ sentence is the formula $\downarrow x.\mathsf{G} f(\varphi, x)$, where $f(\varphi, x)$ is a mapping which is homomorphic with respect to

[2] The well-formedness constraint ensures that a formula captures only branching regular requirements. As an example, the formula $\exists\mathsf{F}\downarrow x.\mathsf{G}^-(\neg\mathsf{X}^-\top \rightarrow \forall\mathsf{F}(x \wedge p))$ is *not* well-formed and requires that there is a level of the computation tree such that each node in the level satisfies p. This represents a non-regular context-free branching requirement (see, e.g., [ACZ06]).

the Boolean connectives, and over proposition letters and modalities behaves in the following way:

$$\begin{aligned}
f(p, x) &= \mathsf{G}^{-}((\mathsf{F}^{-} x) \rightarrow p),\\
f(\langle \mathsf{B} \rangle \psi, x) &= \mathsf{X}^{-}\mathsf{F}^{-}(f(\psi, x) \wedge \mathsf{F}^{-} x),\\
f(\langle \overline{\mathsf{B}} \rangle \psi, x) &= \exists(\mathsf{XF} f(\psi, x)) \wedge \mathsf{F}^{-} x,\\
f(\langle \mathsf{E} \rangle \psi, x) &= \downarrow y.\mathsf{F}^{-}\big(x \wedge \mathsf{XF}\downarrow x.\mathsf{F}(y \wedge f(\psi, x))\big),\\
f(\langle \overline{\mathsf{E}} \rangle \psi, x) &= \downarrow y.\mathsf{F}^{-}\big((\mathsf{XF} x) \wedge \downarrow x.\mathsf{F}(y \wedge f(\psi, x))\big),
\end{aligned}$$

where y is a fresh variable.

Clearly $\downarrow x.\mathsf{G} f(\varphi, x)$ is well-formed. The formula $f(\varphi, x)$ intuitively states that φ holds over an interval of the current path that starts at the position (associated with the variable) x and ends at the current position.

More formally, let $\mathcal{K}$ be a Kripke structure, $[h, i]$ be an interval of positions, g be a valuation assigning to the variable x the position h, and π be an initial infinite path. By a straightforward induction on the structure of φ, one can show that $\mathcal{K}, \pi, g, i \models f(\varphi, x)$ if and only if $\mathcal{C}(\mathcal{K}), \mathcal{C}(\pi, h, i) \models_{\mathrm{st}} \varphi$, where $\mathcal{C}(\pi, h, i)$ denotes the trace of the computation tree $\mathcal{C}(\mathcal{K})$ starting from $\pi(0, h)$ and leading to $\pi(0, i)$. Hence, $\mathcal{K}$ is a model of $\downarrow x.\mathsf{G} f(\varphi, x)$ if, for each initial trace ρ of $\mathcal{C}(\mathcal{K})$, we have that $\mathcal{C}(\mathcal{K}), \rho \models_{\mathrm{st}} \varphi$. □

Let LTL_p be the past extension of LTL, obtained by adding the past modalities X^{-} and U^{-}. By exploiting the well-known separation theorem for LTL_p over finite and infinite words [Gab87], which states that any LTL_p formula can be effectively converted into an equivalent Boolean combination of LTL formulas and pure past LTL_p formulas, we can prove that, under the hypothesis of well-formedness, the extensions of CTL^{*} (resp., finitary CTL^{*}) used to encode $\mathsf{HS}_{\mathrm{ct}}$ formulas do not increase the expressive power of CTL^{*} (resp., finitary CTL^{*}). Such a result is the fundamental step to prove, together with Proposition 6.3.8, that CTL^{*} subsumes $\mathsf{HS}_{\mathrm{ct}}$. In addition, paired with Corollary 6.3.7, it will allow us to state the main result of the section, namely, that $\mathsf{HS}_{\mathrm{ct}}$ and finitary CTL^{*} have the same expressiveness.

Let us now show that the well-formed fragment of hybrid CTL^{*}_{lp} (resp., finitary hybrid CTL^{*}_{lp}) is not more expressive than CTL^{*} (resp., finitary CTL^{*}). Once more, we focus on the well-formed fragment of hybrid CTL^{*}_{lp} omitting the similar proof for the finitary variant.

We start with some additional definitions and auxiliary results. A *pure past* LTL_p formula is an LTL_p formula which does not contain occurrences of future temporal modalities. Given two formulas φ and φ' of hybrid CTL^{*}_{lp}, we say that φ and φ' are *congruent* if, for every Kripke structure $\mathcal{K}$, initial infinite path π, valuation g, and current position i, it holds $\mathcal{K}, \pi, g, i \models \varphi$ if and only if $\mathcal{K}, \pi, g, i \models \varphi'$ (note that congruence is a *stronger* requirement than equivalence).

As usual, for a formula φ of hybrid CTL^*_{lp} with one free variable x, we write $\varphi(x)$. Moreover, since the satisfaction relation depends only on the variables occurring free in the given formula, for $\varphi(x)$ we use the notation $\mathcal{K}, \pi, i \models \varphi(x \leftarrow h)$ to mean that $\mathcal{K}, \pi, g, i \models \varphi$ for any valuation g assigning h to the unique free variable x. For a formula φ of hybrid CTL^*_{lp}, let $\exists SubF(\varphi)$ denote the set of subformulas of φ of the form $\exists\psi$ which do not occur in the scope of the path quantifier $\exists$.

Finally, for technical reasons, we introduce *simple* hybrid CTL^*_{lp} formulas.

Definition 6.3.9. Given a variable x, a *simple* hybrid CTL^*_{lp} formula ψ with respect to x is a hybrid CTL^*_{lp} formula satisfying the constraints:

- x is the unique variable occurring in ψ;
- ψ does *not* contain occurrences of the binder modalities and past temporal modalities;
- $\exists SubF(\psi)$ consists of CTL^* formulas.

Intuitively, a *simple* hybrid CTL^*_{lp} (over $\mathcal{AP}$) formula ψ with respect to x can be seen as a CTL^* formula over the set of proposition letters $\mathcal{AP} \cup \{x\}$ such that x does not occur in the scope of $\exists$. The next lemma, proved in Appendix D.3, shows that ψ can be further simplified whenever it is paired with the formula $\mathsf{F}^- x$.

Lemma 6.3.10. *Let ψ be a* simple *hybrid CTL^*_{lp} formula with respect to x. Then $(F^- x) \wedge \psi$ is congruent to a formula of the form $(F^- x) \wedge \xi$, where ξ is a Boolean combination of the atomic formula x and CTL^* formulas.*

The next lemma states an important technical property of well formed formulas, which will be exploited in Theorem 6.3.14 to prove that the set of sentences of the well-formed fragment of hybrid CTL^*_{lp} has the same expressiveness as CTL^*. Intuitively, if the hybrid features of the language do not occur in the scope of existential path quantifiers, it is possible to remove the occurrences of the binder $\downarrow$ and to suitably separate past and future modalities. The result is obtained by exploiting the equivalence of FO and LTL_p over infinite words and by applying the separation theorem for LTL_p over infinite words [Gab87], that we recall here for completeness.

Theorem 6.3.11 (LTL_p separation over infinite words)**.** *Any LTL_p formula ψ can be effectively transformed into a formula*

$$\psi' = \bigvee_{i=1}^{t} (\psi_{p,i} \wedge \psi_{f,i}),$$

for some $t \geq 1$, where $\psi_{p,i}$ is a pure past LTL_p formula and $\psi_{f,i}$ is an LTL formula, such that for all infinite words w over $2^{\mathcal{AP}}$ and $i \geq 0$, it holds that $w, i \models \psi$ if and only if $w, i \models \psi'$.

Lemma 6.3.12. *Let $(F^{-}x) \wedge \exists\varphi(x)$ (resp., $\exists\varphi$) be a well-formed formula (resp., well-formed sentence) of hybrid CTL^{*}_{lp} such that $\exists SubF(\varphi)$ consists of CTL^{*} formulas. Then $(F^{-}x) \wedge \exists\varphi(x)$ (resp., $\exists\varphi$) is congruent to a well-formed formula of hybrid CTL^{*}_{lp} which is a Boolean combination of CTL^{*} formulas and (formulas that correspond to)* pure past *LTL_p formulas over the set of proposition letters $\mathcal{AP} \cup \exists SubF(\varphi) \cup \{x\}$ (resp., $\mathcal{AP} \cup \exists SubF(\varphi)$).*

Proof. We focus on well-formed formulas of the form $(\mathsf{F}^{-}x) \wedge \exists\varphi(x)$. The case of well-formed sentences of the form $\exists\varphi$ is similar, and thus omitted.

Let $\overline{\mathcal{AP}} = \mathcal{AP} \cup \exists SubF(\varphi) \cup \{x\}$. By hypothesis, $\exists SubF(\varphi)$ is a set of CTL^{*} formulas, that is, they are devoid of any hybrid feature.

Given a Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$, an initial infinite path π, and $h \geq 0$, we denote by $\pi_{\overline{\mathcal{AP}},h}$ the infinite word over $2^{\overline{\mathcal{AP}}}$, which, for every position $i \geq 0$, is defined as follows:

- $\pi_{\overline{\mathcal{AP}},h}(i) \cap \mathcal{AP} = \mu(\pi(i))$;
- $\pi_{\overline{\mathcal{AP}},h}(i) \cap \exists SubF(\varphi) = \{\psi \in \exists SubF(\varphi) \mid \mathcal{K}, \pi, i \models \psi\}$;
- $x \in \pi_{\overline{\mathcal{AP}},h}(i)$ if and only if $i = h$.

By using a fresh position variable *present* to represent the current position, the formula $\varphi(x)$ can be easily converted into an FO formula $\varphi_{\mathsf{FO}}(present)$ over $\overline{\mathcal{AP}}$ having *present* as its unique free variable, such that for all Kripke structures $\mathcal{K}$, initial infinite paths π, and positions i and h, we have:

$$\mathcal{K}, \pi, i \models \varphi(x \leftarrow h) \text{ if and only if } \pi_{\overline{\mathcal{AP}},h} \models \varphi_{\mathsf{FO}}(present \leftarrow i). \tag{6.1}$$

To this end, it suffices to map any proposition letter $\overline{p} \in \overline{\mathcal{AP}}$ into a unary predicate $\overline{p}$, and all the operators $\mathsf{X}, \mathsf{X}^{-}, \mathsf{U}, \mathsf{U}^{-}, \downarrow$ into FO formulas expressing their semantics.

By the equivalence of FO and LTL_p and the separation theorem for LTL_p over infinite words, starting from the FO formula $\varphi_{\mathsf{FO}}(present)$, one can construct an LTL_p formula φ_{LTL_p} over $\overline{\mathcal{AP}}$ of the form

$$\varphi_{\mathsf{LTL}_p} = \bigvee_{i \in I} (\varphi_{p,i} \wedge \varphi_{f,i}) \tag{6.2}$$

such that $\varphi_{p,i}$ is a pure past LTL_p formula, $\varphi_{f,i}$ is an LTL formula, and for all infinite words w over $2^{\overline{\mathcal{AP}}}$ and $i \geq 0$, it holds that:

$$w, i \models \varphi_{\mathsf{LTL}_p} \text{ if and only if } w \models \varphi_{\mathsf{FO}}(present \leftarrow i). \tag{6.3}$$

The LTL_p formula φ_{LTL_p} over $\overline{\mathcal{AP}}$ corresponds to a hybrid CTL^*_{lp} formula $\varphi_{\mathsf{LTL}_p}(x)$ over $\mathcal{AP}$ (note that the only hybrid feature is the possible occurrence of the variable x). By definition of the infinite words $\pi_{\overline{\mathcal{AP}},h}$, one can easily show by structural induction that for all Kripke structures $\mathcal{K}$, initial infinite paths π, and positions i and h:

$$\pi_{\overline{\mathcal{AP}},h}, i \models \varphi_{\mathsf{LTL}_p} \text{ if and only if } \mathcal{K}, \pi, i \models \varphi_{\mathsf{LTL}_p}(x \leftarrow h), \tag{6.4}$$

the latter being a hybrid CTL^*_{lp} formula. Thus, by (6.1), (6.3), and (6.4), we obtain that $\varphi(x)$ and $\varphi_{\mathsf{LTL}_p}(x)$ are congruent.

Since in (6.2), for each $i \in I$, $\varphi_{p,i}$ is a pure past LTL_p formula over $\overline{\mathcal{AP}}$, $\exists\varphi_{p,i}(x)$ is trivially congruent to $\varphi_{p,i}(x)$. As a consequence, we have that $(\mathsf{F}^-x) \wedge \exists\varphi(x)$ is congruent to $(\mathsf{F}^-x) \wedge \bigvee_{i\in I}(\varphi_{p,i}(x) \wedge \exists\varphi_{f,i}(x))$, which is congruent to $\bigvee_{i\in I}(\varphi_{p,i}(x) \wedge (\mathsf{F}^-x) \wedge \exists\varphi_{f,i}(x))$, which is in turn congruent to $\bigvee_{i\in I}(\varphi_{p,i}(x) \wedge \exists((\mathsf{F}^-x) \wedge \varphi_{f,i}(x)))$.

Now, $\varphi_{f,i}(x)$ is a *simple* hybrid CTL^*_{lp} formula with respect to x, and $\exists x$ (resp., $\exists\neg x$) is trivially congruent to x (resp., $\neg x$). By Lemma 6.3.10 and some simple manipulation steps, we can prove the following sequence of equivalences:

$$\bigvee_{i\in I}\Big(\varphi_{p,i}(x) \wedge \exists((\mathsf{F}^-x) \wedge \varphi_{f,i}(x))\Big) =$$

(Lemma 6.3.10 and disjunctive normal form)

$$\bigvee_{i\in I}\Big(\varphi_{p,i}(x) \wedge \exists\big((\mathsf{F}^-x) \wedge \bigvee_{j\in J}(\tilde{x}_{i,j} \wedge \psi_{i,j})\big)\Big) = \quad (\mathsf{F}^-x \text{ is a pure past } \mathsf{LTL}_p \text{ formula})$$

$$\bigvee_{i\in I}\Big(\varphi_{p,i}(x) \wedge (\mathsf{F}^-x) \wedge \exists \bigvee_{j\in J}(\tilde{x}_{i,j} \wedge \psi_{i,j})\Big) =$$

(Distributive property of $\wedge$ over $\vee$)

$$(\mathsf{F}^-x) \wedge \bigvee_{i\in I}\Big(\varphi_{p,i}(x) \wedge \exists \bigvee_{j\in J}(\tilde{x}_{i,j} \wedge \psi_{i,j})\Big) =$$

(Distributive property of $\exists$ over $\vee$ and $\tilde{x}_{i,j}$ is a pure past LTL_p formula)

$$(\mathsf{F}^-x) \wedge \bigvee_{i\in I}\Big(\varphi_{p,i}(x) \wedge \bigvee_{j\in J}(\tilde{x}_{i,j} \wedge \exists\psi_{i,j})\Big) =$$

(Distributive property of $\wedge$ over $\vee$)

$$(\mathsf{F}^-x) \wedge \bigvee_{i\in I}\bigvee_{j\in J}\Big(\varphi_{p,i}(x) \wedge \tilde{x}_{i,j} \wedge \exists\psi_{i,j}\Big)$$

where $\tilde{x}_{i,j}$ is either x, $\neg x$, or $\top$.

Hence, $(\mathsf{F}^-x) \wedge \exists\varphi(x)$ is congruent to a formula of the form

$$(\mathsf{F}^-x) \wedge \bigvee_{i\in I'}(\psi_{p,i}(x) \wedge \exists\psi_i),$$

for some I', where $\psi_{p,i}(x)$ corresponds to a *pure past* LTL_p formula over $\overline{\mathcal{AP}}$ ($= \mathcal{AP} \cup \exists SubF(\varphi) \cup \{x\}$) and ψ_i is a CTL^* formula. □

The following lemma generalizes the separation result given by Lemma 6.3.12 to any well-formed formula of the form $(\mathsf{F}^- x) \wedge \exists \varphi(x)$, that is, to formulas where $\varphi(x)$ is unconstrained. Its proof is by induction, and exploits Lemma 6.3.12 (Appendix D.4).

Lemma 6.3.13. *Let $(\mathsf{F}^- x) \wedge \exists \varphi(x)$ (resp., $\exists \varphi$) be a well-formed formula (resp., well-formed sentence) of hybrid CTL^*_{lp}. Then there exists a finite set $\mathcal{H}$ of CTL^* formulas of the form $\exists \psi$, such that $(\mathsf{F}^- x) \wedge \exists \varphi(x)$ (resp., $\exists \varphi$) is congruent to a well-formed formula of hybrid CTL^*_{lp} which is a Boolean combination of CTL^* formulas and (formulas that correspond to)* pure past *LTL_p formulas over the set of proposition letters $\mathcal{AP} \cup \mathcal{H} \cup \{x\}$ (resp., $\mathcal{AP} \cup \mathcal{H}$).*

We are now ready to prove that the well-formed sentences of hybrid CTL^*_{lp} can be expressed in CTL^*.

Theorem 6.3.14. *The set of sentences of the well-formed fragment of hybrid CTL^*_{lp} has the same expressiveness as CTL^*.*

Proof. Let φ be a well-formed sentence of hybrid CTL^*_{lp}. To prove the thesis, we construct a CTL^* formula which is equivalent to φ.

Since φ is equivalent to $\neg\exists\neg\varphi$ and $\neg\exists\neg\varphi$ is well-formed, by Lemma 6.3.13 one can convert $\neg\exists\neg\varphi$ into a congruent hybrid CTL^*_{lp} formula which is a Boolean combination of CTL^* formulas and formulas θ which can be seen as pure past LTL_p formulas over the set of proposition letters $\mathcal{AP} \cup \mathcal{H}$, where $\mathcal{H}$ is a set of CTL^* formulas of the form $\exists\psi$.

Since the past temporal modalities in such LTL_p formulas θ refer to the initial position of the initial infinite paths, one can replace θ with an equivalent CTL^* formula $f(\theta)$, where the mapping f is inductively defined as follows:

- $f(p) = p$ for all $p \in \mathcal{AP} \cup \mathcal{H}$;
- f is homomorphic with respect to the Boolean connectives;
- $f(\mathsf{X}^-\theta) = \bot$ and $f(\theta_1 \mathsf{U}^- \theta_2) = f(\theta_2)$.

The resulting CTL^* formula is equivalent to $\neg\exists\neg\varphi$, as required. □

By an easy adaptation of the proof of Theorem 6.3.14, where one exploits the separation theorem for LTL_p over finite words [Gab87], it is possible to characterize also the expressiveness of well-formed *finitary* hybrid CTL^*_{lp}.

Theorem 6.3.15. *The set of sentences of the well-formed fragment of finitary hybrid CTL^*_{lp} has the same expressiveness as finitary CTL^*.*

Together with Proposition 6.3.8, Theorem 6.3.14 (resp., Theorem 6.3.15) allows us to conclude that CTL^* (resp., finitary CTL^*) subsumes HS_{ct}.

Finally, by exploiting Corollary 6.3.7, we can state the main result of the section, namely, HS_{ct} and finitary CTL^* have the same expressiveness.

Theorem 6.3.16. *$CTL^* \geq HS_{ct}$. Moreover, HS_{ct} is as expressive as finitary CTL^*.*

6.4 Expressiveness comparison of HS_{lin}, HS_{st}, HS_{ct}

In this section, we compare the expressiveness of the three semantic variants of HS, namely, HS_{lin}, HS_{st}, and HS_{ct}. The resulting picture was anticipated in Figure 6.4. Here we give the proofs of the depicted results.

We start showing that HS_{st} is *not* subsumed by HS_{ct}. As a matter of fact, we show that HS_{st} is sensitive to backward unwinding of finite Kripke structures, allowing us to sometimes discriminate finite Kripke structures with the same computation tree (these structures are always indistinguishable by HS_{ct}).

Let us consider, for instance, the two finite Kripke structures $\mathcal{K}_1$ and $\mathcal{K}_2$ of Figure 6.8, whose forward and backward unwinding is shown in Figure 6.9. Since $\mathcal{K}_1$ and $\mathcal{K}_2$ have the same computation tree, no HS formula φ under the computation-tree-based semantics can distinguish $\mathcal{K}_1$ and $\mathcal{K}_2$, that is, $\mathcal{K}_1 \models_{ct} \varphi$ if and only if $\mathcal{K}_2 \models_{ct} \varphi$. On the other hand, the requirement "*each state reachable from the initial one where p holds has a predecessor where p holds as well*" can be expressed, under the state-based semantics, by the HS formula

$$\psi = \langle \mathrm{E} \rangle (p \wedge \mathit{length}_1) \rightarrow \langle \mathrm{E} \rangle (\mathit{length}_1 \wedge \langle \overline{\mathrm{A}} \rangle (p \wedge \neg \mathit{length}_1)).$$

It is easy to see that $\mathcal{K}_1 \models_{st} \psi$: for any initial trace ρ of $\mathcal{K}_1$, we have $\mathcal{K}_1, \rho \models_{st} \langle \mathrm{E} \rangle (p \wedge \mathit{length}_1)$ iff $\rho = s_0 s_1^k$ for $k \geq 1$; the length-1 suffix s_1 is *met-by* $s_1 s_1$, and $\mathcal{K}_1, s_1 s_1 \models_{st} p \wedge \neg \mathit{length}_1$. On the contrary, in $\mathcal{K}_2$ there is an initial trace, $s'_0 s'_1$, for which $\mathcal{K}_2, s'_0 s'_1 \models_{st} \langle \mathrm{E} \rangle (p \wedge \mathit{length}_1)$; however the only traces that meet the length-1



Figure 6.8: The finite Kripke structures $\mathcal{K}_1$ and $\mathcal{K}_2$.

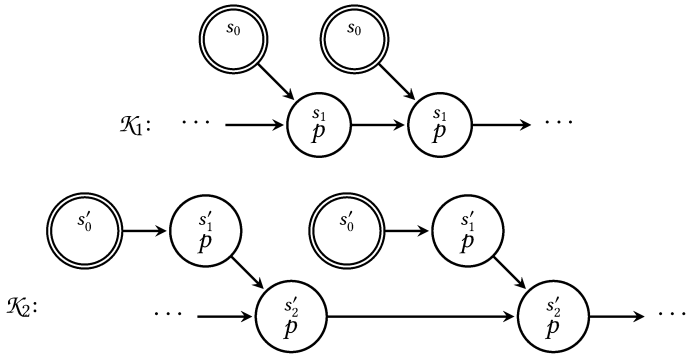


Figure 6.9: Forward and backward unwinding of $\mathcal{K}_1$ and $\mathcal{K}_2$ of Figure 6.8.

suffix s'_1 are s'_1 itself and $s'_0 s'_1$, but neither of them model $p \wedge \neg \mathit{length}_1$. Therefore $\mathcal{K}_2 \not\models_{\mathrm{st}} \psi$. This allows us to prove the following proposition.

Proposition 6.4.1. *$HS_{\mathrm{ct}} \not\geq HS_{\mathrm{st}}$.*

Since, as stated by Theorem 6.3.16, $\mathrm{HS}_{\mathrm{ct}}$ and finitary CTL* have the same expressiveness and finitary CTL* is subsumed by $\mathrm{HS}_{\mathrm{st}}$ (see Corollary 6.3.7), by Proposition 6.4.1 the next corollary immediately follows.

Corollary 6.4.2. *HS_{st} is more expressive than HS_{ct}.*

In the following, we focus on the comparison of $\mathrm{HS}_{\mathrm{lin}}$ with $\mathrm{HS}_{\mathrm{st}}$ and $\mathrm{HS}_{\mathrm{ct}}$ showing that $\mathrm{HS}_{\mathrm{lin}}$ is incomparable with both $\mathrm{HS}_{\mathrm{st}}$ and $\mathrm{HS}_{\mathrm{ct}}$.

The fact that $\mathrm{HS}_{\mathrm{lin}}$ does not subsume either $\mathrm{HS}_{\mathrm{st}}$ or $\mathrm{HS}_{\mathrm{ct}}$ can be easily proved as follows. Consider the CTL formula $\forall \mathsf{G} \exists \mathsf{F} p$ asserting that from each state reachable from the initial one, it is possible to reach a state where p holds. It is well-known that this formula is not LTL-definable (see [BK08], Theorem 6.21). Thus, by Corollary 6.2.4, there is no equivalent $\mathrm{HS}_{\mathrm{lin}}$ formula. On the other hand, the requirement $\forall \mathsf{G} \exists \mathsf{F} p$ can be trivially expressed under the state-based (resp., computation-tree-based) semantics by the HS formula $\langle \overline{\mathrm{B}} \rangle \langle \mathrm{E} \rangle p$, proving the following result.

Proposition 6.4.3. *$HS_{\mathrm{lin}} \not\geq HS_{\mathrm{st}}$ and $HS_{\mathrm{lin}} \not\geq HS_{\mathrm{ct}}$.*

To prove the converse, namely, that $\mathrm{HS}_{\mathrm{lin}}$ is not subsumed either by $\mathrm{HS}_{\mathrm{st}}$ or by $\mathrm{HS}_{\mathrm{ct}}$, we will show that the LTL formula $\mathsf{F} p$ (equivalent to the CTL formula $\forall \mathsf{F} p$) cannot be expressed in either $\mathrm{HS}_{\mathrm{ct}}$ or $\mathrm{HS}_{\mathrm{st}}$ (Proposition 6.4.5). The proof is rather involved and requires a number of definitions and intermediate results; we

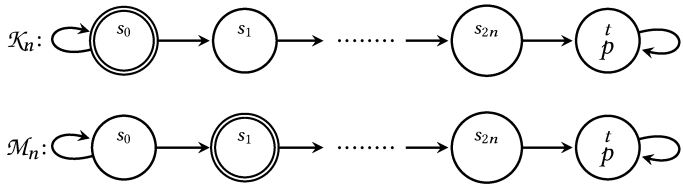


Figure 6.10: The finite Kripke structures $\mathcal{K}_n$ and $\mathcal{M}_n$ with $n \geq 1$.

consider only the state-based semantics, as the case of the computation-tree-based one is very similar.

Let us start by defining two families of Kripke structures $(\mathcal{K}_n)_{n\geq 1}$ and $(\mathcal{M}_n)_{n\geq 1}$ over $\{p\}$ such that for all $n \geq 1$, the LTL formula $\mathsf{F}p$ distinguishes $\mathcal{K}_n$ and $\mathcal{M}_n$, and for every HS formula ψ of size at most n, ψ does *not* distinguish $\mathcal{K}_n$ and $\mathcal{M}_n$ under the state-based semantics.

For a given $n \geq 1$, the Kripke structures $\mathcal{K}_n$ and $\mathcal{M}_n$ are depicted in Figure 6.10. Notice that the Kripke structure $\mathcal{M}_n$ differs from $\mathcal{K}_n$ only in that its initial state is s_1 instead of s_0. Formally, $\mathcal{K}_n = (\{p\}, S_n, R_n, \mu_n, s_0)$ and $\mathcal{M}_n = (\{p\}, S_n, R_n, \mu_n, s_1)$, with $S_n = \{s_0, s_1, \ldots, s_{2n}, t\}$, $R_n = \{(s_0, s_0), (s_0, s_1), \ldots, (s_{2n-1}, s_{2n}), (s_{2n}, t), (t, t)\}$, $\mu(s_i) = \emptyset$ for all $0 \leq i \leq 2n$, and $\mu(t) = \{p\}$.

It is immediate to see that $\mathcal{K}_n \not\models \mathsf{F}p$ and $\mathcal{M}_n \models \mathsf{F}p$.

On the contrary, with the next Lemma 6.4.4, we are going to prove that $\mathcal{K}_n \models_{\mathrm{st}} \psi$ if and only if $\mathcal{M}_n \models_{\mathrm{st}} \psi$ for all balanced $\mathsf{HS}_{\mathrm{st}}$ formulas ψ of length at most n, with $n \geq 1$. An $\mathsf{HS}_{\mathrm{st}}$ formula ψ is *balanced* if, for each subformula $\langle \mathrm{B} \rangle \theta$ (resp., $\langle \overline{\mathrm{B}} \rangle \theta$), θ has the form $\theta_1 \wedge \theta_2$ with $|\theta_1| = |\theta_2|$. Proving the result for balanced $\mathsf{HS}_{\mathrm{st}}$ formulas allows us to state it for any $\mathsf{HS}_{\mathrm{st}}$ formula, since it is possible to trivially convert an $\mathsf{HS}_{\mathrm{st}}$ formula ψ into a balanced one (by using conjunctions of $\top$) which is equivalent to ψ under any of the considered HS semantic variants.

Lemma 6.4.4. *For all $n \in \mathbb{N}^+$ and balanced $\mathsf{HS}_{\mathrm{st}}$ formulas ψ, with $|\psi| \leq n$, it holds that $\mathcal{K}_n \models_{\mathrm{st}} \psi$ if and only if $\mathcal{M}_n \models_{\mathrm{st}} \psi$.*

The complete proof is in Appendix D.5.

As an immediate consequence of Lemma 6.4.4 and of the fact that, for each $n \geq 1$, $\mathcal{K}_n \not\models \mathsf{F}p$ and $\mathcal{M}_n \models \mathsf{F}p$, we get the desired undefinability result.

Proposition 6.4.5. *The LTL formula $\mathsf{F}p$ (equivalent to the CTL formula $\forall\mathsf{F}p$) cannot be expressed in either $\mathsf{HS}_{\mathrm{ct}}$ or $\mathsf{HS}_{\mathrm{st}}$.*

The next proposition follows from Corollary 6.2.4 and Proposition 6.4.5.

Proposition 6.4.6. *$HS_{st} \not\geq HS_{lin}$ and $HS_{ct} \not\geq HS_{lin}$.*

Putting together Proposition 6.4.3 and Proposition 6.4.6, we finally obtain the incomparability result.

Theorem 6.4.7. *HS_{lin} and HS_{st} are expressively incomparable, and so are HS_{lin} and HS_{ct}.*

The proved results also allow us to establish the expressiveness relations between HS_{st}, HS_{ct} and the standard branching temporal logics CTL and CTL*.

Corollary 6.4.8. *The following expressiveness results hold:*

1. *HS_{st} and CTL* are expressively incomparable;*
2. *HS_{st} and CTL are expressively incomparable;*
3. *HS_{ct} and finitary CTL* are less expressive than CTL*;*
4. *HS_{ct} and CTL are expressively incomparable.*

Proof.

1. By Proposition 6.4.5 and the fact that CTL* is not sensitive to unwinding.
2. Again, by Proposition 6.4.5 and by the observation that CTL is not sensitive to unwinding.
3. By Theorem 6.3.16, HS_{ct} is subsumed by CTL*, and HS_{ct} and finitary CTL* have the same expressiveness. Hence, by Proposition 6.4.5, the result follows.
4. Thanks to Proposition 6.4.5, it suffices to show that there exists a HS_{ct} formula which cannot be expressed in CTL.
   Let us consider the CTL* formula $\varphi = \exists\big(((p_1 \mathsf{U} p_2) \vee (q_1 \mathsf{U} q_2))\mathsf{U} r\big)$ over the set of propositions $\{p_1, p_2, q_1, q_2, r\}$. It is shown in [EH86] that φ cannot be expressed in CTL. Clearly, if we replace the path quantifier $\exists$ in φ with the finitary path quantifier $\exists_f$, we obtain an equivalent formula of finitary CTL*. Thus, since HS_{ct} and finitary CTL* have the same expressiveness (Theorem 6.3.16), the result follows. □

6.5 Conclusions

In this chapter we have compared HS MC to MC for point-based temporal logics as far as it concerns expressiveness (and succinctness). To this end, we have taken into consideration three semantic variants of HS, namely, HS_{st}, HS_{ct}, and

$\mathsf{HS}_{\mathsf{lin}}$, under the homogeneity assumption. We have investigated their expressiveness and contrasted them with the point-based temporal logics LTL, CTL, finitary CTL*, and CTL*.

The resulting picture is as follows: $\mathsf{HS}_{\mathsf{lin}}$ and $\mathsf{HS}_{\mathsf{ct}}$ turn out to be as expressive as LTL and finitary CTL*, respectively. Moreover, $\mathsf{HS}_{\mathsf{lin}}$ is at least exponentially more succinct than LTL. $\mathsf{HS}_{\mathsf{st}}$ is expressively incomparable with $\mathsf{HS}_{\mathsf{lin}}$/LTL, CTL, and CTL*, but it is strictly more expressive than $\mathsf{HS}_{\mathsf{ct}}$/finitary CTL*. We believe it possible to fill the expressiveness gap between $\mathsf{HS}_{\mathsf{ct}}$ and CTL* by considering abstract interval models, induced by Kripke structures, featuring worlds also for infinite paths/intervals, and extending the semantics of HS modalities accordingly.

It is finally worth noting that the decidability of the MC problem for (full) $\mathsf{HS}_{\mathsf{ct}}$ and $\mathsf{HS}_{\mathsf{lin}}$ immediately follows, as a byproduct, from the proved results.

Part III

MC for HS relaxing homogeneity

7

MC for HS and its fragments extended with regular expressions

The references for this chapter are [Boz+17a; Boz+17b; Boz+18d].

Contents

7.1 Preliminaries 156
7.2 MC for full HS 160
7.3 The fragments $\overline{\text{A}}\text{A}\overline{\text{B}}\text{B}\overline{\text{E}}$ and $\overline{\text{A}}\text{A}\overline{\text{E}}\text{B}\overline{\text{E}}$ 165
7.3.1 Exponential small-model property for $\overline{\text{A}}\text{A}\overline{\text{B}}\text{B}\overline{\text{E}}$ 165
7.3.2 **AEXP_{pol}** MC algorithm for $\overline{\text{A}}\text{A}\overline{\text{B}}\text{B}\overline{\text{E}}$ 169
7.3.3 **AEXP_{pol}**-hardness of MC for $\text{B}\overline{\text{E}}$ 174
7.4 The fragments $\overline{\text{A}}\text{A}\overline{\text{B}}\text{B}$ and $\overline{\text{A}}\text{A}\overline{\text{E}}\text{E}$ 183
7.4.1 Exponential small-model property for $\overline{\text{A}}\text{A}\overline{\text{B}}\text{B}$ and $\overline{\text{A}}\text{A}\overline{\text{E}}\text{E}$ 183
7.4.2 **PSPACE** MC algorithm for $\text{B}\overline{\text{B}}$ 186
7.4.3 **PSPACE**-completeness of MC for $\overline{\text{A}}\text{A}\overline{\text{B}}\text{B}$ 190
7.5 Conclusions 192

A GOOD BALANCING of *expressiveness* and *complexity* in the choice of the system model and of the specification language is a key factor for the effective exploitation of MC. Various improvements to both of them have been proposed in the literature. As for the latter we recall in particular the extensions of LTL with promptness, that make it possible to bound the delay with which a liveness request is fulfilled (see, e.g., [KPV09]). Another possible direction is *adding regular expressions*, that allow one to enrich the expressive power of existing logics. This has been investigated, for instance, in the cases of LTL [LS07]

Table 7.1: Complexity of MC for HS and its fragments (†local MC). In red, the new results proved in this chapter.

	Homogeneity	Regular expressions	Endpoints [LM13; LM14; LM16]
Full HS, BE	nonelementary **EXPSPACE**-hard	nonelementary **EXPSPACE**-hard	BE+epistemic†: **PSPACE** BE†: **P**
$A\overline{A}B\overline{B}E$, $A\overline{A}E\overline{B}$	$\in$ **EXPSPACE** $\in$ $\mathbf{AEXP_{pol}}$ **PSPACE**-hard	nonelem. **PSPACE**-hard $\mathbf{AEXP_{pol}}$-complete	
$A\overline{ABE}$	**PSPACE**-complete	$\in$ $\mathbf{AEXP_{pol}}$ **PSPACE**-hard	
$A\overline{A}B\overline{B}$, $B\overline{B}$, $\overline{B}$, $A\overline{A}E\overline{E}$, $E\overline{E}$, $\overline{E}$	**PSPACE**-complete	**PSPACE**-complete	$A\overline{B}$+epistemic: nonelementary
$A\overline{A}B$, $A\overline{A}E$, AB, $\overline{A}E$	$\mathbf{P^{NP}}$-complete	**PSPACE**-complete	
$A\overline{A}$, $\overline{A}B$, AE, A, $\overline{A}$	$\in$ $\mathbf{P^{NP[}}O(\log^2 n)\mathbf{]}$ $\mathbf{P^{NP[}}O(\log n)\mathbf{]}$-hard	**PSPACE**-complete	
Prop, B, E	**co-NP**-complete	**PSPACE**-complete	

and CTL [Mat+11]. In this chapter, we study the MC problem for *HS extended with regular expressions*, which are exploited as a means for *relaxing the homogeneity assumption*, that otherwise limits the expressive power of HS.

As we have already said (Section 2.6), in [LM16] Lomuscio and Michaliszyn propose to use regular expressions to define the labeling of proposition letters over intervals in terms of the component states—thus relaxing homogeneity, that can be trivially "encoded" by regular expressions, as shown later. In that work the authors prove decidability of MC with regular expressions for some very restricted fragments of epistemic HS, giving some rough upper bounds to its computational complexity (see the fourth column of Table 7.1). In this chapter, we define interval labeling via regular expressions in a way that can be shown to be equivalent to that of [LM16], and we give a detailed picture of decidability and complexity for HS with regular expressions, which was still missing. The results are summarized in the third column of Table 7.1.

It is interesting to compare the complexity of MC for HS fragments extended with regular expressions with the same fragments under the homogeneity assumption. The rich spectrum of complexities for the less expressive fragments of HS under homogeneity (last four rows in the table) collapses to **PSPACE**-completeness in the case of the corresponding fragments with regular expressions, witnessing that using regular expressions increases the expressive power of (syntactically) small fragments of HS. Whether or not there exists an elementary algorithm for full HS remains an open issue, just like in the case of full HS under homogeneity. The main results of the chapter are summarized in the following account of the contents of the next sections.

Organization of the chapter.

- In Section 7.1, we start by recalling the notions of regular expression and finite-state automaton, and then give syntax and semantics of HS with regular expressions.
- In Section 7.2, we prove that MC for (full) HS extended with regular expressions (under the state-based semantics) is decidable, by exploiting an automata-theoretic approach and the notion of $\mathcal{K}$-NFA, a particular version of finite-state automaton. Moreover, the problem can be shown to be in **P** when it is restricted to system models, assuming the formula to be of constant length.
- Then, in Section 7.3, we study the problems of MC for the two (syntactically) maximal (symmetric) fragments $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}\overline{\mathsf{E}}$ and $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{B}}\overline{\mathsf{E}}$ with regular expressions, proving that both problems are $\mathbf{AEXP}_{\mathbf{pol}}$-complete. $\mathbf{AEXP}_{\mathbf{pol}}$ denotes the complexity class of problems decided by *exponential-time bounded* alternating Turing machines making a *polynomially bounded number of alternations*; such a class captures the exact complexity of some relevant problems [BDP15; FR75], such as, for instance, the first-order theory of real addition with order [FR75]. First, we recall (Chapter 4) that settling the exact complexity of these fragments under the homogeneity assumption is a difficult open question. Moreover, considering that $\mathbf{AEXP}_{\mathbf{pol}} \subseteq \mathbf{AEXP} = \mathbf{EXPSPACE}$ and that HS under homogeneity is subsumed by HS with regular expressions (as we will see later), the results proved in this chapter improve the complexity upper bounds for the fragments $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}\overline{\mathsf{E}}$ and $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{B}}\overline{\mathsf{E}}$ given in Section 4.2. More in detail, we preliminarily establish an exponential small-model property for $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}\overline{\mathsf{E}}$ (Section 7.3.1): for each interval (trace), it is possible to find an interval of bounded exponential length that is indistinguishable with respect to the fulfillment of $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}\overline{\mathsf{E}}$ formulas (respectively, $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{B}}\overline{\mathsf{E}}$ formulas). Such a property allows us to devise an MC procedure belonging to the class $\mathbf{AEXP}_{\mathbf{pol}}$ (Section 7.3.2). Finally, the matching lower bounds are obtained by polynomial-time reductions from the so-called *alternating multi-tiling problem*, showing that they already hold for the fragments $\mathsf{B}\overline{\mathsf{E}}$ and $\mathsf{E}\overline{\mathsf{B}}$ of $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}\overline{\mathsf{E}}$ and $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{B}}\overline{\mathsf{E}}$, respectively (Section 7.3.3).
- Finally, in Section 7.4, we show that formulas of HS fragments featuring (any subset of) HS modalities for the Allen's relations *meets*, *met-by*, *started-by*, and *starts* ($\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}$) can be checked in polynomial working space (MC for all these is **PSPACE**-complete). In particular, in Section 7.4.1 we prove a small-model theorem for $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}$ (and the symmetric $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{E}}$) with regular expressions, which is then exploited in Sections 7.4.2 and 7.4.3 to devise a **PSPACE** MC algorithm for $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}$ (and $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{E}}$). Moreover, in Section 7.4.3, we prove that MC for the purely propositional fragment of HS, denoted as

Prop, is hard for **PSPACE**, which is enough to conclude that MC for any sub-fragment of $A\overline{A}B\overline{B}$ or $A\overline{A}E\overline{E}$ is complete for **PSPACE**. Hence, relaxing the homogeneity assumption via regular expressions comes at no cost for $A\overline{A}B\overline{B}$, $A\overline{A}E\overline{E}$, $B\overline{B}$, $E\overline{E}$, $\overline{B}$ and $\overline{E}$—that remain in **PSPACE**—while $A\overline{A}B$ and $A\overline{A}E$ and their sub-fragments increase their complexity to **PSPACE** (see Table 7.1 once more).

7.1 Preliminaries

We start by recalling the notion of regular expressions over finite words. Since, as it will be clear later, we are interested in expressing requirements over finite words over $2^{\mathcal{AP}}$, here we consider *proposition-based* regular expressions (denoted as REs), where atomic expressions are propositional (Boolean) formulas over $\mathcal{AP}$, instead of just letters over an alphabet. Formally, the set of REs r over $\mathcal{AP}$ is defined by the grammar:

$$r ::= \varepsilon \mid \phi \mid r \cup r \mid r \cdot r \mid r^*,$$

where ϕ is a propositional formula over $\mathcal{AP}$. The length $|r|$ of an RE r is the number of subexpressions of r. An RE r denotes a language $\mathcal{L}(r)$ of finite words over $2^{\mathcal{AP}}$ defined as:

- $\mathcal{L}(\varepsilon) = \{\varepsilon\}$,
- $\mathcal{L}(\phi) = \{A \in 2^{\mathcal{AP}} \mid A \text{ satisfies } \phi\}$,
- $\mathcal{L}(r_1 \cup r_2) = \mathcal{L}(r_1) \cup \mathcal{L}(r_2)$,
- $\mathcal{L}(r_1 \cdot r_2) = \mathcal{L}(r_1) \cdot \mathcal{L}(r_2)$,
- $\mathcal{L}(r^*) = (\mathcal{L}(r))^*$.

By well-known results, the class of RE over $\mathcal{AP}$ captures the class of regular languages of finite words over $2^{\mathcal{AP}}$.

Example 7.1.1. An example of RE is $r_1 = (\mathbf{p} \wedge \mathbf{s}) \cdot \mathbf{s}^* \cdot (\mathbf{p} \wedge \mathbf{s})$ that intuitively denotes the set of finite words where both $\mathbf{p}$ and $\mathbf{s}$ hold true on the endpoints, and $\mathbf{s}$ continuously holds in all internal symbols/sets of $2^{\mathcal{AP}}$. The RE $r_2 = (\neg\mathbf{p})^*$ denotes the set of finite words such that $\mathbf{p}$ does not hold in any position.

We also recall the standard notion of *non-deterministic finite state automaton* (NFA), which is a tuple $\mathcal{A} = (\Sigma, Q, Q_0, \Delta, F)$, where Σ is a finite alphabet, Q is a finite set of states, $Q_0 \subseteq Q$ is the set of initial states, $\Delta : Q \times \Sigma \to 2^Q$ is the transition function (or, equivalently, $\Delta \subseteq Q \times \Sigma \times Q$), and $F \subseteq Q$ is the set of accepting states. An NFA $\mathcal{A}$ is *complete* if, for all $(q, \sigma) \in Q \times \Sigma$, $\Delta(q, \sigma) \neq \emptyset$.

Given a finite word w over Σ, with $|w| = n$, and two states $q, q' \in Q$, a *run* (or *computation*) of $\mathcal{A}$ from q to q' over w is a finite sequence of states $q_1, \ldots, q_{n+1}$ such that $q_1 = q$, $q_{n+1} = q'$, and $q_{i+1} \in \Delta(q_i, w(i))$ for all $i \in [1, n]$. The language $\mathcal{L}(\mathcal{A})$ *accepted by* $\mathcal{A}$ consists of the set of finite words w over Σ such that there is a run over w from some initial state to some accepting state.

A *deterministic finite state automaton* (DFA) is an NFA $\mathcal{D} = (\Sigma, Q, Q_0, \Delta, F)$ such that Q_0 is a singleton, and for all $(q, c) \in Q \times \Sigma$, $\Delta(q, c)$ is a singleton. In the following, in the case of a DFA, we will denote the transition function Δ as δ.

Remark 7.1.2. By well-known results, given an RE r over $\mathcal{AP}$, one can construct, in a compositional way, an NFA $\mathcal{A}_r$ with alphabet $2^{\mathcal{AP}}$, whose number of states is at most $2|r|$, such that $\mathcal{L}(\mathcal{A}_r) = \mathcal{L}(r)$. We call $\mathcal{A}_r$ the *canonical* NFA associated with r.
Note that, though the number of edges of $\mathcal{A}_r$ may be exponential in $|\mathcal{AP}|$ (edges are labelled by assignments $A \in 2^{\mathcal{AP}}$ satisfying propositional formulas ϕ of r), we can avoid explicitly storing edges, as they can be recovered in polynomial time from r.
In Figure 7.1, we depict the canonical NFA $\mathcal{A}_{r_1}$ associated with the RE r_1 of Example 7.1.1. We can avoid storing the edges of $\mathcal{A}_{r_1}$ by remembering which propositional formulas of r_1 they are associated with.

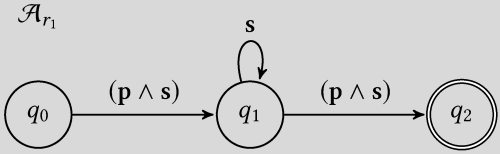


Figure 7.1: The canonical NFA $\mathcal{A}_{r_1}$ associated with the RE r_1 of Example 7.1.1.

In the previous chapters, HS formulas are evaluated over intervals which correspond to the traces of a Kripke structure $\mathcal{K}$. The approach followed is subject to two restrictions: (*i*) the set of proposition letters of HS formulas and the set $\mathcal{AP}$ of proposition letters of the Kripke structure coincide, and (*ii*) a proposition letter holds over an interval if and only if it holds over all its sub-intervals (*homogeneity assumption*). Here we adopt a more general and expressive approach.

Let $\mathcal{P}_u$ be a finite set of *abstract (uninterpreted) interval properties*. An abstract interval property $p_u \in \mathcal{P}_u$ denotes a regular language of finite words over $2^{\mathcal{AP}}$. More specifically, every p_u is a proposition-based regular expression over $\mathcal{AP}$, and it *occurs as a proposition letter* in HS formulas. Thus, hereafter, an HS formula φ over $\mathcal{AP}$ is an HS formula whose proposition letters (i.e., atomic formulas) are REs r over $\mathcal{AP}$. For this reason, we define the size (or length) $|\varphi|$ of

φ as the number of non-atomic subformulas of φ plus $\sum_{r \in \text{Spec}} |r|$, where Spec is the set of REs occurring in φ.

We now define the semantics of an HS formula φ over $\mathcal{AP}$ on a trace ρ of a (finite) Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$. For the purpose we need the following definition.

Definition 7.1.3 (Labelling sequence induced by a trace)**.** A trace $\rho \in \text{Trc}_{\mathcal{K}}$, with $|\rho| = n$, induces a finite word over $2^{\mathcal{AP}}$, denoted as $\mu(\rho)$ and called *labeling sequence*, defined as

$$\mu(\rho(1)) \cdots \mu(\rho(n)).$$

Now, the satisfaction relation $\mathcal{K}, \rho \models \varphi$ can be defined inductively as follows (we omit the standard clauses for Boolean connectives):

- $\mathcal{K}, \rho \models r$ if and only if $\mu(\rho) \in \mathcal{L}(r)$ for each RE r over $\mathcal{AP}$,
- $\mathcal{K}, \rho \models \langle \text{B} \rangle \varphi$ if and only if there exists $\rho' \in \text{Pref}(\rho)$ such that $\mathcal{K}, \rho' \models \varphi$,
- $\mathcal{K}, \rho \models \langle \text{E} \rangle \varphi$ if and only if there exists $\rho' \in \text{Suff}(\rho)$ such that $\mathcal{K}, \rho' \models \varphi$,
- $\mathcal{K}, \rho \models \langle \overline{\text{B}} \rangle \varphi$ if and only if $\mathcal{K}, \rho' \models \varphi$ for some trace ρ' such that $\rho \in \text{Pref}(\rho')$,
- $\mathcal{K}, \rho \models \langle \overline{\text{E}} \rangle \varphi$ if and only if $\mathcal{K}, \rho' \models \varphi$ for some trace ρ' such that $\rho \in \text{Suff}(\rho')$.

As in the previous chapters, we say that $\mathcal{K}$ is a *model* of φ, denoted as $\mathcal{K} \models \varphi$, if, for all *initial* traces ρ of $\mathcal{K}$, it holds that $\mathcal{K}, \rho \models \varphi$. The *MC problem* for HS is the problem of checking, given a finite Kripke structure $\mathcal{K}$ and an HS formula φ, whether or not $\mathcal{K} \models \varphi$. Again, the problem is not trivially decidable since the set $\text{Trc}_{\mathcal{K}}$ of traces of $\mathcal{K}$ is infinite.

With reference to Chapter 6, we recall that the considered *state-based semantics* provides a *branching-time setting both in the past and in the future*. In particular, while modalities for $\langle \text{B} \rangle$ and $\langle \text{E} \rangle$ are linear-time (as they allow us to select prefixes and suffixes of the current trace only), modalities for $\langle \text{A} \rangle$ and $\langle \overline{\text{B}} \rangle$ (respectively, $\langle \overline{\text{A}} \rangle$ and $\langle \overline{\text{E}} \rangle$) are branching-time in the future (respectively, in the past) since they enable us to nondeterministically extend a trace in the future (respectively, in the past).

As shown in Chapter 6, under the considered semantic version, the logics HS and CTL* are expressively incomparable already under the homogeneity assumption. However, under such an assumption, the use of the past branching-time modalities $\langle \overline{\text{A}} \rangle$ and $\langle \overline{\text{E}} \rangle$ is necessary for capturing requirements which cannot be expressed in CTL*. For instance, the constraint "*each state reachable from the initial one where p holds has a predecessor where p holds as well*" cannot be expressed in CTL*, but can be easily stated in the fragment $\overline{\text{A}}\text{E}$ (see Section 6.4).

Conversely, in the more expressive setting based on regular expressions, the future branching-time modalities $\langle \text{A} \rangle$ and $\langle \overline{\text{B}} \rangle$ are already sufficient for capturing

requirements which cannot be expressed in CTL*, such as the following branching-time bounded response property: "*for each state reachable from the initial one, where a request* req *occurs, there is a computation starting from this state such that the request is followed by a response* res *after an* even number *of steps*". This requirement can be expressed in the fragment $\mathsf{A}\overline{\mathsf{B}}$ as follows:

$$[\mathrm{A}](\mathrm{req} \rightarrow \langle \overline{\mathrm{B}} \rangle (\mathrm{req} \cdot (\top \cdot \top)^* \cdot \mathrm{res})).$$

Note that it says nothing about the possible occurrence of a response *res* after an *odd* number of steps.

Before moving on, we show that in this setting it is easy to force homogeneity: we just have to impose that all regular expressions in the formula have the form $p \cdot p^*$, for some $p \in \mathcal{AP}$. Additionally, labelling of intervals by endpoints can be easily captured by regular expressions having the form:

$$\bigcup_{(i,j)\in I} (q_i \cdot \top^* \cdot q_j) \cup \bigcup_{i \in I'} q_i,$$

for some suitable sets of indexes $I \subseteq \{1, \ldots, |S|\}^2$ and $I' \subseteq \{1, \ldots, |S|\}$, where $q_i \in \mathcal{AP}$ is a letter labeling the state $s_i \in S$ of $\mathcal{K}$, only.

Example 7.1.4 (Adapted from [LM16])**.** With this (toy) example we want to compare the effectiveness of regular expressions as rules for defining interval labelling, to homogeneity and to endpoint-based labelling.
In Figure 7.2, a Kripke structure representing a printer is shown. In s_0 the printer starts printing a sheet; in s_1 the process is ongoing, and it ends in s_2. The printer then prints the subsequent sheet, by "moving" back to s_0.
Imagine we want to label the process of printing a *single sheet* by p (i.e., only the trace $s_0s_1s_2$). Under homogeneity, if $s_0s_1s_2$ is labeled by p, then s_0, s_1, s_2, s_0s_1, s_1s_2 must be all labeled by p as well, against our idea. In the endpoint-based approach, in order for p to label $s_0s_1s_2$, p must also label all traces $(s_0s_1s_2)^n$ for $n \in \mathbb{N}^+$, as the endpoints of all these are s_0, s_2. Thus "several, consecutive sheets" are labelled p.
Conversely, we just write the proposition-based RE $\mathbf{p}_{\mathrm{st}} \cdot (\neg \mathbf{p}_{\mathrm{end}} \wedge \neg \mathbf{p}_{\mathrm{st}})^* \cdot \mathbf{p}_{\mathrm{end}}$ to capture precisely the trace $s_0s_1s_2$.

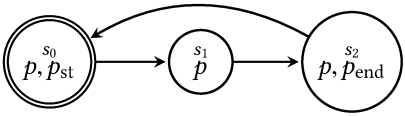


Figure 7.2: Kripke structure representing a printer

7.2 MC for full HS

In this section, we develop an automata-theoretic approach to the MC problem for full HS with regular expressions.

Given a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ and an HS formula φ over $\mathcal{AP}$, we compositionally construct an NFA over the set of states S of $\mathcal{K}$ accepting the set of traces ρ of $\mathcal{K}$ such that $\mathcal{K}, \rho \models \varphi$. The size of the resulting NFA is nonelementary, but it is just *linear in the size of* $\mathcal{K}$. To prove that the nonelementary blow-up does not depend on $\mathcal{K}$, we introduce a special subclass of NFAs, called $\mathcal{K}$-NFA, which intuitively represents the "synchronization" of an NFA with the Kripke structure $\mathcal{K}$. In this way, a $\mathcal{K}$-NFA may only accept traces of $\mathcal{K}$.

Definition 7.2.1 ($\mathcal{K}$-NFA). A $\mathcal{K}$-NFA is an NFA $\mathcal{A} = (S, Q, Q_0, \Delta, F)$ over S satisfying the following conditions:

- the set Q of states has the form $M \times S$ (M is called the *main component*, or the set of *main states*);
- $Q_0 \cap F = \emptyset$, that is, the empty word ε is not accepted;
- for all $(q, s) \in M \times S$ and $s' \in S$, we have $\Delta((q, s), s') = \emptyset$ if $s' \neq s$, and $\Delta((q, s), s) \subseteq M \times R(s)$[a].

[a] We recall that $R(s)$ is the set of successors of s in $\mathcal{K}$.

It is worth noticing that, for all words $\rho \in S^+$, if there is a run of the $\mathcal{K}$-NFA over ρ, then ρ is a trace of $\mathcal{K}$. In the following, we construct a $\mathcal{K}$-NFA $\mathcal{A}$ accepting the traces ρ of $\mathcal{K}$ such that $\mathcal{K}, \rho \models \varphi$.

In a standard automata-theoretic approach, an automaton accepting the set of models of φ would be first defined, and then intersected with $\mathcal{K}$. In the following construction, the synchronization with $\mathcal{K}$ is instead implicitly associated with the construction of the $\mathcal{K}$-NFA itself. Such a choice is motivated by the fact that proposition letters in the formula φ (the base case in the construction) are regular expressions which have to be synchronized with the traces of $\mathcal{K}$. Such a synchronization is then maintained along the whole process of $\mathcal{K}$-NFA construction.

The recursive step for dealing with negation in φ is noteworthy, since it is not just a pure complementation of the $\mathcal{K}$-NFA under construction. As a matter of fact, only the synchronized NFA-component (for the regular expressions of φ) has to be complemented, whereas the synchronized $\mathcal{K}$-component does not. For this reason, the size of the final $\mathcal{K}$-NFA is nonelementary, but linear in the size of $\mathcal{K}$.

In order to prove the main result of the section (stated in Theorem 7.2.5), we preliminarily describe the composition steps to build the required $\mathcal{K}$-NFA. In particular, we give: (*i*) in Proposition 7.2.2 the basic step to deal with propositions

associated with regular expressions, (*ii*) in Proposition 7.2.3 the closure of $\mathcal{K}$-NFAs under language operations corresponding to HS modalities, and (*iii*) in Proposition 7.2.4 the closure of $\mathcal{K}$-NFAs under Boolean operations.

In the following, let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure over $\mathcal{AP}$.

Proposition 7.2.2. *Let $\mathcal{A}$ be an NFA over $2^{\mathcal{AP}}$ with n states. One can construct, in polynomial time, a $\mathcal{K}$-NFA $\mathcal{A}_{\mathcal{K}}$ with at most $n+1$ main states accepting the set of traces ρ of $\mathcal{K}$ such that $\mu(\rho) \in \mathcal{L}(\mathcal{A})$.*

Proof. Let $\mathcal{A} = (2^{\mathcal{AP}}, Q, Q_0, \Delta, F)$. By using an additional state, we can assume $\varepsilon \notin \mathcal{L}(\mathcal{A})$, that is, $Q_0 \cap F = \emptyset$. Then, $\mathcal{A}_{\mathcal{K}} = (S, Q \times S, Q_0 \times S, \Delta', F \times S)$, where for all $(q, s) \in Q \times S$ and $s' \in S$, it holds that $\Delta'((q, s), s') = \emptyset$ if $s' \neq s$, and $\Delta'((q, s), s) = \Delta(q, \mu(s)) \times R(s)$. Since $R(s) \neq \emptyset$ for all $s \in S$, the thesis follows. □

We now define the operations on languages of finite words over S corresponding to the HS modalities $\langle \mathrm{B} \rangle$, $\langle \overline{\mathrm{B}} \rangle$, $\langle \mathrm{E} \rangle$, and $\langle \overline{\mathrm{E}} \rangle$. Given a language $\mathcal{L}$ over S, we define the following languages of traces of $\mathcal{K}$:

- $\langle \mathrm{B} \rangle_{\mathcal{K}}(\mathcal{L}) = \{\rho \in \mathrm{Trc}_{\mathcal{K}} \mid \exists\, \rho' \in \mathcal{L} \cap S^+ \text{ and } \rho'' \in S^+ \text{ such that } \rho = \rho' \cdot \rho''\}$;
- $\langle \overline{\mathrm{B}} \rangle_{\mathcal{K}}(\mathcal{L}) = \{\rho \in \mathrm{Trc}_{\mathcal{K}} \mid \exists\, \rho' \in S^+ \text{ such that } \rho \cdot \rho' \in \mathcal{L} \cap \mathrm{Trc}_{\mathcal{K}}\}$;
- $\langle \mathrm{E} \rangle_{\mathcal{K}}(\mathcal{L}) = \{\rho \in \mathrm{Trc}_{\mathcal{K}} \mid \exists\, \rho'' \in \mathcal{L} \cap S^+ \text{ and } \rho' \in S^+ \text{ such that } \rho = \rho' \cdot \rho''\}$;
- $\langle \overline{\mathrm{E}} \rangle_{\mathcal{K}}(\mathcal{L}) = \{\rho \in \mathrm{Trc}_{\mathcal{K}} \mid \exists\, \rho' \in S^+ \text{ such that } \rho' \cdot \rho \in \mathcal{L} \cap \mathrm{Trc}_{\mathcal{K}}\}$.

We show that $\mathcal{K}$-NFAs are closed under the above defined language operations $\langle \mathrm{B} \rangle_{\mathcal{K}}(\cdot)$, $\langle \mathrm{E} \rangle_{\mathcal{K}}(\cdot)$, $\langle \overline{\mathrm{B}} \rangle_{\mathcal{K}}(\cdot)$ and $\langle \overline{\mathrm{E}} \rangle_{\mathcal{K}}(\cdot)$.

Proposition 7.2.3. *Given a $\mathcal{K}$-NFA $\mathcal{A}$ with n main states, one can construct, in polynomial time, $\mathcal{K}$-NFAs with $n+1$ main states accepting the languages $\langle \mathrm{B} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$, $\langle \mathrm{E} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$, $\langle \overline{\mathrm{B}} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$ and $\langle \overline{\mathrm{E}} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$, respectively.*

Proof. Let $\mathcal{A} = (S, M \times S, Q_0, \Delta, F)$, where M is the set of main states.

Language $\langle \mathrm{B} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$. Let $\mathcal{A}_{\langle \mathrm{B} \rangle}$ be the NFA over S given by

$$\mathcal{A}_{\langle \mathrm{B} \rangle} = (S, (M \cup \{q_{acc}\}) \times S, Q_0, \Delta', \{q_{acc}\} \times S),$$

where $q_{acc} \notin M$ is a fresh main state, and for all $(q, s) \in (M \cup \{q_{acc}\}) \times S$ and $s' \in S$, we have $\Delta'((q, s), s') = \emptyset$, if $s' \neq s$, and

$$\Delta'((q, s), s) = \begin{cases} \Delta((q, s), s) & \text{if } (q, s) \in (M \times S) \setminus F \\ \Delta((q, s), s) \cup (\{q_{acc}\} \times R(s)) & \text{if } (q, s) \in F \\ \{q_{acc}\} \times R(s) & \text{if } q = q_{acc}. \end{cases}$$

Given an input word ρ, from an initial state (q_0, s) of $\mathcal{A}$, the automaton $\mathcal{A}_{\langle \mathrm{B} \rangle}$ simulates the behavior of $\mathcal{A}$ from (q_0, s) over ρ. When $\mathcal{A}$ is in an accepting state (q_f, s) and the current input symbol is s, $\mathcal{A}_{\langle \mathrm{B} \rangle}$ can additionally choose to move to a state in $\{q_{acc}\} \times R(s)$, which is accepting for $\mathcal{A}_{\langle \mathrm{B} \rangle}$ (a prefix of ρ belongs to $\mathcal{L}(\mathcal{A})$). From such states, $\mathcal{A}_{\langle \mathrm{B} \rangle}$ accepts if and only if the remaining part of the input is a trace of $\mathcal{K}$. Since $\mathcal{A}$ is a $\mathcal{K}$-NFA, $\mathcal{A}_{\langle \mathrm{B} \rangle}$ is a $\mathcal{K}$-NFA by construction. Moreover, a word ρ over S is accepted by $\mathcal{A}_{\langle \mathrm{B} \rangle}$ *if and only if* ρ is a trace of $\mathcal{K}$ having some proper prefix ρ' in $\mathcal{L}(\mathcal{A})$ (note that $\rho' \neq \varepsilon$ since $\mathcal{A}$ is a $\mathcal{K}$-NFA). Thus, $\mathcal{L}(\mathcal{A}_{\langle \mathrm{B} \rangle}) = \langle \mathrm{B} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$.

Language $\langle \overline{\mathrm{B}} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$. Let $\mathcal{A}_{\langle \overline{\mathrm{B}} \rangle}$ be the NFA over S given by

$$\mathcal{A}_{\langle \overline{\mathrm{B}} \rangle} = (S, (M \cup \{q_0'\}) \times S, \{q_0'\} \times S, \Delta', F'),$$

where $q_0' \notin M$ is a fresh main state and Δ' and F' are defined as follows:

- for all $(q, s) \in (M \cup \{q_0'\}) \times S$, $s' \in S$, we have $\Delta'((q, s), s') = \emptyset$, if $s' \neq s$, and

$$\Delta'((q, s), s) = \begin{cases} \bigcup_{(q_0, s) \in Q_0} \Delta((q_0, s), s) & \text{if } q = q_0' \\ \Delta((q, s), s) & \text{otherwise.} \end{cases}$$

- The set F' of accepting states is the set of states (q, s) of $\mathcal{A}$ such that there exists a run of $\mathcal{A}$ from (q, s) to some state in F over some non-empty word.

Note that the set F' can be computed in time polynomial in the size of $\mathcal{A}$. Since $\mathcal{A}_{\langle \overline{\mathrm{B}} \rangle}$ essentially simulates $\mathcal{A}$, and $\{q_0'\} \times S$ and F' are disjoint, by construction it easily follows that $\mathcal{A}_{\langle \overline{\mathrm{B}} \rangle}$ is a $\mathcal{K}$-NFA. Moreover, $\mathcal{A}_{\langle \overline{\mathrm{B}} \rangle}$ accepts a word ρ *if and only if* ρ is a non-empty proper prefix of some word accepted by $\mathcal{A}$. Thus, since $\mathcal{A}$ is a $\mathcal{K}$-NFA, we obtain that $\mathcal{L}(\mathcal{A}_{\langle \overline{\mathrm{B}} \rangle}) = \langle \overline{\mathrm{B}} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$.

The constructions for $\langle \mathrm{E} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$ and $\langle \overline{\mathrm{E}} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$—which are symmetric to the ones for $\langle \mathrm{B} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$ and $\langle \overline{\mathrm{B}} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$—can be found in E.1. □

Now we show that $\mathcal{K}$-NFAs are closed under Boolean operations.

Proposition 7.2.4. *Given two $\mathcal{K}$-NFAs $\mathcal{A}$ and $\mathcal{A}'$ with n and n' main states, respectively, one can construct:*

- *a $\mathcal{K}$-NFA with $n + n'$ main states accepting $\mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{A}')$ in time $O(n + n')$;*
- *a $\mathcal{K}$-NFA with $2^{n+1} + 1$ main states accepting $\mathrm{Trc}_{\mathcal{K}} \setminus \mathcal{L}(\mathcal{A})$ in time $2^{O(n)}$.*

Proof. The construction for union is standard and thus omitted. The construction for complementation follows.

Let $\mathcal{A} = (S, M \times S, Q_0, \Delta, F)$. First, we need a preliminary construction. Let us consider the NFA

$$\mathcal{A}'' = (S, (M \cup \{q_{acc}\}) \times S, Q_0, \Delta'', \{q_{acc}\} \times S),$$

where $q_{acc} \notin M$ is a fresh main state, and for all $(q, s) \in (M \cup \{q_{acc}\}) \times S$ and $s' \in s$, we have $\Delta''((q, s), s') = \emptyset$, if $s' \neq s$, and

$$\Delta''((q, s), s) = \begin{cases} \Delta((q, s), s) \cup (\{q_{acc}\} \times S) & \text{if } q \in M \text{ and } \Delta((q, s), s) \cap F \neq \emptyset \\ \Delta((q, s), s) & \text{if } q \in M \text{ and } \Delta((q, s), s) \cap F = \emptyset \\ \emptyset & \text{if } q = q_{acc}. \end{cases}$$

Note that $\mathcal{L}(\mathcal{A}'') = \mathcal{L}(\mathcal{A})$, but $\mathcal{A}''$ is actually *not* a $\mathcal{K}$-NFA.

Next, we show that it is possible to construct in time $2^{O(n)}$ a *weak* $\mathcal{K}$-NFA $\mathcal{A}_c$ with 2^{n+1} main states accepting $(\mathrm{Trc}_{\mathcal{K}} \setminus \mathcal{L}(\mathcal{A}'')) \cup \{\varepsilon\}$, where a *weak* $\mathcal{K}$-NFA is just a $\mathcal{K}$-NFA without the requirement that the empty word ε is not accepted. Thus, since a weak $\mathcal{K}$-NFA can be easily converted into an equivalent $\mathcal{K}$-NFA by using an additional main state, and $\mathcal{L}(\mathcal{A}'') = \mathcal{L}(\mathcal{A})$, the result follows. The weak $\mathcal{K}$-NFA $\mathcal{A}_c$ is given by $\mathcal{A}_c = (S, 2^{\tilde{M}} \times S, Q_{0,c}, \Delta_c, F_c)$, where $\tilde{M} = M \cup \{q_{acc}\}$, and $Q_{0,c}$, F_c and Δ_c are defined as follows:

- $Q_{0,c} = \{(P, s) \in 2^M \times S \mid P = \{q \in M \mid (q, s) \in Q_0\}\}$;
- $F_c = \{(P, s) \in 2^M \times S\}$;
- for all $(P, s) \in 2^{\tilde{M}} \times S$ and $s' \in S$, we have $\Delta_c((P, s), s') = \emptyset$, if $s' \neq s$, and

$$\Delta_c((P, s), s) = \bigcup_{s' \in R(s)} \left\{ (\{q' \in \tilde{M} \mid (q', s') \in \bigcup_{p \in P} \Delta''(p, s)\}, s') \right\}.$$

By construction, $\mathcal{A}_c$ is a weak $\mathcal{K}$-NFA not accepting words in $S^+ \setminus \mathrm{Trc}_{\mathcal{K}}$. Since $Q_{0,c} \subseteq F_c$, we have $\varepsilon \in \mathcal{L}(\mathcal{A}_c)$. Let $\rho \in \mathrm{Trc}_{\mathcal{K}}$ with $|\rho| = k$. To conclude the proof, we have to show that $\rho \in \mathcal{L}(\mathcal{A}'')$ if and only if $\rho \notin \mathcal{L}(\mathcal{A}_c)$.

Assuming that $\rho \in \mathcal{L}(\mathcal{A}'')$, we prove by contradiction that $\rho \notin \mathcal{L}(\mathcal{A}_c)$. Let us assume that there is a run of $\mathcal{A}_c$ over ρ having the form $(P_0, s_0) \cdots (P_k, s_k)$ such that $(P_0, s_0) \in Q_{0,c}$ and $(P_k, s_k) \in F_c$ implying that $q_{acc} \notin P_k$. By construction, $P_0 = \{q \in M \mid (q, s_0) \in Q_0\}$, and for all $i \in [0, k-1]$, $s_i = \rho(i)$ and $P_{i+1} = \{p \in \tilde{M} \mid (p, s_{i+1}) \in \Delta''(q, s_i)$ for some $q \in P_i\}$. Since $\rho \in \mathcal{L}(\mathcal{A}'')$, there is $s \in S$, $(q_0, s_0) \in Q_0$ and an accepting run of $\mathcal{A}''$ over ρ having the form $(q_0, s_0) \cdots (q_{k-1}, s_{k-1})(q_k, s)$ where $q_k = q_{acc}$. By definition of the transition function Δ'' of $\mathcal{A}''$, we can also assume that $s = s_k$. It follows that $q_i \in P_i$ for all $i \in [0, k]$, which is a contradiction since $q_{acc} \notin P_k$. Therefore $\rho \notin \mathcal{L}(\mathcal{A}_c)$.

As for the converse direction, let us assume that $\rho \notin \mathcal{L}(\mathcal{A}_c)$. We have to show that $\rho \in \mathcal{L}(\mathcal{A}'')$. By construction, there exists some run of $\mathcal{A}_c$ over ρ starting from

an initial state (recall that $R(s) \neq \emptyset$ for all $s \in S$). Moreover, each of these runs has the form $(P_0, s_0) \cdots (P_k, s_k)$ such that $P_0 = \{q \in M \mid (q, s_0) \in Q_0\}$, $q_{acc} \in P_k$, and for all $i \in [0, k-1]$, $s_i = \rho(i)$ and $P_{i+1} = \{p \in M \mid (p, s_{i+1}) \in \Delta(q, s_i)$ for some $q \in P_i\}$. It easily follows that there is an accepting run of $\mathcal{A}''$ over ρ from some initial state in $P_0 \times \{s_0\}$, thus proving the thesis. □

An MC algorithm for full HS can be built as follows. Let φ be an HS formula. First of all, we convert φ into an equivalent formula, called *existential form of* φ, that makes use of negations, disjunctions, and the existential modalities $\langle \mathrm{B} \rangle$, $\langle \overline{\mathrm{B}} \rangle$, $\langle \mathrm{E} \rangle$, and $\langle \overline{\mathrm{E}} \rangle$, only. For all $h \geq 1$, let HS_h denote the syntactical HS fragment consisting only of formulas φ such that the *nesting depth of negation in the existential form of* φ *is at most* h. Moreover $\neg\mathsf{HS}_h$ is the set of formulas φ such that $\neg\varphi \in \mathsf{HS}_h$. Now, given an HS formula φ (in existential form), checking whether $\mathcal{K} \not\models \varphi$ reduces to checking the existence of an initial trace ρ of $\mathcal{K}$ such that $\mathcal{K}, \rho \models \neg\varphi$. By exploiting Proposition 7.2.2, 7.2.3 and 7.2.4, we can build in a compositional way (driven by the structure of $\neg\varphi$) a $\mathcal{K}$-NFA $\mathcal{A}$ accepting the set of initial traces ρ such that $\mathcal{K}, \rho \models \neg\varphi$ and check $\mathcal{A}$ for emptiness.

For all $h, n \geq 0$, let $\mathrm{Tower}(h, n)$ denote a tower of exponentials of height h and argument n, that is, $\mathrm{Tower}(0, n) = n$ and $\mathrm{Tower}(h+1, n) = 2^{\mathrm{Tower}(h,n)}$. Moreover, let h-**EXPTIME** denote the class of languages decided by deterministic Turing machines whose number of computation steps is bounded by functions of n in $O(\mathrm{Tower}(h, n^c))$, for some constant $c \geq 1$. Note that 0-**EXPTIME** is **P**.

The next theorem states the main result of the section.

Theorem 7.2.5. *There exists a constant c such that, given a finite Kripke structure $\mathcal{K}$ and an HS formula φ, one can construct a $\mathcal{K}$-NFA with $O(|\mathcal{K}| \cdot \mathrm{Tower}(h, |\varphi|^c))$ states accepting the set of traces ρ of $\mathcal{K}$ such that $\mathcal{K}, \rho \models \varphi$, being h the nesting depth of negation in the existential form of φ. Moreover, for each $h \geq 0$, the MC problem for $\neg\mathsf{HS}_h$ is in h-**EXPTIME**. Additionally, for a constant-length formula, the MC problem is in* **P**.

On one hand, this algorithm can be adapted to HS under homogeneity in a straightforward way, as an alternative to the one presented in Section 2.3. On the other, the complexity lower bound proved in Section 2.4 for MC of HS formulas under homogeneity immediately propagates to the complexity of MC for HS extended with regular expressions.

Theorem 7.2.6. *The MC problem for HS formulas extended with regular expressions over finite Kripke structures is* **EXPSPACE***-hard (under polynomial-time reductions).*

We now focus on the complexity of HS fragments with regular expressions.

7.3 The fragments $A\overline{A}B\overline{BE}$ and $A\overline{A}E\overline{BE}$

In this section, we focus on the syntactically maximal fragments $A\overline{A}B\overline{BE}$ and $A\overline{A}E\overline{BE}$ showing that they feature a lower computational complexity with respect to the general case of full HS.

First of all, in Section 7.3.1 we prove that they feature an exponential small-model property, stating that if ρ is a trace of a finite Kripke structure $\mathcal{K}$ and ψ is an $A\overline{A}B\overline{BE}$ formula, then there is a trace ρ' such that $\mathcal{K}, \rho \models \psi$ if and only if $\mathcal{K}, \rho' \models \psi$, and $|\rho'|$ is exponential in the B-nesting depth of ψ.

In Section 7.3.2, we exploit this small-model property to design a MC algorithm for $A\overline{A}B\overline{BE}$ belonging to the complexity class $\mathbf{AEXP}_{\mathbf{pol}}$, namely, the class of problems decidable by singly exponential-time bounded alternating Turing machines with a polynomial-bounded number of alternations.

Finally, in Section 7.3.3, we show that MC for $A\overline{A}B\overline{BE}$ (actually the smaller fragment $B\overline{E}$ suffices) is hard for $\mathbf{AEXP}_{\mathbf{pol}}$, hence proving completeness for that class.

7.3.1 Exponential small-model property for $A\overline{A}B\overline{BE}$

Here we prove the *exponential small-model property* for $A\overline{A}B\overline{BE}$, which will be used as the basic step to prove that the MC problem for $A\overline{A}B\overline{BE}$ belongs to $\mathbf{AEXP}_{\mathbf{pol}}$. The case of $A\overline{A}E\overline{BE}$ is, as usual, completely symmetric and thus omitted.

Let us consider a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ and a finite set $\text{Spec} = \{r_1, \ldots, r_H\}$ of (proposition-based) regular expressions over $\mathcal{AP}$. The small-model guarantees that for each $h \geq 0$ and trace ρ of $\mathcal{K}$, it is possible to build another trace ρ' of $\mathcal{K}$, of bounded exponential length, which is indistinguishable from ρ with respect to the fulfilment of any $A\overline{A}B\overline{BE}$ formula φ having atomic formulas in Spec and B-nesting depth at most h.

In order to prove the result, we resort again to the notion of *h-prefix bisimilarity* between a pair of traces ρ and ρ' of $\mathcal{K}$, already introduced in Section 4.2, but adapted here to account for regular expressions. As proved by Proposition 7.3.6 below, h-prefix bisimilarity is a sufficient condition for two traces ρ and ρ' to be indistinguishable with respect to the fulfillment of any $A\overline{A}B\overline{BE}$ formula φ over Spec with $\text{Nest}_B(\varphi) \leq h$.

Then we recall (and adapt) also the *h-prefix sampling* of a trace: for a given trace ρ, the *h-prefix sampling* of ρ is a subset of ρ-positions that allows us to build another trace ρ' having single exponential length (both in h and $|\text{Spec}|$, where $|\text{Spec}|$ is defined as $\sum_{r \in \text{Spec}} |r|$) such that ρ and ρ' are h-prefix bisimilar.

Even though we start from adaptations of already presented notions, as it will be clear in the next section the resulting algorithm for $A\overline{A}B\overline{BE}$ MC with regular expressions is completely novel w.r.t. that for the same fragment under homogeneity (Section 4.2).

For a regular expression r_ℓ in Spec, with $\ell \in [1, H]$, let $\mathcal{A}_\ell = (2^{\mathcal{AP}}, Q_\ell, Q^0_\ell, \Delta_\ell, F_\ell)$ be the *canonical (complete)* NFA accepting $\mathcal{L}(r_\ell)$ (recall that $|Q_\ell| \leq 2|r_\ell|$, Remark 7.1.2). W.l.o.g., we assume that the sets of states of these automata are pairwise disjoint.

Prefix bisimilarity now uses the notion of *summary* of a trace ρ of $\mathcal{K}$, namely, a tuple "recording" the initial and final states of ρ, and, for each automaton $\mathcal{A}_\ell$, with $\ell \in [1, H]$, the pairs of states $q, q' \in Q_\ell$ such that some run of $\mathcal{A}_\ell$ over $\mu(\rho)$ takes from q to q'.

Definition 7.3.1 (Summary of a trace)**.** Let ρ be a trace of $\mathcal{K}$ with $|\rho| = n$. The summary $\mathcal{S}(\rho)$ of ρ (w.r.t. Spec) is the triple $(\rho(1), \Pi, \rho(n))$, where

$$\Pi = \{(q, q') \mid q, q' \in Q_\ell \text{ for some } \ell \in [1, H], \text{ and there is a run of } \mathcal{A}_\ell \text{ from } q \text{ to } q' \text{ over } \mu(\rho)\}.$$

Note that the number of summaries is at most $|S|^2 \cdot 2^{(2|\text{Spec}|)^2}$. The following result can be easily proved.

Proposition 7.3.2. *Let $h \geq 0$, and ρ and ρ' be two traces of $\mathcal{K}$ such that $\mathcal{S}(\rho) = \mathcal{S}(\rho')$. Then, for all regular expressions $r \in$ Spec and traces ρ_L and ρ_R of $\mathcal{K}$ such that $\rho_L \star \rho$ and $\rho \star \rho_R$ are defined, the next properties hold:*

1. *$\mu(\rho) \in \mathcal{L}(r)$ if and only if $\mu(\rho') \in \mathcal{L}(r)$;*
2. *$\mathcal{S}(\rho_L \star \rho) = \mathcal{S}(\rho_L \star \rho')$;*
3. *$\mathcal{S}(\rho \star \rho_R) = \mathcal{S}(\rho' \star \rho_R)$.*

We now introduce the "new version" of *prefix bisimilarity* between a pair of traces ρ and ρ' of $\mathcal{K}$.

Definition 7.3.3 (Prefix bisimilarity)**.** Let $h \geq 0$. Two traces ρ and ρ' of $\mathcal{K}$ are *h-prefix bisimilar* (w.r.t. Spec) if the next conditions inductively hold:

- for $h = 0$: $\mathcal{S}(\rho) = \mathcal{S}(\rho')$;
- for $h > 0$: $\mathcal{S}(\rho) = \mathcal{S}(\rho')$ and for each proper prefix ν of ρ (resp., proper prefix ν' of ρ'), there exists a proper prefix ν' of ρ' (resp., proper prefix ν of ρ) such that ν and ν' are $(h-1)$-prefix bisimilar.

Property 7.3.4. For all $h \geq 0$, h-prefix bisimilarity is an equivalence relation over $\text{Trc}_{\mathcal{K}}$.

The h-prefix bisimilarity of two traces ρ and ρ' is preserved by left (resp., right) $\star$-concatenation with another trace of $\mathcal{K}$.

Proposition 7.3.5. *Let $h \geq 0$, and ρ and ρ' be two h-prefix bisimilar traces of $\mathcal{K}$. Then, for all traces ρ_L and ρ_R of $\mathcal{K}$ such that $\rho_L \star \rho$ and $\rho \star \rho_R$ are defined, the following properties hold:*

1. *$\rho_L \star \rho$ and $\rho_L \star \rho'$ are h-prefix bisimilar;*
2. *$\rho \star \rho_R$ and $\rho' \star \rho_R$ are h-prefix bisimilar.*

The proof can be found in Appendix E.2. By exploiting Proposition 7.3.2 and 7.3.5, we can prove that h-prefix bisimilarity preserves the satisfiability of $\mathsf{A\overline{A}B\overline{BE}}$ formulas over Spec having B-nesting depth at most h.

Proposition 7.3.6. *Let $h \geq 0$, and ρ and ρ' be two h-prefix bisimilar traces of $\mathcal{K}$. Then, for each $\mathsf{A\overline{A}B\overline{BE}}$ formula ψ over Spec with $\mathsf{Nest_B}(\psi) \leq h$, we have that*

$$\mathcal{K}, \rho \models \psi \iff \mathcal{K}, \rho' \models \psi.$$

The proof is in Appendix E.3.

In the following, by analogy with Section 4.2, we show how a trace ρ, whose length exceeds a suitable exponential bound—precisely, $(|S| \cdot 2^{(2|\mathsf{Spec}|)^2})^{h+2}$—can be contracted preserving h-prefix bisimilarity and, consequently, satisfiability of formulas φ with $\mathsf{Nest_B}(\varphi) \leq h$. The basic contraction step of ρ is performed by choosing the subset of ρ-positions called *h-prefix sampling* (PS_h). A contraction can be performed whenever there are two positions $\ell < \ell'$ satisfying $\mathcal{S}(\rho(1, \ell)) = \mathcal{S}(\rho(1, \ell'))$ in between two consecutive positions in the linear ordering of PS_h. We will prove that by taking the contraction $\rho' = \rho(1, \ell) \cdot \rho(\ell' + 1, |\rho|)$, we obtain a trace of $\mathcal{K}$ which is h-prefix bisimilar to ρ. The basic contraction step can then be iterated over ρ' until the length bound is reached.

The notion of h-prefix sampling is, again, inductively defined using the definition of *prefix-skeleton sampling* (now accounting for trace summaries). We recall that, for a set I of natural numbers, by "two consecutive elements of I" we mean a pair of elements $i, j \in I$ such that $i < j$ and $I \cap [i, j] = \{i, j\}$.

Definition 7.3.7 (Prefix-skeleton sampling)**.** Let ρ be a trace of $\mathcal{K}$. Given two ρ-positions i and j, with $i \leq j$, the *prefix-skeleton sampling of ρ in the interval $[i, j]$* is the *minimal* set P of ρ-positions in the interval $[i, j]$ satisfying the conditions:

- $i, j \in P$;
- for each $k \in [i + 1, j - 1]$, the minimum position $k' \in [i + 1, j - 1]$ such that $\mathcal{S}(\rho(1, k')) = \mathcal{S}(\rho(1, k))$ belongs to P.

Example 7.3.8. An example of prefix-skeleton sampling P of a trace ρ in the interval $[i, j]$ is shown in Figure 7.3. Assuming that $\mathcal{S}(\rho(1, u)) = \mathcal{S}_1$ for $u \in \{i{+}1, i{+}2, i{+}3, i{+}5, i{+}7, i{+}10\}$, $\mathcal{S}(\rho(1, u')) = \mathcal{S}_2$ for $u' \in \{i{+}4, i{+}8\}$, and $\mathcal{S}(\rho(1, u'')) = \mathcal{S}_3$ for $u'' \in \{i{+}6, i{+}9\}$, we have that $P = \{i, i{+}1, i{+}4, i{+}6, j\}$.

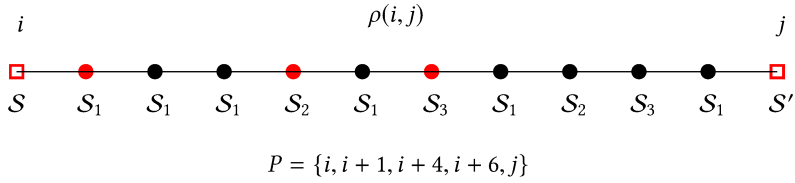


Figure 7.3: Example of prefix-skeleton sampling P of a trace ρ in the interval $[i, j]$.

Note that, as an immediate consequence of Definition 7.3.7, the prefix-skeleton sampling P of (any) trace ρ in $[i, j]$ is such that $|P| \leq (|S| \cdot 2^{(2|\mathrm{Spec}|)^2}) + 2$.

Definition 7.3.9 (h-prefix sampling and h-sampling word)**.** Let $h \geq 0$. The *h-prefix sampling of a trace ρ* of $\mathcal{K}$ is the *minimal* set PS_h of ρ-positions inductively satisfying the following conditions:

- for $h = 0$: $PS_0 = \{1, |\rho|\}$;
- for $h > 0$: (i) $PS_h \supseteq PS_{h-1}$ and (ii) for all pairs of consecutive positions $i, j \in PS_{h-1}$, the prefix-skeleton sampling of ρ in the interval $[i, j]$ belongs to PS_h.

Let $i_1 < \ldots < i_N$ be the ordered sequence of positions in PS_h (note that $i_1 = 1$ and $i_N = |\rho|$). The *h-sampling word of ρ* is the sequence of summaries $\mathcal{S}(\rho(1, i_1)) \cdots \mathcal{S}(\rho(1, i_N))$.

We can state the following upper bound to the cardinality of prefix samplings.

Property 7.3.10. Let $h \geq 0$. The h-prefix sampling PS_h of a (any) trace ρ of $\mathcal{K}$ is such that $|PS_h| \leq (|S| \cdot 2^{(2|\mathrm{Spec}|)^2})^{h+1}$.

As stated by the following lemma, for two traces, the property of having the same h-sampling word is a sufficient condition to guarantee that they are h-prefix bisimilar. The proof is in Appendix E.4.

Lemma 7.3.11. *For $h \geq 0$, any two traces ρ and ρ' of $\mathcal{K}$ having the same h-sampling word are h-prefix bisimilar.*

The sufficient condition of Lemma 7.3.11 allows us to finally state the exponential small-model property for $A\overline{A}B\overline{BE}$. In the proof of Theorem 7.3.12 below, it is shown how to derive from any trace ρ of $\mathcal{K}$, an h-prefix bisimilar trace ρ' *induced by* ρ (Definition 4.1.1) such that $|\rho'| \leq (|S| \cdot 2^{(2|\mathrm{Spec}|)^2})^{h+2}$. By Proposition 7.3.6, ρ' is indistinguishable from ρ w.r.t. the fulfillment of any $A\overline{A}B\overline{BE}$ formula φ over the set of atomic formulas in Spec such that $\mathrm{Nest}_B(\varphi) \leq h$.

Theorem 7.3.12 (Exponential small-model property for $A\overline{A}B\overline{BE}$)**.** *Let ρ be a trace of $\mathcal{K}$ and $h \geq 0$. Then there exists a trace ρ' induced by ρ, whose length is at most $(|S| \cdot 2^{(2|Spec|)^2})^{h+2}$, which is h-prefix bisimilar to ρ. In particular, for every $A\overline{A}B\overline{BE}$ formula ψ with atomic formulas in Spec and $\mathrm{Nest}_B(\psi) \leq h$, it holds that*

$$\mathcal{K}, \rho \models \psi \iff \mathcal{K}, \rho' \models \psi.$$

Proof. We show that if $|\rho| > (|S| \cdot 2^{(2|\mathrm{Spec}|)^2})^{h+2}$, then there exists a trace ρ' induced by ρ such that $|\rho'| < |\rho|$, and ρ and ρ' have the same h-sampling word.

Assume that $|\rho| > (|S| \cdot 2^{(2|\mathrm{Spec}|)^2})^{h+2}$. Let $PS_h : 1 = i_1 < \ldots < i_N = |\rho|$ be the h-prefix sampling of ρ. By Property 7.3.10, $|PS_h| \leq (|S| \cdot 2^{(2|\mathrm{Spec}|)^2})^{h+1}$. Since the number of distinct summaries (w.r.t. Spec) associated with the prefixes of ρ is at most $|S| \cdot 2^{(2|\mathrm{Spec}|)^2}$, there must be two consecutive positions i_j and i_{j+1} in PS_h such that for some $\ell, \ell' \in [i_j + 1, i_{j+1} - 1]$ with $\ell < \ell'$, $\mathcal{S}(\rho(1, \ell)) = \mathcal{S}(\rho(1, \ell'))$. It easily follows that the sequence ρ' given by $\rho' = \rho(1, \ell) \cdot \rho(\ell' + 1, |\rho|)$ is a trace induced by ρ such that $|\rho'| < |\rho|$, and ρ and ρ' have the same h-sampling word. Now, by Lemma 7.3.11, ρ and ρ' are h-prefix bisimilar and by applying Proposition 7.3.6 we get that $\mathcal{K}, \rho \models \psi \iff \mathcal{K}, \rho' \models \psi$. Now, if $|\rho'| \leq (|S| \cdot 2^{(2|\mathrm{Spec}|)^2})^{h+2}$, the thesis follows, otherwise a sequence of contraction steps as shown above can be performed, until the length of the contracted trace fulfills the requirement. □

7.3.2 $\mathbf{AEXP_{pol}}$ MC algorithm for $A\overline{A}B\overline{BE}$

In this section, taking advantage of the exponential small-model property proved previously, we design a MC algorithm for $A\overline{A}B\overline{BE}$ formulas with regular expressions belonging to the complexity class $\mathbf{AEXP_{pol}}$. We recall that $\mathbf{AEXP_{pol}}$ is the class of problems solvable by singly exponential-time bounded alternating Turing machines (ATMs, for short) performing at most a polynomial-bounded number of alternations. More formally, an ATM $\mathcal{M}$ (we refer to [CKS81] for standard syntax and semantics of ATMs) is *singly exponential-time bounded* if there exists an integer constant $c \geq 1$ such that, for each input α, any computation starting on α halts after at most $2^{|\alpha|^c}$ steps. The ATM $\mathcal{M}$ makes a *polynomial-bounded number of alternations* if there exists an integer constant $c' \geq 1$ such that, for

all inputs α and computations π starting from α, the number of alternations of existential and universal configurations along π is at most $|\alpha|^{c'}$.

In the sequel, we restrict ourselves w.l.o.g. to $\overline{\text{A}}\overline{\text{A}}\text{B}\overline{\text{B}}\overline{\text{E}}$ formulas in *negation normal form* (NNF). For φ in NNF, the *dual* of φ, denoted as $\widetilde{\varphi}$, is defined as the NNF of $\neg\varphi$. The complexity measure of an $\overline{\text{A}}\overline{\text{A}}\text{B}\overline{\text{B}}\overline{\text{E}}$ formula φ that we will consider is the standard *alternation depth*, denoted by $\Upsilon(\varphi)$, between the existential $\langle \text{X} \rangle$ and universal modalities $[\text{X}]$ (and vice versa) occurring in the NNF of φ, *for* $X \in \{\overline{B}, \overline{E}\}$. Note that the definition does not consider the modalities associated with the Allen's relations in $\{A, \overline{A}, B\}$. Moreover, let FMC be the set of pairs $(\mathcal{K}, \varphi)$ consisting of a finite Kripke structure $\mathcal{K}$ and an $\overline{\text{A}}\overline{\text{A}}\text{B}\overline{\text{B}}\overline{\text{E}}$ formula φ such that $\mathcal{K} \models \varphi$ (i.e., $\mathcal{K}$ is a model of φ).

The following theorem states the complexity upper bound of MC for $\overline{\text{A}}\overline{\text{A}}\text{B}\overline{\text{B}}\overline{\text{E}}$ formulas with regular expressions.

Theorem 7.3.13. *One can construct a singly exponential-time bounded ATM accepting* FMC *whose number of alternations on an input* $(\mathcal{K}, \varphi)$ *is at most* $\Upsilon(\varphi) + 2$.

To prove the assertion of Theorem 7.3.13, we define a procedure in the remaining part of the section. Such a procedure can be easily translated into an ATM (the translation is omitted).

We start with some auxiliary notation. Let us fix a finite Kripke structure $\mathcal{K}$ with set of states S and an $\overline{\text{A}}\overline{\text{A}}\text{B}\overline{\text{B}}\overline{\text{E}}$ formula φ in NNF. Let $h = \text{Nest}_{\text{B}}(\varphi)$, and Spec be the set of regular expressions occurring in φ. A *certificate* of $(\mathcal{K}, \varphi)$ is a trace ρ of $\mathcal{K}$ whose length is less than $(|S| \cdot 2^{(2|\text{Spec}|)^2})^{h+2}$ (the bound for the exponential small-model property of Theorem 7.3.12). A $\overline{B}$*-witness* (resp., $\overline{E}$*-witness*) of a certificate ρ for $(\mathcal{K}, \varphi)$ is a certificate ρ' of $(\mathcal{K}, \varphi)$ such that ρ' is h-prefix bisimilar to a trace having the form $\rho \star \rho''$ (resp., $\rho'' \star \rho$) for some *certificate* ρ'' of $(\mathcal{K}, \varphi)$ with $|\rho''| > 1$. Finally, by $\text{SD}(\varphi)$ we denote the set consisting of the subformulas ψ of φ and the *duals* $\widetilde{\psi}$.

The results stated in Section 7.3.1 are used to prove the properties of certificates, which are listed in the following Proposition 7.3.14, and then exploited in the MC algorithm.

Proposition 7.3.14. *Let* $\mathcal{K}$ *be a finite Kripke structure,* φ *be an* $\overline{\text{A}}\overline{\text{A}}\text{B}\overline{\text{B}}\overline{\text{E}}$ *formula in NNF, and* ρ *be a certificate for* $(\mathcal{K}, \varphi)$. *The next properties hold:*

1. *for each* $\langle \text{X} \rangle \psi \in SD(\varphi)$, *with* $X \in \{\overline{B}, \overline{E}\}$, *it holds* $\mathcal{K}, \rho \models \langle \text{X} \rangle \psi$ *if and only if there exists an* X*-witness* ρ' *of* ρ *for* $(\mathcal{K}, \varphi)$ *such that* $\mathcal{K}, \rho' \models \psi$;
2. *for each trace having the form* $\rho \star \rho'$ *(resp.,* $\rho' \star \rho$*) such that* ρ' *is a certificate for* $(\mathcal{K}, \varphi)$, *one can construct in time singly exponential in*

the size of $(\mathcal{K}, \varphi)$, a certificate ρ'' which is h-prefix bisimilar to $\rho \star \rho'$ (resp., $\rho' \star \rho$), with $h = \mathrm{Nest}_{\mathrm{B}}(\varphi)$.

Proof. (1.) Let $\langle \mathrm{X} \rangle \psi \in \mathrm{SD}(\varphi)$ with $X \in \{\overline{B}, \overline{E}\}$, $h = \mathrm{Nest}_{\mathrm{B}}(\varphi)$, and ρ be a certificate for $(\mathcal{K}, \varphi)$. Let us assume that $X = \overline{E}$ (the case for $X = \overline{B}$ is similar).

First we assume that there exists an $\overline{E}$-witness ρ' of ρ for $(\mathcal{K}, \varphi)$ such that $\mathcal{K}, \rho' \models \psi$. Hence ρ' is h-prefix bisimilar to a trace having the form $\rho'' \star \rho$, with $|\rho''| > 1$. Since $\langle \overline{\mathrm{E}} \rangle \psi \in \mathrm{SD}(\varphi)$, it holds that $\mathrm{Nest}_{\mathrm{B}}(\langle \overline{\mathrm{E}} \rangle \psi) \leq h$. By Proposition 7.3.6 we have that $\mathcal{K}, \rho'' \star \rho \models \psi$ and, then, $\mathcal{K}, \rho \models \langle \overline{\mathrm{E}} \rangle \psi$.

To prove the converse implication, we assume that $\mathcal{K}, \rho \models \langle \overline{\mathrm{E}} \rangle \psi$. Then, there exists a trace having the form $\rho'' \star \rho$ with $|\rho''| > 1$ such that $\mathcal{K}, \rho'' \star \rho \models \psi$. By Theorem 7.3.12 there exists a certificate ν for $(\mathcal{K}, \varphi)$ which is h-prefix bisimilar to ρ''. By Proposition 7.3.5, $\nu \star \rho$ is h-prefix bisimilar to $\rho'' \star \rho$. By applying Proposition 7.3.6 we deduce that $\mathcal{K}, \nu \star \rho \models \psi$. By applying again Theorem 7.3.12, there exists a certificate ρ' for $(\mathcal{K}, \varphi)$ which is h-prefix bisimilar to $\nu \star \rho$ such that $\mathcal{K}, \rho' \models \psi$. Thus, since ρ' is an $\overline{E}$-witness of ρ for $(\mathcal{K}, \varphi)$, (1.) follows.

(2.) From the trace $\rho \star \rho'$ (resp., $\rho' \star \rho$), where both ρ and ρ' are certificates for $(\mathcal{K}, \varphi)$, we first compute the h-prefix sampling of $\rho \star \rho'$ (resp., $\rho' \star \rho$), where $h = \mathrm{Nest}_{\mathrm{B}}(\varphi)$. Then, proceeding as in the proof of Theorem 7.3.12, we extract from $\rho \star \rho'$ (resp., $\rho' \star \rho$) a trace which is h-prefix bisimilar to $\rho \star \rho'$ (resp., $\rho' \star \rho$). Since $|\rho|$ and $|\rho'|$ are singly exponential in the sizes of $(\mathcal{K}, \varphi)$, (2.) follows. □

Let $\mathrm{A}\overline{\mathrm{A}}(\varphi)$ be the set of formulas in $\mathrm{SD}(\varphi)$ having the form $\langle \mathrm{X} \rangle \psi'$ or $[\mathrm{X}] \psi'$, with $X \in \{A, \overline{A}\}$. An $\mathrm{A}\overline{\mathrm{A}}$-labeling Lab for $(\mathcal{K}, \varphi)$ is a mapping associating with each state s of $\mathcal{K}$ a maximally consistent set of subformulas of $\mathrm{A}\overline{\mathrm{A}}(\varphi)$. More precisely, for all $s \in S$, $Lab(s)$ is such that for all $\psi, \widetilde{\psi} \in \mathrm{A}\overline{\mathrm{A}}(\varphi)$, $Lab(s) \cap \{\psi, \widetilde{\psi}\}$ is a singleton. We say that Lab is *valid* if, for all states $s \in S$ and $\psi \in Lab(s)$, we have $\mathcal{K}, s \models \psi$ (we consider s as a length-1 trace). Note that if Lab is valid, then

- for each trace ρ of $\mathcal{K}$ and $\langle \mathrm{A} \rangle \psi' \in \mathrm{A}\overline{\mathrm{A}}(\varphi)$ (resp., $\langle \overline{\mathrm{A}} \rangle \psi' \in \mathrm{A}\overline{\mathrm{A}}(\varphi)$), it holds that $\mathcal{K}, \rho \models \langle \mathrm{A} \rangle \psi'$ (resp., $\mathcal{K}, \rho \models \langle \overline{\mathrm{A}} \rangle \psi'$) if and only if $\langle \mathrm{A} \rangle \psi' \in Lab(\mathrm{lst}(\rho))$ (resp., $\langle \overline{\mathrm{A}} \rangle \psi' \in Lab(\mathrm{fst}(\rho))$).
- Analogously, for each trace ρ of $\mathcal{K}$ and $[\mathrm{A}] \psi' \in \mathrm{A}\overline{\mathrm{A}}(\varphi)$ (resp., $[\overline{\mathrm{A}}] \psi' \in \mathrm{A}\overline{\mathrm{A}}(\varphi)$), it holds that $\mathcal{K}, \rho \models [\mathrm{A}] \psi'$ (resp., $\mathcal{K}, \rho \models [\overline{\mathrm{A}}] \psi'$) if and only if $[\mathrm{A}] \psi' \in Lab(\mathrm{lst}(\rho))$ (resp., $[\overline{\mathrm{A}}] \psi' \in Lab(\mathrm{fst}(\rho))$).

Finally, a *well-formed set for* $(\mathcal{K}, \varphi)$ is a finite set $\mathcal{W}$ consisting of pairs (ψ, ρ) such that $\psi \in \mathrm{SD}(\varphi)$ and ρ is a certificate of $(\mathcal{K}, \varphi)$. $\mathcal{W}$ is said to be *universal* if each formula occurring in $\mathcal{W}$ has the form $[\mathrm{X}] \psi$, with $X \in \{\overline{B}, \overline{E}\}$. The *dual* $\widetilde{\mathcal{W}}$ of $\mathcal{W}$ is the well-formed set obtained by replacing each pair $(\psi, \rho) \in \mathcal{W}$ with $(\widetilde{\psi}, \rho)$. A well-formed set $\mathcal{W}$ is *valid* if, for each $(\psi, \rho) \in \mathcal{W}$, it holds that $\mathcal{K}, \rho \models \psi$.

We can now introduce the procedure `check`, reported in Algorithm 7.1, that defines the ATM required to prove the assertion of Theorem 7.3.13. The procedure

Algorithm 7.1 `check`$(\mathcal{K}, \varphi)$

1: existentially choose an A$\overline{\text{A}}$-labeling Lab for $(\mathcal{K}, \varphi)$
2: **for** each state s and $\psi \in Lab(s)$ **do**
3: **Case** $\psi = \langle \text{A} \rangle \psi'$ (resp., $\psi = \langle \overline{\text{A}} \rangle \psi'$)
4: existentially choose a certificate ρ with $\text{fst}(\rho) = s$ (resp., $\text{lst}(\rho) = s$)
5: `checkTrue`$_{(\mathcal{K}, \varphi, Lab)}(\{(\psi', \rho)\})$
6: **case** $\psi = [\text{A}] \psi'$ (resp., $\psi = [\overline{\text{A}}] \psi'$)
7: universally choose a certificate ρ with $\text{fst}(\rho) = s$ (resp., $\text{lst}(\rho) = s$)
8: `checkTrue`$_{(\mathcal{K}, \varphi, Lab)}(\{(\psi', \rho)\})$
9: **EndCase**
10: universally choose a certificate ρ for $(\mathcal{K}, \varphi)$ with $\text{fst}(\rho) = s_0$ ◃ s_0 is the initial state of $\mathcal{K}$
11: `checkTrue`$_{(\mathcal{K}, \varphi, Lab)}(\{(\varphi, \rho)\})$

`check` takes a pair $(\mathcal{K}, \varphi)$ as input, being $\mathcal{K}$ a finite Kripke structure and φ an A$\overline{\text{A}}$B$\overline{\text{B}}\overline{\text{E}}$ formula in NNF, and: (1) it guesses an A$\overline{\text{A}}$-labeling Lab for $(\mathcal{K}, \varphi)$ (line 1); (2) it checks that Lab is valid (lines 2–9); (3) for every certificate ρ starting from the initial state, it verifies that $\mathcal{K}, \rho \models \varphi$ (lines 10–11). To perform steps (2) and (3), it exploits the auxiliary ATM procedure `checkTrue`, reported in Algorithm 7.2.

The procedure `checkTrue` takes as input a well-formed set $\mathcal{W}$ for $(\mathcal{K}, \varphi)$ and, assuming that the current A$\overline{\text{A}}$-labeling Lab for $(\mathcal{K}, \varphi)$ is valid, checks whether $\mathcal{W}$ is valid. For each pair $(\psi, \rho) \in \mathcal{W}$ such that ψ does not have the form $[\text{X}] \psi'$, with $X \in \{\overline{B}, \overline{E}\}$, `checkTrue` directly checks whether or not $\mathcal{K}, \rho \models \psi$ (lines 4–29).

In order to allow for a deterministic choice of the current element of the iteration (line 2), we assume that the set $\mathcal{W}$ is implemented as an ordered data structure. At each iteration of the while loop in `checkTrue`, the current pair $(\psi, \rho) \in \mathcal{W}$ is processed according to the semantics of HS, exploiting the guessed A$\overline{\text{A}}$-labeling Lab for modalities $\langle \text{A} \rangle$, $\langle \overline{\text{A}} \rangle$, $[\text{A}]$ and $[\overline{\text{A}}]$ (lines 10–15), and $\langle \overline{\text{E}} \rangle$-witnesses and $\langle \overline{\text{B}} \rangle$-witnesses (guaranteed by Proposition 7.3.14) for $\langle \overline{\text{E}} \rangle$ and $\langle \overline{\text{B}} \rangle$ (lines 26–28). The processing is either deterministic or based on an existential choice, and the currently processed pair (ψ, ρ) is either removed from $\mathcal{W}$, or replaced with pairs (ψ', ρ') such that ψ' is a strict subformula of ψ (this is the case of Boolean connectives and modalities $\langle \text{B} \rangle$, $[\text{B}]$, $\langle \overline{\text{E}} \rangle$ and $\langle \overline{\text{B}} \rangle$, at lines 16–28).

At the end of the while loop, the resulting well formed set $\mathcal{W}$ is either empty or universal. In the former case, the procedure accepts (lines 30–31). In the latter, there is a switch in the current operation mode (line 33). For each element (ψ, ρ) in the dual of $\mathcal{W}$—note that the root modality of ψ is either $\langle \overline{\text{E}} \rangle$ or $\langle \overline{\text{B}} \rangle$—the auxiliary ATM procedure `checkFalse` (reported in Appendix E.5) is invoked, which accepts the input $\{(\psi, \rho)\}$ if and only if $\mathcal{K}, \rho \not\models \psi$. The procedure `checkFalse` is the "dual" of `checkTrue`, as it is simply obtained from `checkTrue` by switching *accept* and

Algorithm 7.2 $\texttt{checkTrue}_{(\mathcal{K},\varphi,Lab)}(\mathcal{W})$

while $\mathcal{W}$ is *not* universal **do**
  deterministically select $(\psi, \rho) \in \mathcal{W}$ s.t. ψ is not of the form $[\overline{\mathrm{E}}]\,\psi'$ and $[\overline{\mathrm{B}}]\,\psi'$
  $\mathcal{W} \leftarrow \mathcal{W} \setminus \{(\psi, \rho)\}$
  Case $\psi = r$ with $r \in \mathsf{RE}$
    if $\rho \notin \mathcal{L}(r)$ **then**
      reject the input
  case $\psi = \neg r$ with $r \in \mathsf{RE}$
    if $\rho \in \mathcal{L}(r)$ **then**
      reject the input
  case $\psi = \langle \mathrm{A} \rangle\,\psi'$ or $\psi = [\mathrm{A}]\,\psi'$
    if $\psi \notin Lab(\mathrm{lst}(\rho))$ **then**
      reject the input
  case $\psi = \langle \overline{\mathrm{A}} \rangle\,\psi'$ or $\psi = [\overline{\mathrm{A}}]\,\psi'$
    if $\psi \notin Lab(\mathrm{fst}(\rho))$ **then**
      reject the input
  case $\psi = \psi_1 \vee \psi_2$
    existentially choose $i = 1, 2$
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\psi_i, \rho)\}$
  case $\psi = \psi_1 \wedge \psi_2$
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\psi_1, \rho), (\psi_2, \rho)\}$
  case $\psi = \langle \mathrm{B} \rangle\,\psi'$
    existentially choose $\rho' \in \mathrm{Pref}(\rho)$
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\psi', \rho')\}$
  case $\psi = [\mathrm{B}]\,\psi'$
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\psi', \rho') \mid \rho' \in \mathrm{Pref}(\rho)\}$
  case $\psi = \langle \mathrm{X} \rangle\,\psi'$ with $X \in \{\overline{E}, \overline{B}\}$
    existentially choose an X-*witness* ρ' of ρ for $(\mathcal{K}, \varphi)$
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\psi', \rho')\}$
  EndCase
if $\mathcal{W} = \emptyset$ **then**
  accept the input
else
  universally choose $(\psi, \rho) \in \widetilde{\mathcal{W}}$
  $\texttt{checkFalse}_{(\mathcal{K},\varphi,Lab)}(\{(\psi, \rho)\})$

reject, by switching existential and universal choices, and by converting the last call to `checkFalse` into `checkTrue`. Thus `checkFalse` accepts an input $\mathcal{W}$ if and only if $\mathcal{W}$ is *not* valid.

Notice that the number of alternations of the ATM `check` between existential and universal choices is clearly the number of switches between the calls to the procedures `checkTrue` and `checkFalse`, plus 2, i.e. $\Upsilon(\varphi) + 2$.

The correctness of the procedure `check` and its complexity bound is stated by the following proposition, that immediately implies Theorem 7.3.13.

Proposition 7.3.15. *The ATM* `check` *is a singly exponential-time bounded ATM accepting* FMC*, whose number of alternations on input* $(\mathcal{K}, \varphi)$ *is at most* $\Upsilon(\varphi) + 2$.

The proof of Proposition 7.3.15 is given in details in Appendix E.6: it exploits the exponential small-model property for $\mathsf{A\overline{A}B\overline{BE}}$ (Theorem 7.3.12) which allows us to consider only certificates, that are singly exponential in the size of the input $(\mathcal{K}, \varphi)$, instead of traces of arbitrary length.

Clearly, this algorithm improves on the complexity of the one presented in Section 4.2 for $\mathsf{A\overline{A}B\overline{BE}}$ under homogeneity. This concludes the section.

7.3.3 $\mathbf{AEXP_{pol}}$-hardness of MC for $\mathsf{B\overline{E}}$

In this section we prove that the MC problem for the fragment $\mathsf{B\overline{E}}$, extended with regular expressions, is $\mathbf{AEXP_{pol}}$-hard (implying the $\mathbf{AEXP_{pol}}$-hardness of $\mathsf{A\overline{A}B\overline{BE}}$). The result is obtained by a polynomial-time reduction from a variant of the domino-tiling problem for grids with exponential-length rows and columns, called *alternating multi-tiling problem*.

An instance of this problem is a tuple $\mathcal{I} = (n, D, D_0, H, V, M, D_{acc})$, where: n is a positive *even* natural number encoded in unary; D is a non-empty finite set of *domino types*; $D_0 \subseteq D$ is a set of *initial domino types*; $H \subseteq D \times D$ and $V \subseteq D \times D$ are the *horizontal* and *vertical matching relations*, resp.; $M \subseteq D \times D$ is the *multi-tiling matching relation*; $D_{acc} \subseteq D$ is a set of *accepting domino types*.

A *tiling of* $\mathcal{I}$ is a map assigning a domino type to each cell of a $2^n \times 2^n$ squared grid coherently with the horizontal and vertical matching relations. Formally, a tiling of $\mathcal{I}$ is a map $f : [0, 2^n - 1] \times [0, 2^n - 1] \to D$ such that:

- for all $i, j \in [0, 2^n - 1] \times [0, 2^n - 1]$ with $j < 2^n - 1$, $(f(i, j), f(i, j + 1)) \in H$ (*row-adjacency requirement*);
- for all $i, j \in [0, 2^n - 1] \times [0, 2^n - 1]$ with $i < 2^n - 1$, $(f(i, j), f(i + 1, j)) \in V$ (*column-adjacency requirement*).

The *initial condition* $Init(f) = f(0,0)f(0,1)\cdots f(0,2^n-1)$ of the tiling f is the content of the first row of f. A *multi-tiling of* $\mathcal{I}$ is a tuple $(f_1,\ldots,f_n)$ of n tilings which are coherent w.r.t. the multi-tiling matching relation M, that is, such that:

- for all $i,j \in [0,2^n-1]\times[0,2^n-1]$ and $\ell \in [1,n-1]$, $(f_\ell(i,j), f_{\ell+1}(i,j)) \in M$ (*multi-cell requirement*), and
- $f_n(2^n-1,j) \in D_{acc}$ for some $j \in [0,2^n-1]$ (*acceptance requirement*).

The *alternating multi-tiling problem* for an instance $\mathcal{I}$ is checking whether

$$\forall w_1 \in (D_0)^{2^n}, \exists w_2 \in (D_0)^{2^n}, \ldots, \forall w_{n-1} \in (D_0)^{2^n}, \exists w_n \in (D_0)^{2^n}$$

such that there is a multi-tiling $(f_1,\ldots,f_n)$, where for all $i \in [1,n]$, $Init(f_i) = w_i$. See Figure 7.4 for a visual representation of the alternating multi-tiling problem.

The following complexity result holds (the proof is in Appendix E.7).

Theorem 7.3.16. *The alternating multi-tiling problem is* $\mathbf{AEXP}_{\mathbf{pol}}$*-complete*

The fact that the MC problem for the fragment $B\overline{E}$ with regular expressions is $\mathbf{AEXP}_{\mathbf{pol}}$-hard is an immediate corollary of the following result.

Theorem 7.3.17. *One can construct, in time polynomial in the size of* $\mathcal{I}$*, a finite Kripke structure* $\mathcal{K}_\mathcal{I}$ *and a* $B\overline{E}$ *formula* $\varphi_\mathcal{I}$ *over the set of proposition letters*

$$\mathcal{AP} = D \cup (\{r,c\}\times\{0,1\}) \cup \{\bot, end\},$$

such that $\mathcal{K}_\mathcal{I} \models \varphi_\mathcal{I}$ *if and only if* $\mathcal{I}$ *is a* positive *instance of the alternating multi-tiling problem.*

The rest of this section is devoted to the construction of the Kripke structure $\mathcal{K}_\mathcal{I}$ and the $B\overline{E}$ formula $\varphi_\mathcal{I}$ proving Theorem 7.3.17. Let $\mathcal{AP}$ be as above. The Kripke structure $\mathcal{K}_\mathcal{I}$ is given by $\mathcal{K}_\mathcal{I} = (\mathcal{AP}, S, R, \mu, s_0)$, where $S = \mathcal{AP}$, $s_0 = end$, μ is the identity mapping (we identify a singleton set $\{p\}$ with p), and $R = \{(s,s') \mid s \in \mathcal{AP}\setminus\{end\}, s' \in \mathcal{AP}\}$. Note that the initial state *end* has no successors,[1] and that a trace of $\mathcal{K}_\mathcal{I}$ can be identified with its induced labeling sequence.

The construction of the $B\overline{E}$ formula $\varphi_\mathcal{I}$ is based on a suitable encoding of multi-tilings which is described in the following. The symbols $\{r\}\times\{0,1\}$ and $\{c\}\times\{0,1\}$ in $\mathcal{AP}$ are used to encode the values of two n-bits counters numbering the 2^n rows and columns, respectively, of a tiling.

[1] This violates Definition 2.1.1, but we define the state *end* to have no successors only for technical convenience.

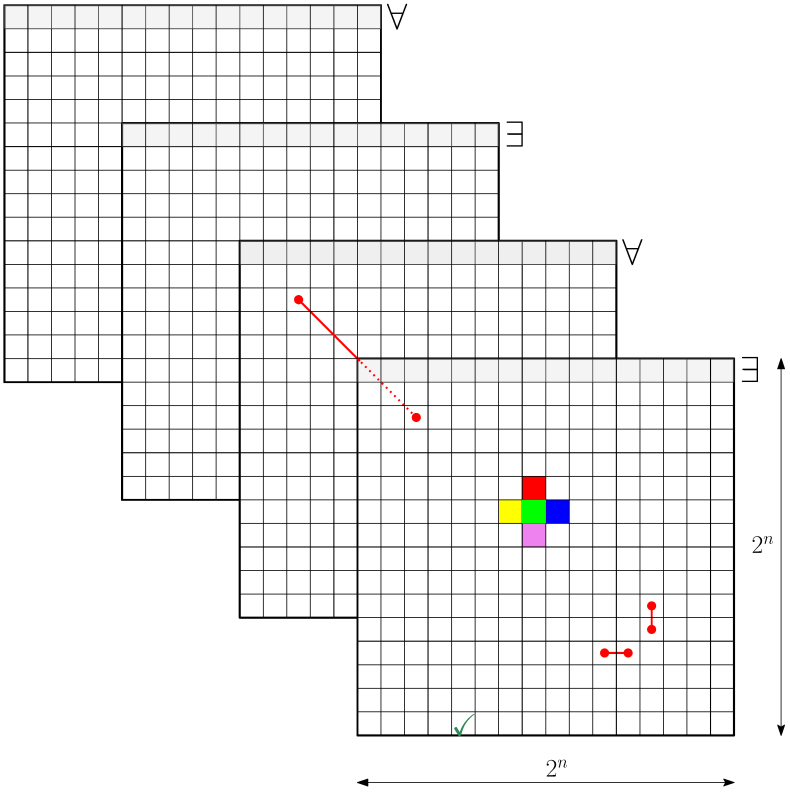


Figure 7.4: The alternating multi-tiling problem (for $n = 4$). The red lines represent the row-adjacency, column-adjacency, and multi-cell requirements. The green tick denotes the acceptance requirement. The quantifiers ∀/∃ associated with the first rows of each tiling mean that the content of these rows has to be universally (resp., existentially) selected, if they belong to an odd (resp., even) tiling.

For a multi-tiling $F = (f_1, \ldots, f_n)$ and for all $i, j \in [0, 2^n - 1]$, the (i, j)-th *multi-cell* $(f_1(i, j), \ldots, f_n(i, j))$ of F is encoded by the word C of length $3n$ over $\mathcal{AP}$, called *multi-cell code*, given by

$$C = d_1 \cdots d_n(r, b_1) \cdots (r, b_n)(c, b'_1) \cdots (c, b'_n),$$

where $b_1 \cdots b_n$ and $b'_1 \cdots b'_n$ are the binary encodings of the row number i and column number j, resp., and for all $\ell \in [1, n]$, $d_\ell = f_\ell(i, j)$ (i.e., the content of the (i, j)-th cell of component f_ℓ). The *content* of C is $d_1 \cdots d_n$. Since F is a multi-tiling, the following well-formedness requirement must be satisfied by the encoding C: for all $\ell \in [1, n-1]$, $(d_\ell, d_{\ell+1}) \in M$. We call such words *well-formed multi-cell codes*.

Definition 7.3.18 (Multi-tiling codes)**.** A *multi-tiling code* is a finite word w over $\mathcal{AP}$ obtained by concatenating well-formed multi-cell codes in such a way that the following conditions hold:

- for all $i, j \in [0, 2^n - 1]$, there is a multi-cell code in w with row number i and column number j *(completeness requirement)*;
- for all multi-cell codes C and C' occurring in w, if C and C' have the same row number and column number, then C and C' have the same content *(uniqueness requirement)*;
- for all multi-cell codes C and C' in w having the same row number (resp., column number), column numbers (resp., row numbers) j and $j + 1$, resp., and contents $d_1 \cdots d_n$ and $d'_1 \cdots d'_n$, resp., it holds that $(d_\ell, d'_\ell) \in H$ (resp. $(d_\ell, d'_\ell) \in V$) for all $\ell \in [1, n]$ *(row-adjacency requirement)* (resp., *(column-adjacency requirement)*);
- there is a multi-cell code in w with row number $2^n - 1$ whose content is in $D^{n-1} \cdot d_{acc}$ for some $d_{acc} \in D_{acc}$ *(acceptance requirement)*.

Finally, we encode the initial conditions of the components of a multi-tiling. An *initial cell code* encodes a cell of the first row of a tiling and is a word w of length $n + 1$ having the form $w = d(c, b_1) \cdots (c, b_n)$, where $d \in D_0$ and $b_1, \ldots, b_n \in \{0, 1\}$. We say that d is the *content* of w and the integer in $[0, 2^n - 1]$ encoded by $b_1 \cdots b_n$ is the *column number* of w.

Definition 7.3.19 (Multi-initialization codes)**.** An *initialization code* is a finite word w over $\mathcal{AP}$ which is the concatenation of initial cell codes such that:

- for all $i \in [0, 2^n - 1]$, there is an initial cell code in w with column number i;
- for all initial cell codes C and C' occurring in w, if C and C' have the same column number, then C and C' have the same content.

A *multi-initialization code* is a finite word over $\mathcal{AP}$ having the form $\bot \cdot w_n \cdots \bot \cdot w_1 \cdot end$ such that for all $\ell \in [1, n]$, w_ℓ is an initialization code.

Definition 7.3.20 (Initialized multi-tiling codes). An *initialized multi-tiling code* is a finite word over $\mathcal{AP}$ having the form $\bot \cdot w \cdot \bot \cdot w_n \cdots \bot \cdot w_1 \cdot end$ such that w is a multi-tiling code, $\bot \cdot w_n \cdots \bot \cdot w_1 \cdot end$ is a multi-initialization code, and the following requirement holds:

- for each multi-cell code in w having row number 0, column number i, and content $d_1 \cdots d_n$, and for all $\ell \in [1, n]$, there is an initial cell code in w_ℓ having column number i and content d_ℓ *(initialization coherence requirement).*

Before proving Theorem 7.3.17, we sketch the idea for the construction of the $\mathrm{B}\overline{\mathrm{E}}$ formula $\varphi_{\mathcal{I}}$ which guarantees that $\mathcal{K}_{\mathcal{I}} \models \varphi_{\mathcal{I}}$ if and only if $\mathcal{I}$ is a positive instance of the alternating multi-tiling problem.

We preliminarily observe that since the initial state of $\mathcal{K}_{\mathcal{I}}$ has no successors, the only initial trace of $\mathcal{K}_{\mathcal{I}}$ is the trace *end* having length 1. To guess a trace corresponding to an initialized multi-tiling code, $\mathcal{K}_{\mathcal{I}}$ is unraveled backward starting from *end*, exploiting the modality $\overline{\mathrm{E}}$. The structure of the formula $\varphi_{\mathcal{I}}$ is

$$\varphi_{\mathcal{I}} = [\overline{\mathrm{E}}](\varphi_1 \rightarrow \langle\overline{\mathrm{E}}\rangle(\varphi_2 \wedge (\ldots([\overline{\mathrm{E}}](\varphi_{n-1} \rightarrow \langle\overline{\mathrm{E}}\rangle(\varphi_n \wedge \langle\overline{\mathrm{E}}\rangle\, \varphi_{IMT})))\ldots))).$$

It features $n + 1$ unravelling steps starting from the initial trace *end*. The first n steps are used to guess a sequence of n initialization codes. Intuitively, each formula φ_i is used to constrain the i-th unravelling to be an initialization code, in such a way that at depth n in the formula a multi-initialization code is under evaluation. The last unravelling step (the innermost in the formula) is used to guess the multi-tiling code. The innermost formula φ_{IMT} is evaluated over a trace corresponding to an initialized multi-tiling code, and checks its structure: multi-cell codes are "captured" by regular expressions (encoding in particular their row and column numbers and contents). The completeness, uniqueness, row- and column-adjacency requirements for the multi-tiling of Definition 7.3.18 are enforced by the combined use of the $[\overline{\mathrm{E}}]$ modality and regular expressions.

The intuition of such a technique is graphically depicted in Figure 7.5, where w is a multi-tiling code. Since the problem is to check constraints between pairs of multi-cell codes occurring in arbitrary positions of w, we use the following trick. A copy of two multi-cell codes C and C' (see Figure 7.5) are generated next to each other, as backward extensions of the initialized multi-tiling code, by means of modality $[\overline{\mathrm{E}}]$. We then check that both C and C' occur in (arbitrary positions of) w, and, if this is the case, the required constraint is checked against the generated copies C and C', taking advantage of their adjacency. The initialization coherence

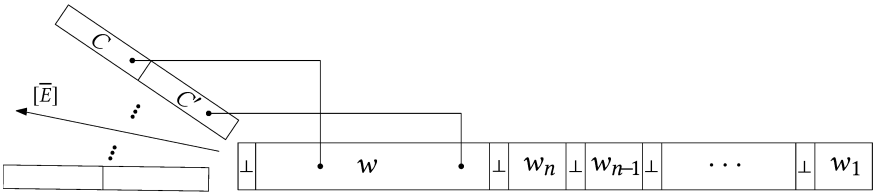


Figure 7.5: Checking constraints between pairs of multi-cell codes C and C' in an initialized multi-tiling code.

requirement of Definition 7.3.20 is guaranteed in an analogous way, by comparing initial cell codes and multi-cell codes.

Note that the first $n-1$ occurrences of alternations between universal and existential modalities $[\overline{E}]$ and $\langle\overline{E}\rangle$ correspond to the alternations of universal and existential quantifications in the definition of alternating multi-tiling problem.

The correctness of the construction of $\varphi_{\mathcal{I}}$ is stated by the next proposition.

Proposition 7.3.21. *One can build, in time polynomial in the size of $\mathcal{I}$, $n+1$ $B\overline{E}$ formulas $\varphi_{IMT}, \varphi_1, \ldots, \varphi_n$ with $\Upsilon(\varphi_{IMT}) = \Upsilon(\varphi_1) = \ldots = \Upsilon(\varphi_n) = 0$, fulfilling the following conditions:*

- *for all finite words ρ over $\mathcal{AP}$ having the form $\rho = \rho' \cdot \bot \cdot w_n \cdots \bot \cdot w_1 \cdot end$ such that $\rho' \neq \varepsilon$ and $\bot \cdot w_n \cdots \bot \cdot w_1 \cdot end$ is a multi-initialization code, it holds that $\mathcal{K}_{\mathcal{I}}, \rho \models \varphi_{IMT}$ if and only if ρ is an initialized multi-tiling code;*
- *for all $\ell \in [1, n]$ and words ρ having the form $\rho = \rho' \cdot \bot \cdot w_{\ell-1} \cdots \bot \cdot w_1 \cdot end$ such that $\rho' \neq \varepsilon$ and $w_j \in (\mathcal{AP} \setminus \{\bot\})^*$ for all $j \in [1, \ell-1]$, it holds that $\mathcal{K}_{\mathcal{I}}, \rho \models \varphi_\ell$ if and only if ρ' has the form $\rho' = \bot \cdot w_\ell$, where w_ℓ is an initialization code.*

Proof. Since each state of the Kripke structure $\mathcal{K}_{\mathcal{I}}$ is labeled by exactly one proposition letter of $\mathcal{AP}$, in the proof we exploit the standard regular expressions, where atomic expressions are single letters of $\mathcal{AP}$. Evidently, a standard regular expression can be converted into a proposition-based one, where each $p \in \mathcal{AP}$ is replaced by the formula $p \wedge \bigwedge_{p' \in \mathcal{AP} \setminus \{p\}} \neg p'$.

Let us focus on the construction of the $B\overline{E}$ formula φ_{IMT} (as $\varphi_1, \ldots, \varphi_n$ are simpler). First, we define a $B\overline{E}$ formula φ_{MT} ensuring the following property:

- for all finite words ρ over $\mathcal{AP}$ having the form $\rho = \rho' \cdot \bot \cdot w_n \cdots \bot \cdot w_1 \cdot end$ such that $\rho' \neq \varepsilon$ and $\bot \cdot w_n \cdots \bot \cdot w_1 \cdot end$ is a multi-initialization code, it holds $\mathcal{K}_{\mathcal{I}}, \rho \models \varphi_{MT}$ if and only if $\rho' = \bot \cdot w$ for some multi-tiling code w.

In order to build φ_{MT}, we need some auxiliary formulas and regular expressions.

- A regular expression $r_{mc} = D^n \cdot (\{r\} \times \{0,1\})^n \cdot (\{c\} \times \{0,1\})^n$ capturing the multi-cell codes.
- A B formula ψ_{comp} requiring that for each word $C \cdot \bot \cdot C_1 \cdots C_N \cdot \bot$ such that $C, C_1, \ldots, C_N$ are multi-cell codes, there is $i \in [1, N]$ such that C and C_i have the same row number and column number.

$$\psi_{comp} = \langle \mathrm{B} \rangle \Big((r_{mc} \cdot \bot \cdot (r_{mc})^+) \wedge \bigwedge_{i \in [1,n]} \bigvee_{b \in \{0,1\}} (\mathcal{AP}^{n+i-1} \cdot (r,b) \cdot \mathcal{AP}^+ \cdot (r,b) \cdot \mathcal{AP}^{2n-i}) \wedge \bigwedge_{i \in [1,n]} \bigvee_{b \in \{0,1\}} (\mathcal{AP}^{2n+i-1} \cdot (c,b) \cdot \mathcal{AP}^+ \cdot (c,b) \cdot \mathcal{AP}^{n-i}) \Big)$$

- A propositional formula $\psi_=$ requiring that for each word having as a proper prefix $C \cdot C'$ such that C and C' are multi-cell codes, C and C' have the same row number and column number.

$$\psi_= = \bigwedge_{i \in [1,n]} \bigvee_{b \in \{0,1\}} (\mathcal{AP}^{n+i-1} \cdot (r,b) \cdot \mathcal{AP}^{3n-1} \cdot (r,b) \cdot \mathcal{AP}^+) \wedge \bigwedge_{i \in [1,n]} \bigvee_{b \in \{0,1\}} (\mathcal{AP}^{2n+i-1} \cdot (c,b) \cdot \mathcal{AP}^{3n-1} \cdot (c,b) \cdot \mathcal{AP}^+)$$

- A propositional formula $\psi_{r,inc}$ (resp., $\psi_{c,inc}$) requiring that for each word having as a proper prefix $C \cdot C'$ such that C and C' are multi-cell codes, C and C' have the same column number (resp., the same row number), and there is $h \in [0, 2^n - 2]$ such that C and C' have row numbers (resp., column numbers) h and $h+1$, resp.. We consider the formula $\psi_{r,inc}$ (the definition of $\psi_{c,inc}$ is similar).

$$\psi_{r,inc} = \bigwedge_{i \in [1,n]} \bigvee_{b \in \{0,1\}} (\mathcal{AP}^{2n+i-1} \cdot (c,b) \cdot \mathcal{AP}^{3n-1} \cdot (c,b) \cdot \mathcal{AP}^+) \wedge \bigvee_{i \in [1,n]} \Big(\bigwedge_{j \in [1,i-1]} (\mathcal{AP}^{n+j-1} \cdot (r,1) \cdot \mathcal{AP}^{3n-1} \cdot (r,0) \cdot \mathcal{AP}^+) \wedge (\mathcal{AP}^{n+i-1} \cdot (r,0) \cdot \mathcal{AP}^{3n-1} \cdot (r,1) \cdot \mathcal{AP}^+) \wedge \bigwedge_{j \in [i+1,n]} \bigvee_{b \in \{0,1\}} (\mathcal{AP}^{n+j-1} \cdot (r,b) \cdot \mathcal{AP}^{3n-1} \cdot (r,b) \cdot \mathcal{AP}^+) \Big)$$

- A B formula ψ_{double} requiring that for each word $C \cdot C' \cdot \bot \cdot C_1 \cdots C_N \cdot \bot$ such that $C, C', C_1, \ldots, C_N$ are multi-cell codes, there are $i, j \in [1, N]$ such that $C = C_i$ and $C' = C_j$; $\psi_{double} = \theta \wedge \theta'$, where θ (resp., θ') requires that there is $i \in [1, N]$ such that $C_i = C$ (resp., $C_i = C'$). We consider θ' (the definition of θ is similar).

$$\theta' = \langle \mathrm{B} \rangle \Big((r_{mc} \cdot r_{mc} \cdot \bot \cdot (r_{mc})^+) \wedge \bigwedge_{i \in [1,n]} \bigvee_{d \in D} (\mathcal{AP}^{3n+i-1} \cdot d \cdot \mathcal{AP}^+ \cdot d \cdot \mathcal{AP}^{3n-i}) \wedge$$
$$\bigwedge_{i \in [1,n]} \bigvee_{b \in \{0,1\}} (\mathcal{AP}^{4n+i-1} \cdot (r,b) \cdot \mathcal{AP}^+ \cdot (r,b) \cdot \mathcal{AP}^{2n-i}) \wedge$$
$$\bigwedge_{i \in [1,n]} \bigvee_{b \in \{0,1\}} (\mathcal{AP}^{5n+i-1} \cdot (c,b) \cdot \mathcal{AP}^+ \cdot (c,b) \cdot \mathcal{AP}^{n-i}) \Big)$$

- A B formula ψ_{not_unique} requiring that for each word $C \cdot C' \cdot \bot \cdot C_1 \cdots C_N \cdot \bot$ such that $C, C', C_1, \ldots, C_N$ are multi-cell codes, the next properties hold:
  - C and C' have the same row and column numbers, but different content;
  - there are $i, j \in [1, N]$ such that $C = C_i$ and $C' = C_j$.

  The construction of ψ_{not_unique} is based on the formulas ψ_{double} and $\psi_{=}$:

$$\psi_{not_unique} = \psi_{double} \wedge \psi_{=} \wedge \bigvee_{i \in [1,n]} \bigvee_{d,d' \in D : d \neq d'} (\mathcal{AP}^{i-1} \cdot d \cdot \mathcal{AP}^{3n-1} \cdot d' \cdot \mathcal{AP}^+).$$

- A B formula ψ_{row} (resp., ψ_{col}) requiring that for each word $C \cdot C' \cdots C_1 \cdots C_N \cdot \bot$ such that $C, C', C_1, \ldots, C_N$ are multi-cell codes, the next condition holds.
  - Let us denote by $d_1 \cdots d_n$ the content of C and by $d'_1 \cdots d'_n$ the content of C'. Whenever (1) there are $i, j \in [1, N]$ such that $C = C_i$ and $C' = C_j$, and (2) C and C' have the same row number and column numbers h and $h+1$, resp. (resp., C and C' have the same column number and row numbers h and $h+1$, resp.) for some $h \in [0, 2^n - 2]$, then it holds that $(d_\ell, d'_\ell) \in H$ (resp., $(d_\ell, d'_\ell) \in V$), for all $\ell \in [1, N]$.

  We focus on ψ_{row} (the definition of ψ_{col} is similar):

$$\psi_{row} = (\psi_{double} \wedge \psi_{c,inc}) \longrightarrow \bigwedge_{i \in [1,n]} \bigvee_{(d,d') \in H} (\mathcal{AP}^{i-1} \cdot d \cdot \mathcal{AP}^{3n-1} \cdot d' \cdot \mathcal{AP}^+).$$

Finally, the $B\overline{E}$ formula φ_{MT} is defined as follows:

$$\neg(\mathcal{AP}^* \cdot \bot \cdot \mathcal{AP}^*)^{n+2} \wedge \langle \mathrm{B} \rangle \Big(\underbrace{(\bot \cdot (r_{mc})^+ \cdot \bot) \wedge \neg \bigvee_{(d,d') \in D^2 \setminus M} (\mathcal{AP}^+ \cdot d \cdot d' \cdot \mathcal{AP}^+)}_{\text{Concatenation of well-formed multi-cell codes}} \wedge$$
$$\underbrace{[\overline{\mathrm{E}}]((r_{mc} \cdot \bot \cdot (r_{mc})^+ \cdot \bot) \longrightarrow \psi_{comp})}_{\text{Completeness requirement of Definition 7.3.18}} \wedge$$
$$\underbrace{[\overline{\mathrm{E}}]((r_{mc} \cdot r_{mc} \cdot \bot \cdot (r_{mc})^+ \cdot \bot) \longrightarrow \neg\psi_{not_unique})}_{\text{Uniqueness requirement of Definition 7.3.18}} \wedge$$

$$\underbrace{[\overline{\mathrm{E}}]((r_{mc} \cdot r_{mc} \cdot \bot \cdot (r_{mc})^+ \cdot \bot) \longrightarrow (\psi_{row} \wedge \psi_{col}))}_{\text{Row-adjacency and column-adjacency requirements of Definition 7.3.18}} \quad \wedge$$

$$\underbrace{\Big(\bigvee_{d_{acc} \in D_{acc}} (\mathcal{AP}^+ \cdot d_{acc} \cdot (r,1)^n \cdot \mathcal{AP}^+)\Big)}_{\text{Acceptance requirement of Definition 7.3.18}}.$$

The B$\overline{\mathrm{E}}$ formula φ_{IMT} is given by $\varphi_{MT} \wedge \varphi_{coh}$, where φ_{coh} ensures the initialization coherence requirement of Definition 7.3.20. In order to define φ_{coh}, we need some auxiliary formulas and regular expressions.

- A regular expression $r_{ic} = D_0 \cdot (\{c\} \times \{0,1\})^n$ capturing the initial cell codes.
- A B formula ψ_{single} requiring that for each word $C \cdot \bot \cdot C_1 \cdots C_N \cdot \bot$ such that $C, C_1, \ldots, C_N$ are multi-cell codes, there is $i \in [1, N]$ such that $C = C_i$ and the row number of C is 0. The definition of ψ_{single} is similar to ψ_{double}.
- A B formula ψ_{coh} requiring that for each word $C \cdot \bot \cdot C_1 \cdots C_N \cdot \bot \cdot w_n \cdots \bot \cdot w_1 \cdot \mathit{end}$ such that $C, C_1, \ldots, C_N$ are multi-cell codes and $\bot \cdot w_n \cdots \bot \cdot w_1 \cdot \mathit{end}$ is a multi-initialization code, the following constraint holds: if there is $i \in [1, N]$ such that $C = C_i$, the row number of C is 0 and the content of C is $d_1 \cdots d_n$, then for all $\ell \in [1, n]$, there exists an initial code in w_ℓ having the same column number as C and content d_ℓ.

$$\psi_{coh} = \big(\langle \mathrm{B} \rangle([(\mathcal{AP} \setminus \{\bot\})^+ \cdot \bot \cdot (\mathcal{AP} \setminus \{\bot\})^+ \cdot \bot] \wedge \psi_{single})\big) \longrightarrow \bigwedge_{\ell \in [1,n]} \psi_\ell;$$

$$\psi_\ell = \langle \mathrm{B} \rangle \Big(\big[(\mathcal{AP} \setminus \{\bot\})^+ \cdot (\bot \cdot (\mathcal{AP} \setminus \{\bot\})^+)^{n-\ell+1} \cdot \bot \cdot r_{ic}^+\big] \wedge \bigwedge_{i \in [1,n]} \bigvee_{b \in \{0,1\}} (\mathcal{AP}^{2n+i-1} \cdot (c,b) \cdot \mathcal{AP}^+ \cdot (c,b) \cdot \mathcal{AP}^{n-i}) \wedge \bigvee_{d \in D} (\mathcal{AP}^{\ell-1} \cdot d \cdot \mathcal{AP}^+ \cdot d \cdot \mathcal{AP}^n) \Big).$$

The B$\overline{\mathrm{E}}$ formula φ_{coh} is then $\varphi_{coh} = [\overline{\mathrm{E}}]\big([r_{mc} \cdot (\bot \cdot (\mathcal{AP} \setminus \{\bot\})^+)^{n+1}] \rightarrow \psi_{coh}\big)$.

This concludes the proof of Proposition 7.3.21. □

Recall that $\varphi_{\mathcal{I}} = [\overline{\mathrm{E}}](\varphi_1 \rightarrow \langle \overline{\mathrm{E}} \rangle(\varphi_2 \wedge (\ldots([\overline{\mathrm{E}}](\varphi_{n-1} \rightarrow \langle \overline{\mathrm{E}} \rangle(\varphi_n \wedge \langle \overline{\mathrm{E}} \rangle \varphi_{IMT}))) \ldots)))$, where $\varphi_{IMT}, \varphi_1, \ldots, \varphi_n$ are the formulas defined in Proposition 7.3.21. Since the initial state of $\mathcal{K}_{\mathcal{I}}$ has no successors and the only initial trace has length 1 and corresponds to the proposition letter *end*, by Definitions 7.3.18–7.3.20, we have that $\mathcal{K}_{\mathcal{I}} \models \varphi_{\mathcal{I}}$ if and only if $\mathcal{I}$ is a positive instance of the alternating multi-tiling problem, proving Theorem 7.3.17. This result, combined with Theorem 7.3.13, implies the following.

Corollary 7.3.22. *The MC problem for* $\mathsf{A\overline{A}B\overline{B}E}$ *(and* $\mathsf{A\overline{A}E\overline{B}E}$*) formulas extended with regular expressions over finite Kripke structures is* $\mathbf{AEXP}_{\mathbf{pol}}$*-complete.*

7.4 The fragments $\mathsf{A\overline{A}B\overline{B}}$ and $\mathsf{A\overline{A}E\overline{E}}$

In this section we show that the two symmetric fragments $\mathsf{A\overline{A}B\overline{B}}$ and $\mathsf{A\overline{A}E\overline{E}}$, extended with regular expressions, feature a better complexity, showing MC for them to be in **PSPACE**. To this end we first prove, in Section 7.4.1, that they feature an *exponential small-model property*, that is, if a trace ρ of a finite Kripke structure $\mathcal{K}$ satisfies a formula φ of $\mathsf{A\overline{A}B\overline{B}}$/$\mathsf{A\overline{A}E\overline{E}}$, then there is always a trace π, whose length is exponential in the sizes of φ and $\mathcal{K}$, starting from and leading to the same states as ρ, that satisfies φ. Therefore, without loss of generality, one can limit the verification of traces of $\mathcal{K}$ to those having at most exponential length. It is worth recalling that, in Section 4.1, we proved a *polynomial small-model property* in the sizes of the $\mathsf{A\overline{A}B\overline{B}}$/$\mathsf{A\overline{A}E\overline{E}}$ formula φ and the Kripke structure $\mathcal{K}$ *under the homogeneity assumption*.

Then, in Section 7.4.2 and 7.4.3 we provide a **PSPACE** MC algorithm which exploits the exponential small-model property. Such an algorithm is completely different from the one presented in Section 4.1 for the MC problem of the same fragments under the homogeneity assumption, which can exploit the aforementioned polynomial small-model property. As a matter of fact, unlike that of Section 4.1, this algorithm cannot store even a single—possibly exponential-length—trace, being bound to use polynomial working space. For this reason it visits the (exponential-length) traces of the input Kripke structure $\mathcal{K}$ by means of a *binary reachability* technique that allows it to use logarithmic space in the length of traces, hence guaranteeing the **PSPACE** complexity upper bound. The surprising fact is that both the algorithm of Section 4.1 and the one presented here use polynomial working space (independently of the different size small-models), thus showing that relaxing the homogeneity assumption comes at no additional computational cost for the fragments $\mathsf{A\overline{A}B\overline{B}}$ and $\mathsf{A\overline{A}E\overline{E}}$.

Finally, in Section 7.4.3 we prove the **PSPACE**-completeness of MC for $\mathsf{A\overline{A}B\overline{B}}$ and $\mathsf{A\overline{A}E\overline{E}}$ with regular expressions.

7.4.1 Exponential small-model property for $\mathsf{A\overline{A}B\overline{B}}$ and $\mathsf{A\overline{A}E\overline{E}}$

In this section we prove the exponential small-model property for the fragments $\mathsf{A\overline{A}B\overline{B}}$ and $\mathsf{A\overline{A}E\overline{E}}$ (actually, we focus only on $\mathsf{A\overline{A}B\overline{B}}$ being the case for $\mathsf{A\overline{A}E\overline{E}}$

symmetric). Most results are just adaptations—with the aim of accounting for regular expressions—of those already presented in Section 4.1.

Given a DFA $\mathcal{D} = (\Sigma, Q, q_0, \delta, F)$, we denote by $\mathcal{D}(w)$ (resp., $\mathcal{D}_q(w)$) the state reached by the computation of $\mathcal{D}$ from q_0 (resp., $q \in Q$) over the word $w \in \Sigma^*$.

We now consider *well-formedness* of induced traces (recall Definition 4.1.1) w.r.t. a set of DFAs:[2] a well formed trace π induced by ρ preserves the states of the computations of the DFAs reached by reading prefixes of ρ and π bounded by corresponding positions. Hereafter, for $i \in [1, |\rho|]$, ρ^i denotes the prefix $\rho(1, i)$.

Definition 7.4.1 (Well-formed trace w.r.t. a set of DFAs)**.** Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure, $\rho \in \mathrm{Trc}_{\mathcal{K}}$ be a trace, and $\mathcal{D}^s = (2^{\mathcal{AP}}, Q^s, q_0^s, \delta^s, F^s)$ for $s = 1, \ldots, k$, be DFAs. A trace $\pi \in \mathrm{Trc}_{\mathcal{K}}$ *induced by* ρ is $(q^1_{\ell_1}, \ldots, q^k_{\ell_k})$*-well-formed w.r.t.* ρ, with $q^s_{\ell_s} \in Q^s$ for all $s = 1, \ldots, k$, if and only if:

- for all π-positions j, with corresponding ρ-positions i_j, and all $s = 1, \ldots, k$, it holds that $\mathcal{D}^s_{q^s_{\ell_s}}(\mu(\pi^j)) = \mathcal{D}^s_{q^s_{\ell_s}}(\mu(\rho^{i_j}))$.

It is easy to see that, for $q^s_{\ell_s} \in Q^s$, $s = 1, \ldots, k$, the $(q^1_{\ell_1}, \ldots, q^k_{\ell_k})$-well-formedness relation is *transitive*.

It is possible to show that a trace, whose length exceeds a suitable exponential threshold, induces a shorter, well-formed trace. Such a contraction pattern (Proposition 7.4.2) represents a "basic step" in a contraction process which will allow us to prove the exponential small-model property for $\overline{\mathsf{AA}}\overline{\mathsf{BB}}$.

Let us consider an $\overline{\mathsf{AA}}\overline{\mathsf{BB}}$ formula φ and let $r_1, \ldots, r_k$ be the RE's over $\mathcal{AP}$ in φ. Let $\mathcal{D}^1, \ldots, \mathcal{D}^k$ be the DFAs such that $\mathcal{L}(\mathcal{D}^t) = \mathcal{L}(r_t)$, for $t = 1, \ldots, k$, where $|Q^t| \leq 2^{2|r_t|}$ (see Remark 7.1.2). We denote $Q^1 \times \ldots \times Q^k$ by $Q(\varphi)$, and $\mathcal{D}^1, \ldots, \mathcal{D}^k$ by $\mathcal{D}(\varphi)$.

Proposition 7.4.2. *Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure, φ be an $\overline{\mathsf{AA}}\overline{\mathsf{BB}}$ formula with* RE*'s $r_1, \ldots, r_k$ over $\mathcal{AP}$, $\rho \in \mathrm{Trc}_{\mathcal{K}}$ be a trace, and $(q^1, \ldots, q^k) \in Q(\varphi)$. There exists a trace $\pi \in \mathrm{Trc}_{\mathcal{K}}$, which is $(q^1, \ldots, q^k)$-well-formed w.r.t. ρ, such that $|\pi| \leq |S| \cdot 2^{2\sum_{\ell=1}^{k} |r_\ell|}$.*

The proof, which is an adaptation of that of Proposition 4.1.3, can be found in Appendix E.8.

The next step is to determine some conditions for contracting traces while preserving the equivalence w.r.t. the satisfiability of the considered $\overline{\mathsf{AA}}\overline{\mathsf{BB}}$ formula. In the following, we restrict ourselves again to formulas in NNF.

[2] Another variant of well-formedness of induced traces, independent of DFAs, was already given in Definition 4.1.1.

For a trace ρ and a formula φ of $\overline{\text{A}}\text{A}\overline{\text{B}}\overline{\text{B}}$ (in NNF), we fix a set of distinguished ρ-positions, called *witness positions* (recall Definition 4.1.4), each one corresponding to the minimum prefix of ρ which satisfies a formula ψ occurring in φ as a subformula of the form $\langle \text{B} \rangle \psi$ (provided that $\langle \text{B} \rangle \psi$ is satisfied by ρ). Such set is denoted by $Wt(\varphi, \rho)$, and we have $|Wt(\varphi, \rho)| \leq |\varphi| - 1$. We can show that, when a contraction is performed in between a pair of *consecutive* witness positions (thus no witness position is ever removed), we get a trace induced by ρ (according to Definition 4.1.1) equivalent with respect to the satisfiability of φ.

We are now ready to state the exponential small-model property for $\overline{\text{A}}\text{A}\overline{\text{B}}\overline{\text{B}}$.

Theorem 7.4.3 (Exponential small-model for $\overline{\text{A}}\text{A}\overline{\text{B}}\overline{\text{B}}$)**.** *Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure, $\sigma, \rho \in \text{Trc}_{\mathcal{K}}$, and φ be an $\overline{\text{A}}\text{A}\overline{\text{B}}\overline{\text{B}}$ formula in NNF, with RE's $r_1, \ldots, r_u$ over $\mathcal{AP}$, such that $\mathcal{K}, \sigma \star \rho \models \varphi$. There exists $\pi \in \text{Trc}_{\mathcal{K}}$, induced by ρ, such that $\mathcal{K}, \sigma \star \pi \models \varphi$ and $|\pi| \leq |S| \cdot (|\varphi| + 1) \cdot 2^{2 \sum_{\ell=1}^{u} |r_\ell|}$.*

The theorem holds in particular if $|\sigma| = 1$, and thus $\sigma \star \rho = \rho$ and $\sigma \star \pi = \pi$. In this case, if $\mathcal{K}, \rho \models \varphi$, then $\mathcal{K}, \pi \models \varphi$, where π is induced by ρ and $|\pi| \leq |S| \cdot (|\varphi| + 1) \cdot 2^{2 \sum_{\ell=1}^{u} |r_\ell|}$. The proof, which is an adaptation of the one of Theorem 4.1.5, can be found in Appendix E.9.

The exponential small-model property allows us to devise a trivial *exponential working space* algorithm for $\overline{\text{A}}\text{A}\overline{\text{B}}\overline{\text{B}}$ (and $\overline{\text{A}}\text{A}\overline{\text{E}}\overline{\text{E}}$)—as already anticipated, we will actually present a polynomial space one in the next sections—which basically unravels the Kripke structure and checks all subformulas of the input formula. At every step it can consider traces not longer than $O(|S| \cdot |\varphi| \cdot 2^{2 \sum_{\ell=1}^{u} |r_\ell|})$. Conversely, the following example shows that the exponential small-model is strict, that is, there exist a formula and a Kripke structure such that the shortest trace satisfying the formula has exponential length in the size of the formula itself. This is the case even for purely propositional formulas (of the HS fragment Prop).

Example 7.4.4. Let pr_i be the i-th smallest prime number. It is well-known that $pr_i \in O(i \log i)$. Let $w^{\otimes k}$ denote the string obtained by concatenating k times w. Let us fix some $n \in \mathbb{N}$, and let $\mathcal{K} = (\{p\}, \{s\}, R, \mu, s)$ be the trivial Kripke structure having only one state with a self-loop, where $R = \{(s, s)\}$, and $\mu(s) = \{p\}$. The shortest trace satisfying

$$\psi = \bigwedge_{i=1}^{n} (p^{\otimes (pr_i)})^*$$

is $\rho = s^{\otimes (pr_1 \cdots pr_n)}$, since its length is the least common multiple of $pr_1, \ldots, pr_n$, which is, indeed, $pr_1 \cdots pr_n$. It is immediate to check that the length of ψ is $O(n \cdot pr_n) = O(n^2 \log n)$. On the other hand, the length of ρ is $pr_1 \cdots pr_n \geq 2^n$.

In the following, we will exploit the exponential small-model property of the two symmetrical fragments $\mathsf{A\overline{A}B\overline{B}}$ and $\mathsf{A\overline{A}E\overline{E}}$ to prove the **PSPACE**-completeness of their MC problems. First, in Section 7.4.2, we will provide a **PSPACE** MC algorithm for $\mathsf{B\overline{B}}$ (resp., $\mathsf{E\overline{E}}$). Then, in Section 7.4.3, we will show that the *meets* and *met-by* modalities A and $\overline{\mathsf{A}}$ can be suitably encoded by regular expressions without increasing the complexity of $\mathsf{B\overline{B}}$ (resp., $\mathsf{E\overline{E}}$).

7.4.2 PSPACE MC algorithm for $\mathsf{B\overline{B}}$

In this section, to start with, we describe a **PSPACE** MC algorithm for $\mathsf{B\overline{B}}$ formulas, extended with regular expressions. W.l.o.g., we assume that the processed formulas do not contain occurrences of the universal modalities $[\mathsf{B}]$ and $[\overline{\mathsf{B}}]$. Moreover, for a formula ψ, we denote by $\mathsf{Subf}_{\langle \mathsf{B} \rangle}(\psi) = \{\varphi \mid \langle \mathsf{B} \rangle \varphi \text{ is a subformula of } \psi\}$. In such an algorithm, Φ represents the overall formula to be checked, while the parametric formula ψ ranges over its subformulas.

Due to the result of the previous section, the algorithm can consider only traces having length bounded by the exponential small-model property. Note that an algorithm required to work in polynomial space cannot explicitly store the DFAs for the regular expressions occurring in Φ (their states are *exponentially* many in the length of the associated regular expressions). For this reason, while checking a formula against a trace, the algorithm just stores the *current states* of the computations of the DFAs associated with the regular expressions in Φ, from the respective initial states (in the following such states are denoted—with a little abuse of notation—again by $\mathcal{D}(\Phi)$, and called the *"current configuration" of the DFAs*) and calculates on-the-fly the successor states in the DFAs, once they have read some state of $\mathcal{K}$ used to extend the considered trace (this can be done by exploiting a *succinct* encoding of the NFAs for the regular expressions of Φ, see again Remark 7.1.2 in Section 7.1).

A call to the recursive procedure $\texttt{Check}(\mathcal{K}, \psi, s, G, \mathcal{D}(\Phi))$ (Algorithm 7.3) verifies the satisfiability of a subformula ψ of Φ w.r.t. any trace ρ fulfilling the next conditions:

1. $G \subseteq \mathsf{Subf}_{\langle \mathsf{B} \rangle}(\psi)$ is the set of formulas that hold true on at least a prefix of ρ;
2. after reading $\mu(\rho(1, |\rho| - 1))$ the current configuration of the DFAs for the regular expressions of Φ is $\mathcal{D}(\Phi)$;
3. the last state of ρ is s.

Intuitively, since the algorithm cannot store the already checked portion of a trace (whose length could be exponential), the relevant information is *summarized* in a triple $(G, \mathcal{D}(\Phi), s)$. Hereafter, the set of all possible summarizing triples $(\overline{G}, \overline{\mathcal{D}(\Phi)}, \overline{s})$, where $\overline{G} \subseteq \mathsf{Subf}_{\langle \mathsf{B} \rangle}(\psi)$, $\overline{\mathcal{D}(\Phi)}$ is any current configuration of the DFAs for the regular expressions of Φ, and $\overline{s}$ is a state of $\mathcal{K}$, is denoted by $\mathsf{Conf}(\mathcal{K}, \psi)$.

Algorithm 7.3 `Check`$(\mathcal{K}, \psi, s, G, \mathcal{D}(\Phi))$

1: **if** $\psi = r$ **then** ◁ r is a regular expression
2: **if** the current state of the DFA for r in `advance`$(\mathcal{D}(\Phi), \mu(s))$ is final **then**
3: **return** $\top$
4: **else**
5: **return** $\bot$
6: **else if** $\psi = \neg\psi'$ **then**
7: **return not** `Check`$(\mathcal{K}, \psi', s, G, \mathcal{D}(\Phi))$
8: **else if** $\psi = \psi_1 \wedge \psi_2$ **then**
9: **return** `Check`$(\mathcal{K}, \psi_1, s, G \cap \mathsf{Subf}_{\langle \mathrm{B} \rangle}(\psi_1), \mathcal{D}(\Phi))$ **and** `Check`$(\mathcal{K}, \psi_2, s, G \cap \mathsf{Subf}_{\langle \mathrm{B} \rangle}(\psi_2), \mathcal{D}(\Phi))$
10: **else if** $\psi = \langle \mathrm{B} \rangle \psi'$ **then**
11: **if** $\psi' \in G$ **then**
12: **return** $\top$
13: **else**
14: **return** $\bot$
15: **else if** $\psi = \langle \overline{\mathrm{B}} \rangle \psi'$ **then**
16: **for** each $b \in \{1, \ldots, |S| \cdot (2|\psi'|+1) \cdot 2^{2\sum_{\ell=1}^{u} |r_\ell|} - 1\}$ and each $(G', \mathcal{D}(\Phi)', s') \in \mathsf{Conf}(\mathcal{K}, \psi)$ **do** ◁ $r_1, \ldots, r_u$ are the regular expressions of ψ'
17: **if** `Reach`$(\mathcal{K}, \psi', (G, \mathcal{D}(\Phi), s), (G', \mathcal{D}(\Phi)', s'), b)$ **and** `Check`$(\mathcal{K}, \psi', s', G', \mathcal{D}(\Phi)')$ **then**
18: **return** $\top$
19: **return** $\bot$

Let us consider in detail the body of the procedure. First, `advance`$(\mathcal{D}(\Phi), \mu(s))$, invoked at line 2, updates the current configuration of the DFAs after reading the symbol $\mu(s)$. If ψ is a regular expression r (lines 1–5), we just check whether the (computation of the) DFA associated with r is in a final state (i.e., the summarized trace is accepted). Boolean connectives are easily dealt with recursively (lines 6–9). If ψ has the form $\langle \mathrm{B} \rangle \psi'$ (lines 10–14), then ψ' has to hold over a proper prefix of the summarized trace, i.e. ψ' must belong to G.

The only involved case is $\psi = \langle \overline{\mathrm{B}} \rangle \psi'$ (lines 15–19): we have to unravel the Kripke structure $\mathcal{K}$ to find an *extension* ρ' of ρ, summarized by the triple $(G', \mathcal{D}(\Phi)', s')$, satisfying ψ'. The idea is checking whether there exists a summarized trace $(G', \mathcal{D}(\Phi)', s')$, suitably extending $(G, \mathcal{D}(\Phi), s)$, namely, such that:

1. $\mathcal{D}(\Phi)'$ and s' are *synchronously* reachable from $\mathcal{D}(\Phi)$ and s, respectively;
2. $G' \supseteq G$ contains any formula of $\mathsf{Subf}_{\langle \mathrm{B} \rangle}(\psi')$ satisfied by a prefix of the extension;
3. the extension $(G', \mathcal{D}(\Phi)', s')$ satisfies ψ'.

Algorithm 7.4 `Reach`$(\mathcal{K}, \psi, (G_1, \mathcal{D}(\Phi)_1, s_1), (G_2, \mathcal{D}(\Phi)_2, s_2), b)$

1: **if** $b = 1$ **then**
2: **return** `Compatible`$(\mathcal{K}, \psi, (G_1, \mathcal{D}(\Phi)_1, s_1), (G_2, \mathcal{D}(\Phi)_2, s_2))$
3: **else** ◃ $b \geq 2$
4: $b' \leftarrow \lfloor b/2 \rfloor$
5: **for** each $(G_3, \mathcal{D}(\Phi)_3, s_3) \in \mathsf{Conf}(\mathcal{K}, \psi)$ **do**
6: **if** `Reach`$(\mathcal{K}, \psi, (G_1, \mathcal{D}(\Phi)_1, s_1), (G_3, \mathcal{D}(\Phi)_3, s_3), b')$ **and** `Reach`$(\mathcal{K}, \psi, (G_3, \mathcal{D}(\Phi)_3, s_3), (G_2, \mathcal{D}(\Phi)_2, s_2), b - b')$ **then**
7: **return** $\top$
8: **return** $\bot$

Algorithm 7.5 `Compatible`$(\mathcal{K}, \psi, (G_1, \mathcal{D}(\Phi)_1, s_1), (G_2, \mathcal{D}(\Phi)_2, s_2))$

1: **if** $(s_1, s_2) \in R$ **and** `advance`$(\mathcal{D}(\Phi)_1, \mu(s_1)) = \mathcal{D}(\Phi)_2$ **and** $G_1 \subseteq G_2$ **then**
2: **for** each $\varphi \in (G_2 \setminus G_1)$ **do**
3: $G \leftarrow G_1 \cap \mathsf{Subf}_{\langle \mathrm{B} \rangle}(\varphi)$
4: **if** `Check`$(\mathcal{K}, \varphi, s_1, G, \mathcal{D}(\Phi)_1) = \bot$ **then**
5: **return** $\bot$
6: **for** each $\varphi \in (\mathsf{Subf}_{\langle \mathrm{B} \rangle}(\psi) \setminus G_2)$ **do**
7: $G \leftarrow G_1 \cap \mathsf{Subf}_{\langle \mathrm{B} \rangle}(\varphi)$
8: **if** `Check`$(\mathcal{K}, \varphi, s_1, G, \mathcal{D}(\Phi)_1) = \top$ **then**
9: **return** $\bot$
10: **return** $\top$
11: **else**
12: **return** $\bot$

In order to check (1.), i.e., synchronous reachability, we can exploit the exponential small-model property and consider only the unravelling of $\mathcal{K}$ starting from s having depth at most $|S| \cdot (2|\psi'| + 1) \cdot 2^{2 \sum_{\ell=1}^{u} |r_\ell|} - 1$[3]. The verification of (1.) and (2.) is performed by the procedure `Reach` (Algorithm 7.4), which accepts as input two summarized traces and a bound b on the depth of the unravelling of $\mathcal{K}$. The proposed reachability algorithm is reminiscent of the binary reachability technique of Savitch's theorem [GJ79].

The procedure `Reach` proceeds recursively (lines 3–8) by halving at each step the value b of the length bound, until it gets called over two states s_1 and s_2 which are adjacent in a trace. At each halving step, an intermediate summarizing triple is generated to be associated with the split point. At the base of recursion (for $b = 1$, lines 1–2), the auxiliary procedure `Compatible` (Algorithm 7.5) is invoked.

[3] The factor 2 of $|\psi'|$ is added since the exponential small-model for $\mathrm{A}\overline{\mathrm{A}}\mathrm{B}\overline{\mathrm{B}}$ requires a formula to be in NNF.

At line 1, `Compatible` checks whether there is an edge between s_1 and s_2 ($(s_1, s_2) \in R$), and if, at the considered step, the current configuration of the DFAs $\mathcal{D}(\Phi)_1$ is transformed into the configuration $\mathcal{D}(\Phi)_2$ (i.e., s_2 and $\mathcal{D}(\Phi)_2$ are synchronously reachable from s_1 and $\mathcal{D}(\Phi)_1$). At lines 2–9, `Compatible` checks that each formula φ in $(G_2 \setminus G_1)$, where $G_2 \supseteq G_1$, is satisfied by a trace summarized by $(G_1, \mathcal{D}(\Phi)_1, s_1)$ (lines 2–5). Intuitively, $(G_1, \mathcal{D}(\Phi)_1, s_1)$ summarizes the maximal prefix of $(G_2, \mathcal{D}(\Phi)_2, s_2)$, and thus a subformula satisfied by a prefix of a trace summarized by $(G_2, \mathcal{D}(\Phi)_2, s_2)$ either belongs to G_1 or it is satisfied by the trace summarized by $(G_1, \mathcal{D}(\Phi)_1, s_1)$. Moreover, (lines 6–9) `Compatible` checks that G_2 is maximal (i.e., no subformula that must be in G_2 has been forgotten). Note that by exploiting this binary reachability technique, the recursion depth of `Reach` is logarithmic in the length of the trace to be visited, hence it can use only polynomial space.

Theorem 7.4.5, proved in Appendix E.10, establishes the soundness of `Check`.

Theorem 7.4.5. *Let Φ be a $\mathrm{B\overline{B}}$ formula, ψ be a subformula of Φ, and $\rho \in \mathrm{Trc}_{\mathcal{K}}$ be a trace with $s = \mathrm{lst}(\rho)$. Let G be the subset of formulas in $\mathrm{Subf}_{\langle \mathrm{B} \rangle}(\psi)$ that hold on some proper prefix of ρ. Let $\mathcal{D}(\Phi)$ be the current configuration of the DFAs associated with the regular expressions in Φ after reading $\mu(\rho(1, |\rho| - 1))$. Then $\mathtt{Check}(\mathcal{K}, \psi, s, G, \mathcal{D}(\Phi)) = \top \iff \mathcal{K}, \rho \models \psi$.*

Algorithm 7.6 reports the main MC procedure `CheckAux`$(\mathcal{K}, \Phi)$ for $\mathrm{B\overline{B}}$. It starts constructing the NFAs and the initial states of the DFAs for the regular expressions of Φ (line 1). Then `CheckAux` invokes the procedure `Check` two times (line 2): the former to check the special case of the trace s_0 consisting of the initial state of $\mathcal{K}$ only, and the latter for all right-extensions of s_0 (i.e., the initial traces having length at least 2). Notice that the NFAs and DFAs for the regular expressions of $\langle \overline{\mathrm{B}} \rangle \neg\Phi$, $\neg\Phi$ and Φ are the same (i.e. $\mathcal{D}(\Phi)_0 = \mathcal{D}(\langle \overline{\mathrm{B}} \rangle \neg\Phi)_0 = \mathcal{D}(\neg\Phi)_0$), allowing us to simultaneously apply the result of Theorem 7.4.5 for both the invocations of `Check` at line 2.

The next theorem, proved in Appendix E.11, establishes the soundness and completeness of `CheckAux`.

○

Algorithm 7.6 `CheckAux`$(\mathcal{K}, \Phi)$

1: `create`$(\mathcal{D}(\Phi)_0)$ ◁ Creates the (succinct) NFAs and the initial states of the DFAs for all the REs in Φ
2: **if** `Check`$(\mathcal{K}, \neg\Phi, s_0, \emptyset, \mathcal{D}(\Phi)_0)$ **or** `Check`$(\mathcal{K}, \langle \overline{\mathrm{B}} \rangle \neg\Phi, s_0, \emptyset, \mathcal{D}(\Phi)_0)$ **then**
3: **return** $\bot$
4: **else**
5: **return** $\top$

Theorem 7.4.6. *Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure, and Φ be a $\mathrm{B}\overline{\mathrm{B}}$ formula. Then, $\texttt{CheckAux}(\mathcal{K}, \Phi) = \top \iff \mathcal{K} \models \Phi$.*

The next corollary states the upper bound to the complexity of MC for $\mathrm{B}\overline{\mathrm{B}}$.

Corollary 7.4.7. *The MC problem for $\mathrm{B}\overline{\mathrm{B}}$ formulas extended with regular expressions over finite Kripke structures is in **PSPACE**.*

Proof. The procedure `CheckAux` decides the problem using *polynomial working space* basically due to two facts. The first one is the number of simultaneously active recursive calls of `Check`, which is $O(|\Phi|)$. The second is the space (in bits) used for any call of `Check`, that is,

$$O\Big(|\Phi| + |S| + \sum_{\ell=1}^{u} |r_\ell| + \underbrace{\log(|S| \cdot |\Phi| \cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|})}_{(1)} + \underbrace{(|\Phi| + |S| + \sum_{\ell=1}^{u} |r_\ell|)}_{(2)} \cdot \underbrace{\log(|S| \cdot |\Phi| \cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|})}_{(3)}\Big),$$

In particular, (1) $O(\log(|S| \cdot |\Phi| \cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|}))$ bits are used for the bound b on the trace length, (3) for *each subformula* $\langle \overline{\mathrm{B}} \rangle \psi'$ of Φ at most $O(\log(|S| \cdot |\Phi| \cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|}))$ recursive calls of `Reach` may be simultaneously active (the recursion depth of `Reach` is logarithmic in b), and (2) each call of `Reach` requires $O(|\Phi| + |S| + \sum_{\ell=1}^{u} |r_\ell|)$ bits. □

Finally, since a Kripke structure can be unravelled against the direction of its edges, and a language $\mathcal{L}$ is regular if and only if its reversed version $\mathcal{L}^{\mathrm{Rev}} = \{w(|w|) \cdots w(1) \mid w \in \mathcal{L}\}$ is, the proposed algorithm can be easily modified to deal with the symmetric fragment $\mathrm{E}\overline{\mathrm{E}}$, proving that also the MC problem for $\mathrm{E}\overline{\mathrm{E}}$ is in **PSPACE**.

7.4.3 PSPACE-completeness of MC for $\mathrm{A}\overline{\mathrm{A}}\mathrm{B}\overline{\mathrm{B}}$

We now show that the algorithm `CheckAux` can be used as a basic engine to design a **PSPACE** MC algorithm for the bigger fragment $\mathrm{A}\overline{\mathrm{A}}\mathrm{B}\overline{\mathrm{B}}$.

The idea is that, being proposition letters (related with) regular expressions, modalities $\langle \mathrm{A} \rangle$ and $\langle \overline{\mathrm{A}} \rangle$ do not augment the expressiveness of the fragment $\mathrm{B}\overline{\mathrm{B}}$. In particular, we will show that the occurrences of modalities $\langle \mathrm{A} \rangle$ and $\langle \overline{\mathrm{A}} \rangle$ in an $\mathrm{A}\overline{\mathrm{A}}\mathrm{B}\overline{\mathrm{B}}$ formula can suitably be "absorbed" and replaced by fresh proposition letters.

We recall that $\mathcal{K}, \rho \models \langle \mathrm{A} \rangle \psi$ if and only if there exists a trace $\tilde{\rho} \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{lst}(\rho) = \mathrm{fst}(\tilde{\rho})$ and $\mathcal{K}, \tilde{\rho} \models \psi$. An immediate consequence is that, for any $\rho' \in \mathrm{Trc}_{\mathcal{K}}$ with $\mathrm{lst}(\rho) = \mathrm{lst}(\rho')$, $\mathcal{K}, \rho \models \langle \mathrm{A} \rangle \psi \iff \mathcal{K}, \rho' \models \langle \mathrm{A} \rangle \psi$ and similarly for the symmetrical modality $\langle \overline{\mathrm{A}} \rangle$ with respect to the first state of the trace. In general, if two traces have the same final state (resp., first state), either both of them satisfy a formula $\langle \mathrm{A} \rangle \psi$ (resp., $\langle \mathrm{A} \rangle \psi$), or none of them does.

As a consequence, for a formula $\langle \mathrm{A} \rangle \psi$ (resp., $\langle \overline{\mathrm{A}} \rangle \psi$), we can determine the subset $S_{\langle \mathrm{A} \rangle \psi}$ (resp., $S_{\langle \overline{\mathrm{A}} \rangle \psi}$) of the set of states S of the Kripke structure such that, for any $\rho \in \mathrm{Trc}_{\mathcal{K}}$, $\mathcal{K}, \rho \models \langle \mathrm{A} \rangle \psi$ (resp., $\mathcal{K}, \rho \models \langle \overline{\mathrm{A}} \rangle \psi$) if and only if $\mathrm{lst}(\rho) \in S_{\langle \mathrm{A} \rangle \psi}$ (resp., $\mathrm{fst}(\rho) \in S_{\langle \overline{\mathrm{A}} \rangle \psi}$).

Now, for a formula $\langle \mathrm{A} \rangle \psi$ (resp., $\langle \overline{\mathrm{A}} \rangle \psi$), we provide a regular expression $r_{\langle \mathrm{A} \rangle \psi}$ (resp., $r_{\langle \overline{\mathrm{A}} \rangle \psi}$) characterizing the set of traces which model the formula. To this end we identify the states in S by a set of fresh proposition letters $\{q_s \mid s \in S\}$ and we replace the Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ by $\mathcal{K}' = (\mathcal{AP}', S, R, \mu', s_0)$, with $\mathcal{AP}' = \mathcal{AP} \cup \{q_s \mid s \in S\}$ and $\mu'(s) = \{q_s\} \cup \mu(s)$ for any $s \in S$. The regular expressions $r_{\langle \mathrm{A} \rangle \psi}$ and $r_{\langle \overline{\mathrm{A}} \rangle \psi}$ are

$$r_{\langle \mathrm{A} \rangle \psi} = \top^* \cdot \Big(\bigcup_{s \in S_{\langle \mathrm{A} \rangle \psi}} q_s \Big) \quad \text{and} \quad r_{\langle \overline{\mathrm{A}} \rangle \psi} = \Big(\bigcup_{s \in S_{\langle \overline{\mathrm{A}} \rangle \psi}} q_s \Big) \cdot \top^*.$$

By definition $\mathcal{K}, \rho \models r_{\langle \mathrm{A} \rangle \psi}$ if and only if $\mathrm{lst}(\rho) \in S_{\langle \mathrm{A} \rangle \psi}$, if and only if $\mathcal{K}, \rho \models \langle \mathrm{A} \rangle \psi$.

We can now sketch the procedure for "reducing" the MC problem for $\mathsf{A\overline{A}B\overline{B}}$ to the MC problem for $\mathsf{B\overline{B}}$: we iteratively rewrite a formula Φ of $\mathsf{A\overline{A}B\overline{B}}$ until it gets converted to an (equivalent) formula of $\mathsf{B\overline{B}}$. At each step, we select an occurrence of a subformula of Φ, either having the form $\langle \mathrm{A} \rangle \psi$ or $\langle \overline{\mathrm{A}} \rangle \psi$, devoid of any occurrence of modalities $\langle \mathrm{A} \rangle$ and $\langle \overline{\mathrm{A}} \rangle$ in ψ. For such an occurrence, say $\langle \mathrm{A} \rangle \psi$, we have to compute the set $S_{\langle \mathrm{A} \rangle \psi}$. For that purpose we can run a variant `CheckAux'`$(\mathcal{K}, \Psi, s)$ of the MC procedure `CheckAux`$(\mathcal{K}, \Psi)$, which invokes `Check` at line 2 on the additional parameter (state) s, instead of s_0. For each $s \in S$, we invoke `CheckAux'`$(\mathcal{K}, \neg\psi, s)$, deciding that $s \in S_{\langle \mathrm{A} \rangle \psi}$ if and only if the procedure returns $\bot$. Then we *replace* $\langle \mathrm{A} \rangle \psi$ in Φ with the regular expression $r_{\langle \mathrm{A} \rangle \psi}$, obtaining a formula Φ'. To deal with subformulas of the form $\langle \overline{\mathrm{A}} \rangle \psi$, we have to introduce a slight variant of the procedure `Check`, which finds traces leading to (and not starting from) a given state. Now, if the resulting formula Φ' is in $\mathsf{B\overline{B}}$, the rewriting process ends; otherwise, we can perform another rewriting step over Φ'.

Considering that the sets $S_{\langle \mathrm{A} \rangle \psi}$, $S_{\langle \overline{\mathrm{A}} \rangle \psi}$ and the regular expressions $r_{\langle \mathrm{A} \rangle \psi}$ and $r_{\langle \overline{\mathrm{A}} \rangle \psi}$ have a size linear in $|S|$, we can conclude with the following result.

Theorem 7.4.8. *The MC problem for $\mathsf{A\overline{A}B\overline{B}}$ formulas extended with regular expressions over finite Kripke structures is in **PSPACE**.*

By symmetry we can show that the MC problem for $\mathsf{A\overline{A}E\overline{E}}$ is also in **PSPACE**.

The **PSPACE**-hardness of MC for $\mathsf{B\overline{B}}$ and $\mathsf{A\overline{A}B\overline{B}}$ directly follows from that of the smallest fragment Prop (the purely propositional fragment of HS), which is stated by Theorem 7.4.9: in Appendix E.12, we prove that Prop is hard for **PSPACE** by a reduction from the **PSPACE**-complete *universality problem for regular expressions* [GJ79], namely, the problem of deciding, for a (standard) regular expression r with $\mathcal{L}(r) \subseteq \Sigma^*$ and $|\Sigma| \geq 2$, whether $\mathcal{L}(r) = \Sigma^*$ or not.

Theorem 7.4.9. *The MC problem for* Prop *formulas extended with regular expressions over finite Kripke structures is* ***PSPACE****-hard (under polynomial-time reductions).*

By Theorem 7.4.8 and Theorem 7.4.9 we obtain the following complexity result.

Theorem 7.4.10. *The MC problem for formulas of any (proper or improper) sub-fragment of* $\mathsf{A\overline{A}B\overline{B}}$ *(and* $\mathsf{A\overline{A}E\overline{E}}$*) extended with regular expressions over finite Kripke structures is* ***PSPACE****-complete.*

7.5 Conclusions

In this chapter we have studied the MC problem for HS extended with regular expressions used to define interval labelling. The approach, stemming from [LM16], generalizes both the one of the previous chapters, in which we enforce the homogeneity principle, and of [LM13; LM14] where labeling is endpoint-based. In the general case, MC for (full) HS turns out to be nonelementarily decidable—the proof exploits an automata-theoretic approach based on the notion of $\mathcal{K}$-NFA—but, for a constant-length formula, it is in **P**. Moreover, the problem is **EXPSPACE**-hard (the hardness follows from that of BE under homogeneity).

We have also investigated the MC problem for two maximal fragments of HS, namely $\mathsf{A\overline{A}B\overline{B}E}$ and $\mathsf{A\overline{A}E\overline{B}E}$ with regular expressions, and we have showed that it is $\mathbf{AEXP_{pol}}$-complete. The complexity upper bound has been proved by providing an alternating algorithm which performs an exponential number of computation steps, but only polynomially many alternations (in the length of the formula to be checked). Conversely, the lower bound has been shown by a reduction from the $\mathbf{AEXP_{pol}}$-complete alternating multi-tiling problem. In this way, we have also improved the known complexity result for the same fragments under the homogeneity assumption.

Finally, we have proved that $\mathsf{A\overline{A}B\overline{B}}$ and $\mathsf{A\overline{A}E\overline{E}}$ and all their sub-fragments are **PSPACE**-complete. The bedrock is a small-model property that allows us to restrict the verification of formulas of $\mathsf{A\overline{A}B\overline{B}}$/$\mathsf{A\overline{A}E\overline{E}}$ to traces having at most exponential length. The matching complexity lower bound has been proved by a reduction from the **PSPACE**-complete universality problem for regular expressions.

Part IV

Interval-based system models

8 Interval-based system models: timelines

The references for this chapter are [Boz+18b; Boz+18a; Boz+18c].

Contents

8.1 The TP Problem . . . 198
8.2 TP over dense temporal domains is an undecidable problem . . . 202
8.3 Decidability of future TP with simple trigger rules . . . 210
8.3.1 Timed automata and the logic MTL . . . 210
8.3.2 Reduction to existential MC for TAs against MTL . . . 213
8.4 Future TP, simple trigger rules, non-singular intervals: hardness 222
8.5 TP with trigger-less rules only is NP-complete . . . 225
8.6 MC for MITL over timelines . . . 231
8.7 Conclusions . . . 235

In all the previous chapters, we have assumed finite Kripke structures as system models. On the positive side, these labelled state-transition graphs are simple, in that no "feasibility check" needs to be performed over them and the described system (the latter exists by definition of the structure itself); moreover they are commonly employed for several industrial purposes, being many modeling languages translated into them, before the MC process (running phase) starts. On the negative side, they are "inherently point-based", as they make explicit how a system evolves *state-by-state* (i.e., how from a state it can move to another one, according to the transition function), and describe which are the atomic properties (proposition letters) that hold true at every state.

In this chapter, we study a possible replacement of Kripke structures by a more expressive model, which allows us to describe systems in terms of their *interval-based* behaviour and properties. We first identified as natural candidates *interval graphs* [Fis85], on which there has been a good deal of work, also from the algorithmic point of view; however, we immediately realized that they are too basic to capture meaningful properties of systems. We then turned to a different kind of structures, called *timelines*, which have been fruitfully exploited in temporal planning for quite a long time. For this reason, we want now to start a short digression on timeline-based planning, and come back to MC later, explaining why the former is a sort of "necessary condition" for the latter, which can then be solved straightforwardly once the former has, under certain conditions.

Timeline-based planning (TP for short) can be viewed as an alternative to the classic action-based approach to planning. Action-based planning aims at determining a sequence of actions that, given the initial state of the world and a goal, transforms, step by step, the state of the world until we reach a state satisfying the goal. TP focuses on what has to happen in order to satisfy the goal instead of what an agent has to do, and thus it can be considered as a more declarative approach with respect to action-based planning: it models the planning domain as a set of independent (but interacting) components, each one consisting of a number of *state variables*. The evolution of the values of state variables over time is described by means of a set of *timelines* (in turn these are sequences of time intervals called *tokens*), and it is governed by a set of transition functions, one for each state variable, and a set of synchronization rules, that constrain the temporal relations among (values of) state variables. This standard lexicon of TP—that some readers, from our experience, may find misleading—will be formally defined and become clearer in the next sections. In the meanwhile, looking (ahead) at Figure 8.1 may give an intuition.

TP has been successfully exploited in a number of application domains, for instance, in space missions, constraint solving, activity scheduling (see, e.g., [Bar+12; Ces+07; Chi+10; FJ03; Jón+00; Mus94]), but a systematic study of its expressiveness and complexity has been undertaken only very recently. The temporal domain is commonly assumed to be *discrete*. In [Gig+16], Gigante et al. showed that TP with bounded temporal relations and token durations, and no temporal horizon, is **EXPSPACE**-complete and expressive enough to capture action-based temporal planning. Later, in [Gig+17], they proved that **EXPSPACE**-completeness still holds for TP with unbounded interval relations, and that the problem becomes **NEXPTIME**-complete if an upper bound to the temporal horizon is added.

In the following sections we will study TP over a *dense temporal domain* (without having recourse to any form of artificial discretization, which is quite a common trick). The reason why we assume this different version of time domain is to avoid discreteness in system descriptions, which can then be abstracted at a higher level, enabling us to neglect details which are unnecessary, and paving the

way for a really interval-based MC: in this respect TP can be regarded, as we said, as a sort of necessary condition for MC, the former playing the role of a "feasibility check" of the system description (which is not immediately feasible by definition—as opposed to Kripke structures—given the presence of synchronization rules). Moreover, if both the system model and the specifications (temporal formulas) can be translated into a common formalism (in our case, as we will show, *timed automata*), "adding" MC on top of TP is just a matter of technical aspects. This is why we shall now focus mainly on TP, and come back to MC at the end of the chapter.

In the next sections, we will study suitable restrictions on the TP problem that allow us to recover its decidability: as a matter of fact, the first result we establish is a negative one, namely, that TP over dense time, in its general formulation, is *undecidable*. Then we will also show how to obtain better computational complexities, which are appropriate to the practical exploitation of timeline-based TP and MC, by suitably constraining the logical structure of synchronization rules.

In the general case, a synchronization rule allows a universal quantification over the tokens of a timeline (such a quantification is called *trigger*). When a token is "selected" by a trigger, the rule allows us to compare tokens of the timelines both preceeding (past) and following (future) the trigger token. The first restriction we consider consists in limiting the comparison to tokens in the future with respect to the trigger (*future semantics* of trigger rules). The second imposes that the name of a non-trigger token appears exactly once in the constraints set by the rule (*simple* trigger rules): this syntactical restriction avoids comparisons of multiple token time-events with a non-trigger reference time-event. Better complexity results can be obtained by restricting also the type of *intervals* used in rules in order to compare tokens.

We now describe the organization of this chapter, outlining in particular which are the complexity results implied by the aforementioned restrictions of TP.

Organization of the chapter.

- In Section 8.1 we start by introducing the TP framework, providing some background knowledge on it.
- In Section 8.2 we prove that TP is *undecidable* in the general case, by a reduction from the *halting problem for Minsky 2-counter machines*. The section is concluded commenting on *non-primitive recursive-hardness* of TP under the future semantics of trigger rules (this is formally proved in Appendix F.2).
- In Section 8.3, we establish that future TP with simple trigger rules is *decidable* (in non-primitive recursive time), and show membership in **EXPSPACE** (respectively, **PSPACE**) under the restriction to *non-singular intervals* (respectively, intervals of the forms $[0, a]$ and $[b, +\infty[$).

- Matching complexity lower bounds for the last two restrictions are given in Section 8.4.
- In Section 8.5 we outline an **NP** planning algorithm for TP with *trigger-less rules only* (which disallow the universal quantification/trigger and have a purely existential form) stemming from the results of the previous sections. With a trivial hardness proof, we also show TP with trigger-less rules to be **NP**-complete.
- In Section 8.6, we finally tackle the MC problem for systems described as timelines, where property specifications are given in terms of formulas of *Metric Interval Temporal Logic* (MITL), a timed logic which extends LTL. The reason why here we drop HS and employ the latter is the following: enriching system models claims for extensions of the property specification language; in our case, timelines would naturally require a timed extension of HS which, however, has not been studied yet (in the literature, only *metric* extensions of HS have been proposed over the natural numbers [Bre+13]). Thus the well-known and thoroughly studied MITL comes to the rescue as a sort of "approximation" of HS; moreover, it allows us to link the world of interval-based models/timelines with that of *timed automata*, another famous formalism used in planning as well as in system verification, that will be heavily used for many results we are about to prove.

8.1 The TP Problem

As in the previous chapters, let $\mathbb{N}$ be the set of natural numbers and $\mathbb{R}_+$ be the set of non-negative real numbers; moreover, $Intv$ denotes the set of intervals of $\mathbb{R}_+$ whose endpoints are in $\mathbb{N} \cup \{\infty\}$, and $Intv_{(0,\infty)}$ is the set of non-singular[1] intervals $I \in Intv$ such that either I is unbounded, or I is left-closed with left endpoint 0. The latter intervals I can be replaced by expressions of the form $\sim n$, for some $n \in \mathbb{N}$ and $\sim \in \{<, \leq, >, \geq\}$.

We now introduce notation and basic notions of the TP framework as presented in [COU16; Gig+16]. In TP, domain knowledge is encoded by a set of state variables, whose behaviour over time is described by transition functions and synchronization rules.

Definition 8.1.1 (State variable)**.** A *state variable* x is a triple

$$x = (V_x, T_x, D_x),$$

where

[1] An interval of the form $[a, a]$, for $a \in \mathbb{N}$, is called *singular*.

- V_x is the *finite domain* of the state variable x,
- $T_x : V_x \to 2^{V_x}$ is the *value transition function*, which maps each $v \in V_x$ to the (possibly empty) set of successor values, and
- $D_x : V_x \to Intv$ is the *constraint (or duration) function* that maps each $v \in V_x$ to an interval of $Intv$.

A *token* for a variable x is a pair (v, d) consisting of a value $v \in V_x$ and a duration $d \in \mathbb{R}_+$ such that $d \in D_x(v)$. Intuitively, a token for x represents an interval of time where the state variable x takes value v. In order to clarify the variable to which a token refers, we shall often denote (v, d) as (x, v, d).

The behavior of the state variable x is specified by means of a *timeline*, which is a non-empty sequence of tokens $\pi = (v_0, d_0) \cdots (v_n, d_n)$ consistent with the value transition function T_x, namely, such that $v_{i+1} \in T_x(v_i)$ for all $0 \leq i < n$. The *start time* $\mathrm{s}(\pi, i)$ and the *end time* $\mathrm{e}(\pi, i)$ of the i-th token of the timeline π are defined respectively as follows:

$$\mathrm{s}(\pi, i) = 0 \text{ if } i = 0, \qquad \mathrm{s}(\pi, i) = \sum_{h=0}^{i-1} d_h \text{ otherwise,}$$

and

$$\mathrm{e}(\pi, i) = \sum_{h=0}^{i} d_h.$$

See Figure 8.1 for an example.

Given a finite set SV of state variables, a *multi-timeline* of SV is a mapping Π assigning to each state variable $x \in SV$ a timeline for x.

Multi-timelines of SV can be constrained by a set of *synchronization rules*, which relate tokens, possibly belonging to different timelines, through temporal constraints on the start/end times of tokens (time-point constraints) and on the difference between start/end times of tokens (interval constraints). The synchronization rules exploit an alphabet $\Sigma = \{o, o_0, o_1, o_2, \ldots\}$ of token names to refer to the tokens along a multi-timeline, and are based on the notions of *atom* and *existential statement*.

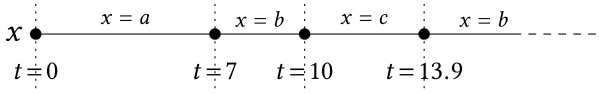


Figure 8.1: An example of timeline $(a, 7)(b, 3)(c, 3.9)\cdots$ for the state variable $x = (V_x, T_x, D_x)$, where $V_x = \{a, b, c, \ldots\}$, $b \in T_x(a)$, $c \in T_x(b)$, $b \in T_x(c)\ldots$ and $D_x(a) = [5, 8]$, $D_x(b) = [1, 4]$, $D_x(c) = [2, \infty[\ldots$

Definition 8.1.2 (Atom)**.** An *atom* ρ is either a clause of the form $o_1 \leq_I^{e_1,e_2} o_2$ (*interval atom*), or of the forms $o_1 \leq_I^{e_1} n$ or $n \leq_I^{e_1} o_1$ (*time-point atom*), where $o_1, o_2 \in \Sigma$, $I \in Intv$, $n \in \mathbb{N}$, and $e_1, e_2 \in \{\mathrm{s}, \mathrm{e}\}$.

An atom ρ is evaluated with respect to a Σ-*assignment* λ_Π for a given multi-timeline Π, which is a mapping assigning to each token name $o \in \Sigma$ a pair $\lambda_\Pi(o) = (\pi, i)$ such that π is a timeline of Π and $0 \leq i < |\pi|$ is a position along π (intuitively, (π, i) represents the token of Π referenced by the name o).

An interval atom $o_1 \leq_I^{e_1,e_2} o_2$ *is satisfied by* λ_Π if $e_2(\lambda_\Pi(o_2)) - e_1(\lambda_\Pi(o_1)) \in I$. A point atom $o \leq_I^e n$ (respectively, $n \leq_I^e o$) *is satisfied by* λ_Π if $n - e(\lambda_\Pi(o)) \in I$ (respectively, $e(\lambda_\Pi(o)) - n \in I$).

Definition 8.1.3 (Existential statement)**.** An *existential statement* $\mathcal{E}$ for a finite set SV of state variables is a statement of the form

$$\mathcal{E} = \exists o_1[x_1 = v_1] \cdots \exists o_n[x_n = v_n].C,$$

where C is a conjunction of atoms, $o_i \in \Sigma$, $x_i \in SV$, $v_i \in V_{x_i}$, for $1 \leq i \leq n$. The elements $o_i[x_i = v_i]$ are called *quantifiers*. A token name used in C, but not occurring in any quantifier, is said to be *free*.

Given a Σ-assignment λ_Π for a multi-timeline Π of SV, we say that λ_Π *is consistent with the existential statement* $\mathcal{E}$ if, for each quantifier $o_i[x_i = v_i]$, we have $\lambda_\Pi(o_i) = (\pi, h)$, where $\pi = \Pi(x_i)$ and the h-th token of π has value v_i. A multi-timeline Π of SV *satisfies* $\mathcal{E}$ if there exists a Σ-assignment λ_Π for Π consistent with $\mathcal{E}$ such that each atom in C is satisfied by λ_Π.

We can now introduce synchronization rules, which constrain tokens, possibly belonging to different timelines.

Definition 8.1.4 (Synchronization rule)**.** A *synchronization rule* $\mathcal{R}$ for a finite set SV of state variables is a rule of one of the forms

$$o_0[x_0 = v_0] \rightarrow \mathcal{E}_1 \vee \mathcal{E}_2 \vee \ldots \vee \mathcal{E}_k, \qquad \top \rightarrow \mathcal{E}_1 \vee \mathcal{E}_2 \vee \ldots \vee \mathcal{E}_k,$$

where $o_0 \in \Sigma$, $x_0 \in SV$, $v_0 \in V_{x_0}$, and $\mathcal{E}_1, \ldots, \mathcal{E}_k$ are *existential statements*. In rules of the first form (which are called *trigger rules*), the quantifier $o_0[x_0 = v_0]$ is called *trigger*; we require that only o_0 may appear free in $\mathcal{E}_i$, for all $1 \leq i \leq n$. In rules of the second form (*trigger-less rules*), we require that no token name appears free.
A trigger rule $\mathcal{R}$ is *simple* if, for each existential statement $\mathcal{E}$ of $\mathcal{R}$ and each token name o distinct from the trigger, there is at most one *interval atom* of $\mathcal{E}$ where o occurs.

Intuitively, the trigger $o_0[x_0 = v_0]$ acts as a universal quantifier, which states that *for all* the tokens of the timeline for x_0, where x_0 takes the value v_0, at least one of the existential statements $\mathcal{E}_i$ must be satisfied. As an example,

$$o_0[x_0 = v_0] \rightarrow \exists o_1[x_1 = v_1].o_0 \leq^{\mathrm{e,s}}_{[2,\infty[} o_1$$

states that after *every* token for x_0 with value v_0 there exists a token for x_1 with value v_1 *starting* at least 2 time instants after the *end* of the former. Trigger-less rules simply assert the satisfaction of some existential statement. The intuitive meaning of *simple* trigger rules is that they disallow simultaneous comparisons of multiple time-events (start/end times of tokens) with a non-trigger reference time-event.

The semantics of synchronization rules is formally defined as follows.

Definition 8.1.5 (Semantics of synchronization rules)**.** Let Π be a multi-timeline of a set SV of state variables.
Given a *trigger-less rule* $\mathcal{R}$ of SV, Π *satisfies* $\mathcal{R}$ if Π satisfies some existential statement of $\mathcal{R}$.
Given a *trigger rule* $\mathcal{R}$ of SV with trigger $o_0[x_0 = v_0]$, Π *satisfies* $\mathcal{R}$ if, for every position i of the timeline $\pi = \Pi(x_0)$ for x_0 such that $\pi(i) = (v_0, d)$, there are an existential statement $\mathcal{E}$ of $\mathcal{R}$ and a Σ-assignment λ_Π for Π consistent with $\mathcal{E}$ such that $\lambda_\Pi(o_0) = (\pi, i)$ and λ_Π satisfies all the atoms of $\mathcal{E}$.

In the paper, we shall also focus on a stronger notion of satisfaction of trigger rules, called *satisfaction under the future semantics*: it requires that all non-trigger tokens selected by some quantifier do not start *strictly before* the start time of the trigger token.

Definition 8.1.6 (Future semantics of trigger rules)**.** A multi-timeline Π of SV satisfies a trigger rule

$$\mathcal{R} = o_0[x_0 = v_0] \rightarrow \mathcal{E}_1 \vee \mathcal{E}_2 \vee \ldots \vee \mathcal{E}_k$$

under the future semantics if Π satisfies the trigger rule obtained from $\mathcal{R}$ by replacing each existential statement

$$\mathcal{E}_i = \exists o_1[x_1 = v_1] \cdots \exists o_n[x_n = v_n].C$$

by

$$\mathcal{E}'_i = \exists o_1[x_1 = v_1] \cdots \exists o_n[x_n = v_n].\Big(C \wedge \bigwedge_{i=1}^{n} o_0 \leq^{\mathrm{s,s}}_{[0,+\infty[} o_i\Big).$$

Table 8.1: Decidability and complexity of restrictions of the TP problem.

	TP problem	Future TP problem
Unrestricted	Undecidable	(Decidable?) Non-primitive recursive-hard
Simple trigger rules	Undecidable	Decidable (non-primitive recursive)
Simple trigger rules, non-singular intervals	?	**EXPSPACE**-complete
Simple trigger rules, intervals in $Intv_{(0,\infty)}$	?	**PSPACE**-complete
Trigger-less rules only	**NP**-complete	//

A *TP domain* $P = (SV, R)$ is specified by a finite set SV of state variables and a finite set R of synchronization rules for SV modeling their admissible behaviors. Trigger-less rules can be used to express initial, as well as intermediate conditions and the goals of the problem, while trigger rules are much more powerful and useful, for instance, to specify invariants and response requirements.

A *plan for* $P = (SV, R)$ is a multi-timeline of SV satisfying all the rules in R. A *future plan for* P is defined in a similar way, but we require satisfaction under the future semantics of *all* trigger rules.

In the next sections we will study the following decision problems:

TP problem Given a TP domain $P = (SV, R)$, is there a plan for P?

Future TP problem Given a TP domain $P = (SV, R)$, is there a *future* plan for P?

Table 8.1 summarizes all the decidability and complexity results described in the following about the mentioned problems: we will consider mixes of restrictions on TP involving trigger rules with future semantics, simple trigger rules, and intervals in atoms (of trigger rules) which are non-singular or in $Intv_{(0,\infty)}$.

8.2 TP over dense temporal domains is an undecidable problem

In this section, we start by settling an important negative result, namely, we show that the TP problem, in its full generality, is undecidable over dense temporal domains, even when a single state variable is involved. Undecidability is proved via a reduction from the halting problem for *Minsky 2-counter machines* [Min67]. The proof somehow resembles the one for the satisfiability problem of Metric Temporal Logic (which will be formally introduced later, in Section 8.3) with both past and future temporal modalities, interpreted on dense time [AH93].

As a preliminary step, we give a short account of Minsky 2-counter machines. A Minsky 2-counter machine (or just *counter machine* for short) is a

tuple $M = (\mathsf{Inst}, \ell_{init}, \ell_{halt})$ consisting of a finite set Inst of labeled instructions $\ell : \imath$, where ℓ is a label and $\imath$ is an instruction for either

- *increment* of counter h: $c_h := c_h + 1$; `goto` ℓ_r, or
- *decrement* of counter h: `if` $c_h > 0$ `then` $c_h := c_h - 1$; `goto` ℓ_s `else goto` ℓ_t,

where $h \in \{1, 2\}$, $\ell_s \neq \ell_t$, and ℓ_r (respectively, ℓ_s, ℓ_t) is either a label of an instruction in Inst or the halting label ℓ_{halt}. Moreover, $\ell_{init} \in \mathsf{Inst}$ is the label of a designated ("initial") instruction.

An *M-configuration* is a triple of the form $C = (\ell, n_1, n_2)$, where ℓ is the label of an instruction (intuitively, which is the one to be executed next), and $n_1, n_2 \in \mathbb{N}$ are the current values of the two counters c_1 and c_2, respectively.

M induces a transition relation, denoted by $\xrightarrow{M}$, over pairs of M-configurations:

- for an instruction with label ℓ incrementing c_1, we have $(\ell, n_1, n_2) \xrightarrow{M} (\ell_r, n_1 + 1, n_2)$, and
- for an instruction decrementing c_1, we have $(\ell, n_1, n_2) \xrightarrow{M} (\ell_s, n_1 - 1, n_2)$ if $n_1 > 0$, and $(\ell, 0, n_2) \xrightarrow{M} (\ell_t, 0, n_2)$ otherwise.

The analogous for instructions changing the value of c_2.

An *M-computation* is a *finite* sequence $C_1, \ldots, C_k$ of M-configurations such that $C_i \xrightarrow{M} C_{i+1}$ for all $1 \leq i < k$. M *halts* if there exists an M-computation starting at $(\ell_{init}, 0, 0)$ and leading to (ℓ_{halt}, n_1, n_2), for some $n_1, n_2 \in \mathbb{N}$. Given a counter machine M, the *halting problem for M* is to decide whether M halts, and it was proved to be *undecidable* by Minsky [Min67].

The rest of the section is devoted to showing the following result.

Theorem 8.2.1. *The TP problem over dense temporal domains is undecidable (even when a single state variable is involved).*

Proof. We prove the thesis by a reduction from the halting problem for Minsky 2-counter machines. Let us introduce the following notational conventions:

- for increment instructions $\ell : c_h := c_h + 1$; `goto` ℓ_r, we define $c(\ell) = c_h$ and $succ(\ell) = \ell_r$;
- for decrement instructions $\ell :$ `if` $c_h > 0$ `then` $c_h := c_h - 1$; `goto` ℓ_r `else goto` ℓ_s, we define $c(\ell) = c_h$, $\mathrm{dec}(\ell) = \ell_r$, and $\mathrm{zero}(\ell) = \ell_s$.

Moreover, let InstLab be the set of instruction labels, including ℓ_{halt}, and let Inc (resp., Dec) be the set of labels for increment (resp., decrement) instructions. We consider a counter machine $M = (\mathsf{Inst}, \ell_{init}, \ell_{halt})$ assuming without loss of generality that no instruction of M leads to ℓ_{init}, and that ℓ_{init} is the label of an increment instruction. To prove the thesis, we build in polynomial time a state variable $x_M = (V, T, D)$ and a finite set R_M of synchronization rules over x_M such

that M halts if and only if there is a timeline for x_M which satisfies all the rules in R_M, that is, a plan for $P = (\{x_M\}, R_M)$.

Encoding of M-computations. First, we define a suitable encoding of a computation of M as a timeline for x_M. For such an encoding we exploit the finite set of symbols $V = V_{main} \cup V_{check}$ corresponding to the finite domain of the state variable x_M. The sets of *main* values V_{main} and *check* values V_{check} are defined as

$$V_{main} = \bigcup_{\ell \in \mathsf{Inc} \cup \{\ell_{halt}\}} \bigcup_{h \in \{1,2\}} \Big(\{\ell\} \cup \{(\ell, c_h)\} \Big) \cup \bigcup_{\ell \in \mathsf{Dec}} \bigcup_{\ell' \in \{\mathsf{zero}(\ell), \mathsf{dec}(\ell)\}} \bigcup_{h \in \{1,2\}} \Big(\{(\ell, \ell')\} \cup \{(\ell, \ell', c_h)\} \cup \{(\ell, \ell', (c_h, \#))\} \Big)$$

and

$$V_{check} = \bigcup_{\ell \in \mathsf{InstLab}} \bigcup_{i,h \in \{1,2\}} \bigcup_{op_i \in \{\mathsf{inc}_i, \mathsf{dec}_i, \mathsf{zero}_i\}} \Big(\{(\ell, op_i)\} \cup \{(\ell, op_i, c_h)\} \cup \{(\ell, op_i, (c_h, \#))\} \Big).$$

For each $h \in \{1, 2\}$, we denote by V_{c_h} the set of V-values v having the form $v = (\ell, c)$, $v = (\ell, \ell', c)$, or $v = (\ell, op, c)$, where $c \in \{c_h, (c_h, \#)\}$: if $c = c_h$, we say that v is an *unmarked* V_{c_h}-value; otherwise ($c = (c_h, \#)$), v is a *marked* V_{c_h}-value.

An M-configuration is encoded by a finite word over V consisting of the concatenation of a *check*-code and a *main*-code. The *main*-code w_{main} for a M-configuration (ℓ, n_1, n_2), where the instruction label $\ell \in \mathsf{Inc} \cup \{\ell_{halt}\}$, $n_1 \geq 0$, and $n_2 \geq 0$, has the form:

$$w_{main} = \ell \cdot \underbrace{(\ell, c_1) \cdots (\ell, c_1)}_{n_1 \text{ times}} \cdot \underbrace{(\ell, c_2) \cdots (\ell, c_2)}_{n_2 \text{ times}}.$$

In the case of a *decrement* instruction label $\ell \in \mathsf{Dec}$ such that $c(\ell) = c_1$, the *main*-code w'_{main} has one of the following two forms, depending on whether the value of c_1 in the encoded configuration is equal to, or greater than zero.

$$(\ell, \mathsf{zero}(\ell)) \cdot \underbrace{(\ell, \mathsf{zero}(\ell), c_2) \cdots (\ell, \mathsf{zero}(\ell), c_2)}_{n_2 \text{ times}},$$

$$(\ell, \mathsf{dec}(\ell)) \cdot (\ell, \mathsf{dec}(\ell), (c_1, \#)) \cdot \underbrace{(\ell, \mathsf{dec}(\ell), c_1) \cdots (\ell, \mathsf{dec}(\ell), c_1)}_{n_1 \text{ times}} \cdot \underbrace{(\ell, \mathsf{dec}(\ell), c_2) \cdots (\ell, \mathsf{dec}(\ell), c_2)}_{n_2 \text{ times}}.$$

In the first case, w'_{main} encodes the configuration $(\ell, 0, n_2)$ and in the second case the configuration $(\ell, n_1 + 1, n_2)$. Note that, in the second case, there is exactly one occurrence of a *marked* V_{c_1}-value which intuitively "marks" the unit of the counter which will be removed by the decrement. Analogously, the *main*-code for a *decrement* instruction label ℓ with $c(\ell) = c_2$ has two forms symmetric with respect to the previous cases.

The *check*-code is used to trace both an M-configuration C and the type of instruction associated with the configuration C_p preceding C in the considered computation. The type of instruction is given by the symbols inc_i, dec_i, and zero_i, with $i \in \{1, 2\}$: inc_i (resp., dec_i, zero_i) means that C_p is associated with an instruction incrementing the counter c_i (resp., decrementing c_i with c_i greater than 0 in C_p, decrementing c_i with c_i equal to 0 in C_p).

The *check*-code for an instruction label $\ell \in \mathsf{InstLab}$ and an inc_1-operation has the following form

$$(\ell, \mathsf{inc}_1) \cdot (\ell, \mathsf{inc}_1, (c_1, \#)) \cdot \underbrace{(\ell, \mathsf{inc}_1, c_1) \cdots (\ell, \mathsf{inc}_1, c_1)}_{n_1 \text{ times}} \cdot \underbrace{(\ell, \mathsf{inc}_1, c_2) \cdots (\ell, \mathsf{inc}_1, c_2)}_{n_2 \text{ times}},$$

and encodes the configuration $(\ell, n_1 + 1, n_2)$. Note that there is exactly one occurrence of a *marked* V_{c_1}-value which intuitively represents the unit added to the counter by the increment operation.

The *check*-code for an instruction label $\ell \in \mathsf{InstLab}$ and an operation $op_1 \in \{\mathsf{dec}_1, \mathsf{zero}_1\}$ for the counter c_1 has the form

$$(\ell, op_1) \cdot \underbrace{(\ell, op_1, c_1) \cdots (\ell, op_1, c_1)}_{n_1 \text{ times}} \cdot \underbrace{(\ell, op_1, c_2) \cdots (\ell, op_1, c_2)}_{n_2 \text{ times}},$$

where we require that $n_1 = 0$ if $op_1 = \mathsf{zero}_1$. The *check*-code for a label $\ell \in \mathsf{InstLab}$ and an operation associated with the counter c_2 is defined in a similar way.

A *configuration*-code is a word $w = w_{check} \cdot w_{main}$ such that w_{check} is a *check*-code, w_{main} is a *main*-code, and w_{check} and w_{main} are associated with the same instruction label. The configuration code is *well-formed* if w_{check} and w_{main} encode the same configuration.

Figure 8.2 depicts the encoding of a configuration-code for the instruction ℓ_{i+1}. The check-code for the instruction ℓ_{i+1} is associated with an increment of the counter c_1 (the type of instruction ℓ_i).

A *computation*-code is a sequence of configuration-codes $\pi = w^1_{check} \cdot w^1_{main} \cdots w^n_{check} \cdot w^n_{main}$ such that, for all $1 \leq j < n$, the following holds (we assume ℓ_i to be the instruction label associated with the configuration code $w^i_{check} \cdot w^i_{main}$):

- $\ell_j \neq \ell_{halt}$;
- if $\ell_j \in \mathsf{Inc}$ with $c(\ell_j) = c_h$, then $\ell_{j+1} = succ(\ell_j)$ and w^{j+1}_{check} is associated with the operation inc_h;

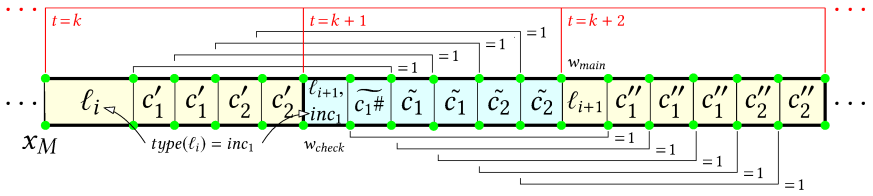


Figure 8.2: A fragment of a computation code with a configuration code for an instruction ℓ_{i+1}. Main-codes are highlighted in yellow and check-codes in cyan. Each square can also be seen as a token of a timeline for x_M (tokens are decorated with their start time and their temporal constraints). In the figure, for $h \in \{1, 2\}$, the symbols c'_h, $\tilde{c_h}$, $\widetilde{c_h\#}$, and c''_h, stand respectively for (ℓ_i, c_h), (ℓ_{i+1}, inc_1, c_h), $(\ell_{i+1}, inc_1, (c_h, \#))$, and (ℓ_{i+1}, c_h).

- if $\ell_j \in$ Dec with $c(\ell_j) = c_h$, and the first symbol of w^j_{main} is $(\ell_j, \text{zero}(\ell_j))$ (resp., $(\ell_j, \text{dec}(\ell_j))$), then $\ell_{j+1} = \text{zero}(\ell_j)$ (resp., $\ell_{j+1} = \text{dec}(\ell_j)$) and w^{j+1}_{check} is associated with the operation zero_h (resp., dec_h).

The computation-code π is *well-formed* if, additionally, each configuration-code in π is *well-formed* and, for all $1 \leq j < n$, the following holds (we assume (ℓ_i, n^i_1, n^i_2) to be the configuration encoded by $w^i_{check} \cdot w^i_{main}$):

- if $\ell_j \in$ Inc, with $c(\ell_j) = c_h$, then $n^{j+1}_h = n^j_h + 1$ and $n^{j+1}_{3-h} = n^j_{3-h}$;
- if $\ell_j \in$ Dec, with $c(\ell_j) = c_h$, then $n^{j+1}_{3-h} = n^j_{3-h}$. Moreover, if w^{j+1}_{check} is associated with dec_h, then $n^{j+1}_h = n^j_h - 1$.

Clearly, a well-formed computation code π encodes a computation of the Minsky 2-counter machine.

A computation-code π is *initial* if it starts with the prefix $(\ell_{init}, \text{zero}_1) \cdot \ell_{init}$, and it is *halting* if it leads to a configuration-code associated with the halting label ℓ_{halt}. The counter machine M halts if and only if there is an initial and halting well-formed computation-code.

Definition of x_M and R_M. Let us show now how to reduce the problem of checking the existence of an initial and halting well-formed computation-code to a TP problem for the state variable x_M.

The idea is to define a timeline where the sequence of values of its tokens is a well-formed computation-code. The durations of tokens are suitably exploited to guarantee well-formedness of computation-codes. We refer the reader again to Figure 8.2 for an intuition. Each symbol of the computation-code is associated with a token having a positive duration. The overall duration of the sequence of tokens corresponding to a check-code or a main-code amounts exactly to one time unit. To allow for the encoding of arbitrarily large values of counters in one time unit, the duration of such tokens is not fixed (taking advantage of the dense

temporal domain). In two adjacent check/main-codes, the time elapsed between the start times of corresponding elements in the representation of the value of a counter (see elements in Figure 8.2 connected by horizontal lines) amounts exactly to one time unit. Such a constraint allows us to compare the values of counters in adjacent codes, either checking for equality, or simulating (by using marked symbols) increment and decrement operations. Note that there is a single *marked* token c_1 in the check-code—that represents the unit added to c_1 by the instruction ℓ_i—which does not correspond to any of the c_1's of the preceding main-code.

We now formally define a state variable x_M and a set R_M of synchronization rules for x_M such that the untimed part of any timeline (i.e., neglecting tokens' durations) for x_M satisfying the rules in R_M is (represents) an initial and halting well-formed computation-code. Thus, M halts if and only if there exists a timeline for x_M satisfying the rules in R_M.

As for x_M, we let $x_M = (V, T, D)$ where, for each $v \in V$, $D(v) =]0, 1]$. This sets the *strict time monotonicity* constraint, namely, the duration of a token along a timeline is always greater than zero and less than or equal to 1.

The value transition function T of x_M ensures the following requirement.

Claim 8.2.2. The untimed part of any timeline for x_M whose first token has value $(\ell_{init}, \mathsf{zero}_1)$ is a prefix of some initial computation-code. Moreover, $(\ell_{init}, \mathsf{zero}_1) \notin T(v)$ for all $v \in V$.

It is a straightforward task to define T in such a way that the previous requirement is fulfilled (for details, see Appendix F.1).

Finally, the synchronization rules in R_M ensure the following requirements.

- *Initialization:* every timeline starts with two tokens, the first one having value $(\ell_{init}, \mathsf{zero}_1)$, and the second having value ℓ_{init}. By Claim 8.2.2 and the fact that no instruction of M leads to ℓ_{init}, it suffices to require that a timeline has a token with value $(\ell_{init}, \mathsf{zero}_1)$ and a token with value ℓ_{init}. This is ensured by the following two trigger-less rules:

$$\top \rightarrow \exists\, o[x_M = (\ell_{init}, \mathsf{zero}_1)].\ \top$$

and

$$\top \rightarrow \exists\, o[x_M = \ell_{init}].\ \top\,.$$

- *Halting:* every timeline leads to a configuration-code associated with the halting label. By the rules for initialization and Claim 8.2.2, it suffices to require that a timeline has a token with value ℓ_{halt}. This is ensured by the following trigger-less rule:

$$\top \rightarrow \exists\, o[x_M = \ell_{halt}].\ \top\,.$$

- *1-Time distance between consecutive control values:* a *control V-value* corresponds to the first symbol of a *main*-code or a *check*-code, i.e., it is an element in $V \setminus (V_{c_1} \cup V_{c_2})$. We require that the difference of the start times of two consecutive tokens along a timeline having a control V-value is exactly 1. Formally, for each pair tk and tk' of tokens along a timeline such that tk and tk' have a control V-value, tk precedes tk', and there is no token between tk and tk' having a control V-value, it holds that $\mathrm{s}(tk') - \mathrm{s}(tk) = 1$ (we write this with a little abuse of notation). By Claim 8.2.2, strict time monotonicity, and the halting requirement, it suffices to ensure that each token tk having a control V-value distinct from ℓ_{halt} is eventually followed by a token tk' such that tk' has a control V-value and $\mathrm{s}(tk') - \mathrm{s}(tk) = 1$. To this aim, for each $v \in V_{\mathrm{con}} \setminus \{\ell_{halt}\}$, being V_{con} the set of control V-values, we write the following trigger rule:

$$o[x_M = v] \rightarrow \bigvee_{u \in V_{\mathrm{con}}} \exists\, o'[x_M = u].\, o \leq^{\mathrm{s,s}}_{[1,1]} o'.$$

- *Well-formedness of configuration-codes:* we need to guarantee that for each configuration-code $w_{check} \cdot w_{main}$ occurring along a timeline and each counter c_h, the value of c_h along the *main*-code w_{main} and the *check*-code w_{check} coincide. By Claim 8.2.2, strict time monotonicity, initialization, and 1-Time distance between consecutive control values, it suffices to ensure that (i) each token tk with a V_{c_h}-value in V_{check} is eventually followed by a token tk' with a V_{c_h}-value such that $\mathrm{s}(tk') - \mathrm{s}(tk) = 1$, and vice versa (ii) each token tk with a V_{c_h}-value in V_{main} is eventually preceded by a token tk' with a V_{c_h}-value such that $\mathrm{s}(tk) - \mathrm{s}(tk') = 1$. As for the former requirement, for each $v \in V_{c_h} \cap V_{check}$, we write the rule:

$$o[x_M = v] \rightarrow \bigvee_{u \in V_{c_h}} \exists\, o'[x_M = u].\, o \leq^{\mathrm{s,s}}_{[1,1]} o'.$$

  For the latter, for each $v \in V_{c_h} \cap V_{main}$, we have the rule:

$$o[x_M = v] \rightarrow \bigvee_{u \in V_{c_h}} \exists\, o'[x_M = u].\, o' \leq^{\mathrm{s,s}}_{[1,1]} o.$$

- *Increment and decrement:* we need to guarantee that the increment and decrement instructions are correctly simulated. By Claim 8.2.2 and the previously defined synchronization rules, we can assume that the untimed part π of a timeline is an initial and halting computation-code such that all configuration-codes occurring in π are well-formed.
  Let $w_{main} \cdot w_{check}$ be a subword occurring in π such that w_{main} (resp., w_{check}) is a *main*-code (resp., *check*-code). Let ℓ_{main} (resp., ℓ_{check}) be the instruction label associated with w_{main} (resp., w_{check}) and for $i = 1, 2$, let n_i^{main}

(resp., n_i^{check}) be the value of counter c_i encoded by w_{main} (resp., w_{check}). Let $c_h = c(\ell_{main})$. By construction $\ell_{main} \neq \ell_{halt}$, end either $\ell_{main} \in \mathsf{Inc}$ and $\ell_{check} = succ(\ell_{main})$, or $\ell_{main} \in \mathsf{Dec}$ and $\ell_{check} \in \{\mathsf{zero}(\ell_{main}), \mathsf{dec}(\ell_{main})\}$. Moreover, if $\ell_{main} \in \mathsf{Dec}$ and $\ell_{check} = \mathsf{zero}(\ell_{main})$, then $n_h^{check} = n_h^{main} = 0$. Thus, it remains to ensure the following two requirements:

(*) if $\ell_{main} \in \mathsf{Inc}$, then $n_h^{check} = n_h^{main} + 1$ and $n_{3-h}^{check} = n_{3-h}^{main}$;

(**) if $\ell_{main} \in \mathsf{Dec}$, then $n_{3-h}^{check} = n_{3-h}^{main}$, and whenever $\ell_{check} = \mathsf{dec}(\ell_{main})$, then $n_h^{check} = n_h^{main} - 1$.

First we observe that, if $\ell_{main} \in \mathsf{Inc}$, our encoding ensures that all $V_{c_{3-h}}$-values in w_{main} and in w_{check} are unmarked, all V_{c_h}-values in w_{main} are unmarked, and there is exactly one marked V_{c_h}-value in w_{check}. If instead $\ell_{main} \in \mathsf{Dec}$, our encoding ensures that all $V_{c_{3-h}}$-values in w_{main} and in w_{check} are unmarked, all V_{c_h}-values in w_{check} are unmarked, and in case $\ell_{check} = \mathsf{dec}(\ell_{main})$, then there is exactly one marked V_{c_h}-value in w_{main}. Thus, by strict time monotonicity and 1-Time distance between consecutive control values, it follows that requirements (*) and (**) are captured by the following rules, where U_{c_i} denotes the set of *unmarked* V_{c_i}-values, for $i = 1, 2$, and V_{init} (resp., V_{halt}) is the set of V-values associated with the label ℓ_{init} (resp., ℓ_{halt}). For each $v \in (U_{c_i} \cap V_{main}) \setminus V_{halt}$, we have the rule:

$$o[x_M = v] \rightarrow \bigvee_{u \in U_{c_i}} \exists\, o'[x_M = u].\ o \leq^{\mathsf{s},\mathsf{s}}_{[1,1]} o'.$$

For each $v \in (U_{c_i} \cap V_{check}) \setminus V_{init}$, we have the rule:

$$o[x_M = v] \rightarrow \bigvee_{u \in U_{c_i}} \exists\, o'[x_M = u].\ o' \leq^{\mathsf{s},\mathsf{s}}_{[1,1]} o.$$

This concludes the proof of the theorem. □

It is worth observing that all the above trigger rules are *simple*, hence *undecidability of the TP problem holds also under the restriction to simple trigger rules.*

In order to ensure the well-formedness of configuration-codes and the increment/decrement requirements, a one-to-one correspondence between (suitable) pairs of tokens in main- and check-codes is enforced thanks to the above trigger rules. Whereas most of such rules are (already) satisfied under the future semantics (as the extra conjoined atoms added by Definition 8.1.6 would be "subsumed" by already-existing ones), some rules are not (the second ones of the well-formedness and increment/decrement requirements are unsatisfiable under the future semantics). As a result, intuitively, having only rules under the future semantics, we can only force the presence, for every token with value c_h (for $h = 1, 2$), of another token with value c_h starting exactly one time instant later,

in the following main-/check-code. However, we cannot prevent extra "spurious" tokens to appear moving from a code to the following one. This is the reason why, with only rules under the future semantics, we lose the ability of encoding computations of (exact) Minsky machines. Only *gainy counter machines* [DL09]—a variant of Minsky machines whose counters may "erroneously" increase—can be captured, thus proving, as a consequence, *non-primitive recursive-hardness* of the future TP problem (the halting problem for gainy counter machines is known to be non-primitive recursive [DL09]).

Theorem 8.2.3. *The future TP problem, even with* one state variable*, is non-primitive recursive-hard also under one of the following two assumptions:* either (1) *the trigger rules are simple,* or (2) *the intervals are in* $Intv_{(0,\infty)}$[a].

[a] We refer to intervals in rules' atoms and in the constraint functions of state variables.

Since this result is just an adaptation of the previous one (apart from some technicalities), we report its proof in Appendix F.2.

In the next section, we will show that future TP with simple trigger rules is indeed decidable in non-primitive recursive time.

8.3 Decidability of future TP with simple trigger rules

In this section, we show that the decidability of the TP problem can be recovered assuming that the trigger rules are *simple* and *interpreted under the future semantics*. Moreover, under the additional assumption that intervals in trigger rules are non-singular (respectively, are in $Intv_{(0,\infty)}$), the problem is in **EXPSPACE** (respectively, in **PSPACE**). The decidability status of *future TP with arbitrary trigger rules remains an open problem.*

The rest of this section is organized as follows: in Section 8.3.1, we recall Timed Automata (TA) [AD94] and Metric Temporal logic (MTL) [Koy90]. In Section 8.3.2, we reduce the future TP problem with simple trigger rules to the *existential MC problem* for TAs against MTL over *finite timed words*. The latter problem is known to be decidable [OW07].

8.3.1 Timed automata and the logic MTL

We start by recalling the notion of timed automaton (TA) [AD94] and the logic MTL [Koy90].

Let Σ be a finite alphabet. A *timed word* w over Σ is a *finite* word $w = (a_0, \tau_0) \cdots (a_n, \tau_n)$ over $\Sigma \times \mathbb{R}_+$ (τ_i is called a *timestamp*, and intuitively represents the time at which the "event" a_i occurs) such that $\tau_i \leq \tau_{i+1}$ for all $0 \leq i < n$ (*monotonicity* requirement). The timed word w is also denoted by (σ, τ), where σ is the finite (untimed) word $a_0 \cdots a_n$ and τ is the sequence of timestamps $\tau_0, \ldots, \tau_n$. A *timed language* over Σ is a set of timed words over Σ.

Timed Automata (TA). Let C be a finite set of clocks. A clock valuation $val : C \to \mathbb{R}_+$ for C is a function assigning a non-negative real value to each clock in C. Given a value $t \in \mathbb{R}_+$ and a set $Res \subseteq C$ (that we call *reset set*), $(val + t)$ and $val[Res]$ denote the valuations for C defined respectively as follows: for all $c \in C$, $(val + t)(c) = val(c) + t$, and $val[Res](c) = 0$ if $c \in Res$ and $val[Res](c) = val(c)$ otherwise.

A *clock constraint* θ over C is a Boolean combination of atomic formulas of the form $c \in I$ or $c - c' \in I$, where $c, c' \in C$ and $I \in Intv$. Given a clock valuation val and a clock constraint θ, val is said to satisfy θ, written $val \models \theta$, if θ evaluates to true after replacing each occurrence of a clock c in θ by $val(c)$, and interpreting Boolean connectives and membership to intervals in the standard way. We denote by $\Phi(C)$ the set of all possible clock constraints over C.

Definition 8.3.1 (Timed automaton TA). A TA over Σ is a tuple $\mathcal{A} = (\Sigma, Q, q_0, C, \Delta, F)$, where Q is a finite set of (control) states, $q_0 \in Q$ is the initial state, C is a finite set of clocks, $F \subseteq Q$ is the set of accepting states, and $\Delta \subseteq Q \times \Sigma \times \Phi(C) \times 2^C \times Q$ is the transition relation.
The *maximal constant of* $\mathcal{A}$ is the greatest integer occurring as an endpoint of some interval in the clock constraints of the transitions of $\mathcal{A}$.

Intuitively, in a TA $\mathcal{A}$, while transitions are instantaneous, time can elapse in a control state. The clocks progress at the same speed and can be reset independently of each other when a transition is executed, in such a way that each clock keeps track of the time elapsed since the last reset. Moreover, clock constraints are used as guards of transitions to restrict the behavior of the automaton.

A configuration of $\mathcal{A}$ is a pair (q, val), where $q \in Q$ and val is a clock valuation for C. A run r of $\mathcal{A}$ on a timed word $w = (a_0, \tau_0) \cdots (a_n, \tau_n)$ over Σ is a sequence of configurations $r = (q_0, val_0) \cdots (q_{n+1}, val_{n+1})$ starting at the initial configuration (q_0, val_0), where $val_0(c) = 0$ for all $c \in C$ (*initiation requirement*), and

- for all $0 \leq i \leq n$ we have (*consecution requirement*): (i) $(q_i, a_i, \theta, Res, q_{i+1}) \in \Delta$ for some $\theta \in \Phi(C)$ and reset set Res, (ii) $(val_i + \tau_i - \tau_{i-1}) \models \theta$ and (iii) $val_{i+1} = (val_i + \tau_i - \tau_{i-1})[Res]$ (we let $\tau_{-1} = 0$).

The intuitive behavior of the TA $\mathcal{A}$ is the following. Assume that $\mathcal{A}$ is on state $q \in Q$ after reading the symbol (a', τ_i) at time τ_i and, at that time, the

clock valuation is *val*. On reading (a, τ_{i+1}), $\mathcal{A}$ chooses a transition of the form $\delta = (q, a, \theta, Res, q') \in \Delta$ such that the constraint θ is fulfilled by $(val + t)$, with $t = \tau_{i+1} - \tau_i$. The control then changes from q to q' and *val* is updated in such a way as to record the amount of time elapsed t in the clock valuation, and to reset the clocks in *Res*, namely, *val* is updated to $(val + t)[Res]$.

A run r is *accepting* if $q_{n+1} \in F$. The *timed language* $\mathcal{L}_T(\mathcal{A})$ of $\mathcal{A}$ is the set of timed words w over Σ such that there is an accepting run of $\mathcal{A}$ on w.

As shown in [AD94], given two TAs $\mathcal{A}_1$, with s_1 states and k_1 clocks, and $\mathcal{A}_2$, with s_2 states and k_2 clocks, the union (resp., intersection) automaton $\mathcal{A}_\vee$ (resp., $\mathcal{A}_\wedge$) such that $\mathcal{L}_T(\mathcal{A}_\vee) = \mathcal{L}_T(\mathcal{A}_1) \cup \mathcal{L}_T(\mathcal{A}_2)$ (resp., $\mathcal{L}_T(\mathcal{A}_\wedge) = \mathcal{L}_T(\mathcal{A}_1) \cap \mathcal{L}_T(\mathcal{A}_2)$) can be effectively calculated, and has $s_1 + s_2$ states (resp., $s_1 \cdot s_2$ states) and $k_1 + k_2$ clocks (resp., $k_1 + k_2$ clocks).

The logic MTL. Let us now recall the framework of Metric Temporal Logic (MTL) [Koy90], a well-known timed linear-time temporal logic which extends standard LTL with time constraints on the until modality.

Given a finite set $\mathcal{AP}$ of proposition letters, the set of MTL formulas φ over $\mathcal{AP}$ is defined by the following grammar:

$$\varphi ::= \top \mid p \mid \varphi \vee \varphi \mid \neg\varphi \mid \varphi \mathsf{U}_I \varphi,$$

where $p \in \mathcal{AP}$, $I \in Intv$, and U_I is the *strict timed until* MTL modality.

MTL formulas over $\mathcal{AP}$ are interpreted over timed words over $2^{\mathcal{AP}}$. Given an MTL formula φ, a timed word $w = (\sigma, \tau)$ over $2^{\mathcal{AP}}$, and a position $0 \leq i < |w|$, the satisfaction relation $(w, i) \models \varphi$—meaning that φ holds at position i of w—is defined as follows (we omit the clauses for Boolean connectives):

- $(w, i) \models p \iff p \in \sigma(i)$,
- $(w, i) \models \varphi_1 \mathsf{U}_I \varphi_2 \iff$ there exists $j > i$ such that $(w, j) \models \varphi_2$, $\tau_j - \tau_i \in I$, and $(w, k) \models \varphi_1$ for all $i < k < j$.

A *model of* φ is a timed word w over $2^{\mathcal{AP}}$ such that $(w, 0) \models \varphi$. The *timed language* $\mathcal{L}_T(\varphi)$ of φ is the set of models of φ.

The *existential MC problem for TAs against MTL* is the problem of checking, for a given TA $\mathcal{A}$ over $2^{\mathcal{AP}}$ and an MTL formula φ over $\mathcal{AP}$, whether $\mathcal{L}_T(\mathcal{A}) \cap \mathcal{L}_T(\varphi) \neq \emptyset$.

In MTL, we use standard shortcuts such as: $\mathsf{F}_I\varphi$ for $\varphi \vee (\top \mathsf{U}_I \varphi)$ (*timed eventually* or *timed future*), and $\mathsf{G}_I\varphi$ for $\neg\mathsf{F}_I\neg\varphi$ (*timed always* or *timed globally*).

We also consider two fragments of MTL, namely, MITL (Metric Interval Temporal Logic) and $\mathsf{MITL}_{(0,\infty)}$ [AFH96]: MITL is obtained by allowing only non-singular intervals of *Intv* at the subscript of U, while $\mathsf{MITL}_{(0,\infty)}$ is the fragment of MITL obtained by allowing only intervals in $Intv_{(0,\infty)}$.

The *maximal constant* of an MTL formula φ is the greatest integer occurring as an endpoint of some interval of (the occurrences of) the U_I modality in φ.

8.3.2 Reduction to existential MC for TAs against MTL

We now solve the future TP problem with simple trigger rules by means of an exponential-time reduction to the existential MC problem for TAs against MTL.

In the following, we fix an instance $P = (SV, R)$ of the problem where the trigger rules in R are simple. The *maximal constant* of P, denoted by K_P, is the greatest integer occurring in the atoms of the rules in R and in the constraint functions of the state variables in SV.

The proposed reduction consists of three steps:

1. first, we define an encoding of the multi-timelines of SV by means of timed words over $2^{\mathcal{AP}}$ for a suitable finite set $\mathcal{AP}$ of proposition letters, and show how to construct a TA $\mathcal{A}_{SV}$ over $2^{\mathcal{AP}}$ accepting such encodings;
2. next, we build an MTL formula $\varphi_\forall$ over $\mathcal{AP}$ such that for each multi-timeline Π of SV and encoding w_Π of Π, w_Π is a model of $\varphi_\forall$ if and only if Π satisfies all the trigger rules in R under the future semantics;
3. finally, we construct a TA $\mathcal{A}_\exists$ over $2^{\mathcal{AP}}$ such that for each multi-timeline Π of SV and encoding w_Π of Π, w_Π is accepted by $\mathcal{A}_\exists$ if and only if Π satisfies all the trigger-less rules in R.

Hence, there is a future plan for $P = (SV, R)$ iff $\mathcal{L}_T(\mathcal{A}_{SV}) \cap \mathcal{L}_T(\mathcal{A}_\exists) \cap \mathcal{L}_T(\varphi_\forall) \neq \emptyset$.

For each $x \in SV$, we let $x = (V_x, T_x, D_x)$. Given an interval $I \in \mathit{Intv}$ and a natural number $n \in \mathbb{N}$, let $n + I$ (respectively, $n - I$) denote the set of non-negative real numbers $\tau \in \mathbb{R}_+$ such that $\tau - n \in I$ (respectively, $n - \tau \in I$). Note that $n + I$ (respectively, $n - I$) is a (possibly empty) interval in Intv whose endpoints can be trivially calculated.

For an atom ρ in R involving a time constant (time-point atom), let $I(\rho)$ be the interval in Intv defined as follows:

- if ρ has the form $o \leq_I^e n$ (resp., $n \leq_I^e o$), then $I(\rho) = n - I$ (resp., $I(\rho) = n + I$).

We finally define Intv_R as the set of intervals $J \in \mathit{Intv}$ such that $J = I(\rho)$ for some time-point atom ρ occurring in a trigger rule of R.

Encodings of multi-timelines of SV. We assume that for distinct state variables x and x', the sets V_x and $V_{x'}$ are disjoint. We exploit the following set $\mathcal{AP}$ of proposition letters to encode multi-timelines of SV:

$$\mathcal{AP} = \bigcup_{x \in SV} \mathit{Main}_x \cup \mathit{Deriv},$$

$$\mathit{Main}_x = ((\{\mathit{beg}_x\} \cup V_x) \times V_x) \cup (V_x \times \{\mathit{end}_x\}),$$

$$\mathit{Deriv} = \mathit{Intv}_R \cup \{p_>\} \cup \bigcup_{x \in SV} \bigcup_{v \in V_x} \{\mathit{past}_v^{\mathrm{s}}, \mathit{past}_v^{\mathrm{e}}\}.$$

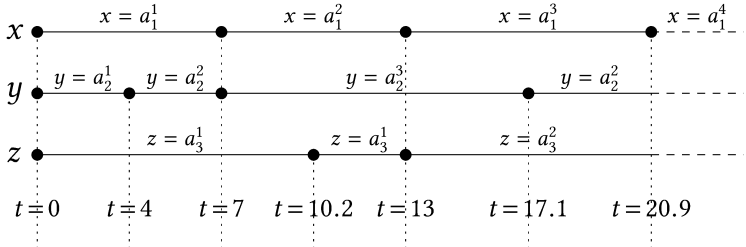


Figure 8.3: An example of multi-timeline of $SV = \{x, y, z\}$, where in particular $V_x = \{a_1^i \mid 1 \le i \le 4\}$, $V_y = \{a_2^i \mid 1 \le i \le 3\}$ and $V_z = \{a_3^i \mid 1 \le i \le 2\}$.
The encoding of the timeline for x depicted in the figure (we show only values in $Main_x$) is $\big(\{(beg_x, a_1^1)\}, 0\big)\big(\{(a_1^1, a_1^2)\}, 7\big)\big(\{(a_1^2, a_1^3)\}, 13\big)\big(\{(a_1^3, a_1^4)\}, 20.9\big)\cdots$
The encoding of the multi-timeline of SV depicted in the figure (we show only values in $Main_x \cup Main_y \cup Main_z$) is $\big(\{(beg_x, a_1^1), (beg_y, a_2^1), (beg_z, a_3^1)\}, 0\big)\big(\{(a_2^1, a_2^2)\}, 4\big)$ $\big(\{(a_1^1, a_1^2), (a_2^2, a_2^3)\}, 7\big)\big(\{(a_3^1, a_3^1)\}, 10.2\big)\big(\{(a_1^2, a_1^3), (a_3^1, a_3^2)\}, 13\big)\big(\{(a_2^3, a_2^2)\}, 17.1\big)\cdots$

Intuitively, we use the propositions in $Main_x$ to encode a token along a timeline for x. The propositions in $Deriv$, as explained below, represent enrichments of the encoding, used for translating simple trigger rules in MTL formulas under the future semantics. The tags beg_x and end_x in $Main_x$ are used to mark the start and the end of a timeline for x.

A token tk with value v along a timeline for x is encoded by two events: the *start-event* (occurring at the start time of tk) and the *end-event* (occurring at the end time of tk). The start-event of tk is specified by a main proposition of the form (v_p, v), where either $v_p = beg_x$ (tk is the first token of the timeline) or v_p is the value of the token for x preceding tk. The end-event of tk is instead specified by a main proposition of the form (v, v_s), where either $v_s = end_x$ (tk is the last token of the timeline) or v_s is the value of the token for x following tk. See Figure 8.3 for an example.

Now we explain the meaning of the proposition letters in $Deriv$. The elements in $Intv_R$ reflect the semantics of the time-point atoms in the trigger rules of R: for each $I \in Intv_R$, I holds at the current position if the current timestamp τ satisfies $\tau \in I$. The tag $p_>$ keeps track of whether the current timestamp is strictly greater than the previous one. Finally, the propositions in $\bigcup_{x \in SV} \bigcup_{v \in V_x} \{past_v^s, past_v^e\}$ keep track of past token events occurring at timestamps *coinciding* with the current timestamp.

We start by defining the encoding of timelines for $x \in SV$. An *encoding of a timeline for* x is a timed word w over $2^{Main_x \cup Deriv}$ of the form

$$w = (\{(beg_x, v_0)\} \cup S_0, \tau_0)(\{(v_0, v_1)\} \cup S_1, \tau_1)\cdots(\{(v_n, end_x)\} \cup S_{n+1}, \tau_{n+1})$$

where, for all $0 \le i \le n+1$, $S_i \subseteq Deriv$, and

- $v_{i+1} \in T_x(v_i)$ for $i < n$;
- $\tau_0 = 0$ and $\tau_{i+1} - \tau_i \in D_x(v_i)$ for $i \leq n$;
- $S_i \cap Intv_R$ is the set of intervals $I \in Intv_R$ such that $\tau_i \in I$;
- $p_> \in S_i$ iff either $i = 0$ or $\tau_i > \tau_{i-1}$;
- for all $v \in V_x$, $past^{\mathrm{s}}_v \in S_i$ (resp., $past^{\mathrm{e}}_v \in S_i$) iff there is $0 \leq h < i$ such that $\tau_h = \tau_i$ and $v = v_h$ (resp., $\tau_h = \tau_i$, $v = v_{h-1}$ and $h > 0$).

Note that the length of w is at least 2. The timed word w encodes the timeline for x of length $n+1$ given by $\pi = (v_0, \tau_1)(v_1, \tau_2 - \tau_1) \cdots (v_n, \tau_{n+1} - \tau_n)$. Note that in the encoding, τ_i and τ_{i+1} represent the start time and the end time of the i-th token of the timeline π ($0 \leq i \leq n$). See the caption of Figure 8.3 for an example of encoding.

Next, we define the encoding of a multi-timeline of SV. For a set $P \subseteq \mathcal{AP}$ and $x \in SV$, let $P[x] = P \setminus \bigcup_{y \in SV \setminus \{x\}} Main_y$. An *encoding of a multi-timeline of SV* is a timed word w over $2^{\mathcal{AP}}$ of the form $w = (P_0, \tau_0) \cdots (P_n, \tau_n)$ such that the following conditions hold:

- for all $x \in SV$, the timed word obtained from $(P_0[x], \tau_0) \cdots (P_n[x], \tau_n)$ by removing the pairs $(P_i[x], \tau_i)$ such that $P_i[x] \cap Main_x = \emptyset$ is an encoding of a timeline for x;
- $P_0[x] \cap Main_x \neq \emptyset$ for all $x \in SV$ (initialization).

See again Figure 8.3 for an example of encoding of a multi-timeline.

We now construct a TA $\mathcal{A}_{SV}$ over $2^{\mathcal{AP}}$ accepting the encodings of the multi-timelines of SV, as shown in the proof of the next proposition.

Proposition 8.3.2. *One can construct in exponential time a TA $\mathcal{A}_{SV}$ over $2^{\mathcal{AP}}$, with $2^{O(\sum_{x \in SV} |V_x|)}$ states, $|SV| + 2$ clocks, and maximal constant $O(K_P)$, such that $\mathcal{L}_T(\mathcal{A}_{SV})$ is the set of encodings of the multi-timelines of SV.*

Proof. Let us fix an ordering $SV = \{x_1, \ldots, x_N\}$ of the state variables. Let $\mathcal{H} = Deriv \setminus (Intv_R \cup \{p_>\})$ and $V'_i = V_{x_i} \cup \{beg_{x_i}, end_{x_i}\}$ for all $1 \leq i \leq N$.

The TA $\mathcal{A}_{SV} = (2^{\mathcal{AP}}, Q, q_0, C, \Delta, F)$ is defined as follows.

- The set of states is given by $Q = V'_1 \times \ldots \times V'_N \times 2^{\mathcal{H}}$. Intuitively, for a state $(v_1, \ldots, v_N, H)$, the i-th component v_i keeps track of the value of the last (start-event for a) token for x_i read so far if $v_i \notin \{beg_{x_i}, end_{x_i}\}$. If $v_i = beg_{x_i}$ (resp., $v_i = end_{x_i}$), then no start-event for a token for x_i has been read so far (resp., no start-event for a token for x_i can be read). Moreover, the last component H of the state keeps track of past token events occurring at a timestamp coinciding with the last timestamp.
- The initial state q_0 is $(beg_{x_1}, \ldots, beg_{x_N}, \emptyset)$.
- The set F of accepting states is the set of all states $(end_{x_1}, \ldots, end_{x_N}, H)$ for any set $H \subseteq \mathcal{H}$.

- The set of clocks C is given by $C = \{c_1, \ldots, c_N, c_>, c_{glob}\}$. We have a clock c_i for each state variable x_i, which is used to check that the duration of a token for x_i with value v is in $D_{x_i}(v)$. Moreover, $c_>$ is a clock which is always reset and is used to capture the meaning of proposition $p_>$, whereas c_{glob} is a clock that measures the current (global) time and is never reset.
- The relation Δ consists of the transitions

$$((v_1, \ldots, v_N, H),\ P,\ \theta_1 \wedge \ldots \wedge \theta_N \wedge \theta_> \wedge \theta_{glob},\ Res,\ (v'_1, \ldots, v'_N, H'))$$

  such that:
  - if $(v_1, \ldots, v_N, H) = q_0$, then $P \cap Main_x \neq \emptyset$ for all $x \in SV$ (this ensures initialization);
  - for all $1 \leq i \leq N$, the following holds:
    - *either* $P \cap Main_{x_i} = \emptyset$, $v'_i = v_i$, $\theta_i = \top$, and $c_i \notin Res$ (intuitively, no event associated with x_i occurs in this case),
    - *or* $P \cap Main_{x_i} = (v_i, v'_i)$ (hence, $v_i \neq end_{x_i}$), $v'_i \in T_{x_i}(v_i)$ if both $v_i \in V_{x_i}$ and $v'_i \in V_{x_i}$; $c_i \in Res$ and $\theta_i = c_i \in D_{x_i}(v_i)$ (resp., $\theta_i = c_i \in [0, 0]$) if $v_i \neq beg_{x_i}$ (resp., if $v_i = beg_{x_i}$);
  - $c_{glob} \notin Res$ and

$$\theta_{glob} = \bigwedge_{I \in P \cap Intv_R} c_{glob} \in I \wedge \bigwedge_{I \in Intv_R \setminus P} (c_{glob} \in \overrightarrow{I} \vee c_{glob} \in \overleftarrow{I}),$$

    where, for each $I \in Intv_R \setminus P$, $\overrightarrow{I}$ and $\overleftarrow{I}$ are (possibly empty) maximal intervals in $\mathbb{R}_+$ disjoint from I (e.g., if $I = [3, 5[$, then $\overleftarrow{I} = [0, 3[$ and $\overrightarrow{I} = [5, +\infty[$). Note that $\overrightarrow{I}, \overleftarrow{I} \in Intv$. Recall that, for each $I \in Intv_R$, I must be in P if and only if the current time (given by c_{glob}) is in I;
  - $c_> \in Res$; moreover, if $(v_1, \ldots, v_N, H) = q_0$, then $p_> \in P$ and $\theta_> = \top$, otherwise, *either* $p_> \in P$ and $\theta_> = c_> \in]0, +\infty[$, *or* $p_> \notin P$ and $\theta_> = c_> \in [0, 0]$;
  - $P \cap \mathcal{H} = \emptyset$ if $p_> \in P$; otherwise $P \cap \mathcal{H} = H$;
  - for all $x \in SV$ and $v \in V_x$, $past^{\mathrm{s}}_v \in H'$ iff *either* $P \cap Main_{x_i}$ is of the form (v', v), *or* $p_> \notin P$ and $past^{\mathrm{s}}_v \in H$;
  - for all $x \in SV$ and $v \in V_x$, $past^{\mathrm{e}}_v \in H'$ iff *either* $P \cap Main_{x_i}$ is of the form (v, v'), *or* $p_> \notin P$ and $past^{\mathrm{e}}_v \in H$.

This concludes the proof. □

Encodings of simple trigger rules by MTL formulas. We now construct an MTL formula $\varphi_\forall$ over $\mathcal{AP}$ capturing the simple trigger rules in R, under the future semantics.

Proposition 8.3.3. *One can construct in linear time an MTL formula $\varphi_\forall$, with maximal constant $O(K_P)$, such that for each multi-timeline Π of SV and encoding w_Π of Π, w_Π is a model of $\varphi_\forall$ iff Π satisfies all the simple trigger rules in R under the future semantics.*
The formula $\varphi_\forall$ is an MITL formula (resp., $MITL_{(0,\infty)}$ formula) if the intervals in the trigger rules are non-singular (resp., belong to $Intv_{(0,\infty)}$).
The formula $\varphi_\forall$ has $O(|R|\cdot N_A\cdot N_\mathcal{E}\cdot(|Intv_R| + (\sum_{x\in SV}|V_x|)^2))$ distinct subformulas, with N_A the maximum number of atoms in a trigger rule of R, and $N_\mathcal{E}$ the maximum number of existential statements in a trigger rule of R.

Proof. We first introduce some auxiliary propositional (Boolean) formulas over $\mathcal{AP}$. Let $x \in SV$ and $v \in V_x$. We denote by $\psi(\mathrm{s}, v)$ and $\psi(\mathrm{e}, v)$ the two propositional formulas over $Main_x$ defined as follows:

$$\psi(\mathrm{s}, v) = (beg_x, v) \vee \bigvee_{u\in V_x}(u, v),$$

$$\psi(\mathrm{e}, v) = (v, end_x) \vee \bigvee_{u\in V_x}(v, u).$$

Intuitively, $\psi(\mathrm{s}, v)$ (resp., $\psi(\mathrm{e}, v)$) states that a start-event (resp., end-event) for a token for x with value v occurs at the current time. We also use the formula

$$\psi_{\neg x} = \neg \bigvee_{m\in Main_x} m$$

asserting that no event for a token for x occurs at the current time. Additionally, given an MTL formula θ, we define the MTL formula

$$EqTime(\theta) = \theta \vee [\neg p_> \mathsf{U}_{\geq 0}(\neg p_> \wedge \theta)]$$

which is satisfied by an encoding of a multi-timeline of SV at the current time if θ eventually holds at a position whose timestamp coincides with the current timestamp.

The MTL formula $\varphi_\forall$ has a conjunct $\varphi_\mathcal{R}$ for each trigger rule $\mathcal{R} \in R$. Let $\mathcal{R}$ be a trigger rule of the form $o_t[x_t = v_t] \rightarrow \mathcal{E}_1 \vee \mathcal{E}_2 \vee \ldots \vee \mathcal{E}_k$. Then $\varphi_\mathcal{R}$ is given by

$$\varphi_\mathcal{R} = \mathsf{G}_{\geq 0}\big(\psi(\mathrm{s}, v_t) \rightarrow \bigvee_{i=1}^{k}\Phi_{\mathcal{E}_i}\big),$$

where $\Phi_{\mathcal{E}_i}$, with $1 \leq i \leq k$, ensures the fulfillment of the existential statement $\mathcal{E}_i$ of $\mathcal{R}$ under the future semantics.

Let $\mathcal{E} \in \{\mathcal{E}_1, \ldots, \mathcal{E}_k\}$, O be the set of token names existentially quantified in $\mathcal{E}$, $\mathbf{A}$ be the set of *interval* atoms in $\mathcal{E}$ and, for each $o \in O$, $val(o)$ be the value

of the token referenced by o in the associated quantifier. In the construction of $\Phi_{\mathcal{E}}$, we crucially exploit the assumption that $\mathcal{R}$ is simple: for each token name $o \in O$, there is at most one atom in $\mathbf{A}$ where o occurs.

For each token name $o \in \{o_t\} \cup O$, we denote by $Intv^{\mathrm{s}}_o$ (resp., $Intv^{\mathrm{e}}_o$) the set of intervals $J \in Intv$ such that $J = I(\rho)$ for some time-point atom ρ occurring in $\mathcal{E}$, which imposes a time constraint on the start time (resp., end time) of the token referenced by o. Note that $Intv^{\mathrm{s}}_o, Intv^{\mathrm{e}}_o \subseteq \mathcal{AP}$, and we exploit the propositional formulas $\xi^{\mathrm{s}}_o = \bigwedge_{I \in Intv^{\mathrm{s}}_o} I$ and $\xi^{\mathrm{e}}_o = \bigwedge_{I \in Intv^{\mathrm{e}}_o} I$ to ensure the fulfillment of the time constraints imposed by the time-point atoms associated with the token o.

The MTL formula $\Phi_{\mathcal{E}}$ is thus given by:

$$\Phi_{\mathcal{E}} = \xi^{\mathrm{s}}_{o_t} \wedge [\psi_{\neg x_t} \mathsf{U}_{\geq 0}(\psi(\mathrm{e}, v_t) \wedge \xi^{\mathrm{e}}_{o_t})] \wedge \bigwedge_{\rho \in \mathbf{A}} \chi_\rho,$$

where, for each atom $\rho \in \mathbf{A}$, the formula χ_ρ captures the future semantics of ρ.

The construction of χ_ρ depends on the form of ρ. We distinguishes four cases.

1. $\rho = o \leq_I^{e_1,e_2} o_t$ and $o \neq o_t$. We assume $0 \in I$ (the other case being simpler). First, assume that $e_2 = \mathrm{s}$. Under the future semantics, ρ holds iff the start time of the trigger token o_t coincides with the e_1-time of token o. Hence, in this case ($e_2 = \mathrm{s}$), χ_ρ is given by:

$$\chi_\rho = \xi^{e_1}_o \wedge \big(past^{e_1}_{val(o)} \vee EqTime(\psi(e_1, val(o)))\big).$$

If instead $e_2 = \mathrm{e}$, then χ_ρ is defined as follows:

$$\begin{aligned}\chi_\rho = \big[\psi_{\neg x_t} \mathsf{U}_{\geq 0}\{\xi^{e_1}_o \wedge \psi(e_1, val(o)) \wedge \psi_{\neg x_t} \wedge (\psi_{\neg x_t} \mathsf{U}_I \psi(\mathrm{e}, v_t))\}\big] \vee \\ \big[(\psi(e_1, val(o)) \vee past^{e_1}_{val(o)}) \wedge \xi^{e_1}_o \wedge \\ \big(EqTime(\psi(\mathrm{e}, v_t)) \vee (\psi_{\neg x_t} \wedge (\psi_{\neg x_t} \mathsf{U}_I \psi(\mathrm{e}, v_t)))\big)\big] \vee \\ \big[\psi_{\neg x_t} \mathsf{U}_{\geq 0}\{\psi(\mathrm{e}, v_t) \wedge EqTime(\psi(e_1, val(o)) \wedge \xi^{e_1}_o)\}\big].\end{aligned}$$

The first disjunct (in square brackets) considers the case where the e_1-event of token o occurs strictly between the start-event and the end-event of the trigger token o_t (along the encoding of a multi-timeline of SV). The second considers the case where the e_1-event of token o precedes the start-event of the trigger token: thus, under the future semantics, it holds that the e_1-time of token o coincides with the start time of the trigger token. Finally, the third disjunct considers the case where the e_1-event of token o follows the end-event of the trigger (hence, the related timestamps must coincide).

2. $\rho = o_t \leq_I^{e_1,e_2} o$ and $o \neq o_t$. We assume $e_1 = \mathrm{e}$ and $0 \in I$ (the other cases being simpler). Then,

$$\chi_\rho = \Big[\psi_{\neg x_t} \mathsf{U}_{\geq 0}(\psi(\mathrm{e}, u_t) \wedge \mathsf{F}_I(\psi(e_2, val(o)) \wedge \xi_o^{e_2}))\Big] \vee \Big[\psi_{\neg x_t} \mathsf{U}_{\geq 0}(\psi(\mathrm{e}, u_t) \wedge past^{e_2}_{val(o)} \wedge \xi_o^{e_2})\Big],$$

where the second disjunct captures the situation where the e_2-time of o coincides with the end time of the trigger token o_t, but the e_2-event of o occurs before the end-event of the trigger token.

3. $\rho = o_t \leq_I^{e_1,e_2} o_t$. This case is straightforward and we omit the details.
4. $\rho = o_1 \leq_I^{e_1,e_2} o_2$, with $o_1 \neq o_t$ and $o_2 \neq o_t$. We assume $o_1 \neq o_2$ and $0 \in I$ (the other cases are simpler). Then,

$$\begin{aligned}\chi_\rho = \Big[past^{e_1}_{val(o_1)} \wedge \xi_o^{e_1} \wedge \mathsf{F}_I(\psi(e_2, val(o_2)) \wedge \xi_o^{e_2})\Big] \vee \\ \Big[\mathsf{F}_{\geq 0}\{\psi(e_1, val(o_1)) \wedge \xi_o^{e_1} \wedge \mathsf{F}_I(\psi(e_2, val(o_2)) \wedge \xi_o^{e_2})\}\Big] \vee \\ \Big[past^{e_1}_{val(o_1)} \wedge \xi_o^{e_1} \wedge past^{e_2}_{val(o_2)} \wedge \xi_o^{e_2}\Big] \vee \\ \Big[past^{e_2}_{val(o_2)} \wedge \xi_o^{e_2} \wedge EqTime(\psi(e_1, val(o_1)) \wedge \xi_o^{e_1})\Big] \vee \\ \Big[\mathsf{F}_{\geq 0}\{\psi(e_2, val(o_2)) \wedge \xi_o^{e_2} \wedge EqTime(\psi(e_1, val(o_1)) \wedge \xi_o^{e_1})\}\Big].\end{aligned}$$

The first two disjuncts handle the cases where (under the future semantics) the e_1-event of token o_1 precedes the e_2-event of token o_2, while the last three disjuncts consider the dual situation. In the latter three cases, the e_1-time of token o_1 and the e_2-time of token o_2 are equal.

Note that the MTL formula $\varphi_\forall$ is an MITL formula (resp., $\mathsf{MITL}_{(0,\infty)}$ formula) if the intervals in the trigger rules are non-singular (resp., belong to $Intv_{(0,\infty)}$). □

Encoding of trigger-less rules by a TA. We now deal with trigger-less rules. We start by noting that an existential statement $\mathcal{E}$ in a trigger-less rule requires the existence of an *a priori bounded number* of temporal events satisfying mutual temporal relations (namely, in the worst case, the start time and end time of all tokens associated with some quantifier of $\mathcal{E}$). Thus we can construct a TA for $\mathcal{E}$ which guesses such a chain of events and then checks the temporal relations by means of suitable clock constraints and clock resets. Finally, by the closure of TAs under language union [AD94], we can build a TA for the whole trigger-less rule. Additionally, exploiting also the closure of TAs under intersection, we construct a TA accepting (encodings of) multi-timelines satisfying all trigger-less rules.

Proposition 8.3.4. *One can construct in exponential time a TA $\mathcal{A}_\exists$ over $2^{\mathcal{AP}}$ such that, for each multi-timeline Π of SV and encoding w_Π of Π, w_Π is accepted by $\mathcal{A}_\exists$ iff Π satisfies all the trigger-less rules in R.*
$\mathcal{A}_\exists$ has $2^{O(N_q)}$ states, $O(N_q)$ clocks and maximal constant $O(K_P)$, where N_q is the overall number of quantifiers in the trigger-less rules of R.

We recall that, in the encoding of multi-timelines of SV, we assume that, for distinct state variables $x, x' \in SV$, the domains V_x and $V_{x'}$ are disjoint.

Proof. Let $\mathcal{E}$ be an existential statement for SV such that no token name appears free in $\mathcal{E}$. We first show how to construct a TA $\mathcal{A}_\mathcal{E}$ over $2^{\mathcal{AP}}$ such that for each multi-timeline Π of SV and encoding w_Π of Π, w_Π is accepted by $\mathcal{A}_\mathcal{E}$ iff Π satisfies $\mathcal{E}$. Then, we exploit the well-known effective closure of TA under language union and language intersection to prove the proposition.

Let O be the set of token names existentially quantified in the existential statement $\mathcal{E}$ and, for each $o \in O$, let $val(o)$ be the value of the token referenced by o in the associated quantifier. For each token name $o \in O$, we denote by $Intv^{\mathrm{s}}_o$ (resp., $Intv^{\mathrm{e}}_o$) the set of intervals $J \in Intv$ such that $J = I(\rho)$ for some time-point atom ρ occurring in $\mathcal{E}$ which imposes a time constraint on the start time (resp., end time) of the token referenced by o.

We first outline the construction of $\mathcal{A}_\mathcal{E}$. We associate two clocks with each token name $o \in O$, namely c^{s}_o and c^{e}_o which, intuitively, are reset when the token chosen for o starts and ends, respectively. The clocks c^{s}_o and c^{e}_o are non-deterministically reset when a start-event for $val(o)$ and the related end-event occur along an encoding of a multi-timeline. The automaton $\mathcal{A}_\mathcal{E}$ ensures that the clocks c^{s}_o and c^{e}_o are reset exactly once. $\mathcal{A}_\mathcal{E}$ moves to an accepting state only if all the clocks c^{s}_o and c^{e}_o for each $o \in O$ have been reset and the time constraints that encode the interval atoms in $\mathcal{E}$ are fulfilled. To deal with time-point atoms, we also exploit, like in the previous proofs, a global clock c_{glob} which measures the current time and is never reset: whenever the clock c^{s}_o (resp., c^{e}_o) is reset, we require that the clock constraint $\bigwedge_{I \in Intv^{\mathrm{s}}_o} c_{glob} \in I$ (resp., $\bigwedge_{I \in Intv^{\mathrm{e}}_o} c_{glob} \in I$) is fulfilled.

The TA $\mathcal{A}_\mathcal{E} = (2^{\mathcal{AP}}, Q, q_0, C, \Delta, F)$ is formally defined as follows.

- The set C of clocks is $\{c_{glob}\} \cup \bigcup_{o \in O} \{c^{\mathrm{s}}_o, c^{\mathrm{e}}_o\}$.
- The set of states is $2^{C \setminus \{c_{glob}\}}$. Intuitively, a state keeps track of the clocks in $C \setminus \{c_{glob}\}$ which have been reset so far.
- The initial state q_0 is $\emptyset$.
- The set of final states F is given by the singleton $\{C \setminus \{c_{glob}\}\}$. In such a state all clocks different from c_{glob} have been reset.
- The transition relation Δ consists of the transitions $(C_1, P, \theta \wedge \theta_{glob}, Res, C_2)$ such that *either* (i) $C_1 = C \setminus \{c_{glob}\}$, $C_2 = C_1$, $Res = \emptyset$, $\theta = \top$, and $\theta_{glob} = \top$ (intuitively $\mathcal{A}_\mathcal{E}$ loops unconditionally in its final state), *or* (ii) $C_1 \subset C \setminus \{c_{glob}\}$, $C_2 \supseteq C_1$ ($\mathcal{A}_\mathcal{E}$ has not reached its final state yet), and the following conditions hold:
  - for each $c^{\mathrm{s}}_o \in C_2 \setminus C_1$, there is a main proposition in P of the form $(v', val(o))$ for some v'.
  - for each $o \in O$, $c^{\mathrm{e}}_o \in C_2 \setminus C_1$ if and only if $c^{\mathrm{s}}_o \in C_1$ and $(val(o), v') \in P$ for some v'.

- if $C_2 \subset C \setminus \{c_{glob}\}$ (in this case $\mathcal{A}_\mathcal{E}$ is not transitioning to its final state), then $\theta = \top$.
  Conversely, if $C_2 = C \setminus \{c_{glob}\}$ (here $\mathcal{A}_\mathcal{E}$ moves to the final state), then $\theta = \bigwedge_{\rho \in \mathbf{A}} code(\rho)$, where $\mathbf{A}$ is the set of *interval* atoms of $\mathcal{E}$ and for each interval atom $\rho \in \mathbf{A}$ of the form $o_1 \leq_I^{e_1,e_2} o_2$, the clock constraint $code(\rho)$ is defined as follows:
  - if $c_{o_2}^{e_2} \notin C_1$ and $c_{o_1}^{e_1} \notin C_1$, then $code(\rho) = c_{o_2}^{e_2} - c_{o_1}^{e_1} \in I$ (in this case, both $c_{o_2}^{e_2}$ and $c_{o_1}^{e_1}$ are reset simultaneously by the transition to the final state C_2, meaning that o_2's e_2-event and o_1's e_1-event have the same timestamp; hence it must be that $c_{o_2}^{e_2} - c_{o_1}^{e_1} = 0 \in I$ for the atom to be satisfied);
  - if $c_{o_2}^{e_2} \in C_1$ and $c_{o_1}^{e_1} \in C_1$, then $code(\rho) = c_{o_1}^{e_1} - c_{o_2}^{e_2} \in I$;
  - if $c_{o_2}^{e_2} \in C_1$ and $c_{o_1}^{e_1} \notin C_1$, then $code(\rho) = c_{o_2}^{e_2} \in [0,0] \wedge c_{o_2}^{e_2} \in I$ (o_2's e_2-event and o_1's e_1-event must have the same timestamp; as before, it must be that $0 \in I$);
  - if $c_{o_2}^{e_2} \notin C_1$ and $c_{o_1}^{e_1} \in C_1$, then $code(\rho) = c_{o_1}^{e_1} \in I$.
- $\theta_{glob} = \bigwedge_{c_o^e \in C_2 \setminus C_1} \bigwedge_{I \in Intv_o^e} c_{glob} \in I$.
- $Res = C_2 \setminus C_1$.

Note that $\mathcal{A}_\mathcal{E}$ has $2^{O(m)}$ states, $O(m)$ clocks and maximal constant $O(K)$, where m is the number of quantifiers in $\mathcal{E}$ and K is the maximal constant in $\mathcal{E}$.

Given a trigger-less rule $\mathcal{R} = \top \rightarrow \mathcal{E}_1 \vee \mathcal{E}_2 \vee \ldots \vee \mathcal{E}_k$, we construct the TA $\mathcal{A}_\mathcal{R}$ resulting from the union of the automata $\mathcal{A}_{\mathcal{E}_1}, \ldots, \mathcal{A}_{\mathcal{E}_k}$. Then the TA $\mathcal{A}_\exists$ is obtained as intersection of the automata $\mathcal{A}_\mathcal{R}$, for all $\mathcal{R} \in R$ being trigger-less rules. By [AD94], $\mathcal{A}_\exists$ has $2^{O(N_q)}$ states, $O(N_q)$ clocks, and maximal constant $O(K_P)$, where N_q is the overall number of quantifiers in the trigger-less rules of R. □

Conclusion of the construction. By applying Proposition 8.3.2, 8.3.3, 8.3.4 and well-known results about TAs and MTL over finite timed words [AD94; OW07], we obtain the main result of this section.

Theorem 8.3.5. *The future TP problem with simple trigger rules is decidable (with non-primitive recursive complexity). Moreover, if the intervals in the atoms of the trigger rules are non-singular (resp., belong to $Intv_{(0,\infty)}$), then the problem is in **EXPSPACE** (resp., in **PSPACE**).*

Proof. Let us consider an instance $P = (SV, R)$ of the problem with maximal constant K_P. Let $N_v = \sum_{x \in SV} |V_x|$, N_q be the overall number of quantifiers in the trigger-less rules of R, N_A the maximum number of atoms in a trigger rule of R, and $N_\mathcal{E}$ the maximum number of existential statements in a trigger rule of R.

By Proposition 8.3.2, 8.3.3, 8.3.4 and the effective closure of TAs under language intersection [AD94], we can build:

- a TA $\mathcal{A}_P$—namely, the intersection of $\mathcal{A}_{SV}$ from Proposition 8.3.2 and $\mathcal{A}_\exists$ from Proposition 8.3.4—having $2^{O(N_q+N_v)}$ states, $O(N_q + |SV|)$ clocks, and maximal constant $O(K_P)$,
- and an MTL formula $\varphi_\forall$ with $O(|R| \cdot N_A \cdot N_\mathcal{E} \cdot (|Intv_R| + N_v^2))$ distinct subformulas and maximal constant $O(K_P)$,

such that there is a future plan for P iff $\mathcal{L}_T(\mathcal{A}_P) \cap \mathcal{L}_T(\varphi_\forall) \neq \emptyset$. By [OW07], checking non-emptiness of $\mathcal{L}_T(\mathcal{A}_P) \cap \mathcal{L}_T(\varphi_\forall)$ is decidable. Thus the first part of the theorem holds.

For the second part, assume that the intervals in the trigger rules are non-singular (resp., belong to $Intv_{(0,\infty)}$). By Proposition 8.3.3, $\varphi_\forall$ is an MITL (resp., $\text{MITL}_{(0,\infty)}$) formula. By [AFH96], one can build a TA $\mathcal{A}_\forall$ accepting $\mathcal{L}_T(\varphi_\forall)$ having

- $2^{O(K_P \cdot |R| \cdot N_A \cdot N_\mathcal{E} \cdot (|Intv_R| + N_v^2))}$ states, $O(K_P \cdot |R| \cdot N_A \cdot N_\mathcal{E} \cdot (|Intv_R| + N_v^2))$ clocks
- (resp., $2^{O(|R| \cdot N_A \cdot N_\mathcal{E} \cdot (|Intv_R| + N_v^2))}$ states, $O(|R| \cdot N_A \cdot N_\mathcal{E} \cdot (|Intv_R| + N_v^2))$ clocks),

and maximal constant $O(K_P)$.

Non-emptiness of a TA $\mathcal{A}$ can be solved by an **NPSPACE** = **PSPACE** search algorithm over the *region automaton* of $\mathcal{A}$,[2] which uses work space *logarithmic* in the number of control states of $\mathcal{A}$ and *polynomial* in the number of clocks and in the length of the encoding of the maximal constant of $\mathcal{A}$ [AD94]. Thus, since $\mathcal{A}_P$, $\mathcal{A}_\forall$, and the intersection $\mathcal{A}_\wedge$ of $\mathcal{A}_P$ and $\mathcal{A}_\forall$ can be constructed on the fly—that is, by looking at their transition relations Δ, one can determine, given a state q, a successor q' and the connecting transition, along with the associated constraints and clocks to reset—and the search in the region automaton of $\mathcal{A}_\wedge$ can be done without explicitly constructing $\mathcal{A}_\wedge$, the result follows. □

In the next section, we consider future TP with simple trigger rules and non-singular intervals in the atoms of trigger rules (resp., intervals in $Intv_{(0,\infty)}$), and prove a matching complexity *lower bound*: **EXPSPACE**-completeness (resp., **PSPACE**-completeness) of the problem follows.

8.4 Future TP with simple trigger rules and non-singular intervals: hardness

In this section, we first consider the future TP problem with simple trigger rules and non-singular intervals, and prove that it is **EXPSPACE**-hard by a polynomial-

[2] The *region automaton* of $\mathcal{A}$ features states of the form (q, r), where q is a state of $\mathcal{A}$ and r a *region*: every region specifies, for each clock c of $\mathcal{A}$, whether its value is integer or not (and, if it is, its value up to K_c, the maximum constant to which c is compared), and the ordering of the fractional parts of the clocks.

time reduction from the *domino-tiling problem for grids with rows of single exponential length* (it has already been presented in Section 2.4), which is known to be **EXPSPACE**-complete [Har12]. Since the reduction is standard, we refer the reader to Appendix F.3 for the details of the construction.

Theorem 8.4.1. *The future TP problem, even with* one state variable, *with simple trigger rules and non-singular intervals is **EXPSPACE**-hard (under polynomial-time reductions).*

By putting together Theorem 8.3.5, **EXPSPACE**-completeness follows.

We now focus on the case with intervals in $Intv_{(0,\infty)}$, proving that the problem is **PSPACE**-hard (and thus **PSPACE**-complete by Theorem 8.3.5) by reducing periodic SAT to it in polynomial time.

The problem *periodic SAT* is defined as follows [Pap94]. We are given a Boolean formula φ in *conjunctive normal form*, defined over two sets of variables, $\Gamma = \{x_1, \ldots, x_n\}$ and $\Gamma^{+1} = \{x_1^{+1}, \ldots, x_n^{+1}\}$, namely,

$$\varphi = \bigwedge_{t=1}^{m} \Big(\bigvee_{x \in (\Gamma \cup \Gamma^{+1}) \cap L_t^+} x \vee \bigvee_{x \in (\Gamma \cup \Gamma^{+1}) \cap L_t^-} \neg x \Big),$$

where m is the number of conjuncts of φ and, for $1 \leq t \leq m$, L_t^+ (resp., L_t^-) is the set of variables occurring non-negated (resp., negated) in the t-th conjunct of φ. Moreover, the formula φ^j, for $j \in \mathbb{N} \setminus \{0\}$, is defined as φ in which we replace each variable $x_i \in \Gamma$ by a fresh one x_i^j, and $x_i^{+1} \in \Gamma^{+1}$ by x_i^{j+1}. Periodic SAT is then the problem of deciding the satisfiability of the (infinite-length) formula

$$\Phi = \bigwedge_{j \in \mathbb{N} \setminus \{0\}} \varphi^j,$$

that is, deciding the existence of a truth assignment of (infinitely many) variables x_i^j, for $i = 1, \ldots, n$, $j \in \mathbb{N} \setminus \{0\}$, satisfying Φ.

Periodic SAT is **PSPACE**-complete [Pap94]; in particular membership to such a class is proved by showing that one can equivalently check the satisfiability of the (finite-length) formula $\Phi_f = \bigwedge_{j=1}^{2^{2n}+1} \varphi^j$. Intuitively, 2^{2n} is the number of possible truth assignments to variables of $\Gamma \cup \Gamma^{+1}$, thus, after $2^{2n} + 1$ copies of φ, we can find a repeated assignment: from that point, we can just loop through the previous assignments.

We now reduce periodic SAT to our problem. Hardness also holds when only a single state variable is involved, and also restricting to intervals of the form $[0, a]$.

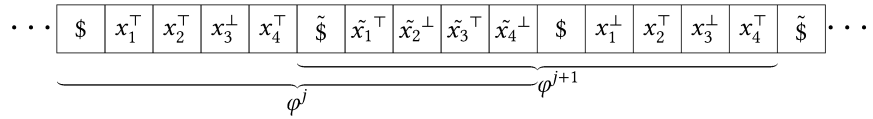


Figure 8.4: Let the formula φ be defined over two sets of variables, $\Gamma = \{x_1, x_2, x_3, x_4\}$ and $\Gamma^{+1} = \{x_1^{+1}, x_2^{+1}, x_3^{+1}, x_4^{+1}\}$. The j-th copy (we assume j is odd) of φ, i.e., φ^j, is satisfied by the assignment $x_1^j \mapsto \top, x_2^j \mapsto \top, x_3^j \mapsto \perp, x_4^j \mapsto \top, x_1^{j+1} \mapsto \top, x_2^{j+1} \mapsto \perp, x_3^{j+1} \mapsto \top, x_4^{j+1} \mapsto \perp$. The analogous for φ^{j+1}.

Theorem 8.4.2. *The future TP problem, even with* one state variable*, with simple trigger rules and intervals* $[0, a]$*,* $a \in \mathbb{N} \setminus \{0\}$*, is* ***PSPACE****-hard (under polynomial-time reductions).*

Proof. Let us define the state variable $y = (V, T, D)$, where

- $V = \{\$, \tilde{\$}, stop\} \cup \{x_i^\top, x_i^\perp, \tilde{x_i}^\top, \tilde{x_i}^\perp \mid i = 1, \ldots, n\}$,
- $T(\$) = \{x_1^\top, x_1^\perp\}$, $T(\tilde{\$}) = \{\tilde{x_1}^\top, \tilde{x_1}^\perp\}$ and $T(stop) = \{stop\}$,
- for $i = 1, \ldots, n-1$, $T(x_i^\top) = T(x_i^\perp) = \{x_{i+1}^\top, x_{i+1}^\perp\}$,
- for $i = 1, \ldots, n-1$, $T(\tilde{x_i}^\top) = T(\tilde{x_i}^\perp) = \{\tilde{x_{i+1}}^\top, \tilde{x_{i+1}}^\perp\}$,
- $T(x_n^\top) = T(x_n^\perp) = \{\tilde{\$}, stop\}$,
- $T(\tilde{x_n}^\top) = T(\tilde{x_n}^\perp) = \{\$, stop\}$, and
- for all $v \in V$, $D(v) = [2, +\infty[$.

Intuitively, we represent an assignment of variables x_i^j by means of a timeline for y: after every occurrence of the symbol \$, n tokens are present, one for each x_i, and the value $x_i^\top$ (resp., $x_i^\perp$) represents a positive (resp., negative) assignment of x_i^j, for some *odd* $j \geq 1$. Then, there is an occurrence of $\tilde{\$}$, after which n more tokens occur, again one for each x_i, and the value $\tilde{x_i}^\top$ (resp., $\tilde{x_i}^\perp$) represents a positive (resp., negative) assignment of x_i^j, for some *even* $j \geq 2$. See Figure 8.4 for an example.

We start with the next simple trigger rules, one for each $v \in V$:

$$o[y = v] \longrightarrow o \leq^{\mathrm{s,e}}_{[0,2]} o.$$

Paired with the constraint function D, they enforce all tokens' durations to be *exactly* 2: intuitively, since we exclude singular intervals, requiring, for instance, that a token o' starts t instants of time after the end of o, with $t \in [\ell, \ell + 1]$ and even $\ell \in \mathbb{N}$, boils down to o' starting *exactly* ℓ instants after the end of o. We also observe that, given the constant token duration, the density of the time domain does not play any role in this proof.

We now add the next rules:

- $\top \longrightarrow \exists o[y = \$].o \geq^{\mathrm{s}}_{[0,1]} 0$;
- $\top \longrightarrow \exists o[y = \tilde{\$}].o \geq^{\mathrm{s}}_{[0,1]} (2^{2n} + 1) \cdot 2(n + 1)$;
- $\top \longrightarrow \exists o[y = stop].o \geq^{\mathrm{s}}_{[0,1]} (2^{2n} + 2) \cdot 2(n + 1)$.

They respectively impose that (i) a token with value \$ starts exactly at $t = 0$ (recall that the duration of every token is 2); (ii) there exists a token with value $\tilde{\$}$ starting at $t = (2^{2n} + 1) \cdot 2(n + 1)$; (iii) a token with value *stop* starts at $t = (2^{2n} + 2) \cdot 2(n + 1)$. We are forcing the timeline to encode truth assignments for variables $x^1_1, \ldots, x^1_n, \ldots, x^{2^{2n}+2}_1, \ldots, x^{2^{2n}+2}_n$: as a matter of fact, we will decide satisfiability of the finite formula $\Phi_f = \bigwedge_{j=1}^{2^{2n}+1} \varphi^j$, which is equivalent to Φ.

We now consider the next rules, that enforce the satisfaction of each φ^j or, equivalently, of φ over the assignments of $(x^j_1, \ldots, x^j_n, x^{j+1}_1, \ldots, x^{j+1}_n)$.

For the t-th conjunct of φ, we define the future simple rule:

$$
\begin{aligned}
o[y = \tilde{\$}] \longrightarrow\ & \\
& \Big(\bigvee_{x_i \in \Gamma \cap L^+_t} \exists o'[y = \tilde{x_i}^{\top}].o \leq^{\mathrm{e,s}}_{[0,4n]} o' \Big) \vee \Big(\bigvee_{x^{+1}_i \in \Gamma^{+1} \cap L^+_t} \exists o'[y = x^{\top}_i].o \leq^{\mathrm{e,s}}_{[0,4n]} o' \Big) \vee \\
& \Big(\bigvee_{x_i \in \Gamma \cap L^-_t} \exists o'[y = \tilde{x_i}^{\bot}].o \leq^{\mathrm{e,s}}_{[0,4n]} o' \Big) \vee \Big(\bigvee_{x^{+1}_i \in \Gamma^{+1} \cap L^-_t} \exists o'[y = x^{\bot}_i].o \leq^{\mathrm{e,s}}_{[0,4n]} o' \Big) \vee \\
& \qquad \exists o''[y = stop].o \leq^{\mathrm{e,s}}_{[0,2n]} o''.
\end{aligned}
$$

Basically, this rule (the rule where the trigger has value \$ being analogous) states that, after every occurrence of $\tilde{\$}$, a token o', making true at least a (positive or negative) literal in the conjunct, must occur by $4n$ time instants (i.e., before the following occurrence of $\tilde{\$}$). The disjunct $\exists o''[y = stop].o \leq^{\mathrm{e,s}}_{[0,2n]} o''$ is present just to avoid evaluating φ on the n tokens before (the first occurrence of) *stop*.

The variable y and all synchronization rules can be generated in time polynomial in $|\varphi|$ (in particular, all interval bounds and time constants of time-point atoms have a value, encoded in binary, in $O(2^{2n})$). □

By Theorem 8.3.5 and Theorem 8.4.2, **PSPACE**-completeness of future TP with simple trigger rules and intervals in $Intv_{(0,\infty)}$ follows.

In the next section we focus on a different restriction of the TP problem, which will allow us to devise a **NP** planning algorithm for it.

8.5 TP with trigger-less rules only is NP-complete

In this section we describe a TP algorithm, for planning domains where only trigger-less rules are allowed, which requires a polynomial number of (non-deterministic) computation steps. We recall that trigger-less rules are useful, for instance, to express initial, intermediate conditions and reachability goals.

We want to start with the following example, with which we highlight that there is no *polynomial-size* plan for some problem instances/domains. Thus, an explicit enumeration of all tokens of a multi-timeline *does not* represent a suitable polynomial-size certificate.

Example 8.5.1. Let us consider the following planning domain. We denote by $p(i)$ the i-th prime number, assuming $p(1) = 1$, $p(2) = 2$, $p(3) = 3$, $p(4) = 5$,.... We define, for $i = 1, \dots, n$, the state variables $x_i = (\{v_i\}, \{(v_i, v_i)\}, D_{x_i})$ with $D_{x_i}(v_i) = [p(i), p(i)]$. The following rule

$$\top \rightarrow \exists o_1[x_1 = v_1] \cdots \exists o_n[x_n = v_n]. \bigwedge_{i=1}^{n-1} o_i \leq^{\mathrm{e,e}}_{[0,0]} o_{i+1}$$

is asking for the existence of a "synchronization point", where n tokens (one for each variable) have their ends aligned. Due to the allowed token durations, the first such time point is $\prod_{i=1}^{n} p(i) \geq 2^{n-1}$. Hence, in any plan, the timeline for x_1 features at least 2^{n-1} tokens: no explicit polynomial-time enumeration of such tokens is possible.

As a consequence, there exists no trivial guess-and-check **NP** algorithm. Conversely, one can easily prove the following result.

Theorem 8.5.2. *The TP problem with trigger-less rules only is **NP**-hard, even with one state variable (under polynomial-time reductions).*

Proof. There is a trivial reduction from the problem of the existence of a Hamiltonian path in a directed graph.

Given a directed graph $G = (V, E)$, with $|V| = n$, we define the state variable $x = (V, E, D_x)$, where $D_x(v) = [1, 1]$ for each $v \in V$. We add the following trigger-less rules, one for each $v \in V$:

$$\top \rightarrow \exists o[x = v].o \geq^{\mathrm{s}}_{[0,n-1]} 0.$$

The rule for $v \in V$ requires that there is a token $(x, v, 1)$ along the timeline for x, which starts no later than $n - 1$. It is easy to check that G contains a Hamiltonian path if and only if there exists a plan for the defined planning domain. □

We now present the aforementioned non-deterministic polynomial-time algorithm, proving that timeline-based planning with trigger-less rules is in **NP**.

We preliminarily have to derive a *finite horizon* (namely, the end time of the last token) for the plans of a (any) instance of TP with trigger-less rules. That is, if an instance $P = (SV, R)$ admits a plan, then P also has a plan whose horizon is

no greater than a given bound. Analogously, we have to calculate a *bound to the maximum number of tokens* in a plan. Both can be obtained from the constructions of the TAs described in the proof of Theorem 8.3.5: since only trigger-less rules are now allowed, we disregard the construction of the MTL formula $\varphi_\forall$, and restrict our attention to the TA $\mathcal{A}_P$ (i.e., the intersection between $\mathcal{A}_{SV}$ for the state variables in SV from Proposition 8.3.2 and $\mathcal{A}_\exists$ for the trigger-less rules in R from Proposition 8.3.4), which has $\alpha_s = 2^{O(N_q + \sum_{x \in SV} |V_x|)}$ states, $\alpha_c = O(N_q + |SV|)$ clocks and maximum constant $\alpha_K = O(K_p)$, where N_q is the overall number of quantifiers in the trigger-less rules of R, and accepts all and only the encodings w_Π of multi-timelines Π of SV satisfying all the trigger-less rules in R.

The language emptiness checking algorithm for TAs executed over $\mathcal{A}_P$ visits the (untimed) region automaton for $\mathcal{A}_P$ [AD94], which features $\alpha = \alpha_s \cdot O(\alpha_c! \cdot 2^{\alpha_c} \cdot 2^{2N_q^2} \cdot (2\alpha_K + 2)^{\alpha_c})$ states[3], trying to find a path, from the initial state to a final state, whose length can clearly be bounded by the number of states. We observe that each edge/transition of the region automaton in such a path corresponds, in the worst case, to the start point of a token for each timeline for the variables in SV (i.e., assuming that all these tokens start simultaneously). This yields a bound on the number of tokens, which is $\alpha \cdot |SV|$. We can also derive a bound on the horizon of the plan, which is $\alpha \cdot |SV| \cdot (\alpha_K + 1)$, as every transition taken in $\mathcal{A}_P$ may let at most $\alpha_K + 1$ time units pass, as α_K accounts in particular for the maximum constant to which a (any) clock is compared.[4]

Having this pair of bounds, we are now ready to describe the two main phases of the algorithm, corresponding to the following pair of observations. On the one hand, (i) each trigger-less rule requires, as we said, the existence of an *a priori bounded number* of temporal events satisfying mutual temporal relations (namely, in the worst case, the start time and end time of all tokens associated with the quantifiers of one of its existential statements). On the other hand, (ii) timelines for different state variables evolve independently of each other. In order to deal with (i), we non-deterministically position such temporal events along timelines; as for (ii), we enforce a correct evolution of each timeline between pairs of "positioned" events, completely independently of the other timelines.

Non-deterministic token positioning. The algorithm starts by non-deterministically selecting, for every trigger-less rule in R, a disjunct—and deleting all the others. Then, for every (left) quantifier $o_i[x_i = v_i]$, it generates the integer part of both the start and the end time of the token for x_i to which o_i is mapped. We

[3] The factor $2^{2N_q^2}$ is present due to diagonal clock constraints in $\mathcal{A}_P$.

[4] Clearly, and unbounded quantity of time units may pass, but after $\alpha_K + 1$ the last region of the region automaton will certainly have been reached.

call such time instants, respectively, $\mathrm{s}_{int}(o_i)$ and $\mathrm{e}_{int}(o_i)$.[5] We observe that all start/end time $\mathrm{s}_{int}(o_i)$ and $\mathrm{e}_{int}(o_i)$, being less or equal to $\alpha \cdot |SV| \cdot (\alpha_K + 1)$ (the finite horizon bound), have an integer part that can be encoded with polynomially many bits (and thus can be generated in polynomial time).

Let us now consider the fractional parts of the start/end time of the tokens associated with quantifiers. We denote them by $\mathrm{s}_{frac}(o_i)$ and $\mathrm{e}_{frac}(o_i)$. The algorithm non-deterministically generates an *order* of all such fractional parts. In particular we have to specify, for every token start/end time, whether it is integer ($\mathrm{s}_{frac}(o_i) = 0$, $\mathrm{e}_{frac}(o_i) = 0$) or not ($\mathrm{s}_{frac}(o_i) > 0$, $\mathrm{e}_{frac}(o_i) > 0$). Every such possibility can be generated in polynomial time.

Some trivial tests should now be performed, namely that, for all o_i, $\mathrm{s}_{int}(o_i) \leq \mathrm{e}_{int}(o_i)$, each token is assigned an end time equal or greater than its start time, and no two tokens for the same variable are overlapping.

It is routine to check that, if we change the start/end time of (some of the) tokens associated with quantifiers, but we leave unchanged (*i*) all the integer parts, (*ii*) zeroness/non-zeroness of fractional parts, and (*iii*) the fractional parts' order, then the satisfaction of the (atoms in the) trigger-less rules does not change. This is due to all the constants being integers.[6] Therefore we can now check whether all rules are satisfied.

Enforcing legal token durations and timeline evolutions. We now continue by checking that: (*i*) all tokens associated with a quantifier have a legal duration, and that (*ii*) there exists a legal timeline evolution between pairs of adjacent such tokens over the same variable (here *adjacent* means that there is no other token associated with a quantifier in between). We will enforce all these requirements as constraints of a *linear problem*, which can be solved in deterministic polynomial time (e.g., using the ellipsoid algorithm). When needed, we use *strict inequalities*, which are not allowed in linear programs. We shall show later how to convert these into non-strict ones.

We start by associating non-negative variables $\alpha_{o_i,s}, \alpha_{o_i,e}$ with the fractional parts of the start/end times $\mathrm{s}_{frac}(o_i)$, $\mathrm{e}_{frac}(o_i)$ of every token for a quantifier $o_i[x_i = v_i]$. First, we add the linear constraints

$$0 \leq \alpha_{o_i,s} < 1, \quad 0 \leq \alpha_{o_i,e} < 1.$$

[5] We can assume w.l.o.g. that all quantifiers refer to distinct tokens. As a matter of fact, the algorithm can non-deterministically choose to make two (or more) quantifiers $o_i[x_i = v_i]$ and $o_j[x_i = v_i]$ over the same variable and value "collapse" to the same token just by rewriting all occurrences of o_j as o_i in the atoms of the rules.

[6] We may observe that, by leaving unchanged all the integer parts and the fractional parts' order, the region of the region graph of the timed automaton does not change.

Then, we also need to enforce that the values of $\alpha_{o_i,s}, \alpha_{o_i,e}$ respect the decided order of the fractional parts: for example,

$$0 = \alpha_{o_i,s} = \alpha_{o_j,s} < \alpha_{o_k,s} < \ldots < \alpha_{o_j,e} < \alpha_{o_i,e} = \alpha_{o_k,e} < \ldots$$

To enforce requirement (i), we set, for all $o_i[x_i = v_i]$,

$$a \leq (\mathrm{e}_{int}(o_i) + \alpha_{o_i,e}) - (\mathrm{s}_{int}(o_i) + \alpha_{o_i,s}) \leq b$$

where $D_{x_i}(v_i) = [a, b]$. Clearly, strict ($<$) inequalities must be used for a left/right open interval.

To enforce requirement (ii), namely that there exists a legal timeline evolution between each pair of adjacent tokens for the same state variable, say $o_i[x_i = v_i]$ and $o_j[x_i = v_j]$, we proceed as follows (for a correct evolution between $t = 0$ and the first token, analogous considerations can be made).

Let us consider each state variable $x_i = (V_i, T_i, D_i)$ as a directed graph $G = (V_i, T_i)$ where D_i is a function associating with each vertex $v \in V_i$ a duration range. We have to decide whether or not there exist

- a path in G, possibly with repeated vertices and edges, $v_0 \cdot v_1 \cdot \cdot \cdot v_{n-1} \cdot v_n$ where $v_0 \in T_i(v_i)$ and v_n with $v_j \in T_i(v_n)$ are non-deterministically generated, and
- a list of non-negative real values $d_0, \ldots, d_n$, such that

$$\sum_{t=0}^{n} d_t = (\mathrm{s}_{int}(o_j) + \alpha_{o_j,s}) - (\mathrm{e}_{int}(o_i) + \alpha_{o_i,e}),$$

and for all $s = 0, \ldots, n$, $d_s \in D_i(v_s)$.

We guess a set of integers $\{\alpha'_{u,v} \mid (u, v) \in T_i\}$. Intuitively, $\alpha'_{u,v}$ is the number of times the solution path traverses (u, v). Since every time an edge is traversed a new token starts, each $\alpha'_{u,v}$ is bounded by the number of tokens, i.e., by $\alpha \cdot |SV|$. Hence the binary encoding of $\alpha'_{u,v}$ can be generated in polynomial time.

We then perform the following deterministic steps.

1. We consider the subset E' of edges of G, $E' = \{(u, v) \in T_i \mid \alpha'_{u,v} > 0\}$. We check whether E' induces a strongly (undirected) connected subgraph of G.
2. We check whether
   - $\sum_{(u,v)\in E'} \alpha'_{u,v} = \sum_{(v,w)\in E'} \alpha'_{v,w}$, for all $v \in V_i \setminus \{v_0, v_n\}$;
   - $\sum_{(u,v_0)\in E'} \alpha'_{u,v_0} = \sum_{(v_0,w)\in E'} \alpha'_{v_0,w} - 1$;
   - $\sum_{(u,v_n)\in E'} \alpha'_{u,v_n} = \sum_{(v_n,w)\in E'} \alpha'_{v_n,w} + 1$.
3. For all $v \in V_i \setminus \{v_0\}$, we define $y_v = \sum_{(u,v)\in E'} \alpha'_{u,v}$ (y_v is the number of times the solution path gets into v). Moreover, $y_{v_0} = \sum_{(v_0,u)\in E'} \alpha'_{v_0,u}$.

4. We define the real non-negative variables z_v, for every $v \in V_i$ (z_v is the total waiting time of the path on the node v), subject to the following constraints:

$$a \cdot y_v \leq z_v \leq b \cdot y_v,$$

where $D_i(v) = [a, b]$ (an analogous constraint should be written for open intervals). Finally we set:

$$\sum_{v \in V_i} z_v = (\mathrm{s}_{int}(o_j) + \alpha_{o_j,s}) - (\mathrm{e}_{int}(o_i) + \alpha_{o_i,e}).$$

Steps (1.) and (2.) together check that the values $\alpha'_{u,v}$ for the arcs specify a directed Eulerian path from v_0 to v_n in a multigraph. Indeed, the following theorem holds.

Theorem 8.5.3. *[Jun13] Let $G' = (V', E')$ be a directed multigraph (E' is a multiset). G' has a (directed) Eulerian path from v_0 to v_n if and only if:*

- *the undirected version of G' is connected, and*
- $|\{(u, v) \in E'\}| = |\{(v, w) \in E'\}|$, *for all* $v \in V' \setminus \{v_0, v_n\}$;
- $|\{(u, v_0) \in E'\}| = |\{(v_0, w) \in E'\}| - 1$;
- $|\{(u, v_n) \in E'\}| = |\{(v_n, w) \in E'\}| + 1$.

Steps (3.) and (4.) evaluate the waiting times of the path in some vertex v with duration interval $[a, b]$. If the solution path visits the vertex y_v times, then every single visit must take at least a and at most b units of time. Hence the overall visitation time is in between $a \cdot y_v$ and $b \cdot y_v$. Vice versa, if the total visitation time is in between $a \cdot y_v$ and $b \cdot y_v$, then it can be slit into y_v intervals, each one falling into $[a, b]$.

The algorithm concludes by solving the linear program given by the variables $\alpha_{o_i,s}$ and $\alpha_{o_i,e}$ for each quantifier $o_i[x_i = v_i]$, and for each pair of adjacent tokens in the same timeline for x_i, for each $v \in V_i$, the variables z_v subject to their constraints.

Finally, in order to conform to linear programming, we have to replace all strict inequalities with non-strict ones. It is straightforward to observe that all constraints involving strict inequalities we have written so far are of (or can easily be converted into) the following forms: $\xi s < \eta q + k$ or $\xi s > \eta q + k$, where s and q are variables, and ξ, η, k are constants. We replace them, respectively, by $\xi s - \eta q - k + \beta_t \leq 0$ and $\xi s - \eta q - k - \beta_t \geq 0$, where β_t is an additional fresh non-negative variable, which is *local* to a single constraint. We observe that the original inequality and the new one are equivalent if and only if β_t is a small enough *positive* number. Moreover, we add another non-negative variable, say r, which is subject to a constraint $r \leq \beta_t$, for each of the introduced variables

β_t (i.e., r is less than or equal to the minimum of all β_t's). Finally, we maximize the value of r when solving the linear program. We have that $\max r > 0$ if and only if there is an admissible solution where the values of all β_t's are positive (and thus the original strict inequalities hold true).

This ends the description of the planning algorithm. We can thus conclude the section with the main result.

Theorem 8.5.4. *The TP problem with trigger-less rules only is **NP**-complete.*

In the next section, using the results on the variants of TP problem, we shall study MC over timelines.

8.6 MC for MITL over timelines

In this section we deal with the ultimate goal of the present chapter: as mentioned, we want to model check systems specified in terms of timelines. More precisely, a system is described as a set of state variables along with a set of synchronization rules over them (a TP domain) $P = (SV, R)$. As mentioned in the introduction, the property specification language we will be assuming is the logic MITL.

We first recall the encoding of multi-timelines already adopted in Section 8.3.2, over which we interpret MITL, that exploits the set $\mathcal{AP}$ of proposition letters

$$\mathcal{AP} = \bigcup_{x \in SV} Main_x \cup Deriv,$$

where in particular $Main_x = ((\{beg_x\} \cup V_x) \times V_x) \cup (V_x \times \{end_x\})$. The tags beg_x and end_x mark the beginning and the end of a timeline for x, and a pair $(v, v') \in Main_x$ represents a transition of the value taken by x from v to v' (a token for x with value v' follows a token with value v). The already introduced formula $\psi(\mathrm{s}, v) = (beg_x, v) \vee \bigvee_{u \in V_x}(u, v)$, states that a start-event for a token for x with value v occurs at the current time. Finally, $EqTime(\theta) = \theta \vee [\neg p_> \mathsf{U}_{\geq 0}(\neg p_> \wedge \theta)]$, where $p_> \in Deriv$, is satisfied by an encoding of a multi-timeline at the current time if θ eventually holds at a position whose timestamp coincides with the current one.

Before formalizing the *MC problem for MITL formulas over timelines*, we want to start with an easy example of a system whose components are described by timelines, over which we check some properties encoded by MITL formulas.

Example 8.6.1. Let us consider the following system, consisting of a *temperature sensor*, a *processing unit*, and a *data transmission unit*. These components are modelled by three state variables, $x_{\mathtt{temp}} = (V_{\mathtt{temp}}, T_{\mathtt{temp}}, D_{\mathtt{temp}})$, $x_{\mathtt{proc}} = (V_{\mathtt{proc}}, T_{\mathtt{proc}}, D_{\mathtt{proc}})$ and $x_{\mathtt{transm}} = (V_{\mathtt{transm}}, T_{\mathtt{transm}}, D_{\mathtt{transm}})$, where

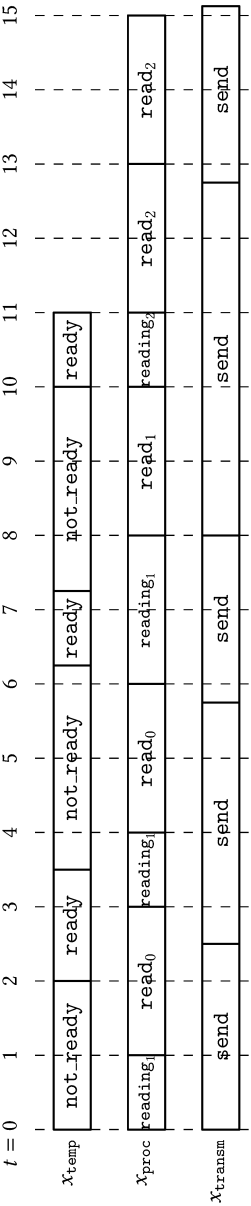


Figure 8.5: Example of computation for the defined system

- $V_{\texttt{temp}} = \{\texttt{ready}, \texttt{not_ready}\}$,
  $T_{\texttt{temp}}(\texttt{ready}) = \{\texttt{not_ready}\}$, $T_{\texttt{temp}}(\texttt{not_ready}) = \{\texttt{ready}\}$,
  $D_{\texttt{temp}}(\texttt{ready}) = [1, 2]$, $D_{\texttt{temp}}(\texttt{not_ready}) = [2, 3]$;
- $V_{\texttt{proc}} = \{\texttt{reading}_1, \texttt{reading}_2, \texttt{read}_0, \texttt{read}_1, \texttt{read}_2\}$,
  $T_{\texttt{proc}}(\texttt{reading}_1) = \{\texttt{read}_0, \texttt{read}_1\}$, $T_{\texttt{proc}}(\texttt{reading}_2) = \{\texttt{read}_1, \texttt{read}_2\}$, $T_{\texttt{proc}}(\texttt{read}_0) = \{\texttt{reading}_1\}$, $T_{\texttt{proc}}(\texttt{read}_1) = \{\texttt{reading}_2\}$, $T_{\texttt{proc}}(\texttt{read}_2) = \{\texttt{read}_2\}$,
  $D_{\texttt{proc}}(\texttt{reading}_1) = D_{\texttt{proc}}(\texttt{reading}_2) = [1, 2]$, $D_{\texttt{proc}}(\texttt{read}_0) = D_{\texttt{proc}}(\texttt{read}_1) = D_{\texttt{proc}}(\texttt{read}_2) = [2, 3]$;
- $V_{\texttt{transm}} = \{\texttt{send}\}$, $T_{\texttt{transm}}(\texttt{send}) = \{\texttt{send}\}$, $D_{\texttt{transm}}(\texttt{send}) = [2, 5]$.

The temperature sensor alternates between the two states `ready` and `not_ready`. In the former, it senses the temperature of the environment and *possibly* sends the temperature value to the processing unit. The purpose of the processing unit is receiving *two* temperature samples from the sensor, and sending the average value to the data transmission unit. While in state $\texttt{read}_i$, for $i = 0, 1, 2$, it has read i samples. While in $\texttt{reading}_j$, for $j = 1, 2$, it is attempting to read the j-th sample. A reading is possible only if the sensor and the processing unit are synchronized: a time interval (token) with value $\texttt{reading}_j$ has to *contain* (Allen relation, see Table 1.1) a token `ready`. Analogously, the processing unit can send data to the transmitter only if a token with value `send` contains one with value $\texttt{read}_2$. The sensor starts in state `not_ready`. This is specified by the trigger-less rule $\top \longrightarrow \exists o[x_{\texttt{temp}} = \texttt{not_ready}].o \leq^{\texttt{s}}_{[0,0]} 0$. The processing unit starts in state $\texttt{reading}_1$: $\top \longrightarrow \exists o[x_{\texttt{proc}} = \texttt{reading}_1].o \leq^{\texttt{s}}_{[0,0]} 0$. (Recall that trigger-less rules may also contain singular intervals at no extra computational cost.) The goal of the system is encoded by the rule $\top \longrightarrow \exists o_1[x_{\texttt{proc}} = \texttt{read}_2]\exists o_2[x_{\texttt{transm}} = \texttt{send}].(o_2 \leq^{\texttt{s,s}}_{[0,+\infty[} o_1 \wedge o_1 \leq^{\texttt{e,e}}_{[0,+\infty[} o_2)$.
Let us now encode the fact that the sensor and the processing unit must be synchronized for the latter to receive a temperature sample. We assume the next simple trigger rule to be *interpreted under the future semantics*.

$$o[x_{\texttt{proc}} = \texttt{reading}_1] \longrightarrow (\exists o_1[x_{\texttt{proc}} = \texttt{read}_0].o \leq^{\texttt{e,s}}_{[0,1]} o_1) \vee$$
$$(\exists o_2[x_{\texttt{proc}} = \texttt{read}_1]\exists o_3[x_{\texttt{temp}} = \texttt{ready}].o \leq^{\texttt{e,s}}_{[0,1]} o_2 \wedge o_3 \leq^{\texttt{e,e}}_{[0,+\infty[} o). \quad (8.1)$$

Let us observe that, due to the future semantics, the token (referenced by the name) o starts no later than o_3: this with the interval atom $o_3 \leq^{\texttt{e,e}}_{[0,+\infty[} o$ force the *contains* Allen relation between o and o_3. An analogous rule can be written for the second temperature sample (where $x_{\texttt{proc}} = \texttt{reading}_2$). In Figure 8.5 we show an example of plan/computation for the system described by $P = (\{x_{\texttt{temp}}, x_{\texttt{proc}}, x_{\texttt{transm}}\}, R)$.

Let us now specify some properties in MITL (more precisely, $\mathsf{MITL}_{(0,\infty)}$) to check on the system model. The idea is that such properties must hold true over *all possible computations (plans)* of the described system, in order for the MC problem to be satisfied.

- $\mathsf{G}_{<2}\ \neg\psi(\mathtt{s}, \mathtt{ready})$. This property holds true in any system computation, as the sensor does not ever get ready by 2 seconds;
- $\mathsf{F}_{\leq 8}\ \psi(\mathtt{s}, \mathtt{read}_1)$. This property is not true in all computations (but it is, e.g., in the one of Figure 8.5), because the sensor and the processing unit may synchronize for the first time after 8 seconds;
- $\mathsf{F}_{\geq 0}\big(\psi(\mathtt{s}, \mathtt{ready}) \wedge (\top\, \mathsf{U}_{>0}\ \psi(\mathtt{s}, \mathtt{ready}))\big)$. This property holds true in any system computation, since the system guarantees, after some time, to eventually send the data via the transmitter. In order for this to happen, the sensor must become ready (at least) twice.
- $\mathsf{G}_{\geq 0}\big(\psi(\mathtt{s}, \mathtt{read}_1) \rightarrow \mathsf{F}_{\leq 3}\ \psi(\mathtt{s}, \mathtt{read}_2)\big)$. This property is not true in all computations as the processing unit, after reading the first sample, may not be able to read the second one by 3 time units (e.g., when the transmitter and the processing unit do not synchronize as soon as possible).
- $\mathsf{G}_{\geq 0}\big(\psi(\mathtt{s}, \mathtt{reading}_1) \wedge (\mathit{EqTime}(\psi(\mathtt{s}, \mathtt{ready})) \vee \mathit{past}^{\mathtt{s}}_{\mathtt{ready}}) \rightarrow \mathsf{F}_{\leq 2}\ \psi(\mathtt{s}, \mathtt{read}_1)\big)$. We recall that the proposition letter $\mathit{past}^{\mathtt{s}}_{\mathtt{ready}}$ is true at the time it is interpreted if there is a past token for $x_{\mathtt{temp}}$ with value $\mathtt{ready}$ starting at the same timestamp. The formula considers a situation where a token with value $\mathtt{reading}_1$ starts together with a token $\mathtt{ready}$. The property expressed by the formula is not true in general, as either (i) the token $\mathtt{reading}_1$ may not contain the token $\mathtt{ready}$, hence $x_{\mathtt{proc}}$ will not move to the state $\mathtt{read}_1$ by 2 time units, or (ii) the token $\mathtt{reading}_1$ is followed by a token $\mathtt{read}_0$. As for the latter case, the system description rule (8.1) states that if there is a transition from state $\mathtt{reading}_1$ to $\mathtt{read}_1$, then the processing unit and the sensors must have synchronized. However, the converse implication need not hold: the two component may fail to communicate anyway (the processing unit remaining in $\mathtt{read}_0$).

Let us now formally define the MC problem for MITL formulas over timelines. As shown in the proof of Theorem 8.3.5, given a system model $P_{\mathrm{sys}} = (SV, R)$, it is possible to build a TA $\mathcal{A}_{\mathrm{sys}}$ that accepts all and only the encodings w_Π of multi-timelines Π of SV satisfying all the rules in R.

Definition 8.6.2 (Model checking)**.** Given a system model $P_{\mathrm{sys}} = (SV, R)$ and a MITL formula φ over $\mathcal{AP}$, the MC problem for MITL formulas over timelines is to decide whether or not $\mathcal{L}_T(\mathcal{A}_{\mathrm{sys}}) \subseteq \mathcal{L}_T(\varphi)$.

We recall that [AFH96] given a MITL (resp., $\mathsf{MITL}_{(0,\infty)}$) formula ψ, where N is the number of distinct subformulas of ψ, and K the largest integer constant appearing in ψ, we can build a TA $\mathcal{A}_\psi$ accepting the models of ψ, with $O(2^{N \cdot K})$ (resp., $O(2^N)$) states, $O(N \cdot K)$ (resp., $O(N)$) clocks, and maximum constant $O(K)$. Deciding its emptiness requires space *logarithmic* in the number of states of $\mathcal{A}_\psi$ and *polynomial* in the number of clocks and in the length of the encoding of K, hence exponential (resp., polynomial) space.

To decide if $\mathcal{L}_T(\mathcal{A}_{\text{sys}}) \subseteq \mathcal{L}_T(\varphi)$, we check whether $\mathcal{L}_T(\mathcal{A}_{\text{sys}}) \cap \mathcal{L}_T(\mathcal{A}_{\neg\varphi}) = \emptyset$ by making the intersection $\mathcal{A}_\wedge$ of $\mathcal{A}_{\text{sys}}$ and $\mathcal{A}_{\neg\varphi}$, and checking for emptiness of its timed language. The size of $\mathcal{A}_\wedge$ is polynomial in those of $\mathcal{A}_{\text{sys}}$ and $\mathcal{A}_{\neg\varphi}$. Moreover $\mathcal{A}_{\text{sys}}$, $\mathcal{A}_{\neg\varphi}$ and $\mathcal{A}_\wedge$ can be built on the fly, and the emptiness test can be done without explicitly constructing them as well. The next result follows by these observations and by Theorem 8.3.5.

Theorem 8.6.3. *The MC problem for MITL formulas over timelines, with simple future trigger rules and non-singular intervals, is in* ***EXPSPACE****.*
The MC problem for $\mathsf{MITL}_{(0,\infty)}$ *formulas over timelines, with simple future trigger rules and intervals in* $\mathit{Intv}_{(0,\infty)}$*, is in* ***PSPACE****.*

Clearly, **EXPSPACE**- and **PSPACE**-completeness of the above MC problems follow by the underlying future TP problems.

8.7 Conclusions

In this chapter we started by considering the timeline-based planning problem (TP) over dense temporal domains. Timelines have been fruitfully used in temporal planning for quite a long time to describe planning domains. Having recourse to dense time is important for expressiveness: in this way one can express interval-based properties of planning domains, can abstract from unnecessary (or even "forced") details often artificially added due to the necessity of discretizing time, and can suitably represent actions with duration, accomplishments and temporally extended goals. Since TP turns out to be undecidable in its general form, we identified and studied "intermediate" decidable cases of the problem, which enforce forms of synchronization rules having lower expressive power than that of general ones.

The final purpose of the chapter is however exploiting timelines as system descriptions/models, which are checked against properties specified in the logic MITL (which replaces HS, in that timed extensions of the latter are not available in literature). TP is a sort of "necessary condition" for timeline-based MC: TP boils down to a feasibility check of the system description; moreover MC can easily be solved once TP has, as both timelines and the property specification language MITL can be translated into timed automata [AD94], which have been studied for a long time and are at the basis of well-known model checkers (e.g., Uppaal [LPY97]).

Acknowledgment I would like to sincerely thank Gerhard Woeginger (RWTH Aachen University) for his fundamental contribution to the "enforcing legal token durations and timeline evolutions" phase of the **NP** planning algorithm for TP with trigger-less rules.

9 Conclusions

The main topic of this thesis is the MC problem for the interval temporal logic HS. The first results we have established regard HS under the homogeneous semantics, interpreted over finite Kripke structures: on one hand, we have shown the decidability of the problem in nonelementary time, and on the other, its **EXPSPACE**-hardness. Then, we have studied many HS fragments under the homogeneity assumption, proposing, for each of these, ad-hoc MC algorithms that rest on concepts and techniques different from one another (e.g., small model properties, trace contractions, Boolean circuits with oracles, finite state automata,...). The complexity of the MC problem for them ranges from **EXPSPACE** down to **PSPACE** and to some of the lowest levels of the polynomial time hierarchy. Still under homogeneity, we have studied the expressive power of three semantic variants of HS: we have compared such variants among themselves, and to the standard point-based logics LTL, CTL and CTL*.

Homogeneity readily enables us to interpret HS formulas over Kripke structures, which are inherently point-based models; however, it limits the expressive power of HS. The first way of relaxing such an assumption, while still having Kripke structures as system models, is by adding regular expressions into HS formulas, in place of vanilla proposition letters. This gives us a means of selecting some traces fulfilling specific propositional patterns, and avoiding all the others. By an automata-theoretic approach, we have shown that MC for full HS extended with regular expressions retains the same complexity upper and lower bounds as those of the homogeneous case. As for the analyzed HS fragments, some of them, when extended with regular expressions, feature an increased computational complexity, while others do not.

Finally, in the last chapter of the thesis we have replaced Kripke structures by timelines, a more expressive type of model with dense temporal domain, which can capture the interval-based behaviour and properties of systems. In order to finally deal with timeline-based MC, we have first considered the timeline-based planning problem, identifying suitable restrictions on it, in order to overcome the undecidability of its general formulation. As for the language for specifying properties of timelines in MC, we have adopted the logic MITL, which represents a sort of "compromise solution", because a timed extension of HS over dense domains is not available from the literature.

In this thesis we have completely worked out several problems; however, we now list some issues that remain still open from all previous chapters.

- The first noticeable one is the precise complexity of MC for full HS (and BE)—with or without regular expressions. We conjecture the problem to be *elementarily decidable* (and, maybe, **EXPSPACE**-complete). The machinery presented in Chapter 3 for D does not generalize to BE, and neither do the results of Section 4.2 on $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}\overline{\mathsf{E}}$/$\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{B}}\overline{\mathsf{E}}$.
- MC for $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}\overline{\mathsf{E}}$ and $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{B}}\overline{\mathsf{E}}$ under homogeneity is in $\mathbf{AEXP}_{\mathbf{pol}}$, but only known to be **PSPACE**-hard; hardness derives from that of $\overline{\mathsf{B}}$/$\overline{\mathsf{E}}$. We recall that, conversely, MC for $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}\overline{\mathsf{E}}$/$\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{B}}\overline{\mathsf{E}}$ with regular expressions is complete for $\mathbf{AEXP}_{\mathbf{pol}}$ (Section 7.3). It is not clear if removing regular expression from such fragments lowers the complexity from $\mathbf{AEXP}_{\mathbf{pol}}$ to other classes "above" **PSPACE** (or to **PSPACE** itself).
- MC for A, $\overline{\mathsf{A}}$, A$\overline{\mathsf{A}}$, $\overline{\mathsf{A}}$B, and AE under homogeneity has complexity in between $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$ and $\mathbf{P}^{\mathbf{NP}[O(\log^2 n)]}$. We do not know if the problem can be solved by less than $O(\log^2 n)$ queries to the **NP** oracle, or a tighter lower bound can be proved, or both (e.g., $\Theta(\log n \log\log n)$ queries may be needed).
- It is not known whether the future timeline-based planning problem with unrestricted trigger rules is decidable, but we conjecture that this is *not* the case. In Section 8.3 decidability has been shown only for *simple* trigger rules, as they allow for translation into MTL/MITL formulas.

Now we would like to conclude the thesis by outlining possible future research directions and themes.

- An interesting problem to study is MC for HS over *visibly pushdown systems* (VPS) that, given their infinite state (configuration) space, allow us to represent *infinite state systems*, hence enabling us to deal, for instance, with *recursive programs*. In this case, HS may be extended with a *binding* and an *unbinding* operator—the former for restricting the valuation of its argument to the sub-intervals of the current context-interval (each interval being a computation trace of the VPS), and the latter for removing the current binding. Such logic may then be compared, as for its expressiveness, to the

context-free linear-time LTL extensions CARET [AEM04], NWTL [Alu+07], and CARET with the *within* modality [Alu+07].

- The current semantic definition of HS—either with or without regular expressions—does not allow us to express a meaningful property of intervals, the *average*. Adding proposition letters in HS such as $p_{\geq 0.5}$, having the intuitive meaning of "p holds true in at least half of the states of the interval being considered", would be fairly natural and give the possibility of expressing a peculiar interval-based property.
- A *timed* extension of HS *over dense domains* would be naturally required by timelines, where time is a fundamental dimension. However, in the literature, only *metric* extensions of HS have been proposed over the natural numbers [Bre+13]. Defining such an extension, and perhaps linking it with known results regarding other timed logics and/or timed automata, is an interesting research theme.
- The *synthesis* problem for HS, a natural evolution of MC, useful, for example, for the development of digital sequential circuits, has already been studied in [MS14], where the authors consider $\overline{\mathsf{A}}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}$ and $\mathsf{A}\overline{\mathsf{A}\mathsf{B}}$ (with an equivalence relation). Synthesis for such fragments is either *decidable and non-primitive recursive-hard*, or *undecidable* (depending on the underlying linear orders). Thus, suitable restrictions should be identified on the fragments, on the semantics, and/or on the models, in order for the results to be useful for practical purposes.

Part V

Appendices

A

Proofs and complements of Chapter 3

Contents

A.1 Proof of Lemma 3.3.3 243
A.2 Proof of Lemma 3.3.5 245
A.3 Proof of Lemma 3.3.7 245
A.4 Proof of Theorem 3.4.3 246
A.5 PSPACE-Hardness of MC for D over finite Kripke structures . . . 246
A.6 PSPACE-Hardness of SAT for D over finite linear orders 253

A.1 Proof of Lemma 3.3.3

Lemma (3.3.3). *Let $\mathcal{G} = (\mathbb{P}_{\mathbb{S}}, \mathcal{L})$. $\mathcal{G}$ is a fulfilling homogeneous compass φ-structure if and only if, for every pair $x, y \in \mathcal{S}$, we have:*

- $\mathcal{L}(x, y-1)\mathcal{L}(x+1, y) \Rightarrow^{D_\varphi} \mathcal{L}(x, y)$ *if* $x < y$, *and*
- $\mathcal{R}eq_D(\mathcal{L}(x, y)) = \emptyset$ *if* $x = y$.

Proof. ($\Rightarrow$) Let us consider $x, y \in \mathcal{S}$. First we note that, since $\mathcal{G}$ is fulfilling, it must be $\mathcal{R}eq_D(\mathcal{L}(x, y)) = \emptyset$ whenever $x = y$. Otherwise, if $x < y$, we consider the labelings $\mathcal{L}(x, y-1)$ and $\mathcal{L}(x+1, y)$. By the homogeneity property of Definition 3.2.8, $\mathcal{L}(x, y) \cap \mathcal{AP} = \mathcal{L}(x, y-1) \cap \mathcal{L}(x+1, y) \cap \mathcal{AP}$: the first condition of Definition 3.3.1 holds. Moreover, since $\mathcal{G}$ is fulfilling, for every $\psi \in \mathcal{R}eq_D(\mathcal{L}(x, y))$ we have that either $\psi \in \mathcal{L}(x, y-1)$, or $\psi \in \mathcal{L}(x+1, y)$, or $\psi \in \mathcal{L}(x', y')$ for some

$x < x' \leq y' < y$. In the first two cases $\psi \in Obs_D(\mathcal{L}(x, y-1)) \cup Obs_D(\mathcal{L}(x+1, y))$. As for the last case, by Lemma 3.2.5, $Obs_D(\mathcal{L}(x', y')) \subseteq Req_D(\mathcal{L}(x, y-1))$ and $Obs_D(\mathcal{L}(x', y')) \subseteq Req_D(\mathcal{L}(x+1, y))$, hence $\psi \in Req_D(\mathcal{L}(x, y-1))$ and $\psi \in Req_D(\mathcal{L}(x+1, y))$. We can conclude that $Req_D(\mathcal{L}(x, y)) \subseteq Obs_D(\mathcal{L}(x, y-1)) \cup Obs_D(\mathcal{L}(x+1, y)) \cup Req_D(\mathcal{L}(x, y-1)) \cup Req_D(\mathcal{L}(x+1, y))$. The converse inclusion ($\supseteq$) follows by Lemma 3.2.5, hence the second condition of Definition 3.3.1 holds. We conclude that $\mathcal{L}(x, y-1)\mathcal{L}(x+1, y) =^{D_\varphi}\!\!\Rightarrow \mathcal{L}(x, y)$.

($\Leftarrow$) Let us consider $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$ such that, for every pair $x, y \in \mathcal{S}$, $x \leq y$, we have $\mathcal{L}(x, y-1)\mathcal{L}(x+1, y) =^{D_\varphi}\!\!\Rightarrow \mathcal{L}(x, y)$ if $x < y$, and $Req_D(\mathcal{L}(x, y)) = \emptyset$ if $x = y$. We have to prove that $\mathcal{G}$ is a homogeneous fulfilling compass φ-structure.

First, we prove consistency w.r.t. the relation D_φ. Let us show that, for all pairs of points (x, y) and (x', y') with $(x', y') \sqsubset (x, y)$, we have $\mathcal{L}(x, y)\ D_\varphi\ \mathcal{L}(x', y')$. The proof is by induction on $\Delta = (x' - x) + (y - y') \geq 1$. If $\Delta = 1$, either $(x', y') = (x+1, y)$ or $(x', y') = (x, y-1)$. Let us consider $(x', y') = (x+1, y)$ (the other case is symmetric). Since $\mathcal{L}(x, y-1)\mathcal{L}(x+1, y) =^{D_\varphi}\!\!\Rightarrow \mathcal{L}(x, y)$, we easily get that $\mathcal{L}(x, y)\ D_\varphi\ \mathcal{L}(x+1, y)$. If $\Delta \geq 2$, since $(x', y') \sqsubset (x, y)$, then $(x', y'+1) \sqsubset (x, y)$ or $(x'-1, y') \sqsubset (x, y)$. We only consider $(x'-1, y') \sqsubset (x, y)$, being the other case symmetric. By the inductive hypothesis, $\mathcal{L}(x, y)\ D_\varphi\ \mathcal{L}(x'-1, y')$. Since $\mathcal{L}(x'-1, y'-1)\mathcal{L}(x', y') =^{D_\varphi}\!\!\Rightarrow \mathcal{L}(x'-1, y')$, we have $\mathcal{L}(x'-1, y')\ D_\varphi\ \mathcal{L}(x', y')$. Let us observe that D_φ is a transitive relation, and thus $\mathcal{L}(x, y)\ D_\varphi\ \mathcal{L}(x', y')$.

We now show that $\mathcal{G}$ is fulfilling. We prove that for every point $(x, y) \in \mathbb{P}_\mathbb{S}$ and for every $\psi \in Req_D(\mathcal{L}(x, y))$, there exists $(x', y') \in \mathbb{P}_\mathbb{S}$, $(x', y') \sqsubset (x, y)$ such that $\psi \in \mathcal{L}(x', y')$. The proof is by induction on $y - x \geq 0$. If $x = y$, we have $Req_D(\mathcal{L}(x, y)) = \emptyset$, hence the thesis holds vacuously. If $y - x \geq 1$, since $\mathcal{L}(x, y-1)\mathcal{L}(x+1, y) =^{D_\varphi}\!\!\Rightarrow \mathcal{L}(x, y)$, we have $Req_D(\mathcal{L}(x, y)) = Req_D(\mathcal{L}(x, y-1)) \cup Req_D(\mathcal{L}(x+1, y)) \cup Obs_D(\mathcal{L}(x, y-1)) \cup Obs_D(\mathcal{L}(x+1, y))$. If $\psi \in Obs_D(\mathcal{L}(x, y-1)) \cup Obs_D(\mathcal{L}(x+1, y))$, the thesis is verified. If $\psi \in Req_D(\mathcal{L}(x+1, y))$ (the case $\psi \in Req_D(\mathcal{L}(x, y-1))$ is symmetric and thus omitted), by the inductive hypothesis, $\psi \in \mathcal{L}(x'', y'')$ for some $(x'', y'') \sqsubset (x+1, y) \sqsubset (x, y)$.

It remains to prove that $\mathcal{G}$ is homogeneous. We have to show that, for every $(x, y) \in \mathbb{P}_\mathbb{S}$ and every $p \in \mathcal{AP}$, $p \in \mathcal{L}(x, y)$ if and only if for every point (x', x'), with $x \leq x' \leq y$, we have $p \in \mathcal{L}(x', x')$. The proof is by induction on the length of the interval (x, y). If $x = y$ the property trivially holds. Let us consider now $y - x > 0$ (inductive step). By the inductive hypothesis, since $(x+1, y)$ and $(x, y-1)$ are shorter than (x, y), we have $p \in \mathcal{L}(x+1, y)$ (resp., $p \in \mathcal{L}(x, y-1)$) if and only if, for every (x', x') with $x+1 \leq x' \leq y$, (resp., $x \leq x' \leq y-1$), $p \in \mathcal{L}(x', x')$. Thus $p \in \mathcal{L}(x+1, y) \cap \mathcal{L}(x, y-1)$ if and only if for every (x', x') with $x \leq x' \leq y$, $p \in \mathcal{L}(x', x')$. Since $\mathcal{L}(x+1, y)\mathcal{L}(x, y-1) =^{D_\varphi}\!\!\Rightarrow \mathcal{L}(x, y)$, we have $\mathcal{L}(x, y) \cap \mathcal{AP} = \mathcal{L}(x+1, y) \cap \mathcal{L}(x, y-1) \cap \mathcal{AP}$. Therefore $p \in \mathcal{L}(x, y)$ if and only if for every (x', x'), with $x \leq x' \leq y$, we have $p \in \mathcal{L}(x', x')$. □

A.2 Proof of Lemma 3.3.5

Lemma (3.3.5). *Let $\mathcal{G} = (\mathbb{P}_{\mathbb{S}}, \mathcal{L})$ be a fulfilling homogeneous compass φ-structure. For every $y \in \mathcal{S}$, row_y is a φ-row.*

Proof. Let $row_y = \mathcal{L}(y,y)^{m_0}\mathcal{L}(y-m_0,y)^{m_1}\cdots\mathcal{L}(y-\sum_{0\le i<n} m_i, y)^{m_n}$ where, for every $0 \le j \le n$, $\mathcal{L}(y-\sum_{0\le i<j} m_j, y)^{m_j}$ is a maximal substring of identical atoms (note that any row_y can be represented w.l.o.g. in this way, for $m_i > 0$). Since $(y,y) \sqsubset \ldots \sqsubset (0,y)$, by Lemma 3.2.5, $\mathcal{R}eq_D(\mathcal{L}(y,y)) \subseteq \mathcal{R}eq_D(\mathcal{L}(y-m_0,y)) \subseteq \ldots \subseteq \mathcal{R}eq_D(\mathcal{L}(y-\sum_{0\le i<n} m_i, y))$. Moreover, by homogeneity, $(\mathcal{L}(y,y)\cap\mathcal{AP}) \supseteq (\mathcal{L}(y-m_0,y)\cap\mathcal{AP}) \supseteq \ldots \supseteq (\mathcal{L}(y-\sum_{0\le i<n} m_i, y)\cap\mathcal{AP})$. By maximality, $\mathcal{L}(y-\sum_{0\le i<j} m_i, y) \neq \mathcal{L}(y-\sum_{0\le i<j-1} m_i, y)$ for every $0 < j \le n$, and thus, since φ-atoms are uniquely determined by a pair $R \subseteq \mathrm{REQ}_\varphi$ and $P \subseteq \mathcal{AP}$ which are monotonically arranged, we can conclude that $\mathcal{L}(y-\sum_{0\le i<j} m_i, y) \neq \mathcal{L}(y-\sum_{0\le i<j'} m_i, y)$ for every $j' < j$. Now we prove that $m_j = 1$ if $\mathcal{L}(y-\sum_{0\le i<j} m_i, y)$ is irreflexive. By contradiction let us suppose that $m_j > 1$; then $\mathcal{L}(y-\sum_{0\le i<j} m_i, y) = \mathcal{L}(y-(\sum_{0\le i<j} m_i)-1, y)$. Since $\mathcal{L}(y-(\sum_{0\le i<j} m_i)-1, y)\ D_\varphi\ \mathcal{L}(y-\sum_{0\le i<j} m_i, y)$, then $\mathcal{L}(y-\sum_{0\le i<j} m_i, y)$ is reflexive (contradiction). Finally we have $\mathcal{R}eq_D(\mathcal{L}(y,y)) = \emptyset$, as $\mathcal{G}$ is fulfilling. □

A.3 Proof of Lemma 3.3.7

Lemma (3.3.7). *Let $\mathcal{G} = (\mathbb{P}_{\mathbb{S}}, \mathcal{L})$, with $\mathcal{R}eq_D(\mathcal{L}(x,x)) = \emptyset$ for all $(x,x) \in \mathbb{P}_{\mathbb{S}}$. $\mathcal{G}$ is a fulfilling homogeneous compass φ-structure if and only if, for each $0 \le y < |\mathcal{S}| - 1$, $row_y \,{=}^{row_\varphi}\!\!\Rightarrow\, row_{y+1}$.*

Proof. ($\Rightarrow$) By Lemma 3.3.5, the rows $row_0, \ldots, row_{|\mathcal{S}|-1}$ of $\mathcal{G}$ are φ-rows. By Lemma 3.3.3, for every $0 \le x \le y$, $\mathcal{L}(x,y)\mathcal{L}(x+1,y+1) {=}^{D_\varphi}\!\!\Rightarrow \mathcal{L}(x,y+1)$. Since $\mathcal{L}(x,y) = row_y(y-x)$, $\mathcal{L}(x+1,y+1) = row_{y+1}((y+1)-(x+1))$, and $\mathcal{L}(x,y+1) = row_{y+1}((y+1)-x)$, we can conclude that $row_y \,{=}^{row_\varphi}\!\!\Rightarrow\, row_{y+1}$.

($\Leftarrow$) Since for each $0 \le y < |\mathcal{S}| - 1$, $row_y \,{=}^{row_\varphi}\!\!\Rightarrow\, row_{y+1}$, then for all $0 \le i \le y$, $row_y(i)row_{y+1}(i) {=}^{D_\varphi}\!\!\Rightarrow row_{y+1}(i+1)$, namely $\mathcal{L}(y-i,y)\mathcal{L}(y+1-i,y+1) {=}^{D_\varphi}\!\!\Rightarrow \mathcal{L}(y-i,y+1)$. Let $x = y-i$ for $0 \le x \le y$. We get $\mathcal{L}(x,y)\mathcal{L}(x+1,y+1) {=}^{D_\varphi}\!\!\Rightarrow \mathcal{L}(x,y+1)$. By Lemma 3.3.3, $\mathcal{G}$ is a fulfilling homogeneous compass φ-structure. □

A.4 Proof of Theorem 3.4.3

Theorem (3.4.3). *Given a finite Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ and a* D *formula φ, there exists an initial trace ρ of $\mathcal{K}$ such that $\mathcal{K}, \rho \models \varphi$ if and only if there exists a path in $G_{\varphi\sim\mathcal{K}} = (\Gamma, \Xi)$ from some node $(s_0, [row]_\sim) \in \Gamma$ to some node $(s, [row']_\sim) \in \Gamma$ such that:*

1. *there exists $row_1 \in [row]_\sim$ with $|row_1| = 1$, and*
2. *there exists $row_2 \in [row']_\sim$ with $\varphi \in row_2(|row_2| - 1)$.*

Proof. Preliminarily we observe that, in (1.), if $|row_1| = 1$, then $\{row_1\} = [row]_\sim$; moreover, in (2.), if for $row_2 \in [row']_\sim$ we have $\varphi \in row_2[|row_2| - 1]$, then for any $row_2' \in [row']_\sim$ we have $\varphi \in row_2'[|row_2'| - 1]$.

($\Longrightarrow$) Let us consider an initial trace ρ such that $\mathcal{K}, \rho \models \varphi$. By Proposition 3.4.2, $\varphi \in \mathcal{L}(0, |\rho| - 1)$ in the fulfilling homogeneous compass φ-structure induced by ρ, $\mathcal{G}_{(\mathcal{K},\rho)} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$. By Lemma 3.3.5 and 3.3.7, $\mathcal{L}(0,0) =^{row_\varphi}\!\!\Rightarrow row_1 =^{row_\varphi}\!\!\Rightarrow \cdots =^{row_\varphi}\!\!\Rightarrow row_{|\rho|-1}$, and $\varphi \in row_{|\rho|-1}[|\rho| - 1]$. By definition of $(\varphi\sim\mathcal{K})$-graph, $(\rho(0), [\mathcal{L}(0,0)]_\sim) \xrightarrow{\Xi} (\rho(1), [row_1]_\sim) \xrightarrow{\Xi} \cdots \xrightarrow{\Xi} (\rho(|\rho| - 1), [row_{|\rho|-1}]_\sim)$ is a path in $\mathcal{G}_{(\mathcal{K},\rho)}$ (since $row_y[0] \cap \mathcal{AP} = \mu(\rho(y))$ for all $0 \le y < |\rho|$) satisfying (1.) and (2.).

($\Longleftarrow$) Let us assume there is a path $(s_0, [row_0]_\sim) \xrightarrow{\Xi} (s_1, [row_1]_\sim) \xrightarrow{\Xi} \cdots \xrightarrow{\Xi} (s_m, [row_m]_\sim)$ in the $(\varphi\sim\mathcal{K})$-graph $G_{\varphi\sim\mathcal{K}} = (\Gamma, \Xi)$, satisfying (1.) and (2.). Hence, by definition of $(\varphi\sim\mathcal{K})$-graph, $\rho = s_0 s_1 \cdots s_m$ is an (initial) trace, $[row_0]_\sim =^{row_\varphi}\!\!\Rightarrow \cdots =^{row_\varphi}\!\!\Rightarrow [row_m]_\sim$, and $\mu(s_y) = row_y[0] \cap \mathcal{AP}$ for all $0 \le y \le m$. Applying repeatedly Lemma 3.3.12 we get that there is a sequence $row_0' =^{row_\varphi}\!\!\Rightarrow \cdots =^{row_\varphi}\!\!\Rightarrow row_m'$ of φ-rows where $row_0' = row_0$, for every $0 \le j \le m$, $row_j' \in [row_j]_\sim$, and $\varphi \in row_m'[|row_m'| - 1]$. We observe that, by Definition 3.3.6, $|row_j'| = |row_{j-1}'| + 1$ for $1 \le j \le m$ and, since $|row_0'| = 1$, we have $|row_j'| = j + 1$. Let us now define $\mathcal{G} = (\mathbb{P}_\mathbb{S}, \mathcal{L})$ where $\mathcal{S} = \{0, \ldots, m\}$ and $\mathcal{L}(x, y) = row_y'[y - x]$ for every $0 \le x \le y \le m$. Note that $\mathcal{R}eq_D(\mathcal{L}(y, y)) = \emptyset$ for every $0 \le y \le m$ (by definition of φ-row). By Lemma 3.3.7, $\mathcal{G}$ is a fulfilling homogeneous compass φ-structure. Since $\mu(s_y) = row_y[0] \cap \mathcal{AP} (= \mathcal{L}(y, y) \cap \mathcal{AP})$ for all $0 \le y \le m$, then $\mathcal{G}$ is precisely the compass φ-structure induced by ρ. Finally, since $\varphi \in row_m'[m] = \mathcal{L}(0, m)$, by Proposition 3.4.2 we can conclude that $\mathcal{K}, \rho \models \varphi$. □

A.5 PSPACE-Hardness of MC for D over finite Kripke structures

In this section we prove the **PSPACE**-hardness of MC for D over finite Kripke structures by a reduction from the **PSPACE**-complete *problem of (non-)universality* of

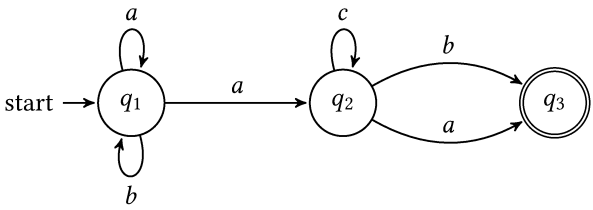


Figure A.1: An example of NFA, where q_1 is the initial state, and q_3 the only final state.

the language of a non-deterministic finite state automaton (NFA) [HK11]. We refer to Section 7.1 for notation and definitions on NFAs.

Let us consider an NFA $\mathcal{A} = (\Sigma, Q, q_1, \Delta, F)$, where, for convenience, $q_1 \in Q$ is the only initial state. The problem of deciding whether $\mathcal{L}(\mathcal{A}) \neq \emptyset$ can be solved by *logarithmic working space* by means of a non deterministic reachability from the initial state of $\mathcal{A}$ to an accepting state. On the other hand, deciding if $\mathcal{L}(\mathcal{A}) \neq \Sigma^*$ (namely, deciding if $\mathcal{L}(\mathcal{A})$ is *non-universal*, i.e., there is some word $w \in \Sigma^*$ such that $w \notin \mathcal{L}(\mathcal{A})$) is more difficult, and it is **PSPACE**-complete [HK11]. This is due to the fact that the shortest word *not accepted* by an NFA can have length exponential in the number of states of the NFA.

It is well-known that, by a subset construction, we can build from $\mathcal{A}$ a DFA $\mathcal{D} = (\Sigma, \tilde{Q}, \tilde{q_1}, \tilde{\delta}, \tilde{F})$ such that $\mathcal{L}(\mathcal{D}) = \mathcal{L}(\mathcal{A})$ and $\tilde{Q} = 2^Q$, $\tilde{q_1} = \{q_1\}$. We say that $\mathcal{D}$ is *equivalent* to $\mathcal{A}$. In order to decide if a word w is accepted by $\mathcal{A}$, it is then possible to build "on the fly" *the* computation of $\mathcal{D}$ over w from $\tilde{q_1}$ (hereafter we will omit "from $\tilde{q_1}$").

For example, let us consider the NFA $\mathcal{A}$ of Figure A.1. The computation of $\mathcal{D}$ (its equivalent DFA) over *aab* is:

$$(Q_1 = \{q_1\}) \xrightarrow{a} (Q_2 = \{q_1, q_2\}) \xrightarrow{a} (Q_3 = \{q_1, q_2, q_3\}) \xrightarrow{b} (Q_4 = \{q_1, q_3\}). \quad \text{(A.1)}$$

The word *aab* is accepted since there exists a final state of $\mathcal{A}$, $q_3 \in Q_4$ (Q_4 is the state of $\mathcal{D}$ reached by the computation). Note that Q_1 is the initial state of $\mathcal{D}$. The computation of $\mathcal{D}$ over *aac* is:

$$(Q_1 = \{q_1\}) \xrightarrow{a} (Q_2 = \{q_1, q_2\}) \xrightarrow{a} (Q_3 = \{q_1, q_2, q_3\}) \xrightarrow{c} (Q'_4 = \{q_2\}), \quad \text{(A.2)}$$

hence *aac* is *not* accepted since Q'_4 does not contain final states of $\mathcal{A}$.

Note that, as a rule, in the computation of $\mathcal{D}$ over w, with $|w| = n$,

$$Q_1 \xrightarrow{w(0)} Q_2 \xrightarrow{w(1)} \cdots \xrightarrow{w(n-2)} Q_n \xrightarrow{w(n-1)} Q_{n+1},$$

we have that the $\mathcal{A}$-state q belongs to Q_i, for $0 \leq i \leq n$, *if and only if* there exists some computation of $\mathcal{A}$ over w,

$$q_1 \overset{w(0)}{\rightarrow} q_2 \overset{w(1)}{\rightarrow} \cdots \overset{w(n-2)}{\rightarrow} q_n \overset{w(n-1)}{\rightarrow} q_{n+1},$$

where $q = q_i$. Moreover, if some $q \in Q_i$ then all $q' \in \Delta(q, w(i-1))$ must be in Q_{i+1}. Conversely, if some $q' \in Q_{i+1}$ then it has to exist some $q \in Q_i$ such that $q' \in \Delta(q, w(i-1))$.

We are ready to reduce the **PSPACE**-complete problem of (non-)universality of the language of an NFA to the MC problem for D over finite Kripke structures, proving that the latter is **PSPACE**-hard.

Given an NFA $\mathcal{A} = (\Sigma, Q, q_1, \Delta, F)$, we build the Kripke structure $\mathcal{K}_\mathcal{A} = (\mathcal{AP}, S, R, \mu, s_0)$, where:

- $S = \{q_i^\top, q_i^\perp, q_i'^\top, q_i'^\perp \mid i = 1, \ldots, |Q|\} \cup \{x_1, x_2, v_1, v_2, v_1', v_2', \widehat{q_1}^\top, \widehat{q_2}^\perp, \ldots, \widehat{q_{|Q|}}^\perp\} \cup \Sigma$;
- $s_0 = v_1'$;
- $\mathcal{AP} = \{q_i, q_i' \mid i = 1, \ldots, |Q|\} \cup \Sigma \cup \{e_1, e_2, f_1, f_2\}$;
- $\mu(q_i^\top) = \mu(q_i'^\top) = \mathcal{AP}$, $\mu(q_i^\perp) = \mathcal{AP} \setminus \{q_i\}$, $\mu(q_i'^\perp) = \mathcal{AP} \setminus \{q_i'\}$, for $1 \leq i \leq |Q|$;
  $\mu(a) = \mathcal{AP} \setminus (\Sigma \setminus \{a\})$ for $a \in \Sigma$;
  $\mu(x_1) = \mathcal{AP} \setminus \{e_1\}$, $\mu(x_2) = \mathcal{AP} \setminus \{e_2\}$, $\mu(v_1) = \mu(v_1') = \mathcal{AP} \setminus \{f_1\}$, $\mu(v_2) = \mu(v_2') = \mathcal{AP} \setminus \{f_2\}$;
  $\mu(\widehat{q_1}^\top) = \mathcal{AP}$, $\mu(\widehat{q_2}^\perp) = \mathcal{AP} \setminus \{q_2\}, \ldots, \mu(\widehat{q_{|Q|}}^\perp) = \mathcal{AP} \setminus \{q_{|Q|}\}$.

The edges R of $\mathcal{K}_\mathcal{A}$ can easily be deduced from Figure A.2, which is an example of Kripke structure built for an NFA with set of states $Q = \{q_1, q_2, q_3\}$, and alphabet $\Sigma = \{a, b, c\}$.

The idea is that the computation of $\mathcal{D}$ on a word, say aab, which we have already seen in (A.1), should be represented by the following initial trace of $\mathcal{K}_\mathcal{A}$:

$$\begin{aligned}
&v_1' v_2' \underbrace{(\widehat{q_1}^\top \widehat{q_2}^\perp \widehat{q_3}^\perp)}_{Q_1} a x_1 x_2 \underbrace{(q_1'^\top q_2'^\top q_3'^\perp)}_{Q_2'} \cdots \\
&v_1 v_2 \underbrace{(q_1^\top q_2^\top q_3^\perp)}_{Q_2} a x_1 x_2 \underbrace{(q_1'^\top q_2'^\top q_3'^\top)}_{Q_3'} \cdots \\
&v_1 v_2 \underbrace{(q_1^\top q_2^\top q_3^\top)}_{Q_3} b x_1 x_2 \underbrace{(q_1'^\top q_2'^\perp q_3'^\top)}_{Q_4'} \cdots \\
&\qquad\qquad v_1 v_2 \underbrace{(q_1^\top q_2^\perp q_3^\top)}_{Q_4}.
\end{aligned} \tag{A.3}$$

The states $v_1, v_1', v_2, v_2', x_1, x_2$ are there only for technical reasons (explained later). A triple of states $(q_1^* q_2^* q_3^*)$ denoted by Q_i, where * stands for $\top$ or $\perp$,

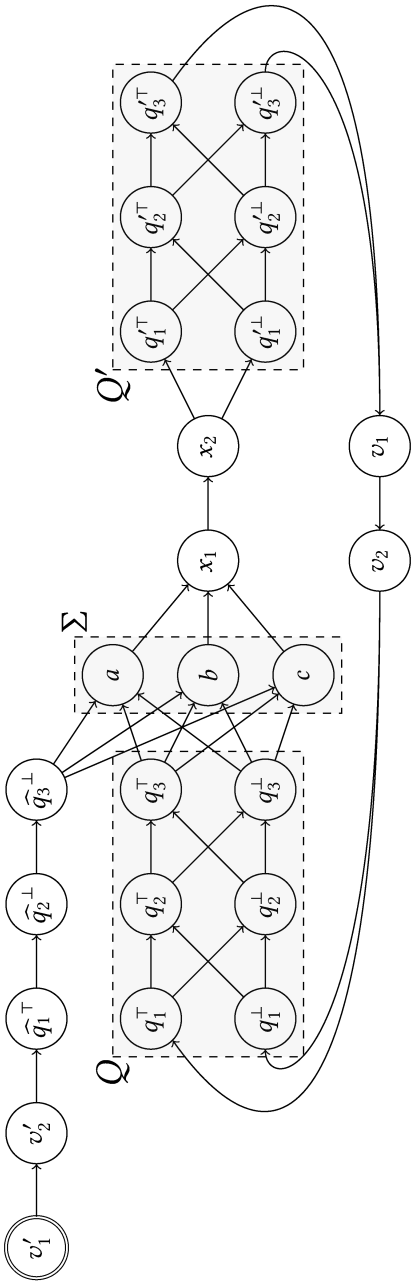


Figure A.2: The Kripke structure $\mathcal{K}_{\mathcal{A}}$ built for an NFA with set of states $Q = \{q_1, q_2, q_3\}$ and alphabet $\Sigma = \{a, b, c\}$

represents a state of $\mathcal{D}$, reached at the $(i-1)$-th step of the computation before reading $w(i-1)$: we have ${q_j}^\top$ if $q_j \in Q_i$, and ${q_j}^\bot$ if $q_j \notin Q_i$. Moreover the subtraces denoted by Q_i and Q'_i must be copies (i.e., ${q_j}^\top \in Q_i$ iff ${q'_j}^\top \in Q'_i$). In between Q_i and Q'_{i+1} in the trace we have $w(i-1) \in \Sigma$. The states ${\widehat{q_1}}^\top$, ${\widehat{q_2}}^\bot$ and ${\widehat{q_3}}^\bot$ of $\mathcal{K}_\mathcal{A}$ are just "copies" of ${q_1}^\top$, ${q_2}^\bot$ and ${q_3}^\bot$ respectively, added to ensure that the first state of the DFA $\mathcal{D}$ is $Q_1 = \{q_1\}$ (represented by ${\widehat{q_1}}^\top {\widehat{q_2}}^\bot {\widehat{q_3}}^\bot$). Finally note that there is an intuitive match between subtraces and proposition letters satisfied. For example,

$$\mathcal{K}_\mathcal{A}, v_1 v_2 ({q_1}^\top {q_2}^\top {q_3}^\top) b x_1 x_2 ({q'_1}^\top {q'_2}^\bot {q'_3}^\top) \models (q_1 \wedge q_2 \wedge q_3) \wedge (q'_1 \wedge \neg q'_2 \wedge q'_3) \wedge (\neg a \wedge b \wedge \neg c).$$

Let us now come to the formula $\Phi_\mathcal{A}$, built from $\mathcal{A}$. We assume the *strict* semantic variant of D. Preliminarily, we define the following formulas, which exploit the auxiliary states $v_1, v'_1, v_2, v'_2, x_1, x_2$ in order to "select" some suitable traces:

$$\varphi_{trans} = \neg f_1 \wedge \neg f_2 \wedge [\mathrm{D}](f_1 \wedge f_2) \wedge \langle \mathrm{D} \rangle \top,$$

$$\varphi_{copy} = \neg e_2 \wedge \neg e_1 \wedge [\mathrm{D}](e_1 \wedge e_2) \wedge \langle \mathrm{D} \rangle \top.$$

We can prove that:

- $\mathcal{K}_\mathcal{A}, \rho \models \varphi_{trans}$ if and only if $\rho = \tilde{v_2} \cdots v_1$ and v_1, v_2 do not occur as internal states of ρ (where $\tilde{v_2}$ can be either v_2 or v'_2);
- $\mathcal{K}_\mathcal{A}, \rho \models \varphi_{copy}$ if and only if $\rho = x_2 \cdots x_1$ and x_1, x_2 do not occur as internal states of ρ.

Moreover the following formulas have an intuitive meaning (in particular $length_{\geq 3}$ is satisfied by a trace ρ if and only if $|\rho| \geq 3$):

$$\varphi_{reject} = \bigwedge_{q_i \in F} \neg q_i,$$

$$length_{\geq 3} = \langle \mathrm{D} \rangle \top.$$

The formula $\Phi_\mathcal{A}$ is defined as follows (for the sake of brevity, for $q_i, q_j \in Q$ and $c \in \Sigma$, we denote $q_j \in \Delta(q_i, c)$ as $(q_i, c, q_j) \in \Delta$).

$$\Phi_\mathcal{A} = \underbrace{[\mathrm{D}] \left(\varphi_{trans} \rightarrow \left(\Big(\bigwedge_{(q_i,a,q'_j)\in\Delta} ((q_i \wedge a) \rightarrow q'_j) \Big) \wedge \Big(\bigwedge_{q'_i \in Q} (q'_i \rightarrow \bigvee_{(q_j,a,q'_i)\in\Delta} (q_j \wedge a)) \Big) \right) \right)}_{(1)} \wedge$$

$$\underbrace{[\mathrm{D}] \left(\varphi_{copy} \rightarrow \bigwedge_{q_i \in Q} (q_i \leftrightarrow q'_i) \right)}_{(2)} \wedge \underbrace{\Big((e_1 \wedge length_{\geq 3} \wedge \varphi_{reject}) \vee \langle \mathrm{D} \rangle \left(\varphi_{copy} \wedge \varphi_{reject} \right) \Big)}_{(3)}$$

Let us now prove the following lemma.

Lemma A.5.1. *$\mathcal{L}(\mathcal{A}) \neq \Sigma^*$ if and only if there exists an initial trace ρ of $\mathcal{K}_\mathcal{A}$ such that $\mathcal{K}_\mathcal{A}, \rho \models \Phi_\mathcal{A}$.*

Proof. ($\Rightarrow$) If $\mathcal{L}(\mathcal{A}) \neq \Sigma^*$, then there is $w \notin \mathcal{L}(\mathcal{A})$. Therefore the computation of $\mathcal{D}$ over w is *not* accepting. Let us consider the initial trace ρ of $\mathcal{K}_\mathcal{A}$ encoding such a computation as explained before, see (A.3). We distinguish two cases:

- $w = \varepsilon$: then we consider $\rho = v_1' v_2' \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\widehat{q_{|Q|}}^{\perp}}$. No strict subtrace satisfies φ_{trans} or φ_{copy}, hence conjuncts (1) and (2) are trivially satisfied. Moreover, since $\varepsilon \notin \mathcal{L}(\mathcal{A})$, $q_1 \notin F$, thus ρ models also $e_1 \wedge \mathit{length}_{\geq 3} \wedge \varphi_{reject}$.
- $w \neq \varepsilon$; then we consider the initial trace ρ of $\mathcal{K}_\mathcal{A}$ encoding the computation over w, w.l.o.g. extended with some $c \in \Sigma$ (any symbol is fine), and finally $x_1 x_2$: its generic form is $\rho = v_1' v_2' \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\widehat{q_{|Q|}}^{\perp}} w(0)(x_1 x_2 {q_1'}^* {q_2'}^* \cdots {q'_{|Q|}}^* \cdot v_1 v_2 {q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c)^+ x_1 x_2$, where * is $^{\perp}$ or $^{\top}$, and $^+$ denotes a positive number of occurrences of the string in brackets. Every strict subtrace satisfying φ_{trans} models the right part of the implication in conjunct (1), which enforces the consistency conditions of a computation. Every strict subtrace satisfying φ_{copy} features $q_i'^{\top}$ if it features $q_i{}^{\top}$, and $q_i'^{\perp}$ if it features $q_i{}^{\perp}$, hence it satisfies $\bigwedge_{q_i \in Q}(q_i \leftrightarrow q_i')$. Finally, the last part of ρ, $x_2 {q_1'}^* {q_2'}^* \cdots {q'_{|Q|}}^* v_1 v_2 {q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c x_1$, models φ_{copy}, and, since w is *not* accepted, it also fulfills φ_{reject}.

Therefore, in both cases, there exists an initial trace ρ such that $\mathcal{K}_\mathcal{A}, \rho \models \Phi_\mathcal{A}$.

($\Leftarrow$) Let us assume there exists an initial trace ρ of $\mathcal{K}_\mathcal{A}$ such that $\mathcal{K}_\mathcal{A}, \rho \models \Phi_\mathcal{A}$. We distinguish some cases, according to the structure of ρ.

1. $\rho = v_1'(v_2')^?$ ($^?$ denotes 0 or 1 occurrences of the string in brackets).
   This trace does not model (3), thus it cannot be the trace we are looking for.
2. $\rho = v_1' v_2' \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{q_j}^{\perp}$ for $j \geq 1$, or $\rho = v_1' v_2' \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\widehat{q_{|Q|}}^{\perp}} c$ for some $c \in \Sigma$.
   No subtrace satisfies φ_{copy}, thus, by the conjunct (3), ρ models φ_{reject}. Hence $q_1 \notin F$, and ε is rejected by $\mathcal{A}$.
3. $\rho = v_1' v_2' \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\widehat{q_{|Q|}}^{\perp}} c x_1 (x_2)^?$
   This trace does not model the conjunct (3).
4. $\rho = v_1' v_2' \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\widehat{q_{|Q|}}^{\perp}} c x_1 x_2 {q_1'}^* {q_2'}^* \cdots {q_j'}^*$ for $j \geq 1$
   This trace does not model the conjunct (3).
5. $\rho = v_1' v_2' \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\widehat{q_{|Q|}}^{\perp}} c x_1 x_2 {q_1'}^* {q_2'}^* \cdots {q'_{|Q|}}^* v_1 (v_2)^?$
   This trace does not model the conjunct (3).
6. $\rho = v_1' v_2' \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\widehat{q_{|Q|}}^{\perp}} c x_1 x_2 {q_1'}^* {q_2'}^* \cdots {q'_{|Q|}}^* v_1 v_2 {q_1}^* {q_2}^* \cdots {q_j}^*$
   This trace does not model the conjunct (3).
7. $\rho = v_1' v_2' \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\widehat{q_{|Q|}}^{\perp}} c x_1 x_2 {q_1'}^* {q_2'}^* \cdots {q'_{|Q|}}^* v_1 v_2 {q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c$
   This trace does not model the conjunct (3).

8. $\rho = v'_1 v'_2 \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\overline{q_{|Q|}}}^{\perp} c x_1 x_2 {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1 v_2 {q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c x_1$
This trace does not model the conjunct (3).
9. $\rho = v'_1 v'_2 \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\overline{q_{|Q|}}}^{\perp} c (x_1 x_2 {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1 v_2 {q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c)^+ x_1 x_2$
We have that, since ρ models the conjunct (1), *all* adjacent pairs of occurrences of ${q_1}^* {q_2}^* \cdots {q_{|Q|}}^* \rightsquigarrow {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^*$ consistently model a transition of $\mathcal{D}$; moreover *all* adjacent pairs of occurrences of ${q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* \rightsquigarrow {q_1}^* {q_2}^* \cdots {q_{|Q|}}^*$ are "copies". Put all together, a legal computation of $\mathcal{D}$ over some string w is encoded. Finally, by the conjunct (3), a strict subtrace $x_2 {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1 v_2 {q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c x_1$ models φ_{reject}. Thus either w (if such subtrace is the last one) or one of its prefixes (if it is not the last one) is rejected by $\mathcal{D}$.
10. $\rho = v'_1 v'_2 \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\overline{q_{|Q|}}}^{\perp} c (x_1 x_2 {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1 v_2 {q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c)^+ x_1 x_2 \cdot \underline{{q'_1}^* {q'_2}^* \cdots {q'_j}^*}$
In this case and in the following ones, we underline the final part of ρ which may be "garbage", namely, it may encode an illegal suffix of a computation, just because it is not forced to "behave correctly" by $\Phi_{\mathcal{A}}$. However, since conjunct (3) is satisfied, a strict subtrace $x_2 {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1 v_2 {q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c x_1$ models φ_{reject} (and this is not part of the garbage). Thus, as before, some word w or one of its prefixes is rejected by $\mathcal{D}$.
11. $\rho = v'_1 v'_2 \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\overline{q_{|Q|}}}^{\perp} c (x_1 x_2 {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1 v_2 {q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c)^+ x_1 x_2 \cdot \underline{{q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1}$
Like the previous case.
12. $\rho = v'_1 v'_2 \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\overline{q_{|Q|}}}^{\perp} c (x_1 x_2 {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1 v_2 {q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c)^+ x_1 x_2 \cdot {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1 v_2$
Like case 9., but a prefix of w is necessarily rejected, such that ρ encodes the computation of $\mathcal{D}$ over w.
13. $\rho = v'_1 v'_2 \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\overline{q_{|Q|}}}^{\perp} c (x_1 x_2 {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1 v_2 {q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c)^+ x_1 x_2 \cdot {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1 v_2 \underline{{q_1}^* {q_2}^* \cdots {q_j}^*}$,
$\rho = v'_1 v'_2 \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\overline{q_{|Q|}}}^{\perp} c (x_1 x_2 {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1 v_2 {q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c)^+ x_1 x_2 \cdot {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1 v_2 \underline{{q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c}$,
$\rho = v'_1 v'_2 \widehat{q_1}^{\top} \widehat{q_2}^{\perp} \cdots \widehat{\overline{q_{|Q|}}}^{\perp} c (x_1 x_2 {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1 v_2 {q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c)^+ x_1 x_2 \cdot {q'_1}^* {q'_2}^* \cdots {q'_{|Q|}}^* v_1 v_2 \underline{{q_1}^* {q_2}^* \cdots {q_{|Q|}}^* c x_1}$.
All like case 12., just with the addition of final garbage, which is not considered, since it is not part of a strict subtrace satisfying φ_{copy}.

Thus, in all possible (legal) cases, some string is rejected by $\mathcal{D}$ (and by $\mathcal{A}$). □

It follows that $\mathcal{L}(\mathcal{A}) = \Sigma^*$ if and only if $\mathcal{K}_{\mathcal{A}} \models \neg\Phi_{\mathcal{A}}$. Since also the *problem of universality* of the language of an NFA is **PSPACE**-complete (because **PSPACE** is

closed under complement), and both $\mathcal{K}_{\mathcal{A}}$ and $\Phi_{\mathcal{A}}$ can be generated in polynomial time, we have proved the following.

Theorem A.5.2. *The MC problem for* D *formulas over finite Kripke structures is* ***PSPACE****-hard (under polynomial-time reductions).*

By slightly modifying $\Phi_{\mathcal{A}}$, we can adapt the proof to the *proper* variant of D.

A.6 PSPACE-Hardness of SAT for D over finite linear orders

In this section we outline a **PSPACE**-hardness proof for the SAT problem for D formulas over finite linear orders.

The construction mimics that of Sections 3.2 and 3.3 of [MM14], in which the authors show that it is possible to build a formula Ψ of D which encodes accepting computations of an NFA. More precisely the set of letters of Ψ equals the union of the alphabet of the NFA and the set of its states (plus some auxiliary letters, to enforce the "orientation" in the linear order, something that D is unaware of), and Ψ is satisfied by all and only the models such that the point-intervals are labeled with an accepting computation of the NFA over the word written in its point-intervals.

The idea is then to exploit Ψ to encode the Kripke structure of the previous section, thus getting a reduction from the problem of non-universality of the language of an NFA to the SAT problem for D. As a matter of fact, a Kripke structure can be regarded as a trivial NFA over a unary alphabet, say $\{a\}$, such that all the states are final, as we are interested only in the structure of traces (i.e., any word/trace is accepted under the only constraint that it exists in the structure).

By an easy adaptation of the results of Sections 3.2 and 3.3 of [MM14] we get the following.

Proposition A.6.1. *Given a Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ devoid of self-loops, there exists a* D *formula $\Psi_{\mathcal{K}}$ whose set of proposition letters is $\mathcal{AP} \cup S \cup Aux$—being Aux a set of auxiliary letters—such that any finite linear order satisfying $\Psi_{\mathcal{K}}$ represents an initial trace of $\mathcal{K}$. Moreover $\Psi_{\mathcal{K}}$ is polynomial in the size of $\mathcal{K}$.*

Every linear order satisfying $\Psi_{\mathcal{K}}$ features states of $\mathcal{K}$ labeling point-intervals (exactly one state for each point). Moreover we can easily force, for each occurrence of some state s of $\mathcal{K}$ along the order, the set of letters $\mu(s)$ to hold on the same position (point). The structure $\mathcal{K}$ in Proposition A.6.1 must not feature self-loops for a technical reason: by fulfilling this requirement, there is no way for a state of

$\mathcal{K}$ to "span" (by homogeneity) more than one point in a linear order satisfying $\Psi_{\mathcal{K}}$. We observe that in [MM14] the authors do not assume homogeneity; however homogeneity does not cause problems in our construction, as, intuitively, all the significant properties stated by $\Psi_{\mathcal{K}}$ are related to point-intervals.

Let us observe that the Kripke structure of the previous section does not contain self-loops. By Lemma A.5.1, the language of an NFA $\mathcal{A}$ is non-universal if and only if there exists an initial trace ρ such that $\mathcal{K}_{\mathcal{A}}, \rho \models \Phi_{\mathcal{A}}$ (the Kripke structure and formula built from $\mathcal{A}$ in the previous section), if and only if (by Proposition A.6.1 applied to $\mathcal{K}_{\mathcal{A}}$) the formula $\Psi_{\mathcal{K}_{\mathcal{A}}} \wedge \Phi_{\mathcal{A}}$ is satisfiable. We have proved the following result.

Theorem A.6.2. *The SAT problem for* D *formulas over finite linear orders is* ***PSPACE****-hard.*

B Proofs and complements of Chapter 4

Contents

B.1 Proof of Lemma 4.1.7 255
B.2 Proof of Lemma 4.1.8 256
B.3 PSPACE-hardness of MC for $\overline{\mathsf{B}}$ and $\overline{\mathsf{E}}$ 257
B.4 Proof of Lemma 4.2.10 260

B.1 Proof of Lemma 4.1.7

Lemma (4.1.7). *Let ψ be an $\mathsf{A}\overline{\mathsf{A}}\mathsf{E}\overline{\mathsf{E}}$ formula, $\mathcal{K}$ be a finite Kripke structure, and $\sigma \in \mathrm{Trc}_{\mathcal{K}}$. Then, $\mathtt{Check}(\mathcal{K}, \psi, \sigma) = 1$ if and only if $\mathcal{K}, \sigma \models \psi$.*

Proof. The proof is by induction on the structure of ψ. The base case where $\psi = p$, for some $p \in \mathcal{AP}$, directly follows from the definition (line 2 of Algorithm 4.2). The cases in which $\psi = \neg\varphi$ and $\psi = \varphi_1 \wedge \varphi_2$ are also trivial and thus omitted. We focus on the remaining cases.

- $\psi = \langle \mathrm{A} \rangle \varphi$. If $\mathcal{K}, \sigma \models \psi$, then there exists a trace $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{lst}(\sigma) = \mathrm{fst}(\rho)$ and $\mathcal{K}, \rho \models \varphi$. By Theorem 4.1.5, there exists a trace $\pi \in \mathrm{Trc}_{\mathcal{K}}$, with $|\pi| \leq |S| \cdot (|\varphi'| + 1)^2$ and $\mathrm{fst}(\pi) = \mathrm{fst}(\rho) (= \mathrm{lst}(\sigma))$, such that $\mathcal{K}, \pi \models \varphi'$, where φ' is the NNF of φ. Thus, being $|\pi| \leq |S| \cdot (2|\varphi| + 1)^2$, such trace π is considered in the for-loop at line 12. By the inductive hypothesis, $\mathtt{Check}(\mathcal{K}, \varphi, \pi) = 1$ and thus $\mathtt{Check}(\mathcal{K}, \psi, \sigma) = 1$.

Vice versa, if $\texttt{Check}(\mathcal{K}, \psi, \sigma) = 1$, then there exists a trace $\pi \in \text{Trc}_{\mathcal{K}}$, with $\text{lst}(\sigma) = \text{fst}(\pi)$, such that $\texttt{Check}(\mathcal{K}, \varphi, \pi) = 1$. By the inductive hypothesis, $\mathcal{K}, \pi \models \varphi$, hence $\mathcal{K}, \sigma \models \psi$.

- $\psi = \langle \overline{\text{A}} \rangle \varphi$ is analogous to the previous case.
- $\psi = \langle \text{E} \rangle \varphi$. If $\mathcal{K}, \sigma \models \psi$, there exists a trace $\pi \in \text{Suff}(\sigma)$ such that $\mathcal{K}, \pi \models \varphi$. By the inductive hypothesis, $\texttt{Check}(\mathcal{K}, \varphi, \pi) = 1$. Since all the proper suffixes of σ are checked (line 17), $\texttt{Check}(\mathcal{K}, \psi, \sigma) = 1$.
  Vice versa, if $\texttt{Check}(\mathcal{K}, \psi, \sigma) = 1$, then for some $\pi \in \text{Suff}(\sigma)$, it holds that $\texttt{Check}(\mathcal{K}, \varphi, \pi) = 1$. By the inductive hypothesis $\mathcal{K}, \pi \models \varphi$ implying that $\mathcal{K}, \sigma \models \psi$.
- $\psi = \langle \overline{\text{E}} \rangle \varphi$. If $\mathcal{K}, \sigma \models \psi$, then there exists a trace $\rho \in \text{Trc}_{\mathcal{K}}$, with $|\rho| \geq 2$, such that $\mathcal{K}, \rho \star \sigma \models \varphi$. By Theorem 4.1.5, there exists a trace $\pi \in \text{Trc}_{\mathcal{K}}$ induced by ρ, with $|\pi| \leq |S| \cdot (|\varphi'| + 1)^2$, such that $\mathcal{K}, \pi \star \sigma \models \varphi'$, where φ' is the NNF of φ. Such trace π is considered in the for-loop at line 22, since $|\pi| \leq |S| \cdot (2|\varphi| + 1)^2$ and $|\pi| \geq 2$ as it is induced by ρ. By the inductive hypothesis, $\texttt{Check}(\mathcal{K}, \varphi, \pi \star \sigma) = 1$ implying that $\texttt{Check}(\mathcal{K}, \psi, \sigma) = 1$.
  Vice versa, if $\texttt{Check}(\mathcal{K}, \psi, \sigma) = 1$, then there exists a trace $\pi \in \text{Trc}_{\mathcal{K}}$, with $|\pi| \geq 2$, such that $\texttt{Check}(\mathcal{K}, \varphi, \pi \star \sigma) = 1$. By the inductive hypothesis, $\mathcal{K}, \pi \star \sigma \models \varphi$, hence $\mathcal{K}, \sigma \models \psi$. □

B.2 Proof of Lemma 4.1.8

Theorem (4.1.8). *Let ψ be an $\text{A}\overline{\text{A}}\text{E}\overline{\text{E}}$ formula and $\mathcal{K}$ be a finite Kripke structure. Then, $\texttt{ModCheck}(\mathcal{K}, \psi) = 1$ if and only if $\mathcal{K} \models \psi$.*

Proof. ($\Leftarrow$) If $\mathcal{K} \models \psi$ then, for all initial traces $\rho \in \text{Trc}_{\mathcal{K}}$, we have that $\mathcal{K}, \rho \models \psi$. By Lemma 4.1.7, it follows that $\texttt{Check}(\mathcal{K}, \psi, \rho) = 1$. Now, since the for-loop at line 1 considers a subset of the initial traces, it holds that $\texttt{ModCheck}(\mathcal{K}, \psi) = 1$.

($\Rightarrow$) If $\texttt{ModCheck}(\mathcal{K}, \psi) = 1$, then, for any initial trace ρ considered by the for-loop at line 1, that is, with $|\rho| \leq |S| \cdot (2|\psi| + 3)^2$, it holds $\texttt{Check}(\mathcal{K}, \psi, \rho) = 1$. Let us assume by contradiction that $\mathcal{K} \not\models \psi$, that is, there exists an initial trace $\rho' \in \text{Trc}_{\mathcal{K}}$ such that $\mathcal{K}, \rho' \models \neg\psi$, or, equivalently, $\mathcal{K}, \rho' \models \overline{\psi}$, where $\overline{\psi}$ is the NNF of $\neg\psi$. Thus, by Theorem 4.1.5, there exists an initial trace σ, with $|\sigma| \leq |S| \cdot (|\overline{\psi}| + 1)^2 \leq |S| \cdot (2|\psi| + 3)^2$, such that $\mathcal{K}, \sigma \models \overline{\psi}$, namely, $\mathcal{K}, \sigma \not\models \psi$. By Lemma 4.1.7, it holds that $\texttt{Check}(\mathcal{K}, \psi, \sigma) = 0$, leading to a contradiction and proving that $\mathcal{K} \models \psi$. □

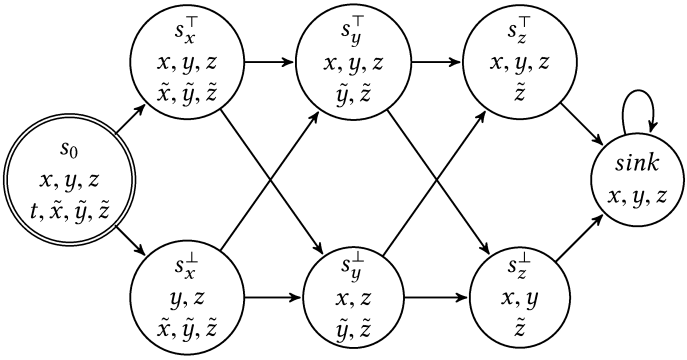


Figure B.1: The finite Kripke structure $\mathcal{K}_{QBF}^{x,y,z}$.

B.3 PSPACE-hardness of MC for $\overline{\mathsf{B}}$ and $\overline{\mathsf{E}}$

In this section, we prove that the MC problem for formulas of $\overline{\mathsf{B}}$ and of $\overline{\mathsf{E}}$, over finite Kripke structures, is **PSPACE**-hard by means of a reduction from the QBF problem, that is, the problem of determining the truth of a *fully-quantified* Boolean formula in *prenex normal form*, which is known to be **PSPACE**-complete (see, for example, [Sip12]). The proof for $\overline{\mathsf{B}}$ can easily be modified to show the **PSPACE**-hardness of the symmetric fragment $\overline{\mathsf{E}}$.

Let $\psi = Q_n x_n Q_{n-1} x_{n-1} \cdots Q_1 x_1 \phi(x_n, x_{n-1}, \ldots, x_1)$ be a quantified Boolean formula where, for $i = 1, \ldots, n$, $Q_i \in \{\exists, \forall\}$ and $\phi(x_n, x_{n-1}, \ldots, x_1)$ is a quantifier-free Boolean formula over the set of variables $Var = \{x_n, \ldots, x_1\}$. We define a Kripke structure $\mathcal{K}_{QBF}^{Var}$, whose initial traces represent all the possible assignments to the variables of Var. For each variable $x \in Var$, $\mathcal{K}_{QBF}^{Var}$ features a pair of states $s_x^\top$ and $s_x^\bot$, that represent a $\top$ and $\bot$ truth assignment to x, respectively. An example of $\mathcal{K}_{QBF}^{Var}$, with $Var = \{x, y, z\}$, is given in Figure B.1.

Formally, let $\mathcal{K}_{QBF}^{Var} = (\mathcal{AP}, S, R, \mu, s_0)$, where:

- $\mathcal{AP} = Var \cup \{t\} \cup \{\tilde{x}_i \mid 1 \leq i \leq n\}$;
- $S = \{s_{x_i}^\ell \mid 1 \leq i \leq n,\ \ell \in \{\bot, \top\}\} \cup \{s_0, sink\}$;
- if $n = 0$, $R = \{(s_0, sink), (sink, sink)\}$;
  if $n > 0$, $R = \{(s_0, s_{x_n}^\top), (s_0, s_{x_n}^\bot)\} \cup \{(s_{x_i}^\ell, s_{x_{i-1}}^m) \mid \ell, m \in \{\bot, \top\},\ 2 \leq i \leq n\} \cup \{(s_{x_1}^\top, sink), (s_{x_1}^\bot, sink), (sink, sink)\}$.
- $\mu(s_0) = Var \cup \{t\} \cup \{\tilde{x}_i \mid x_i \in Var\}$;
  for all $1 \leq i \leq n$, $\mu(s_{x_i}^\top) = Var \cup \{\tilde{x}_j \mid 1 \leq j \leq i\}$ and $\mu(s_{x_i}^\bot) = (Var \setminus \{x_i\}) \cup \{\tilde{x}_j \mid 1 \leq j \leq i\}$;
  $\mu(sink) = Var$.

The QBF formula ψ is reduced to the $\overline{\mathrm{B}}$ formula $\xi = t \rightarrow \xi_n$, where

$$\xi_i = \begin{cases} \phi(x_n, x_{n-1}, \ldots, x_1) & \text{if } i = 0 \\ \langle\overline{\mathrm{B}}\rangle\left(\tilde{x_i} \wedge \xi_{i-1}\right) & \text{if } i > 0 \text{ and } Q_i = \exists \\ [\overline{\mathrm{B}}]\left(\tilde{x_i} \rightarrow \xi_{i-1}\right) & \text{if } i > 0 \text{ and } Q_i = \forall \end{cases}.$$

Note that both $\mathcal{K}_{QBF}^{Var}$ and ξ can be built in polynomial time. In Theorem B.3.2, we shall show the correctness of the reduction, i.e., that ψ is true iff $\mathcal{K}_{QBF}^{Var} \models \xi$. As a preliminary step, we introduce some technical definitions and an auxiliary lemma.

Given a Kripke structure $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ and a $\overline{\mathrm{B}}$ formula χ, we denote by $p\ell(\chi)$ the set of proposition letters occurring in χ and by $\mathcal{K}_{|p\ell(\chi)}$ the structure obtained from $\mathcal{K}$ by restricting the labelling of each state to $p\ell(\chi)$, namely, the Kripke structure $(\overline{\mathcal{AP}}, S, R, \overline{\mu}, s_0)$, where $\overline{\mathcal{AP}} = \mathcal{AP} \cap p\ell(\chi)$ and $\overline{\mu}(s) = \mu(s) \cap p\ell(\chi)$, for all $s \in S$.

Moreover, for $v \in S$, we denote by $reach(\mathcal{K}, v)$ the subgraph of $\mathcal{K}$ induced by the states reachable from v, namely, the Kripke structure $(\mathcal{AP}, S', R', \mu', v)$, where $S' = \{s \in S \mid \text{there exists } \rho \in \mathrm{Trc}_{\mathcal{K}} \text{ with } \mathrm{fst}(\rho) = v \text{ and } \mathrm{lst}(\rho) = s\}$, $R' = R \cap (S' \times S')$, and $\mu'(s) = \mu(s)$, for all $s \in S'$.

As usual, we say that two Kripke structures $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ and $\mathcal{K}' = (\mathcal{AP}', S', R', \mu', s_0')$ are *isomorphic* (written $\mathcal{K} \sim \mathcal{K}'$) if and only if there is a *bijection* $f : S \rightarrow S'$ such that

- $f(s_0) = s_0'$;
- for all $u, v \in S$, $(u, v) \in R \iff (f(u), f(v)) \in R'$;
- for all $v \in S$, $\mu(v) = \mu'(f(v))$.

Finally, if $\mathcal{A}_{\mathcal{K}} = (\mathcal{AP}, \mathbb{I}, A_{\mathbb{I}}, B_{\mathbb{I}}, E_{\mathbb{I}}, \sigma)$ is the abstract interval model induced by a Kripke structure $\mathcal{K}$ and $\rho \in \mathrm{Trc}_{\mathcal{K}}$, we denote $\sigma(\rho)$ by $\mathcal{L}(\mathcal{K}, \rho)$.

Let $\mathcal{K}$ and $\mathcal{K}'$ be two Kripke structures. The following lemma, which is an immediate consequence of Lemma 1 of [MMP15a], states that, for any $\overline{\mathrm{B}}$ formula ψ, if the same set of proposition letters, restricted to $p\ell(\psi)$, holds over two traces $\rho \in \mathrm{Trc}_{\mathcal{K}}$ and $\rho' \in \mathrm{Trc}_{\mathcal{K}'}$, and the subgraphs consisting of the states reachable from, respectively, $\mathrm{lst}(\rho)$ and $\mathrm{lst}(\rho')$ are isomorphic, then ρ and ρ' are equivalent with respect to ψ.

Lemma B.3.1. *Given a $\overline{\mathrm{B}}$ formula ψ, two finite Kripke structures*

$$\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0) \text{ and } \mathcal{K}' = (\mathcal{AP}', S', R', \mu', s_0'),$$

and two traces $\rho \in \mathrm{Trc}_{\mathcal{K}}$, $\rho' \in \mathrm{Trc}_{\mathcal{K}'}$ such that

$$\mathcal{L}(\mathcal{K}_{|p\ell(\psi)}, \rho) = \mathcal{L}(\mathcal{K}'_{|p\ell(\psi)}, \rho')$$

and

$$reach(\mathcal{K}_{|p\ell(\psi)}, \mathrm{lst}(\rho)) \sim reach(\mathcal{K}'_{|p\ell(\psi)}, \mathrm{lst}(\rho')),$$

it holds that $\mathcal{K}, \rho \models \psi \iff \mathcal{K}', \rho' \models \psi$.

Theorem B.3.2. *The MC problem for* $\overline{\mathsf{B}}$ *formulas over finite Kripke structures is* ***PSPACE****-hard (under polynomial-time reductions).*

Proof. Let $\psi = Q_n x_n\, Q_{n-1}x_{n-1} \cdots Q_1 x_1 \phi(x_n, x_{n-1}, \ldots, x_1)$. We prove by induction on the number n of variables of ψ that ψ is true if and only if $\mathcal{K}_{QBF}^{x_n,\cdots,x_1} \models \xi$. In the following, $\phi(x_n, x_{n-1}, \ldots, x_1)\{x_i/v\}$, with $v \in \{\top, \bot\}$, denotes the formula obtained from $\phi(x_n, x_{n-1}, \ldots, x_1)$ by replacing all occurrences of x_i by v. Notice that $\mathcal{K}_{QBF}^{x_n,x_{n-1},\cdots,x_1}$ and $\mathcal{K}_{QBF}^{x_{n-1},\cdots,x_1}$ are isomorphic if restricted to the states $s^{\top}_{x_{n-1}}$, $s^{\bot}_{x_{n-1}}$, $\cdots$, $s^{\top}_{x_1}$, $s^{\bot}_{x_1}$, *sink* (i.e., the initial parts of both Kripke structures are eliminated), and the labelling of states is suitably restricted. Moreover, notice that only the trace s_0 satisfies the proposition letter t in ξ.

Base case ($n = 0$). In this case, $\psi = \phi(\emptyset)$ is a Boolean formula devoid of variables. If ψ is true, then in particular $\mathcal{K}_{QBF}^{\emptyset}, s_0 \models \phi(\emptyset)$ and thus $\mathcal{K}_{QBF}^{\emptyset} \models s \rightarrow \phi(\emptyset)(= \xi)$. Conversely, if $\mathcal{K}_{QBF}^{\emptyset} \models s \rightarrow \phi(\emptyset)$, then $\mathcal{K}_{QBF}^{\emptyset}, s_0 \models \phi(\emptyset)$, and since $\phi(\emptyset)$ has no variables, it must be true.

Case $n \geq 1$. We first prove that if $\psi = Q_n x_n Q_{n-1}x_{n-1} \cdots Q_1 x_1 \phi(x_n, x_{n-1}, \ldots, x_1)$ is true, then $\mathcal{K}_{QBF}^{x_n,\cdots,x_1} \models \xi$.

If $Q_n = \exists$, one possibility is that $Q_{n-1}x_{n-1} \cdots Q_1 x_1 \phi'(x_{n-1}, \ldots, x_1)$ is true, where $\phi'(x_{n-1}, \ldots, x_1) = \phi(x_n, x_{n-1}, \ldots, x_1)\{x_n/\top\}$. By the inductive hypothesis, it holds that $\mathcal{K}_{QBF}^{x_{n-1},\cdots,x_1} \models \xi'$, where $\xi' = t \rightarrow \xi'_{n-1}$ and $\xi'_{n-1} = \xi_{n-1}\{x_n/\top\}$. Thus $\mathcal{K}_{QBF}^{x_{n-1},\cdots,x_1}, s'_0 \models \xi'_{n-1}$ (s'_0 is the initial state of the structure $\mathcal{K}_{QBF}^{x_{n-1},\cdots,x_1}$). By Lemma B.3.1, $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 s^{\top}_{x_n} \models \xi'_{n-1}$. Since every right extension of $s_0 s_{x_n^{\top}}$ models x_n, $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 s^{\top}_{x_n} \models \xi_{n-1}$, and thus $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 \models \langle\overline{\mathsf{B}}\rangle(\tilde{x_n} \wedge \xi_{n-1})(= \xi_n)$. To conclude, $\mathcal{K}_{QBF}^{x_n,\cdots,x_1} \models t \rightarrow \xi_n (= \xi)$. The only other possible case is that $Q_{n-1}x_{n-1} \cdots Q_1 x_1 \phi'(x_{n-1}, \ldots, x_1)$ is true, with $\phi'(x_{n-1}, \ldots, x_1) = \phi(x_n, x_{n-1}, \ldots, x_1)\{x_n/\bot\}$. As before, it follows that $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 s^{\bot}_{x_n} \models \xi_{n-1}\{x_n/\bot\}$ and thus $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 \models \langle\overline{\mathsf{B}}\rangle(\tilde{x_n} \wedge \xi_{n-1})$.

Now, let $Q_n = \forall$. Both formulas $Q_{n-1}x_{n-1} \cdots Q_1 x_1 \phi(x_n, x_{n-1}, \ldots, x_1)\{x_n/\top\}$ and $Q_{n-1}x_{n-1} \cdots Q_1 x_1 \phi(x_n, x_{n-1}, \ldots, x_1)\{x_n/\bot\}$ are true. By reasoning as in the existential case, we have $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 s^{\top}_{x_n} \models \xi_{n-1}$ and $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 s^{\bot}_{x_n} \models \xi_{n-1}$. Thus, $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 \models [\overline{\mathsf{B}}](\tilde{x_n} \rightarrow \xi_{n-1})$ and $\mathcal{K}_{QBF}^{x_n,\cdots,x_1} \models s \rightarrow [\overline{\mathsf{B}}](\tilde{x_n} \rightarrow \xi_{n-1})$.

We now prove that if $\mathcal{K}_{QBF}^{x_n,\cdots,x_1} \models \xi$, then ψ is true.

If $Q_n = \exists$, then $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 \models \langle\overline{\mathrm{B}}\rangle(\tilde{x_n} \wedge \xi_{n-1})$. Thus $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 s_{x_n}^{\top} \models \xi_{n-1}$ or $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 s_{x_n}^{\bot} \models \xi_{n-1}$. In the former case, $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 s_{x_n}^{\top} \models \xi_{n-1}\{x_n/\top\}$ (since every right extension of $s_0 s_{x_n}^{\top}$ models x_n). By Lemma B.3.1, $\mathcal{K}_{QBF}^{x_{n-1},\cdots,x_1}, s_0' \models \xi_{n-1}\{x_n/\top\}$, and $\mathcal{K}_{QBF}^{x_{n-1},\cdots,x_1} \models t \rightarrow \xi_{n-1}\{x_n/\top\}$. By the inductive hypothesis, $Q_{n-1}x_{n-1}\cdots Q_1x_1\phi(x_n, x_{n-1}, \ldots, x_1)\{x_n/\top\}$ is true, thus the formula $\psi = \exists x_n Q_{n-1}x_{n-1}\cdots Q_1x_1\phi(x_n, x_{n-1}, \ldots, x_1)$ is true. In the latter case, we symmetrically have that $Q_{n-1}x_{n-1}..Q_1x_1\phi(x_n, x_{n-1}, \ldots, x_1)\{x_n/\bot\}$ is true, thus the formula $\psi = \exists x_n Q_{n-1}x_{n-1}\cdots Q_1x_1\phi(x_n, x_{n-1}, \ldots, x_1)$ is true.

If $Q_n = \forall$, then $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 \models [\overline{\mathrm{B}}](\tilde{x_n} \rightarrow \xi_{n-1})$. Thus both $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 s_{x_n}^{\top} \models \xi_{n-1}$ and $\mathcal{K}_{QBF}^{x_n,\cdots,x_1}, s_0 s_{x_n}^{\bot} \models \xi_{n-1}$. By reasoning as in the existential case, both $Q_{n-1}x_{n-1}\cdots Q_1x_1\phi(x_n, x_{n-1}, \ldots, x_1)\{x_n/\top\}$ and $Q_{n-1}x_{n-1}\cdots Q_1x_1\phi(x_n, x_{n-1}, \ldots, x_1)\{x_n/\bot\}$ are true, thus $\psi = \forall x_n Q_{n-1}x_{n-1}\cdots Q_1x_1\phi(x_n, x_{n-1}, \ldots, x_1)$ is true. □

B.4 Proof of Lemma 4.2.10

Lemma (4.2.10). *Let $h \geq 1$, ρ be a trace of $\mathcal{K}$, and let i, j be two consecutive ρ-positions in the h-prefix sampling of ρ. Then, for all ρ-positions $n, n' \in [i+1, j]$ such that $\rho(n) = \rho(n')$, it holds that $\rho(1, n)$ and $\rho(1, n')$ are $(h-1)$-prefix bisimilar.*

Proof. The proof is by induction on $h \geq 1$.

Base case: $h = 1$. The 1-prefix sampling of ρ is the prefix-skeleton sampling of ρ in $[1, |\rho|]$. Hence, being i and j consecutive positions in this sampling, for each position $k \in [i, j-1]$, there is $\ell \leq i$ such that $\rho(\ell) = \rho(k)$. Since $\rho(n) = \rho(n')$, it holds that $\mathrm{states}(\rho(1, n)) = \mathrm{states}(\rho(1, n'))$, and thus $\rho(1, n)$ and $\rho(1, n')$ are 0-prefix bisimilar.

Inductive step: $h > 1$. By definition of h-prefix sampling, there are two consecutive positions i', j' in the $(h-1)$-prefix sampling of ρ such that i, j are consecutive positions of the prefix-skeleton sampling of $\rho(i', j')$.

If $i = i'$, then $j = i + 1$, and thus, being $n, n' \in [i+1, j]$, we get that $n = n'$, and the result trivially holds.

Let $i \neq i'$, thus $i > i'$. As in the base case, we easily deduce that $\rho(1, n)$ and $\rho(1, n')$ are 0-prefix bisimilar. It remains to show that for each proper prefix ν of $\rho(1, n)$ (resp., ν' of $\rho(1, n')$), there is a proper prefix ν' of $\rho(1, n')$ (resp., ν of $\rho(1, n)$) such that ν and ν' are $(h-2)$-prefix bisimilar. Let us consider a proper prefix ν of $\rho(1, n)$ (the proof for the other direction is symmetric). It holds that $\nu = \rho(1, m)$, for some $m < n$. We distinguish two cases:

- $m \leq i$. Hence, $\rho(1, m)$ is a proper prefix of $\rho(1, n')$ and the result follows.

- $m > i$. Since i and j are consecutive positions of the prefix-skeleton sampling of $\rho(i', j')$, $i > i'$, and $m \in [i+1, j-1]$ (hence $m < j'$), there is $m' \in [i'+1, i]$ such that $\rho(m') = \rho(m)$ and m' is in the prefix-skeleton sampling of $\rho(i', j')$. Let $v' = \rho(1, m')$. Clearly, v' is a proper prefix of $\rho(1, n')$ (as $n' \geq i+1$). Moreover, since $m, m' \in [i'+1, j']$ and i', j' are consecutive positions in the $(h-1)$-prefix sampling of ρ, by the inductive hypothesis, $v = \rho(1, m)$ and $v' = \rho(1, m')$ are $(h-2)$-prefix bisimilar.

This concludes the proof of Lemma 4.2.10. □

C

Proofs of Chapter 5

Contents

C.1 Proof of Lemma 5.2.2 263
C.2 Proof of Theorem 5.3.3 264
C.3 Proof of Theorem 5.4.2 265

C.1 Proof of Lemma 5.2.2

Lemma (5.2.2). *Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure, ψ be an $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}$ formula, and $V_{\mathsf{A}}(\cdot,\cdot)$, $V_{\overline{\mathsf{A}}}(\cdot,\cdot)$ be two Boolean arrays. We assume that*

1. *for each $\langle \mathsf{A} \rangle \phi \in \mathsf{ModSubf}_{\mathsf{A}\overline{\mathsf{A}}}(\psi)$ and $s' \in S$, $V_{\mathsf{A}}(\phi, s') = \top$ if and only if there exists $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{fst}(\rho) = s'$ and $\mathcal{K}, \rho \models \phi$, and*
2. *for each $\langle \overline{\mathsf{A}} \rangle \phi \in \mathsf{ModSubf}_{\mathsf{A}\overline{\mathsf{A}}}(\psi)$ and $s' \in S$, $V_{\overline{\mathsf{A}}}(\phi, s') = \top$ if and only if there exists $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{lst}(\rho) = s'$ and $\mathcal{K}, \rho \models \phi$.*

Then $\texttt{Oracle}(\mathcal{K}, \psi, s, \textsc{direction}, V_{\mathsf{A}} \cup V_{\overline{\mathsf{A}}})$ features a successful computation (returning $\top$) if and only if:

- *there exists $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{fst}(\rho) = s$ and $\mathcal{K}, \rho \models \psi$, when $\textsc{direction}$ is $\textsc{forward}$;*
- *there exists $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{lst}(\rho) = s$ and $\mathcal{K}, \rho \models \psi$, when $\textsc{direction}$ is $\textsc{backward}$.*

Proof. It is easy to check that if $\tilde{\rho}$ is the trace non-deterministically generated by `A_trace` at line 1 then, for $i = 1, \ldots, |\tilde{\rho}|$, it holds that $\mathcal{K}, \tilde{\rho}(1, i) \models \phi$ if and only if $T[\phi, i] = \top$, either by hypothesis, when ϕ occurs in $\mathsf{ModSubf}_{\mathsf{A}\overline{\mathsf{A}}}(\psi)$ (lines 2–7), or by construction, when ϕ does not occur in $\mathsf{ModSubf}_{\mathsf{A}\overline{\mathsf{A}}}(\psi)$ (lines 8–22).

Let us now assume that the value of the parameter DIRECTION is FORWARD (the proof for the other direction is analogous).

($\Rightarrow$) If $\texttt{Oracle}(\mathcal{K}, \psi, s, \text{FORWARD}, V_{\mathsf{A}} \cup V_{\overline{\mathsf{A}}})$ features a successful computation, it means that there exists a trace $\tilde{\rho} \in \mathrm{Trc}_{\mathcal{K}}$ (generated at line 1) such that $\mathrm{fst}(\tilde{\rho}) = s$ and $T[\psi, |\tilde{\rho}|] = \top$ implying that $\mathcal{K}, \tilde{\rho} \models \psi$.

($\Leftarrow$) If there exists $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{fst}(\rho) = s$ and $\mathcal{K}, \rho \models \psi$, by Theorem 4.1.5 there exists $\tilde{\rho} \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathcal{K}, \tilde{\rho} \models \psi$, $\mathrm{fst}(\tilde{\rho}) = \mathrm{fst}(\rho)$, and $|\tilde{\rho}| \leq |S| \cdot (2|\psi| + 1)^2$. It follows that in some non-deterministic instance of $\texttt{Oracle}(\mathcal{K}, \psi, s, \text{FORWARD}, V_{\mathsf{A}} \cup V_{\overline{\mathsf{A}}})$, $\texttt{A_trace}(\mathcal{K}, s, |S| \cdot (2|\psi|+1)^2, \text{FORWARD})$ returns such $\tilde{\rho}$ (at line 1). Finally, we have that $T[\psi, |\tilde{\rho}|] = \top$ as $\mathcal{K}, \tilde{\rho} \models \psi$, and hence the considered instance of $\texttt{Oracle}(\mathcal{K}, \psi, s, \text{FORWARD}, V_{\mathsf{A}} \cup V_{\overline{\mathsf{A}}})$ is successful. □

C.2 Proof of Theorem 5.3.3

Theorem (5.3.3). *Let ψ be an $\mathsf{A}\overline{\mathsf{A}}$ formula and $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_1)$ be a finite Kripke structure. For every block B of $T_{\mathcal{K}, \neg\psi}$, if B is associated with an $\mathsf{A}\overline{\mathsf{A}}$ formula φ, then*

- *if B is a FORWARD block, for all $i \in \{1, \ldots, |S|\}$, $B(z_i) = \top$ if and only if there exists a trace $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{fst}(\rho) = s_i$ and $\mathcal{K}, \rho \models \varphi$;*
- *if B is a BACKWARD block, for all $i \in \{1, \ldots, |S|\}$, $B(z_i) = \top$ if and only if there exists a trace $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{lst}(\rho) = s_i$ and $\mathcal{K}, \rho \models \varphi$.*

Proof. The proof is by induction on the level $L \geq 1$ of the block B. The proof of the base case, i.e., for $L = 1$, is just a simpler version of the inductive step and it is therefore omitted.

Assume that B is a FORWARD block at level $L \geq 2$ associated with a formula φ (the BACKWARD case is symmetric).

We first prove the implication ($\Leftarrow$). We have to show that if there exists a trace $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{fst}(\rho) = s_i$ (for some $i \in \{1, \ldots, |S|\}$) and $\mathcal{K}, \rho \models \varphi$, then $B(z_i) = \top$ that is, there exists a truth assignment ω to the variables in V satisfying the formula $F_i(Y, V)$ of G_i. In [MMP17], it is proved that if φ is an $\mathsf{A}\overline{\mathsf{A}}$ formula and $\mathcal{K}, \rho \models \varphi$ (as in this case), there exists a trace $\rho' \in \mathrm{Trc}_{\mathcal{K}}$, with $|\rho'| \leq |S|^2 + 2$, such that $\mathrm{fst}(\rho) = \mathrm{fst}(\rho') = s_i$, $\mathrm{lst}(\rho) = \mathrm{lst}(\rho')$, and $\mathcal{K}, \rho' \models \varphi$.

Thus, by Proposition 5.3.2, there exists a truth assignment ω to the variables in V, that satisfies $trace(V_{trace}, V_{last}, V_{\mathcal{AP}})$, such that for all $1 \leq r \leq |\rho'|$ and $1 \leq j \leq |S|$, $\rho(r) = s_j \iff \omega(v_j^r) = \top$ and $\omega(v_j^{|\rho|}) = \omega(v_j)$, and for all $p \in \mathcal{AP}$, $\omega(v_p) = \top \iff \mathcal{K}, \rho' \models p$ ($\star$).

Since $L \geq 2$, it holds that $\mathsf{ModSubf}_{\mathrm{A}\overline{\mathrm{A}}}(\varphi) \neq \emptyset$. Let us consider a FORWARD child B' of B (if any), at a level lower than L, associated with some formula ξ such that $\langle \mathrm{A} \rangle \xi \in \mathsf{ModSubf}_{\mathrm{A}\overline{\mathrm{A}}}(\varphi)$. By the inductive hypothesis, for all j, $B'(z_j) = \top$ if and only if there exists a trace $\overline{\rho} \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{fst}(\overline{\rho}) = s_j$ and $\mathcal{K}, \overline{\rho} \models \xi$. Thus, $\mathcal{K}, \rho' \models \langle \mathrm{A} \rangle \xi$ if and only if there exists $\tilde{\rho} \in \mathrm{Trc}_{\mathcal{K}}$, with $\mathrm{fst}(\tilde{\rho}) = \mathrm{lst}(\rho') = s_j$, for some j, and $\mathcal{K}, \tilde{\rho} \models \xi$ if and only if $B'(z_j) (= y_j^{\xi}) = \top$. Thus if $\mathcal{K}, \rho' \models \langle \mathrm{A} \rangle \xi$, then $y_j^{\xi} = \top$, and $\omega(v_j) \wedge y_j^{\xi} = \top$. Now, to satisfy $F_i(Y, V)$, the truth assignment ω has to be such that $\omega(v_{\langle \mathrm{A} \rangle \xi}) = \top$. If $\mathcal{K}, \rho' \not\models \langle \mathrm{A} \rangle \xi$, then $y_j^{\xi} = \bot$, thus $\bigvee_{u=1}^{|S|} (\omega(v_u) \wedge y_u^{\xi})$ is false, and ω must be such that $\omega(v_{\langle \mathrm{A} \rangle \xi}) = \bot$. To conclude, $\mathcal{K}, \rho' \models \langle \mathrm{A} \rangle \xi$ if and only if $\omega(v_{\langle \mathrm{A} \rangle \xi}) = \top$ ($\star\star$). The symmetric reasoning can be applied to BACKWARD children of B. Since $\mathcal{K}, \rho' \models \varphi$, by ($\star$) and ($\star\star$), we have $\omega(\overline{\varphi}(V_{\mathcal{AP}}, V_{modSubf})) = \top$.

We prove now the implication ($\Rightarrow$). If $B(z_i) = \top$, then there exists a truth assignment ω of V satisfying $F_i(Y,V)$. In particular, ω satisfies $trace(V_{trace}, V_{last}, V_{\mathcal{AP}})$ and v_i^1, thus, by Proposition 5.3.2, there exists a trace $\rho \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{fst}(\rho) = s_i$, $\mathrm{lst}(\rho) = s_j$, for some j, and $\mathcal{K}, \rho \models p \iff \omega(v_p) = \top$, for any $p \in \mathcal{AP}$. By the inductive hypothesis, for all the formulas $\langle \mathrm{A} \rangle \xi \in \mathsf{ModSubf}_{\mathrm{A}\overline{\mathrm{A}}}(\varphi)$, $\mathcal{K}, \rho \models \langle \mathrm{A} \rangle \xi$ if and only if $\omega(v_{\langle \mathrm{A} \rangle \xi}) = \top$, and symmetrically, for all $\langle \overline{\mathrm{A}} \rangle \xi' \in \mathsf{ModSubf}_{\mathrm{A}\overline{\mathrm{A}}}(\varphi)$, $\mathcal{K}, \rho \models \langle \overline{\mathrm{A}} \rangle \xi'$ if and only if $\omega(v_{\langle \overline{\mathrm{A}} \rangle \xi'}) = \top$. Since $\omega(\overline{\varphi}(V_{\mathcal{AP}}, V_{modSubf})) = \top$, then we have $\mathcal{K}, \rho \models \varphi$. □

C.3 Proof of Theorem 5.4.2

Theorem (5.4.2). *Let $\mathcal{I}$ be an instance of SNSAT, with $|\mathcal{I}| = n$, and let $\mathcal{K}_{\mathcal{I}}$ and $\mathcal{F}_{\mathcal{I}}$ be defined as above. For all $0 \leq k \leq n+1$ and all $r = 1, \ldots, n$, it holds that:*

1. *if $k \geq r$, then $v_{\mathcal{I}}(x_r) = \top \iff \mathcal{K}_{\mathcal{I}}, w_{x_r} \models \psi_k$;*
2. *if $k \geq r+1$, then $v_{\mathcal{I}}(x_r) = \bot \iff \mathcal{K}_{\mathcal{I}}, \overline{w_{x_r}} \models \psi_k$.*

Proof. The proof is by induction on $k \geq 0$. If $k = 0$, the thesis trivially holds. Therefore, let us assume that $k \geq 1$. We first prove the ($\Leftarrow$) implication for both (1.) and (2.).

(1.) Assume that $k \geq r$ and $\mathcal{K}_{\mathcal{I}}, w_{x_r} \models \psi_k$. Thus, there exists $\rho \in \mathrm{Trc}_{\mathcal{K}_{\mathcal{I}}}$ such that $\rho = w_{x_r} \cdots s_0$ does not pass through any $\overline{s_m}$, for $1 \leq m \leq r$, and $\mathcal{K}_{\mathcal{I}}, \rho \models \varphi_k$. We show by induction on $1 \leq m \leq r$ that $\omega_\rho(x_m) = v_{\mathcal{I}}(x_m)$.

- Let us consider first the case where ρ passes through w_{x_m}, implying that $\omega_\rho(x_m) = \top$; thus $\mathcal{K}_\mathcal{I}, \rho \models x_m \wedge \neg r_m$ and $\mathcal{K}_\mathcal{I}, \rho \models F_m(x_1, \ldots, x_{m-1}, Z_m)$. If $m = 1$ (base case), since F_1 is satisfiable, then $v_\mathcal{I}(x_1) = \top$. If $m \geq 2$ (inductive step), by the inductive hypothesis $\omega_\rho(x_1) = v_\mathcal{I}(x_1)$, $\ldots$, $\omega_\rho(x_{m-1}) = v_\mathcal{I}(x_{m-1})$. Since $\mathcal{K}_\mathcal{I}, \rho \models F_m(x_1, \ldots, x_{m-1}, Z_m)$ or, equivalently, $F_m(\omega_\rho(x_1), \ldots, \omega_\rho(x_{m-1}), \omega_\rho(Z_m)) = \top$, it holds that $F_m(v_\mathcal{I}(x_1), \ldots, v_\mathcal{I}(x_{m-1}), \omega_\rho(Z_m)) = \top$ and, by definition of $v_\mathcal{I}$, we have $v_\mathcal{I}(x_m) = \top$.
- Conversely, let us consider the case where ρ passes through $\overline{w_{x_m}}$, implying that $\omega_\rho(x_m) = \bot$ and $m < r$, as we are assuming $\mathrm{fst}(\rho) = w_{x_r}$. In this case, the prefix $w_{x_r} \cdots \overline{w_{x_m}}$ of ρ satisfies both $\bigvee_{i=1}^{n} \langle \mathrm{A} \rangle \, p_{\overline{x_i}}$ and $\langle \mathrm{A} \rangle \left(\neg s \wedge \mathit{length}_2 \wedge \langle \mathrm{A} \rangle (\mathit{length}_2 \wedge \neg \psi_{k-1}) \right)$. Therefore, $\mathcal{K}_\mathcal{I}, \overline{w_{x_m}} \cdot \overline{s_m} \models \langle \mathrm{A} \rangle (\mathit{length}_2 \wedge \neg \psi_{k-1})$ and $\mathcal{K}_\mathcal{I}, \overline{s_m} \cdot w_{x_m} \not\models \psi_{k-1}$, with $\psi_{k-1} = \langle \mathrm{A} \rangle \, \varphi_{k-1}$. Hence $\mathcal{K}_\mathcal{I}, w_{x_m} \not\models \psi_{k-1}$. Since $1 \leq m < r$, we have $1 \leq m < r \leq k$, thus $k' = k - 1 \geq m \geq 1$. By the inductive hypothesis (on $k' = k - 1$), we get that $v_\mathcal{I}(x_m) = \bot$.

Therefore $v_\mathcal{I}(x_r) = \omega_\rho(x_r)$ and, since $w_{x_r} \in \mathrm{states}(\rho)$, we have $\omega_\rho(x_r) = \top$ and then $v_\mathcal{I}(x_r) = \top$ proving the thesis.

(2.) Assume that $k \geq r + 1$ and $\mathcal{K}_\mathcal{I}, \overline{w_{x_r}} \models \psi_k$. The proof follows the same steps as the previous case and it is thus only sketched: there exists $\rho \in \mathrm{Trc}_{\mathcal{K}_\mathcal{I}}$ such that $\rho = \overline{w_{x_r}} \cdots s_0$ does not pass through any $\overline{s_m}$, for $1 \leq m \leq r$, and $\mathcal{K}_\mathcal{I}, \rho \models \varphi_k$. The only difference is that the prefix $\overline{w_{x_r}}$ satisfies $\bigvee_{i=1}^{n} \langle \mathrm{A} \rangle \, p_{\overline{x_i}}$, thus, as before, we get $\mathcal{K}_\mathcal{I}, w_{x_r} \not\models \psi_{k-1}$. Now, $k' = k - 1 \geq r \geq 1$ and, by the inductive hypothesis (on $k' = k - 1$), $v_\mathcal{I}(x_r) = \bot$.

We prove now the converse implication ($\Rightarrow$) for both (1.) and (2.).

(1.) Assume that $k \geq r$ and $v_\mathcal{I}(x_r) = \top$. Let us consider the trace $\rho \in \mathrm{Trc}_{\mathcal{K}_\mathcal{I}}$, $\rho = w_{x_r} \cdots s_0$ never passing through any $\overline{s_m}$, for $1 \leq m \leq r$, such that $w_{x_m} \in \mathrm{states}(\rho)$ if $v_\mathcal{I}(x_m) = \top$, and $\overline{w_{x_m}} \in \mathrm{states}(\rho)$ if $v_\mathcal{I}(x_m) = \bot$, for $1 \leq m \leq r$. Such a choice of ρ ensures that $v_\mathcal{I}(x_m) = \omega_\rho(x_m)$. In addition, the choice of ρ has to induce also the proper truth-assignment of private variables, that is, if $v_\mathcal{I}(x_m) = \top$, then for $1 \leq u_m \leq j_m$, $w_{z_m^{u_m}} \in \mathrm{states}(\rho)$ if $F_m(v_\mathcal{I}(x_1), \ldots, v_\mathcal{I}(x_{m-1}), Z_m)$ is satisfied for $z_m^{u_m} = \top$, and $\overline{w_{z_m^{u_m}}} \in \mathrm{states}(\rho)$ otherwise. Note that such a choice of ρ is always possible. We have to show that $\mathcal{K}_\mathcal{I}, \rho \models \varphi_k$, hence $\mathcal{K}_\mathcal{I}, w_{x_r} \models \psi_k$.

- For all $1 \leq m \leq r$ such that $v_\mathcal{I}(x_m) = \top$, it holds that $F_m(v_\mathcal{I}(x_1), \ldots, v_\mathcal{I}(x_{m-1}), Z_m)$ is satisfiable. Hence, by our choice of ρ, $F_m(\omega_\rho(x_1), \ldots, \omega_\rho(x_{m-1}), \omega_\rho(Z_m)) = \top$, or, equivalently, $\mathcal{K}_\mathcal{I}, \rho \models F_m(x_1, \ldots, x_{m-1}, Z_m)$. Thus, $\mathcal{K}_\mathcal{I}, \rho \models \bigwedge_{i=1}^{n} \Big((x_i \wedge \neg r_i) \rightarrow F_i(x_1, \ldots, x_{i-1}, Z_i) \Big)$.
- Conversely, for all $1 \leq m < r$ such that $v_\mathcal{I}(x_m) = \bot$ ($m \neq r$ as, by hypothesis, $v_\mathcal{I}(x_r) = \top$), it holds that $\overline{w_{x_m}} \in \mathrm{states}(\rho)$. Since

$m < r$, we have $k \geq r > m$ and $k - 1 \geq m \geq 1$. By the inductive hypothesis, $\mathcal{K}_\mathcal{I}, w_{x_m} \not\models \psi_{k-1}$. It follows that $\mathcal{K}_\mathcal{I}, \overline{s_m} \cdot w_{x_m} \models \neg\psi_{k-1} \wedge \mathit{length}_2$, $\mathcal{K}_\mathcal{I}, \overline{w_{x_m}} \cdot \overline{s_m} \models \neg s \wedge \mathit{length}_2 \wedge \langle \mathrm{A} \rangle(\neg\psi_{k-1} \wedge \mathit{length}_2)$ and $\mathcal{K}_\mathcal{I}, \overline{w_{x_m}} \models \langle \mathrm{A} \rangle(\neg s \wedge \mathit{length}_2 \wedge \langle \mathrm{A} \rangle(\neg\psi_{k-1} \wedge \mathit{length}_2))$. Hence, $\mathcal{K}_\mathcal{I}, \rho \models [\mathrm{B}]((\bigvee_{i=1}^{n} \langle \mathrm{A} \rangle p_{\overline{x_i}}) \rightarrow \langle \mathrm{A} \rangle(\neg s \wedge \mathit{length}_2 \wedge \langle \mathrm{A} \rangle(\neg\psi_{k-1} \wedge \mathit{length}_2)))$.

Combining the two cases, we can conclude that $\mathcal{K}_\mathcal{I}, \rho \models \varphi_k$.

(2.) Assume that $k \geq r + 1$ and $v_\mathcal{I}(x_r) = \bot$. The proof is as before and it is only sketched. In this case, we choose a trace $\rho = \overline{w_{x_r}} \cdots s_0$. Since $k' = k - 1 \geq r$, by the inductive hypothesis, $\mathcal{K}_\mathcal{I}, w_{x_r} \not\models \psi_{k-1}$, and we can prove that $\mathcal{K}_\mathcal{I}, \overline{w_{x_r}} \models \langle \mathrm{A} \rangle(\neg s \wedge \mathit{length}_2 \wedge \langle \mathrm{A} \rangle(\neg\psi_{k-1} \wedge \mathit{length}_2))$. □



Proofs of Chapter 6

Contents

D.1 Proof of Lemma 6.3.3 269
D.2 Proof of Lemma 6.3.4 271
D.3 Proof of Lemma 6.3.10 272
D.4 Proof of Lemma 6.3.13 273
D.5 Proof of Lemma 6.4.4 274

D.1 Proof of Lemma 6.3.3

Lemma (6.3.3). *Let Σ be a finite alphabet, $b \in \Sigma$, $\Gamma = \Sigma \setminus \{b\}$, $\mathcal{L} \subseteq \Gamma^+$, and ψ be a* BE *formula over Γ such that $\mathcal{L}_{act}(\psi) = \mathcal{L}$. Then, there are* BE *formulas defining (under the action-based semantics) the languages $b\mathcal{L}$, $\Sigma^* b\mathcal{L}$, $\Sigma^* b(\mathcal{L} + \varepsilon)$, $\mathcal{L}b$, $\mathcal{L}b\Sigma^*$, $(\mathcal{L} + \varepsilon)b\Sigma^*$, and $b\mathcal{L}b$.*

Proof. We focus on the cases for the languages $b\mathcal{L}$, $\Sigma^* b\mathcal{L}$, $\Sigma^* b$, and $b\mathcal{L}b$ (for the other languages, the proof is similar: $\Sigma^* b(\mathcal{L} + \varepsilon) = \Sigma^* b\mathcal{L} + \Sigma^* b$, $\mathcal{L}b$ is symmetric to $b\mathcal{L}$, $\mathcal{L}b\Sigma^*$ to $\Sigma^* b\mathcal{L}$, and $(\mathcal{L} + \varepsilon)b\Sigma^*$ to $\Sigma^* b(\mathcal{L} + \varepsilon)$).

Let ψ be a BE formula over Γ such that $\mathcal{L}_{act}(\psi) = \mathcal{L}$.

Language $b\mathcal{L}$. The BE formula defining the language $b\mathcal{L}$ is:

$$(\neg \mathit{length}_1 \wedge \langle \mathrm{B} \rangle\, b \wedge [\mathrm{E}](\neg b \wedge [\mathrm{B}]\, \neg b)) \wedge h_b(\psi), \tag{D.1}$$

where the formula $h_b(\psi)$ is inductively defined on the structure of ψ in the following way. The mapping h_b is homomorphic with respect to the Boolean connectives, while for the atomic actions in Γ and the modalities $\langle \mathrm{E} \rangle$ and $\langle \mathrm{B} \rangle$, it is defined as follows:

- for all $a \in \Gamma$, $h_b(a) = a \vee (\langle \mathrm{B} \rangle\, b \wedge \langle \mathrm{E} \rangle\, a \wedge [\mathrm{E}]\, a)$;
- $h_b(\langle \mathrm{B} \rangle\, \theta) = (\langle \mathrm{B} \rangle\, h_b(\theta) \wedge \neg \langle \mathrm{B} \rangle\, b) \vee \langle \mathrm{B} \rangle (h_b(\theta) \wedge \langle \mathrm{B} \rangle\, b)$;
- $h_b(\langle \mathrm{E} \rangle\, \theta) = (\langle \mathrm{E} \rangle\, h_b(\theta) \wedge \neg \langle \mathrm{B} \rangle\, b) \vee (\langle \mathrm{B} \rangle\, b \wedge \langle \mathrm{E} \rangle \langle \mathrm{E} \rangle\, h_b(\theta))$.

The first conjunct of (D.1) ensures that a word u' in the defined language has length at least 2 and it has the form bu without any occurrence of b in u. The second conjunct $h_b(\psi)$ ensures that u belongs to the language defined by ψ. For atomic actions and temporal modalities, $h_b(\psi)$ is a disjunction of two possible choices; the appropriate one is forced at top level by the first conjunct of (D.1), that constrains one and only one b to occur in the word in the first position.

By a straightforward structural induction on ψ, it can be shown that the following fact holds.

Claim D.1.1. Let $u \in \Gamma^+$, $u' = bu$, and $|u| = n + 1$. Then, for all $i, j \in [0, n]$ with $i \leq j$, $u(i, j) \in \mathcal{L}_{act}(\psi)$ if and only if $u'(\hat{i}, j + 1) \in \mathcal{L}_{act}(h_b(\psi))$, where $\hat{i} = i$ if $i = 0$, and $\hat{i} = i + 1$ otherwise.

By Claim D.1.1, for each $u \in \Gamma^+$, $u \in \mathcal{L}_{act}(\psi)$ if and only if $bu \in \mathcal{L}_{act}(h_b(\psi))$. Therefore, (D.1) captures the language $b\mathcal{L}_{act}(\psi)$.

Languages $\Sigma^* b\mathcal{L}$ and $\Sigma^* b$. Following the proof given for the case of the language $b\mathcal{L}$, with $\mathcal{L} \subseteq \Gamma^+$, one can construct a BE formula φ defining the language $b\mathcal{L}$. Hence, the BE formula $\varphi \vee \langle \mathrm{E} \rangle\, \varphi$ defines $\Sigma^* b\mathcal{L}$. The BE formula defining $\Sigma^* b$ is $b \vee \langle \mathrm{E} \rangle\, b$.

Language $b\mathcal{L}b$. By the proof given for the language $b\mathcal{L}$, with $\mathcal{L} \subseteq \Gamma^+$, one can build a BE formula φ defining the language $b\mathcal{L}$. The BE formula defining the language $b\mathcal{L}b$ is the formula:

$$(\neg \mathit{length}_1 \wedge \neg \mathit{length}_2 \wedge \langle \mathrm{B} \rangle\, b \wedge \langle \mathrm{E} \rangle\, b \wedge [\mathrm{E}]\, [\mathrm{B}]\, \neg b) \wedge k_b(\varphi) \tag{D.2}$$

where the formula $k_b(\varphi)$ is inductively defined on the structure of φ in the following way. The mapping k_b is homomorphic with respect to the Boolean connectives, while for the atomic actions in Σ and the modalities $\langle \mathrm{E} \rangle$ and $\langle \mathrm{B} \rangle$, it is defined as follows:

- for all $a \in \Gamma$, $k_b(a) = a \vee (\langle \mathrm{E} \rangle\, b \wedge \langle \mathrm{B} \rangle\, a \wedge [\mathrm{B}]\, a)$;
- $k_b(b) = b$;

- $k_b(\langle \mathrm{B}\rangle\, \theta) = (\langle \mathrm{B}\rangle\, k_b(\theta) \wedge \neg \langle \mathrm{E}\rangle\, b) \vee (\langle \mathrm{E}\rangle\, b \wedge \langle \mathrm{B}\rangle\, \langle \mathrm{B}\rangle\, k_b(\theta))$.
- $k_b(\langle \mathrm{E}\rangle\, \theta) = (\langle \mathrm{E}\rangle\, k_b(\theta) \wedge \neg \langle \mathrm{E}\rangle\, b) \vee \langle \mathrm{E}\rangle (k_b(\theta) \wedge \langle \mathrm{E}\rangle\, b)$.

The first conjunct of (D.2) ensures that a word u' in the defined language has length at least 3 and it has the form bub without any occurrence of b in u. The second conjunct $k_b(\varphi)$ ensures that bu belongs to the language defined by φ. Similarly to the case of the language $b\mathcal{L}$, for atomic actions (different from b) and temporal modalities, $k_b(\psi)$ is a disjunction of two possible choices; the appropriate one is forced at top level by the first conjunct of (D.2), that constrains one and only one b to occur in the word in the last position.

By a straightforward structural induction on φ, it can be shown that the following fact holds.

Claim D.1.2. Let $u \in \Gamma^+$ and $|bu| = n + 1$. Then, for all $i, j \in [0, n]$ with $i \leq j$, $bu(i, j) \in \mathcal{L}_{act}(\varphi)$ if and only if $bub(i, \hat{j}) \in \mathcal{L}_{act}(k_b(\varphi))$ where $\hat{j} = j$ if $j < n$, and $\hat{j} = n + 1$ otherwise.

By Claim D.1.2, for each $u \in \Gamma^+$, $bu \in \mathcal{L}_{act}(\varphi)$ if and only if $bub \in \mathcal{L}_{act}(k_b(\varphi))$ implying that the formula of (D.2) defines the language $\mathcal{L}_{act}(\varphi)b$. □

D.2 Proof of Lemma 6.3.4

Lemma (6.3.4). *Let Σ and Δ be finite alphabets, $b \in \Sigma$, $\Gamma = \Sigma \setminus \{b\}$, $U_0 = \Gamma^* b$, $h_0 : U_0 \to \Delta$ and $h : U_0^+ \to \Delta^+$ be defined by $h(u_0 u_1 \cdots u_n) = h_0(u_0) \cdots h_0(u_n)$. Assume that, for each $d \in \Delta$, there is a* BE *formula capturing the language $\mathcal{L}_d = \{u \in \Gamma^+ \mid h_0(ub) = d\}$. Then, for each* BE *formula φ over Δ, one can construct a* BE *formula over Σ capturing the language $\Gamma^* b h^{-1}(\mathcal{L}_{act}(\varphi))\Gamma^*$.*

Proof. By hypothesis and Lemma 6.3.3, for each $d \in \Delta$ there exists a BE formula θ_d over Σ defining the language $b\mathcal{L}_d b$, where $\mathcal{L}_d = \{u \in \Gamma^+ \mid h_0(ub) = d\}$. Hence there exists a BE formula $\hat{\theta}_d$ over Σ capturing the language $b\hat{\mathcal{L}}_d b$, where $\hat{\mathcal{L}}_d = \{u \in \Gamma^* \mid h_0(ub) = d\}$ (note that $\mathcal{L}_d = \hat{\mathcal{L}}_d \setminus \{\varepsilon\}$).

Let φ be a BE formula over Δ. By structural induction over φ, we construct a BE formula φ^+ over Σ such that $\mathcal{L}_{act}(\varphi^+) = \Gamma^* b h^{-1}(\mathcal{L}_{act}(\varphi))\Gamma^*$. The formula φ^+ is defined as follows:

- $\varphi = d$ with $d \in \Delta$. We have that $\mathcal{L}_{act}(d) = d^+$ and $\Gamma^* b h^{-1}(\mathcal{L}_{act}(d))\Gamma^*$ is the set of finite words in $\Gamma^* b \Sigma^* b \Gamma^*$ such that each subword $u(i, j)$ of u which

is in $b\Gamma^* b$ is in $b\hat{\mathcal{L}}_d b$ as well. Using the formula $\psi_b := \neg length_1 \wedge \langle \mathrm{B} \rangle\, b \wedge \langle \mathrm{E} \rangle\, b \wedge [\mathrm{E}]\,[\mathrm{B}]\,\neg b$ to define the language $b\Gamma^* b$, φ^+ is defined as follows:

$$\varphi^+ = (\langle \mathrm{G} \rangle\, \psi_b) \wedge [\mathrm{G}](\psi_b \rightarrow \hat{\theta}_d).$$

- $\varphi = \neg\theta$. We have that

$$\Gamma^* bh^{-1}(\mathcal{L}_{act}(\varphi))\Gamma^* = \Gamma^* bh^{-1}(\Delta^+ \setminus \mathcal{L}_{act}(\theta))\Gamma^* = \Gamma^* bh^{-1}(\Delta^+)\Gamma^* \cap \overline{\Gamma^* bh^{-1}(\mathcal{L}_{act}(\theta))\Gamma^*},$$

  where $\Gamma^* bh^{-1}(\Delta^+)\Gamma^*$ restricts the set of "candidate" models to the well-formed ones.
  Thus, taking ψ_b as defined in the previous case, φ^+ is given by:

$$\varphi^+ = (\langle \mathrm{G} \rangle\, \psi_b) \wedge [\mathrm{G}](\psi_b \rightarrow \bigvee_{d \in \Delta} \hat{\theta}_d) \wedge \neg\theta^+,$$

  where, by the inductive hypothesis, $\mathcal{L}_{act}(\theta^+) = \Gamma^* bh^{-1}(\mathcal{L}_{act}(\theta))\Gamma^*$.
- $\varphi = \theta \wedge \psi$. We simply have $\varphi^+ = \theta^+ \wedge \psi^+$.
- $\varphi = \langle \mathrm{B} \rangle\, \theta$. First, we note that $\Gamma^* bh^{-1}(\mathcal{L}_{act}(\langle \mathrm{B} \rangle\, \theta))\Gamma^*$ is the set of finite words in the language $\Gamma^* bh^{-1}(\mathcal{L}_{act}(\theta))h^{-1}(\Delta^+)\Gamma^*$, which is included in the language $\Gamma^* bh^{-1}(\Delta^+)\Gamma^*$ defined by the formula $[\mathrm{G}](\psi_b \rightarrow \bigvee_{d \in \Delta} \hat{\theta}_d)$. Note also that, by the inductive hypothesis, $\Gamma^* bh^{-1}(\mathcal{L}_{act}(\theta))$ is included in the language of θ^+. Thus, φ^+ is given by (for $\xi = (\langle \mathrm{E} \rangle\, b) \wedge \langle \mathrm{B} \rangle(\theta^+ \wedge \langle \mathrm{E} \rangle\, b)$):

$$\varphi^+ = [\mathrm{G}](\psi_b \rightarrow \bigvee_{d \in \Delta} \hat{\theta}_d) \wedge (\xi \vee \langle \mathrm{B} \rangle\, \xi).$$

- $\varphi = \langle \mathrm{E} \rangle\, \theta$. $\Gamma^* bh^{-1}(\mathcal{L}_{act}(\langle \mathrm{E} \rangle\, \theta))\Gamma^*$ is the set $\Gamma^* bh^{-1}(\Delta^+)h^{-1}(\mathcal{L}_{act}(\theta))\Gamma^*$ included in the language $\Gamma^* bh^{-1}(\Delta^+)\Gamma^*$, symmetrically to the previous case. Thus, φ^+ is given by (for $\xi' = (\langle \mathrm{B} \rangle\, b) \wedge \langle \mathrm{E} \rangle(\theta^+ \wedge \langle \mathrm{B} \rangle\, b)$):

$$\varphi^+ = [\mathrm{G}](\psi_b \rightarrow \bigvee_{d \in \Delta} \hat{\theta}_d) \wedge (\xi' \vee \langle \mathrm{E} \rangle\, \xi').$$ □

D.3 Proof of Lemma 6.3.10

Lemma (6.3.10). *Let ψ be a* simple *hybrid CTL^*_{lp} formula with respect to x. Then $(F^- x) \wedge \psi$ is congruent to a formula of the form $(F^- x) \wedge \xi$, where ξ is a Boolean combination of the atomic formula x and CTL^* formulas.*

Proof. Let ψ be a *simple* hybrid CTL^*_{lp} formula with respect to x. From a syntactic point of view, ψ is not, in general, a CTL^* formula due to the occurrences of the free variable x. We show that these occurrences can be separated whenever ψ is paired with F^-x, obtaining a Boolean combination of the atomic formula x and CTL^* formulas.

The base case where $\psi = x$, $\psi = p \in \mathcal{AP}$, or $\psi = \exists\psi'$, is obvious.

As for the inductive step, let ψ be a Boolean combination of simple hybrid CTL^*_{lp} formulas θ, where θ is either $p \in \mathcal{AP}$, the variable x, a CTL^* formula, or a simple hybrid CTL^*_{lp} formula (with respect to x) of the forms $\mathsf{X}\theta_1$ or $\theta_1\mathsf{U}\theta_2$. Thus, we just need to consider the cases where $\theta = \mathsf{X}\theta_1$ or $\theta = \theta_1\mathsf{U}\theta_2$.

Let us consider the case $\theta = \mathsf{X}\theta_1$. Since there are no past temporal modalities in θ_1, $\mathsf{X}\theta_1$ forces the free occurrence of x in ψ to be interpreted in a (strictly) future position. However, ψ is conjunct with the formula F^-x, which turns out to be false when x is associated with a (strictly) future position. Let us denote by $\widehat{\theta}$ the CTL^* formula obtained from θ by replacing each occurrence of x in ψ with $\bot$ (false). Now, when x is mapped to a (strictly) future position, F^-x is false and, when x is mapped to a present/past position, F^-x is true, and θ and $\widehat{\theta}$ are congruent. As a consequence it is clear that $(\mathsf{F}^-x) \wedge \theta$ is congruent to $(\mathsf{F}^-x) \wedge \widehat{\theta}$.

Let us consider the case for $\theta = \theta_1\mathsf{U}\theta_2$. Using the same arguments of the previous case, we have that $(\mathsf{F}^-x) \wedge \theta$ is congruent to $(\mathsf{F}^-x) \wedge (\theta_2 \vee (\theta_1 \wedge \mathsf{X}(\widehat{\theta_1\mathsf{U}\theta_2})))$. By distributivity of $\wedge$ over $\vee$, we get $((\mathsf{F}^-x) \wedge \theta_2) \vee ((\mathsf{F}^-x) \wedge \theta_1 \wedge \mathsf{X}(\widehat{\theta_1\mathsf{U}\theta_2}))$. The thesis follows by applying the inductive hypothesis to $(\mathsf{F}^-x) \wedge \theta_2$ and to $(\mathsf{F}^-x) \wedge \theta_1$, and by factorizing F^-x (note that $\widehat{\theta_1\mathsf{U}\theta_2}$ is a CTL^* formula). □

D.4 Proof of Lemma 6.3.13

Lemma (6.3.13). *Let $(\mathsf{F}^-x) \wedge \exists\varphi(x)$ (resp., $\exists\varphi$) be a well-formed formula (resp., well-formed sentence) of hybrid CTL^*_{lp}. Then there exists a finite set $\mathcal{H}$ of CTL^* formulas of the form $\exists\psi$, such that $(\mathsf{F}^-x) \wedge \exists\varphi(x)$ (resp., $\exists\varphi$) is congruent to a well-formed formula of hybrid CTL^*_{lp} which is a Boolean combination of CTL^* formulas and (formulas that correspond to)* pure past *LTL_p formulas over the set of proposition letters $\mathcal{AP} \cup \mathcal{H} \cup \{x\}$ (resp., $\mathcal{AP} \cup \mathcal{H}$).*

Proof. As in the case of Lemma 6.3.12, we focus on well-formed formulas of the form $(\mathsf{F}^-x) \wedge \exists\varphi(x)$ (the case of well-formed sentences of the form $\exists\varphi$ is similar). The proof is by induction on the nesting depth of the path quantifier $\exists$ in $\varphi(x)$.

In the base case we have $\exists SubF(\varphi) = \emptyset$: we apply Lemma 6.3.12, and the result follows by taking $\mathcal{H} = \emptyset$.

As for the inductive step, we let $\exists\psi \in \exists SubF(\varphi)$. Since $(\mathsf{F}^{-}x) \wedge \exists\varphi(x)$ is well-formed, either ψ is a sentence, or ψ has a unique free variable y and $\exists\psi(y)$ occurs in $\varphi(x)$ in the context $(\mathsf{F}^{-}y) \wedge \exists\psi(y)$. Assume that the latter case holds (the former is similar). By definition of well-formed formula, y is not free in $\varphi(x)$, and $(\mathsf{F}^{-}y) \wedge \exists\psi(y)$ must occur in the scope of some occurrence of $\downarrow y$. By the inductive hypothesis, the thesis holds for $(\mathsf{F}^{-}y) \wedge \exists\psi(y)$. Hence there exists a finite set $\mathcal{H}'$ of CTL^{*} formulas of the form $\exists\theta$ such that $(\mathsf{F}^{-}y) \wedge \exists\psi(y)$ is congruent to a well-formed formula of hybrid CTL^{*}_{lp}, say $\xi(y)$, which is a Boolean combination of CTL^{*} formulas and formulas that correspond to *pure past* LTL_p formulas over the set of proposition letters $\mathcal{AP} \cup \mathcal{H}' \cup \{y\}$.

By replacing each occurrence of $(\mathsf{F}^{-}y) \wedge \exists\psi(y)$ in $\varphi(x)$ with $\xi(y)$, and repeating the procedure for all the formulas in $\exists SubF(\varphi)$, we obtain a well-formed formula of hybrid CTL^{*}_{lp} of the form $(\mathsf{F}^{-}x) \wedge \exists\theta(x)$ which is congruent to $(\mathsf{F}^{-}x) \wedge \exists\varphi(x)$ (note that the congruence relation is closed under substitution) and such that $\exists SubF(\theta)$ consists of CTL^{*} formulas. At this point we can apply Lemma 6.3.12 finally proving the assertion. □

D.5 Proof of Lemma 6.4.4

To prove the lemma, we need some technical definitions. Let ρ be a trace of $\mathcal{K}_n$ (note that $\mathcal{K}_n$ and $\mathcal{M}_n$ feature the same traces). By construction, ρ has the form $\rho' \cdot \rho''$, where ρ' is a (possibly empty) trace visiting only states where p does not hold, and ρ'' is a (possibly empty) trace visiting only the state t, where p holds. We say that ρ' (resp., ρ'') is the $\emptyset$-part (resp., p-part) of ρ.

Let $N_\emptyset(\rho)$, $N_p(\rho)$, and $D_p(\rho)$ be the natural numbers defined as follows:

- $N_\emptyset(\rho) = |\rho'|$ (the length of the $\emptyset$-part of ρ);
- $N_p(\rho) = |\rho''|$ (the length of the p-part of ρ);
- $D_p(\rho) = 0$ if $N_p(\rho) > 0$ (i.e., $\mathrm{lst}(\rho) = t$); otherwise, $D_p(\rho)$ is the length of the minimal trace starting from $\mathrm{lst}(\rho)$ and leading to s_{2n}. Notice that $D_p(\rho)$ is well defined and $0 \leq D_p(\rho) \leq 2n + 1$.

By construction, the following property holds.

Proposition D.5.1. *For all traces ρ and ρ' of $\mathcal{K}_n$, if $D_p(\rho) = D_p(\rho')$, then* $\mathrm{lst}(\rho) = \mathrm{lst}(\rho')$.

Now, for each $h \in [1, n]$, we introduce the notion of *h-compatibility* between traces of $\mathcal{K}_n$. Intuitively, this notion provides a sufficient condition to make two traces indistinguishable under the state-based semantics by means of balanced HS formulas having size at most h.

Definition D.5.2 (h-compatibility)**.** Let $h \in [1, n]$. Two traces ρ and ρ' of $\mathcal{K}_n$ are *h-compatible* if the following conditions hold:

- $N_p(\rho) = N_p(\rho')$;
- either $N_\emptyset(\rho) = N_\emptyset(\rho')$, or $N_\emptyset(\rho) \geq h$ and $N_\emptyset(\rho') \geq h$;
- either $D_p(\rho) = D_p(\rho')$, or $D_p(\rho) \geq h$ and $D_p(\rho') \geq h$.

We denote by $\tilde{\mathcal{R}}(h)$ the binary relation over the set of traces of $\mathcal{K}_n$ such that $(\rho, \rho') \in \tilde{\mathcal{R}}(h)$ if and only if ρ and ρ' are h-compatible.
Notice that $\tilde{\mathcal{R}}(h)$ is an equivalence relation, for all $h \in [1, n]$. Moreover $\tilde{\mathcal{R}}(h) \subseteq \tilde{\mathcal{R}}(h-1)$, for all $h \in [2, n]$, that is, $\tilde{\mathcal{R}}(h)$ is a refinement of $\tilde{\mathcal{R}}(h-1)$.

By construction, the following property, that will be exploited in the proof of Lemma 6.4.4, can be easily shown.

Proposition D.5.3. *For every trace ρ of $\mathcal{K}_n$ starting from s_0 (resp., s_1), there exists a trace ρ' of $\mathcal{K}_n$ starting from s_1 (resp., s_0) such that $(\rho, \rho') \in \tilde{\mathcal{R}}(n)$.*

The following lemma lists some useful properties of $\tilde{\mathcal{R}}(h)$.

Lemma D.5.4. *Let $h \in [2, n]$ and $(\rho, \rho') \in \tilde{\mathcal{R}}(h)$. The next properties hold:*

1. *for each proper prefix σ of ρ, there exists a proper prefix σ' of ρ' such that $(\sigma, \sigma') \in \tilde{\mathcal{R}}(\lfloor \frac{h}{2} \rfloor)$;*
2. *for each trace of the form $\rho \cdot \sigma$, where σ is not empty, there exists a trace of the form $\rho' \cdot \sigma'$ such that σ' is not empty and $(\rho \cdot \sigma, \rho' \cdot \sigma') \in \tilde{\mathcal{R}}(\lfloor \frac{h}{2} \rfloor)$;*
3. *for each proper suffix σ of ρ, there exists a proper suffix σ' of ρ' such that $(\sigma, \sigma') \in \tilde{\mathcal{R}}(h-1)$;*
4. *for each trace of the form $\sigma \cdot \rho$, where σ is not empty, there exists a trace of the form $\sigma' \cdot \rho'$ such that σ' is not empty and $(\sigma \cdot \rho, \sigma' \cdot \rho') \in \tilde{\mathcal{R}}(h)$.*

Proof. We prove (1.) and (2.); (3.) and (4.) easily follow by construction and by definition of h-compatibility.

(1.) We distinguish the following cases:

1. $D_p(\rho) < h$ and $N_\emptyset(\rho) < h$. Since $(\rho, \rho') \in \tilde{\mathcal{R}}(h)$ and $h \in [2, n]$, it holds that $D_p(\rho) = D_p(\rho')$, $N_\emptyset(\rho) = N_\emptyset(\rho')$, and $N_p(\rho) = N_p(\rho')$, and thus $\rho = \rho'$.
2. $D_p(\rho) \geq h$. Since $(\rho, \rho') \in \tilde{\mathcal{R}}(h)$, $D_p(\rho') \geq h$, $N_p(\rho') = N_p(\rho) = 0$, and either $N_\emptyset(\rho') = N_\emptyset(\rho)$, or $N_\emptyset(\rho) \geq h$ and $N_\emptyset(\rho') \geq h$. In both cases, by construction it easily follows that for each proper prefix σ of ρ, there exists a proper prefix σ' of ρ' such that $(\sigma, \sigma') \in \tilde{\mathcal{R}}(h-1) \subseteq \tilde{\mathcal{R}}(\lfloor \frac{h}{2} \rfloor)$.

3. $D_p(\rho) < h$ and $N_\emptyset(\rho) \geq h$. Since $(\rho, \rho') \in \tilde{\mathcal{R}}(h)$, we have that $D_p(\rho') = D_p(\rho)$ (and hence, by Proposition D.5.1, $\mathrm{lst}(\rho) = \mathrm{lst}(\rho')$), $N_p(\rho') = N_p(\rho)$, and $N_\emptyset(\rho') \geq h$.
   Let σ be a proper prefix of ρ. We distinguish the following three subcases:
   a. $N_\emptyset(\sigma) < \lfloor \frac{h}{2} \rfloor$. Since $N_\emptyset(\rho) \geq h$, we have that $D_p(\sigma) \geq \lfloor \frac{h}{2} \rfloor$ and $|\sigma| = N_\emptyset(\sigma)$ (and thus $N_p(\sigma) = 0$). Since $N_\emptyset(\rho') \geq h$, by taking the proper prefix σ' of ρ' having length $N_\emptyset(\sigma)$, we obtain that $(\sigma, \sigma') \in \tilde{\mathcal{R}}(\lfloor \frac{h}{2} \rfloor)$.
   b. $N_\emptyset(\sigma) \geq \lfloor \frac{h}{2} \rfloor$ and $D_p(\sigma) \geq \lfloor \frac{h}{2} \rfloor$. By taking the prefix σ' of ρ' of length $\lfloor \frac{h}{2} \rfloor$, we get that $(\sigma, \sigma') \in \tilde{\mathcal{R}}(\lfloor \frac{h}{2} \rfloor)$.
   c. $N_\emptyset(\sigma) \geq \lfloor \frac{h}{2} \rfloor$ and $D_p(\sigma) < \lfloor \frac{h}{2} \rfloor$. Since $\mathrm{lst}(\rho) = \mathrm{lst}(\rho')$, $N_p(\rho') = N_p(\rho)$, and $N_\emptyset(\rho') \geq h$, there exists a proper prefix σ' of ρ' such that $\mathrm{lst}(\sigma') = \mathrm{lst}(\sigma)$, $N_p(\sigma') = N_p(\sigma)$, and $N_\emptyset(\sigma') \geq \lfloor \frac{h}{2} \rfloor$. Hence $(\sigma, \sigma') \in \tilde{\mathcal{R}}(\lfloor \frac{h}{2} \rfloor)$.

Thus, in all the cases, (1.) holds.

(2.) Let $(\rho, \rho') \in \tilde{\mathcal{R}}(h)$ and σ be a non-empty trace such that $\rho \cdot \sigma$ is a trace. We distinguish the following cases:

1. $D_p(\rho) < h$. Since $(\rho, \rho') \in \tilde{\mathcal{R}}(h)$, we have that $D_p(\rho') = D_p(\rho)$, $N_p(\rho) = N_p(\rho')$, and either $N_\emptyset(\rho') = N_\emptyset(\rho)$, or $N_\emptyset(\rho) \geq h$ and $N_\emptyset(\rho') \geq h$. Hence, $\mathrm{lst}(\rho) = \mathrm{lst}(\rho')$ and, taking $\sigma' = \sigma$, we get that $(\rho \cdot \sigma, \rho' \cdot \sigma') \in \tilde{\mathcal{R}}(h) \subseteq \tilde{\mathcal{R}}(\lfloor \frac{h}{2} \rfloor)$.
2. $D_p(\rho) \geq h$ and $D_p(\sigma) < \lfloor \frac{h}{2} \rfloor$. It follows that $N_\emptyset(\rho \cdot \sigma) \geq \lfloor \frac{h}{2} \rfloor$. Since $D_p(\rho') \geq h$, there exists a trace of the form $\rho' \cdot \sigma'$ such that $D_p(\rho' \cdot \sigma') = D_p(\rho \cdot \sigma)$, $N_p(\rho' \cdot \sigma') = N_p(\rho \cdot \sigma)$, and $N_\emptyset(\rho' \cdot \sigma') \geq \lfloor \frac{h}{2} \rfloor$. Hence, $(\rho \cdot \sigma, \rho' \cdot \sigma') \in \tilde{\mathcal{R}}(\lfloor \frac{h}{2} \rfloor)$.
3. $D_p(\rho) \geq h$ and $D_p(\sigma) \geq \lfloor \frac{h}{2} \rfloor$. Thus $D_p(\rho') \geq h$. If $N_\emptyset(\rho \cdot \sigma) < \lfloor \frac{h}{2} \rfloor$, then $N_\emptyset(\rho) = N_\emptyset(\rho')$. Therefore, there exists a trace of the form $\rho' \cdot \sigma'$ such that $N_\emptyset(\rho' \cdot \sigma') = N_\emptyset(\rho \cdot \sigma)$ and $D_p(\sigma') \geq \lfloor \frac{h}{2} \rfloor$. Otherwise, $N_\emptyset(\rho \cdot \sigma) \geq \lfloor \frac{h}{2} \rfloor$ and there exists a trace of the form $\rho' \cdot \sigma'$ such that $N_\emptyset(\rho' \cdot \sigma') \geq \lfloor \frac{h}{2} \rfloor$ and $D_p(\sigma') = \lfloor \frac{h}{2} \rfloor$. In both cases, $(\rho \cdot \sigma, \rho' \cdot \sigma') \in \tilde{\mathcal{R}}(\lfloor \frac{h}{2} \rfloor)$.

Thus, (2.) holds. □

By exploiting the previous lemma, we can prove the following one.

Lemma D.5.5. *Let $n \in \mathbb{N}$, ψ be a balanced HS_{st} formula with $|\psi| \leq n$, and $(\rho, \rho') \in \tilde{\mathcal{R}}(|\psi|)$. Then $\mathcal{K}_n, \rho \models \psi$ if and only if $\mathcal{K}_n, \rho' \models \psi$.*

Proof. The proof is by induction on $|\psi|$. The cases for the Boolean connectives directly follow from the inductive hypothesis and the fact that $\tilde{\mathcal{R}}(h) \subseteq \tilde{\mathcal{R}}(k)$, for all $h, k \in [1, n]$ with $h \geq k$.

As for the other cases, we proceed as follows:

- $\psi = p$. Since $(\rho, \rho') \in \tilde{\mathcal{R}}(1)$, that is, either $N_\emptyset(\rho) = N_\emptyset(\rho') = 0$ or both $N_\emptyset(\rho) \geq 1$ and $N_\emptyset(\rho') \geq 1$, ρ visits a state where p does not hold if and only if ρ' visits a state where p does not hold, which proves the thesis.

- $\psi = \langle \mathrm{B} \rangle\, \theta$ (resp., $\psi = \langle \overline{\mathrm{B}} \rangle\, \theta$). Since ψ is balanced, θ has the form $\theta = \theta_1 \wedge \theta_2$, with $|\theta_1| = |\theta_2|$. Hence $|\theta_1|, |\theta_2| \leq \lfloor \frac{|\psi|}{2} \rfloor$. We focus on the case $\psi = \langle \mathrm{B} \rangle\, \theta$. Since $\tilde{\mathcal{R}}(|\psi|)$ is an equivalence relation, by symmetry it suffices to show that $\mathcal{K}_n, \rho \models \psi$ implies $\mathcal{K}_n, \rho' \models \psi$. If $\mathcal{K}_n, \rho \models \psi$, then there exists a proper prefix σ of ρ such that $\mathcal{K}_n, \sigma \models \theta_i$, for $i = 1, 2$. Since $(\rho, \rho') \in \tilde{\mathcal{R}}(|\psi|)$, by (1.) of Lemma D.5.4, there exists a proper prefix σ' of ρ' such that $(\sigma, \sigma') \in \tilde{\mathcal{R}}(\lfloor \frac{|\psi|}{2} \rfloor)$. Since $\tilde{\mathcal{R}}(\lfloor \frac{|\psi|}{2} \rfloor) \subseteq \tilde{\mathcal{R}}(|\theta_i|)$, for $i = 1, 2$, by the inductive hypothesis we get that $\mathcal{K}_n, \sigma' \models \theta_i$, for $i = 1, 2$, thus proving that $\mathcal{K}_n, \rho' \models \psi$. The case for $\psi = \langle \overline{\mathrm{B}} \rangle\, \theta$ can be dealt with similarly by exploiting (2.) of Lemma D.5.4.
- $\psi = \langle \mathrm{E} \rangle\, \theta$ (resp., $\psi = \langle \overline{\mathrm{E}} \rangle\, \theta$). We proceed as in the previous case by applying (3.) of Lemma D.5.4 (resp., (4.) of Lemma D.5.4) and the inductive hypothesis. □

We can finally prove Lemma 6.4.4.

Lemma (6.4.4). *For all $n \in \mathbb{N}^+$ and balanced $\mathrm{HS}_{\mathrm{st}}$ formulas ψ, with $|\psi| \leq n$, it holds that $\mathcal{K}_n \models_{\mathrm{st}} \psi$ if and only if $\mathcal{M}_n \models_{\mathrm{st}} \psi$.*

Proof. First, let us assume that $\mathcal{K}_n \not\models_{\mathrm{st}} \psi$. Then, there exists an initial trace ρ of $\mathcal{K}_n$ such that $\mathcal{K}_n, \rho \not\models_{\mathrm{st}} \psi$. By Proposition D.5.3, there exists a trace ρ' of $\mathcal{K}_n$, which is an initial trace for $\mathcal{M}_n$, such that $(\rho, \rho') \in \tilde{\mathcal{R}}(|\psi|)$. By Lemma D.5.5, we have that $\mathcal{K}_n, \rho' \not\models_{\mathrm{st}} \psi$. Since for any trace σ and any $\mathrm{HS}_{\mathrm{st}}$ formula φ, we have that $\mathcal{K}_n, \sigma \models_{\mathrm{st}} \varphi$ if and only if $\mathcal{M}_n, \sigma \models_{\mathrm{st}} \varphi$ ($\mathcal{K}_n$ and $\mathcal{M}_n$ feature exactly the same set of traces with exactly the same labeling; they only differ in the initial state), we can conclude that $\mathcal{M}_n, \rho' \not\models_{\mathrm{st}} \psi$, and thus $\mathcal{M}_n \not\models_{\mathrm{st}} \psi$.

Let us now assume that $\mathcal{M}_n \not\models_{\mathrm{st}} \psi$. Then there exists an initial trace ρ of $\mathcal{M}_n$ such that $\mathcal{M}_n, \rho \not\models_{\mathrm{st}} \psi$. As in the converse direction, we have $\mathcal{K}_n, \rho \not\models_{\mathrm{st}} \psi$, and, by Proposition D.5.3, we can easily find an initial trace ρ' of $\mathcal{K}_n$ such that $(\rho, \rho') \in \tilde{\mathcal{R}}(|\psi|)$. By Lemma D.5.5 we can conclude that $\mathcal{K}_n \not\models_{\mathrm{st}} \psi$. □



Proofs and complements of Chapter 7

Contents

E.1 Completion of the proof of Proposition 7.2.3 279
E.2 Proof of Proposition 7.3.5 280
E.3 Proof of Proposition 7.3.6 281
E.4 Proof of Lemma 7.3.11 282
E.5 Pseudocode of `checkFalse` 283
E.6 Proof of Proposition 7.3.15 283
E.7 Proof of Theorem 7.3.16 290
E.7.1 TM alternation problem 290
E.7.2 $\mathrm{AEXP}_{\mathrm{pol}}$-hardness of the alternating multi-tiling problem . . . 294
E.8 Proof of Proposition 7.4.2 298
E.9 Proof of Theorem 7.4.3 299
E.10 Proof of Theorem 7.4.5 301
E.11 Proof of Theorem 7.4.6 304
E.12 Proof of Theorem 7.4.9 304

E.1 Completion of the proof of Proposition 7.2.3

Language $\langle \mathrm{E} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$. Let us consider the NFA $\mathcal{A}_{\langle \mathrm{E} \rangle}$ over S given by

$$\mathcal{A}_{\langle \mathrm{E} \rangle} = (S, (M \cup \{q_0'\}) \times S, \{q_0'\} \times S, \Delta', F),$$

where $q_0' \notin M$ is a fresh main state, and for all $(q,s) \in (M \cup \{q_0'\}) \times S$ and $s' \in S$, we have $\Delta'((q,s),s') = \emptyset$, if $s' \neq s$, and

$$\Delta((q,s),s) = \begin{cases} \Delta((q,s),s) & \text{if } q \neq q_0' \\ (\{q_0'\} \times R(s)) \cup \{(q_0,s') \in Q_0 \mid s' \in R(s)\} & \text{otherwise.} \end{cases}$$

Starting from an initial state (q_0',s), the automaton $\mathcal{A}_{\langle \mathrm{E} \rangle}$ either remains in a state whose main component is q_0', or moves to an initial state (q_0,s') of $\mathcal{A}$, ensuring at the same time that the portion of the input read so far is faithful to the evolution of $\mathcal{K}$. From the state (q_0,s'), $\mathcal{A}_{\langle \mathrm{E} \rangle}$ simulates the behavior of $\mathcal{A}$. Formally, since $\mathcal{A}$ is a $\mathcal{K}$-NFA, by construction it easily follows that $\mathcal{A}_{\langle \mathrm{E} \rangle}$ is a $\mathcal{K}$-NFA which accepts the set of traces of $\mathcal{K}$ having a non-empty proper suffix in $\mathcal{L}(\mathcal{A})$. Hence, $\mathcal{L}(\mathcal{A}_{\langle \mathrm{E} \rangle}) = \langle \mathrm{E} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$.

Language $\langle \overline{\mathrm{E}} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$. Let us consider the NFA $\mathcal{A}_{\langle \overline{\mathrm{E}} \rangle}$ over S given by

$$\mathcal{A}_{\langle \overline{\mathrm{E}} \rangle} = (S, (M \cup \{q_{acc}\}) \times S, Q_0', \Delta', \{q_{acc}\} \times S),$$

where $q_{acc} \notin M$ is a fresh main state, and Q_0' and Δ' are defined as follows:

- the set Q_0' of initial states is the set of states (q,s) of $\mathcal{A}$ such that there is a run of $\mathcal{A}$ from some initial state to (q,s) over some non-empty word.
- For all $(q,s) \in (M \cup \{q_{acc}\}) \times S$ and $s' \in S$, we have $\Delta'((q,s),s') = \emptyset$, if $s' \neq s$, and

$$\Delta'((q,s),s) = \begin{cases} \Delta((q,s),s) \cup \bigcup_{(q',s') \in F \cap \Delta((q,s),s)} \{(q_{acc},s')\} & \text{if } q \in M \\ \emptyset & \text{if } q = q_{acc}. \end{cases}$$

Note that the set Q_0' can be computed in time polynomial in the size of $\mathcal{A}$. Since $\mathcal{A}_{\langle \overline{\mathrm{E}} \rangle}$ essentially simulates $\mathcal{A}$, and $\mathcal{A}$ is a $\mathcal{K}$-NFA, by construction we easily obtain that $\mathcal{A}_{\langle \overline{\mathrm{E}} \rangle}$ is a $\mathcal{K}$-NFA which accepts the set of words over S which are non-empty proper suffixes of words in $\mathcal{L}(\mathcal{A})$. Thus, since $\mathcal{A}$ is a $\mathcal{K}$-NFA, we obtain that $\mathcal{L}(\mathcal{A}_{\langle \overline{\mathrm{E}} \rangle}) = \langle \overline{\mathrm{E}} \rangle_{\mathcal{K}}(\mathcal{L}(\mathcal{A}))$. □

E.2 Proof of Proposition 7.3.5

Proposition (7.3.5). *Let $h \geq 0$, and ρ and ρ' be two h-prefix bisimilar traces of $\mathcal{K}$. Then, for all traces ρ_L and ρ_R of $\mathcal{K}$ such that $\rho_L \star \rho$ and $\rho \star \rho_R$ are defined, the following properties hold:*

1. *$\rho_L \star \rho$ and $\rho_L \star \rho'$ are h-prefix bisimilar;*
2. *$\rho \star \rho_R$ and $\rho' \star \rho_R$ are h-prefix bisimilar.*

Proof. Let us note first that, since $\mathcal{S}(\rho) = \mathcal{S}(\rho')$, we have $\mathrm{fst}(\rho) = \mathrm{fst}(\rho')$ and $\mathrm{lst}(\rho) = \mathrm{lst}(\rho')$. Hence $\rho_L \star \rho$ (resp., $\rho \star \rho_R$) is defined if and only if $\rho_L \star \rho'$ (resp., $\rho' \star \rho_R$) is defined. The proofs of (1.) and (2.) are by induction on $h \geq 0$.

(1.) Since ρ and ρ' are h-prefix bisimilar, $\mathcal{S}(\rho) = \mathcal{S}(\rho')$. By Proposition 7.3.2, $\mathcal{S}(\rho_L \star \rho) = \mathcal{S}(\rho_L \star \rho')$. Thus, if $h = 0$ (base case), the thesis follows. Now let $h > 0$ (inductive step). Let us assume that ν is a proper prefix of $\rho_L \star \rho$ (the symmetric case, where we consider a proper prefix of $\rho_L \star \rho'$ is similar). We need to show that there exists a proper prefix ν' of $\rho_L \star \rho'$ such that ν and ν' are $(h-1)$-prefix bisimilar. If ν is a prefix of ρ_L, then we set $\nu' = \nu$ and the result trivially follows (note that, since ρ and ρ' are h-prefix bisimilar, it holds that $|\rho| > 1$ if and only if $|\rho'| > 1$). Otherwise, there is a proper prefix ξ of ρ such that $\nu = \rho_L \star \xi$. Since ρ and ρ' are h-prefix bisimilar, there exists a proper prefix ξ' of ρ' such that ξ and ξ' are $(h-1)$-prefix bisimilar. Thus, by setting $\nu' = \rho_L \star \xi'$, by the inductive hypothesis the thesis follows.

(2.) By Proposition 7.3.2, $\mathcal{S}(\rho \star \rho_R) = \mathcal{S}(\rho' \star \rho_R)$. Thus, if $h = 0$, the thesis follows. Now, let us assume that $h > 0$. We proceed by a double induction on $|\rho_R|$. As for the base case, where $|\rho_R| = 1$, the result is obvious. Thus let us assume that $|\rho_R| > 1$. Let ν be a proper prefix of $\rho \star \rho_R$ (the symmetric case, where we consider a proper prefix of $\rho' \star \rho_R$ is similar). We need to show that there exists a proper prefix ν' of $\rho' \star \rho_R$ such that ν and ν' are $(h-1)$-prefix bisimilar. If $\nu = \rho$ or ν is a proper prefix of ρ, then there exists a prefix ν' of ρ' such that ν and ν' are $(h-1)$-prefix bisimilar. Thus, since ν' is a proper prefix of $\rho' \star \rho_R$, the result follows. Otherwise, there exists a proper prefix ξ of ρ_R such that $\nu = \rho \star \xi$. By setting $\nu' = \rho' \star \xi$, and considering the inductive hypothesis on $|\rho_R|$, we obtain that ν and ν' are h-prefix bisimilar, hence $(h-1)$-prefix bisimilar as well, concluding the proof. □

E.3 Proof of Proposition 7.3.6

Proposition (7.3.6). *Let $h \geq 0$, and ρ and ρ' be two h-prefix bisimilar traces of $\mathcal{K}$. Then, for each $\mathsf{AA\overline{B}B\overline{E}}$ formula ψ over Spec with $\mathrm{Nest}_{\mathrm{B}}(\psi) \leq h$, we have that*

$$\mathcal{K}, \rho \models \psi \iff \mathcal{K}, \rho' \models \psi.$$

Proof. We prove the proposition by a nested induction on the structure of the formula ψ and on the B-nesting depth $\mathrm{Nest}_{\mathrm{B}}(\psi)$.

As for the base case, ψ is a regular expression in Spec. Since $\mathcal{S}(\rho) = \mathcal{S}(\rho')$ (as ρ and ρ' are h-prefix bisimilar) the thesis holds by Proposition 7.3.2.

Let us now consider the inductive step. The cases where the root modality of ψ is a Boolean connective directly follow by the inductive hypothesis. As for the

cases where the root modality is $\langle \mathrm{A} \rangle$ or $\langle \overline{\mathrm{A}} \rangle$, the result follows from the fact that, being ρ and ρ' h-prefix bisimilar, we have $\mathrm{fst}(\rho) = \mathrm{fst}(\rho')$ and $\mathrm{lst}(\rho) = \mathrm{lst}(\rho')$. It remains to consider the cases where the root modality is $\langle \mathrm{B} \rangle$, $\langle \overline{\mathrm{B}} \rangle$, or $\langle \overline{\mathrm{E}} \rangle$. We prove the implication $\mathcal{K}, \rho \models \psi \implies \mathcal{K}, \rho' \models \psi$ (the converse is similar). Let $\mathcal{K}, \rho \models \psi$.

- $\psi = \langle \mathrm{B} \rangle \varphi$: since $0 < \mathrm{Nest}_{\mathrm{B}}(\psi) \leq h$, it holds that $h > 0$. As $\mathcal{K}, \rho \models \langle \mathrm{B} \rangle \varphi$, there is a proper prefix ν of ρ such that $\mathcal{K}, \nu \models \varphi$. Since ρ and ρ' are h-prefix bisimilar, there is a proper prefix ν' of ρ' such that ν and ν' are $(h-1)$-prefix bisimilar. Being $\mathrm{Nest}_{\mathrm{B}}(\varphi) \leq h - 1$, by the inductive hypothesis we obtain that $\mathcal{K}, \nu' \models \varphi$. Hence, $\mathcal{K}, \rho' \models \langle \mathrm{B} \rangle \varphi$.
- $\psi = \langle \overline{\mathrm{B}} \rangle \varphi$: since $\mathcal{K}, \rho \models \langle \overline{\mathrm{B}} \rangle \varphi$, there is a trace ρ_R such that $|\rho_R| > 1$ and $\mathcal{K}, \rho \star \rho_R \models \varphi$. By Proposition 7.3.5, $\rho \star \rho_R$ and $\rho' \star \rho_R$ are h-prefix bisimilar. By the inductive hypothesis on the structure of the formula, we obtain that $\mathcal{K}, \rho' \star \rho_R \models \varphi$. Hence, $\mathcal{K}, \rho' \models \langle \overline{\mathrm{B}} \rangle \varphi$.
- $\psi = \langle \overline{\mathrm{E}} \rangle \varphi$: this case is similar to the previous one. □

E.4 Proof of Lemma 7.3.11

Lemma (7.3.11). *For $h \geq 0$, any two traces ρ and ρ' of $\mathcal{K}$ having the same h-sampling word are h-prefix bisimilar.*

Proof. The proof can be immediately derived by a stronger result stated in the following claim.

Claim E.4.1. Let $h \geq 0$, ρ and ρ' be two traces of $\mathcal{K}$, and PS_h and PS'_h be the two h-prefix samplings of ρ and ρ', resp.. Assume that ρ and ρ' have the same h-sampling word, namely there is $N \geq 1$ such that

- $PS_h : i_1 < i_2 < \ldots < i_N$,
- $PS'_h : i'_1 < i'_2 < \ldots < i'_N$, and
- for all $j \in [1, N]$, $\mathcal{S}(\rho(1, i_j)) = \mathcal{S}(\rho'(1, i'_j))$.

Then, for all $j \in [1, N-1]$, $n \in [i_j + 1, i_{j+1}]$ and $n' \in [i'_j + 1, i'_{j+1}]$ such that $\mathcal{S}(\rho[1, n]) = \mathcal{S}(\rho'[1, n'])$, it holds that $\rho(1, n)$ and $\rho'(1, n')$ are h-prefix bisimilar.

Proof. The proof is by induction on $h \geq 0$. For $h = 0$, the result is obvious. Now let us assume that $h > 0$. If $N = 1$ (resp., $N = 2$), then $\rho = \rho'$ and $|\rho| = |\rho'| = N$, and the thesis trivially holds. Now, let us assume that $N > 2$. Since by hypothesis $\mathcal{S}(\rho(1, n)) = \mathcal{S}(\rho'(1, n'))$, we need to show that:

1. for each $m \in [1, n-1]$, there exists $m' \in [1, n'-1]$ such that $\rho(1, m)$ and $\rho'(1, m')$ are $(h-1)$-prefix bisimilar;
2. for each $m' \in [1, n'-1]$, there exists $m \in [1, n-1]$ such that $\rho(1, m)$ and $\rho'(1, m')$ are $(h-1)$-prefix bisimilar;

We only prove (1.), being the proof of (2.) symmetric. We exploit in the proof the following fact that can be easily shown: let $k \in [0, h-1]$ and $1 = x_1 < \ldots < x_r = N$ be the subsequence of $1, \ldots, N$ such that $i_{x_1} < \ldots < i_{x_r}$ is the k-prefix sampling of ρ. Then, $i'_{x_1} < \ldots < i'_{x_r}$ is the k-prefix sampling of ρ'.

Now we prove (1.). Let $m \in [1, n-1]$. If $m = 1$, we set $m' = 1$, and the result follows. Now, let us assume that $m \geq 2$. Since $h > 0$, there must exist $x, y \in [1, N]$ such that $x < y$, $m \in [i_x + 1, i_y]$, and i_x and i_y are two consecutive positions in the $(h-1)$-prefix sampling of ρ. By the fact above, i'_x and i'_y are two consecutive positions in the $(h-1)$-prefix sampling of ρ'. We distinguish two cases:

- $m = i_y$. Since $n \in [i_j + 1, i_{j+1}]$ and $m < n$, it holds that $i_y \leq i_j$. Hence, $i'_y \leq i'_j$ as well. Moreover, since $n' > i'_j$, it holds that $i'_y < n'$. We set $m' = i'_y$. As $\mathcal{S}(\rho(1, i_y)) = \mathcal{S}(\rho'(1, i'_y))$, $m = i_y$, $m' = i'_y$, and i_x and i_y (resp., i'_x and i'_y) are two consecutive positions in the $(h-1)$-prefix sampling of ρ (resp., ρ'), the thesis follows by the inductive hypothesis on h.
- $m \neq i_y$. Hence, $m \in [i_x + 1, i_y - 1]$. Since i_x and i_y are two consecutive positions in the $(h-1)$-prefix sampling of ρ, there must exist $z \in [x+1, y-1]$ such that $i_z \leq m$ and $\mathcal{S}(\rho(1, m)) = \mathcal{S}(\rho(1, i_z))$. Since $i_z \leq m$, $m < n$, and $n \in [i_j + 1, i_{j+1}]$, it holds that $i_z \leq i_j$. Hence, $i'_z \leq i'_j < n'$. We set $m' = i'_z$. As $\mathcal{S}(\rho(1, i_z)) = \mathcal{S}(\rho'(1, i'_z))$, we obtain that $\mathcal{S}(\rho(1, m)) = \mathcal{S}(\rho'(1, m'))$, $m \in [i_x + 1, i_y]$ and $m' \in [i'_x + 1, i'_y]$. Thus, being i_x and i_y (resp., i'_x and i'_y) two consecutive positions in the $(h-1)$-prefix sampling of ρ (resp., ρ'), by the inductive hypothesis on h the result follows. □

This concludes the proof of the claim, and of Lemma 7.3.11 as well. □

E.5 Pseudocode of checkFalse

The pseudocode of procedure checkFalse, the "dual" of checkTrue, is reported in Algorithm E.1 at page 284.

E.6 Proof of Proposition 7.3.15

For technical convenience, given an $\mathrm{A\overline{A}B\overline{B}E}$ formula φ, we consider a slight variant $\Upsilon_w(\varphi)$ of $\Upsilon(\varphi)$. Formally, $\Upsilon_w(\varphi)$ is defined as $\Upsilon(\langle \overline{\mathrm{B}} \rangle \varphi)$ (or, equivalently, as $\Upsilon(\langle \overline{\mathrm{E}} \rangle \varphi)$). Note that for each $\mathrm{A\overline{A}B\overline{B}E}$ formula φ and $X \in \{\overline{E}, \overline{B}\}$, we have $\Upsilon_w([\mathrm{X}]\,\varphi) = \Upsilon_w(\widetilde{[\mathrm{X}]\,\varphi}) + 1$.

Algorithm E.1 $\texttt{checkFalse}_{(\mathcal{K},\varphi,Lab)}(\mathcal{W})$

while $\mathcal{W}$ is *not* universal **do**
  deterministically select $(\psi, \rho) \in \mathcal{W}$ s.t. ψ is not of the form $[\overline{\mathrm{E}}]\,\psi'$ and $[\overline{\mathrm{B}}]\,\psi'$
  $\mathcal{W} \leftarrow \mathcal{W} \setminus \{(\psi, \rho)\}$
  Case $\psi = r$ with $r \in \mathsf{RE}$
    if $\rho \notin \mathcal{L}(r)$ **then**
      accept the input
  case $\psi = \neg r$ with $r \in \mathsf{RE}$
    if $\rho \in \mathcal{L}(r)$ **then**
      accept the input
  case $\psi = \langle \mathrm{A} \rangle\,\psi'$ or $\psi = [\mathrm{A}]\,\psi'$
    if $\psi \notin Lab(\mathrm{lst}(\rho))$ **then**
      accept the input
  case $\psi = \langle \overline{\mathrm{A}} \rangle\,\psi'$ or $\psi = [\overline{\mathrm{A}}]\,\psi'$
    if $\psi \notin Lab(\mathrm{fst}(\rho))$ **then**
      accept the input
  case $\psi = \psi_1 \vee \psi_2$
    universally choose $i = 1, 2$
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\psi_i, \rho)\}$
  case $\psi = \psi_1 \wedge \psi_2$
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\psi_1, \rho), (\psi_2, \rho)\}$
  case $\psi = \langle \mathrm{B} \rangle\,\psi'$
    universally choose $\rho' \in \mathrm{Pref}(\rho)$
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\psi', \rho')\}$
  case $\psi = [\mathrm{B}]\,\psi'$
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\psi', \rho') \mid \rho' \in \mathrm{Pref}(\rho)\}$
  case $\psi = \langle \mathrm{X} \rangle\,\psi'$ with $X \in \{\overline{E}, \overline{B}\}$
    universally choose an X-*witness* ρ' of ρ for $(\mathcal{K}, \varphi)$
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\psi', \rho')\}$
  EndCase
if $\mathcal{W} = \emptyset$ **then**
  reject the input
else
  existentially choose $(\psi, \rho) \in \widetilde{\mathcal{W}}$
  $\texttt{checkTrue}_{(\mathcal{K},\varphi,Lab)}(\{(\psi, \rho)\})$

Let $\mathcal{K}$ be a finite Kripke structure, φ be an $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}\mathsf{E}$ formula in NNF, and $\mathcal{W}$ be a well-formed set for $(\mathcal{K}, \varphi)$. We denote by $\Upsilon_w(\mathcal{W})$ the maximum over the alternation depths $\Upsilon_w(\psi)$, where ψ is a formula occurring in $\mathcal{W}$ (we set $\Upsilon_w(\mathcal{W}) = 0$ if $\mathcal{W} = \emptyset$). For each non-empty *universal* well-formed set $\mathcal{W}$ for $(\mathcal{K}, \varphi)$, we have $\Upsilon_w(\widetilde{\mathcal{W}}) = \Upsilon_w(\mathcal{W}) - 1$.

Now, we can prove Proposition 7.3.15.

Proposition (7.3.15)**.** *The ATM* `check` *is a singly exponential-time bounded ATM accepting* FMC *whose number of alternations on input* $(\mathcal{K}, \varphi)$ *is at most* $\Upsilon(\varphi) + 2$.

Proof. Let us fix an input $(\mathcal{K}, \varphi)$, where φ is an $\mathsf{A}\overline{\mathsf{A}}\mathsf{B}\overline{\mathsf{B}}\mathsf{E}$ formula in NNF.

Note that whenever there is a switch between the procedures `checkTrue` and `checkFalse`, e.g., from `checkTrue` to `checkFalse`, (*i*) the input $\{(\psi, \rho)\}$ of the called procedure is contained in the dual $\widetilde{\mathcal{W}}$ of the currently processed well-formed set $\mathcal{W}$ for $(\mathcal{K}, \varphi)$, and (*ii*) $\mathcal{W}$ is non-empty and universal, hence $\Upsilon_w(\{(\psi, \rho)\}) < \Upsilon_w(\mathcal{W})$. Moreover, a well-formed set $\mathcal{W}$ for $(\mathcal{K}, \varphi)$ contains only formulas ψ such that $\psi \in \mathsf{SD}(\varphi)$.

Additionally, in each iteration of the while loops of procedures `checkTrue` and `checkFalse`, the processed pair (ψ, ρ) in the current well-formed set $\mathcal{W}$ is either removed from $\mathcal{W}$, or it is replaced with pairs (ψ', ρ') such that ψ' is a strict subformula of ψ. This ensures that the algorithm always terminates.

Furthermore, since the number of alternations of the ATM `check` between existential choices and universal choices is evidently the number of switches between the calls to procedures `checkTrue` and `checkFalse` plus 2, and the top calls to `checkTrue` take as input well-formed sets for $(\mathcal{K}, \varphi)$ having the form $\{(\psi, \rho)\}$, where $\psi \in \mathsf{SD}(\varphi)$, we have proved the following result.

Claim E.6.1. The number of alternations of the ATM `check` on input $(\mathcal{K}, \varphi)$ is at most $\Upsilon(\varphi) + 2$.

Next, we prove the following property.

Claim E.6.2. The ATM `check` runs in time singly exponential in the size of the input $(\mathcal{K}, \varphi)$.

Proof. Let us fix an input $(\mathcal{K}, \varphi)$. Let $T(\varphi)$ be the standard tree encoding of φ, where each node is labeled by some subformula of φ. Let $\psi \in \mathsf{SD}(\varphi)$. If ψ is a subformula of φ, we define d_ψ as the maximum over the distances from the root in $T(\varphi)$ of ψ-labeled nodes. If, conversely, ψ is the dual of a subformula of φ, we let $d_\psi = d_{\widetilde{\psi}}$.

Let us denote by $H(\mathcal{K},\varphi)$ the length of a certificate for $(\mathcal{K},\varphi)$. Recall that $H(\mathcal{K},\varphi) = (|S| \cdot 2^{(2|\mathrm{Spec}|)^2})^{h+2}$, where S is the set of states of $\mathcal{K}$, Spec is the set of atomic formulas (regular expressions) occurring in φ, and $h = \mathrm{Nest}_{\mathrm{B}}(\varphi)$.

By Proposition 7.3.14, it follows that each step in an iteration of the while loops in the procedures `checkTrue` and `checkFalse` can be performed in time singly exponential in the size of $(\mathcal{K},\varphi)$. Thus, in order to prove Claim E.6.2, it suffices to show that for all computations π of the ATM `check` starting from the input $(\mathcal{K},\varphi)$, the overall number N_ψ of iterations of the while loops (of procedures `checkTrue` and `checkFalse`) along π, where the formula ψ is processed, is at most $(2^{|\varphi|} \cdot H(\mathcal{K},\varphi))^{d_\psi}$.

The proof is done by induction on d_ψ. As for the base case, we have $d_\psi = 0$. Therefore, $\psi = \varphi$ or $\psi = \widetilde{\varphi}$; by construction of the algorithm, N_φ and $N_{\widetilde{\varphi}}$ are at most equal to 1. Thus the result holds.

As for the inductive step, let us assume that $d_\psi > 0$. We consider the case where ψ is a subformula of φ (the case where $\widetilde{\psi}$ is a subformula of φ is similar). Then, the result follows from the next sequence of inequalities, where $P(\psi)$ denotes the set of nodes of $T(\varphi)$ which are parents of the nodes labeled by ψ, and for each node x, $fo(x)$ denotes the formula labeling x.

$$N_\psi \leq \sum_{x \in P(\psi)} N_{fo(x)} \cdot H(\mathcal{K},\varphi) \leq$$

$$\leq \sum_{x \in P(\psi)} (2^{|\varphi|} \cdot H(\mathcal{K},\varphi))^{d_{fo(x)}} \cdot H(\mathcal{K},\varphi) \leq \left(2^{|\varphi|} \cdot H(\mathcal{K},\varphi)\right)^{d_\psi}.$$

The first inequality directly follows from the construction of the algorithm (note that if $fo(x) = [\mathrm{B}]\,\psi$, the processing of the subformula $fo(x)$ in an iteration of the two while loops generates at most $H(\mathcal{K},\varphi)$ new "copies" of ψ). The second inequality follows by the inductive hypothesis, and the last one from the fact that $|P(\psi)| \leq 2^{|\varphi|}$ and $d_{fo(x)} \leq d_\psi - 1$ for all $x \in P(\psi)$. This concludes the proof of Claim E.6.2. □

It remains to show that the ATM `check` accepts FMC. Let us fix an input $(\mathcal{K},\varphi)$ and let Lab be the $\mathrm{A}\overline{\mathrm{A}}$-labeling initially and existentially guessed by `check` (at line 1). Evidently, after the top calls to `checkTrue`, each configuration of the procedure `check` can be described by a tuple $(\ell, Lab, \mathcal{W}, f)$, where:

- $\mathcal{W}$ is a well-formed set for $(\mathcal{K},\varphi)$,
- $f = \top$ if $\mathcal{W}$ is processed within `checkTrue`, and $f = \bot$ otherwise, and
- ℓ is an instruction label corresponding to one of the instructions of the procedures `checkTrue` and `checkFalse`.

We denote by ℓ_0 the label associated with the while instruction. A *main configuration* is a configuration having label ℓ_0.

Let $Lab_{\mathcal{W}}$ be the restriction of Lab to the set of formulas in $\overline{\mathrm{AA}}(\varphi)$ which are subformulas of formulas occurring in $\mathcal{W}$. In other words, for each state s, $Lab_{\mathcal{W}}(s)$ contains all and only the formulas $\psi \in Lab(s)$ such that either ψ or its dual $\widetilde{\psi}$ is a subformula of some formula occurring in $\mathcal{W}$. $Lab_{\mathcal{W}}$ is said to be *valid* if, for all states s and $\psi \in Lab_{\mathcal{W}}(s)$, it holds $\mathcal{K}, s \models \psi$.

Claim E.6.3. Let $\mathcal{W}$ be a well-formed set for $(\mathcal{K}, \varphi)$ and let us assume that $Lab_{\mathcal{W}}$ is valid. Then

1. the main configuration $(\ell_0, Lab, \mathcal{W}, \top)$ leads to acceptance *if and only if* $\mathcal{W}$ is valid;
2. the main configuration $(\ell_0, Lab, \mathcal{W}, \bot)$ leads to acceptance *if and only if* $\mathcal{W}$ is *not* valid.

Proof. We associate with $\mathcal{W}$ a natural number $\|\mathcal{W}\|$ defined as follows. Let us fix an ordering $\psi_1, \ldots, \psi_k$ of the formulas in $\mathrm{SD}(\varphi)$ such that, for all $i \neq j$, $|\psi_i| > |\psi_j|$ implies $i < j$.

First, we associate with $\mathcal{W}$ a $(k+1)$-tuple $(n_0, n_1, \ldots, n_k)$ of natural numbers defined as: the first component n_0 in the tuple is the alternation depth $\Upsilon_w(\mathcal{W})$ and, for all the other components n_i, with $1 \leq i \leq k$, n_i is the number of elements of $\mathcal{W}$ associated with the formula ψ_i (i.e., the number of elements having the form (ψ_i, ρ)).

Then $\|\mathcal{W}\|$ is the position of the tuple $(n_0, n_1, \ldots, n_k)$ along the total lexicographic ordering over $\mathbb{N}^{k+1}$. Note that if $\mathcal{W}$ is non-empty and universal, since $\Upsilon_w(\widetilde{\mathcal{W}}) < \Upsilon_w(\mathcal{W})$, it holds that $\|\widetilde{\mathcal{W}}\| < \|\mathcal{W}\|$. Moreover, $\|\mathcal{W}\|$ strictly decreases at each iteration of the while loops in the procedures `checkTrue` and `checkFalse` (this is because at each iteration $\Upsilon_w(\mathcal{W})$ does not increase, and an element of $\mathcal{W}$ is replaced with elements associated with smaller formulas).

The proof of Claim E.6.3 is now carried out by induction on $\|\mathcal{W}\|$. As for the base case we have $\|\mathcal{W}\| = 0$, thus $\mathcal{W}$ is empty and clearly valid. By construction `checkTrue` accepts the empty set, while `checkFalse` rejects the empty set. The result holds.

As for the inductive step, let $\|\mathcal{W}\| > 0$, hence $\mathcal{W}$ is not empty. First, we assume that $\mathcal{W}$ is universal. Recall that $\|\widetilde{\mathcal{W}}\| < \|\mathcal{W}\|$. Thus,

- $\mathcal{W}$ is valid $\iff$ for each $(\psi, \rho) \in \widetilde{\mathcal{W}}$, $\{(\psi, \rho)\}$ is not valid $\iff$ (by the inductive hypothesis) for each $(\psi, \rho) \in \widetilde{\mathcal{W}}$, the main configuration $(\ell_0, Lab, \{(\psi, \rho)\}, \bot)$ leads to acceptance $\iff$ (by construction of the algorithm and since $\mathcal{W}$ is universal) the main configuration $(\ell_0, Lab, \mathcal{W}, \top)$ leads to acceptance.
- $\mathcal{W}$ is *not* valid $\iff$ for some $(\psi, \rho) \in \widetilde{\mathcal{W}}$, $\{(\psi, \rho)\}$ is valid $\iff$ (by the inductive hypothesis) for some $(\psi, \rho) \in \widetilde{\mathcal{W}}$, the main configuration $(\ell_0, Lab, \{(\psi, \rho)\}, \top)$ leads to acceptance $\iff$ (by construction of the algorithm

and since $\mathcal{W}$ is universal) the main configuration $(\ell_0, Lab, \mathcal{W}, \bot)$ leads to acceptance.

Hence, (1.) and (2.) of Claim E.6.3 hold if $\mathcal{W}$ is universal.

Now, let us assume that the non-empty set $\mathcal{W}$ is not universal. We consider (2.) of Claim E.6.3 (the proof of (1.) is just the "dual"). Let $(\psi, \rho) \in \mathcal{W}$ be the pair selected by the procedure `checkFalse` in the iteration of the while loop associated with the main configuration $(\ell_0, Lab, \mathcal{W}, \bot)$. Here we examine the cases where either $\psi = \langle \mathrm{A} \rangle \psi'$, or $\psi = [\mathrm{B}] \psi'$, or $\psi = \langle \mathrm{X} \rangle \psi'$ with $X \in \{\overline{B}, \overline{E}\}$ (the other cases are similar or simpler).

- $\psi = \langle \mathrm{A} \rangle \psi'$. We have that $\{(\langle \mathrm{A} \rangle \psi', \rho)\}$ is valid if and only if $\mathcal{K}, \mathrm{lst}(\rho) \models \langle \mathrm{A} \rangle \psi'$. By hypothesis, $Lab_{\mathcal{W}}$ is valid. Hence $\{(\langle \mathrm{A} \rangle \psi', \rho)\}$ is not valid if and only if $\langle \mathrm{A} \rangle \psi' \notin Lab_{\mathcal{W}}(\mathrm{lst}(\rho))$. Let $\mathcal{W}' = \mathcal{W} \setminus \{(\psi, \rho)\}$. Note that $\|\mathcal{W}'\| < \|\mathcal{W}\|$. Then $\mathcal{W}$ is not valid $\iff$ either $\langle \mathrm{A} \rangle \psi' \notin Lab_{\mathcal{W}}(\mathrm{lst}(\rho))$ or $\mathcal{W}'$ is not valid $\iff$ (by the inductive hypothesis) either $\langle \mathrm{A} \rangle \psi' \notin Lab_{\mathcal{W}}(\mathrm{lst}(\rho))$ or the main configuration $(\ell_0, Lab, \mathcal{W}', \bot)$ leads to acceptance $\iff$ (by construction of `checkFalse`) the main configuration $(\ell_0, Lab, \mathcal{W}, \bot)$ leads to acceptance.
- $\psi = [\mathrm{B}] \psi'$. Let $\mathcal{W}' = (\mathcal{W} \setminus \{(\psi, \rho)\}) \cup \{(\psi', \rho') \mid \rho' \in \mathrm{Pref}(\rho)\}$. Note that $\|\mathcal{W}'\| < \|\mathcal{W}\|$. Then $\mathcal{W}$ is not valid $\iff$ $\mathcal{W}'$ is not valid $\iff$ (by the inductive hypothesis) the main configuration $(\ell_0, Lab, \mathcal{W}', \bot)$ leads to acceptance $\iff$ (by construction of `checkFalse`) the main configuration $(\ell_0, Lab, \mathcal{W}, \bot)$ leads to acceptance.
- $\psi = \langle \mathrm{X} \rangle \psi'$ with $X \in \{\overline{B}, \overline{E}\}$. By Proposition 7.3.14(1), $\mathcal{K}, \rho \models \langle \mathrm{X} \rangle \psi'$ if and only if there exists an X-witness ρ' of ρ for $(\mathcal{K}, \varphi)$ such that $\mathcal{K}, \rho' \models \psi'$. Then (2.) of Claim E.6.3 directly follows from the next sequence of equivalences: $\mathcal{W}$ is not valid $\iff$ either $\mathcal{W} \setminus \{(\psi, \rho)\}$ is not valid, or for each X-witness ρ' of ρ for $(\mathcal{K}, \varphi)$, $\{(\psi', \rho')\}$ is not valid $\iff$ for each X-witness ρ' of ρ for $(\mathcal{K}, \varphi)$, $(\mathcal{W} \setminus \{(\psi, \rho)\}) \cup \{(\psi', \rho')\}$ is not valid $\iff$ (by the inductive hypothesis) for each X-witness ρ' of ρ for $(\mathcal{K}, \varphi)$, the main configuration $(\ell_0, Lab, (\mathcal{W} \setminus \{(\psi, \rho)\}) \cup \{(\psi', \rho')\}, \bot)$ leads to acceptance $\iff$ (by construction of the procedure `checkFalse`) the main configuration $(\ell_0, Lab, \mathcal{W}, \bot)$ leads to acceptance.

This concludes the proof of Claim E.6.3. □

By Claim E.6.3 we prove the next result, which finally concludes the proof of Proposition 7.3.15.

Claim E.6.4. The ATM `check` accepts an input $(\mathcal{K}, \varphi)$ if and only if $\mathcal{K} \models \varphi$.

Proof. Let us fix an input $(\mathcal{K}, \varphi)$ and an $\mathsf{A}\overline{\mathsf{A}}$-labeling Lab for $(\mathcal{K}, \varphi)$. A Lab-*guessing* for $(\mathcal{K}, \varphi)$ is a well-formed set $\mathcal{W}$ for $(\mathcal{K}, \varphi)$ which minimally satisfies the following conditions for all states s of $\mathcal{K}$:

- for all certificates ρ for $(\mathcal{K}, \varphi)$ with $\mathrm{fst}(\rho) = s_0$, $(\varphi, \rho) \in \mathcal{W}$;
- for all $\langle \mathsf{A} \rangle \psi \in Lab(s)$ (resp., $\langle \overline{\mathsf{A}} \rangle \psi \in Lab(s)$), there is a certificate ρ for $(\mathcal{K}, \varphi)$ with $\mathrm{fst}(\rho) = s$ (resp., $\mathrm{lst}(\rho) = s$) such that $(\psi, \rho) \in \mathcal{W}$;
- for all $[\mathsf{A}] \psi \in Lab(s)$ (resp., $[\overline{\mathsf{A}}] \psi \in Lab(s)$) and for all certificates ρ for $(\mathcal{K}, \varphi)$ with $\mathrm{fst}(\rho) = s$ (resp., $\mathrm{lst}(\rho) = s$), $(\psi, \rho) \in \mathcal{W}$.

Evidently, by construction of the procedure `check`, for each input $(\mathcal{K}, \varphi)$, it holds:

(*) `check` accepts $(\mathcal{K}, \varphi) \iff$ there are an $\mathsf{A}\overline{\mathsf{A}}$-labeling Lab and a Lab-guessing $\mathcal{W}$ for $(\mathcal{K}, \varphi)$ such that, for all $(\psi, \rho) \in \mathcal{W}$, the main configuration $(\ell_0, Lab, \{(\psi, \rho)\}, \top)$ leads to acceptance.

First we assume that $\mathcal{K} \models \varphi$. Let Lab be the *valid* $\mathsf{A}\overline{\mathsf{A}}$-labeling defined as follows for all states s: for all $\psi \in \mathsf{A}\overline{\mathsf{A}}(\varphi)$, $\psi \in Lab(s)$ if and only if $\mathcal{K}, s \models \psi$. By Theorem 4.2.12, there exists a Lab-guessing $\mathcal{W}$ for $(\mathcal{K}, \varphi)$ such that for all $(\psi, \rho) \in \mathcal{W}$, $\mathcal{K}, \rho \models \psi$. By Claim E.6.3, for all $(\psi, \rho) \in \mathcal{W}$, the main configuration $(\ell_0, Lab, \{(\psi, \rho)\}, \top)$ leads to acceptance. Hence, by (*), `check` accepts $(\mathcal{K}, \varphi)$.

For the converse direction, let us assume that `check` accepts $(\mathcal{K}, \varphi)$. By (*), there exist an $\mathsf{A}\overline{\mathsf{A}}$-labeling Lab and a Lab-guessing $\mathcal{W}$ for $(\mathcal{K}, \varphi)$ such that, for all $(\psi, \rho) \in \mathcal{W}$, the main configuration $(\ell_0, Lab, \{(\psi, \rho)\}, \top)$ leads to acceptance. First we show that Lab is valid.

We fix a state s and a formula $\psi \in Lab(s)$. We need to prove that $\mathcal{K}, s \models \psi$. The proof is by induction on the nesting depth of modalities $\langle \mathsf{A} \rangle$, $\langle \overline{\mathsf{A}} \rangle$, $[\mathsf{A}]$ and $[\overline{\mathsf{A}}]$ in ψ. Assume that $\psi = [\mathsf{A}] \psi'$ for some ψ' (the other cases, where either $\psi = \langle \mathsf{A} \rangle \psi'$, or $\psi = \langle \overline{\mathsf{A}} \rangle \psi'$, or $\psi = [\overline{\mathsf{A}}] \psi'$ are similar). By definition of Lab-guessing, for each certificate ρ for $(\mathcal{K}, \varphi)$ with $\mathrm{fst}(\rho) = s$, $(\psi', \rho) \in \mathcal{W}$. Moreover, by the inductive hypothesis, one can assume that $Lab_{\{(\psi', \rho)\}}$ is valid (note that for the base case, i.e., when ψ' does not contain occurrences of modalities $\langle \mathsf{A} \rangle$, $\langle \overline{\mathsf{A}} \rangle$, $[\mathsf{A}]$, and $[\overline{\mathsf{A}}]$, $Lab_{\{(\psi', \rho)\}}$ is trivially valid). By hypothesis, the main configuration $(\ell_0, Lab, \{(\psi', \rho)\}, \top)$ leads to acceptance. By Claim E.6.3, for each certificate ρ for $(\mathcal{K}, \varphi)$ with $\mathrm{fst}(\rho) = s$, it holds $\mathcal{K}, \rho \models \psi'$. Thus, by Theorem 4.2.12, we obtain that $\mathcal{K}, s \models \psi$. Hence Lab is valid.

Now, by definition of Lab-guessing, for each certificate ρ for $(\mathcal{K}, \varphi)$ with $\mathrm{fst}(\rho) = s_0$, $(\varphi, \rho) \in \mathcal{W}$. Thus, by hypothesis, by Claim E.6.3, and by Theorem 7.3.12, we have $\mathcal{K} \models \varphi$. This concludes the proof of the claim. □

Proposition 7.3.15 has been proved. □

E.7 Proof of Theorem 7.3.16

In this section, we show that

Theorem (7.3.16). *The alternating multi-tiling problem is* $\mathbf{AEXP}_{\mathbf{pol}}$*-complete*

Membership to $\mathbf{AEXP}_{\mathbf{pol}}$ is straightforward. As for $\mathbf{AEXP}_{\mathbf{pol}}$-hardness, we establish it in two steps. In the first, we focus on a variant of the considered problem, called *TM alternation problem*, which is defined in terms of multi-tape deterministic Turing machines, and we prove that it is $\mathbf{AEXP}_{\mathbf{pol}}$-hard. Then, in the second, we provide a polynomial-time reduction from the TM alternation problem to alternating multi-tiling.

E.7.1 TM alternation problem

Let $n \geq 1$. An *n-ary deterministic Turing machine* (TM, for short) is a deterministic Turing machine $\mathcal{M} = (n, I, A, Q, \{q_{acc}, q_{rej}\}, q_0, \delta)$ operating on n ordered semi-infinite tapes and having *only one* read/write head (shared by all tapes), where: I (resp., $A \supset I$) is the input (resp., work) alphabet, A contains the blank symbol $\# \notin I$, Q is the set of states, q_{acc} (resp., q_{rej}) is the terminal accepting (resp., rejecting) state, q_0 is the initial state, and $\delta : Q \times A \to \{\bot\} \cup (Q \times A \times \{\leftarrow, \rightarrow\}) \cup (Q \times \{\text{prev}, \text{next}\})$ is the transition function, where the symbol $\bot$ is for "undefined", and for all $(q, a) \in Q \times A$, $\delta(q, a) = \bot$ if and only if $q \in \{q_{acc}, q_{rej}\}$. In each non-terminal step, if the read/write head scans the k-th cell from the left of the ℓ-th tape (for $\ell \in [1, n]$ and $k \geq 1$) and $(q, a) \in (Q \setminus \{q_{acc}, q_{rej}\}) \times A$ is the current pair state/scanned cell content, one of the following occurs:

- $\delta(q, a) \in Q \times A \times \{\leftarrow, \rightarrow\}$ (*ordinary moves*): $\mathcal{M}$ overwrites the tape cell being scanned, there is a change of state, and the read/write head moves one position to the left ($\leftarrow$) or right ($\rightarrow$) in accordance with $\delta(q, a)$.[1]
- $\delta(q, a) \in Q \times \{\text{prev}, \text{next}\}$ (*jump moves*): if $\delta(q, a) = (q', \text{prev})$ (resp., $\delta(q, a) = (q', \text{next})$) for some $q' \in Q$, then the read/write head jumps to the k-th cell of the $(\ell - 1)$-th tape (resp., $(\ell + 1)$-th tape) and $\mathcal{M}$ moves to state q' if $\ell > 1$ (resp., $\ell < n$); otherwise, $\mathcal{M}$ moves to the rejecting state.

Initially, each tape contains a word in I^* and the read/write head points to the left-most cell of the first tape. Thus, an input of $\mathcal{M}$, called n-ary input, can be described by a tuple $(w_1, \ldots, w_n) \in (I^*)^n$, where for all $i \in [1, n]$, w_i represent the initial content of the i-th tape. $\mathcal{M}$ *accepts* a n-ary input $(w_1, \ldots, w_n) \in (I^*)^n$, written $\mathcal{M}(w_1, \ldots, w_n)$, if the computation of $\mathcal{M}$ from $(w_1, \ldots, w_n)$ is accepting. We consider the following problem.

[1] If the read/write head points to the left-most cell of the ℓ-th tape and $\delta(q, a)$ is of the form $(q', a, \leftarrow)$, then $\mathcal{M}$ moves to the rejecting state.

Definition E.7.1 (TM Alternation problem)**.** An instance of the problem is a tuple $(n, \mathcal{M})$, where $n > 1$ and $\mathcal{M}$ is a *polynomial-time bounded* n-ary deterministic Turing machine with input alphabet I. The instance $(n, \mathcal{M})$ is *positive* if and only if the following holds, where, for $\ell \in [1, n]$, $\mathrm{Q}_\ell = \exists$ if ℓ is odd, and $\mathrm{Q}_\ell = \forall$ otherwise:

$$\mathrm{Q}_1 x_1 \in I^{2^n}. \mathrm{Q}_2 x_2 \in I^{2^n}. \ldots \mathrm{Q}_n x_n \in I^{2^n}. \mathcal{M}(x_1, \ldots, x_n).$$

Note that the quantifications Q_i are restricted to words over I of length 2^n. Thus, even if $\mathcal{M}$ is polynomial-time bounded, it operates on an input whose size is exponential in n.

Proposition E.7.2. *The TM Alternation problem is* $\mathbf{AEXP}_{\mathbf{pol}}$*-complete.*

The proof is standard; however, for completeness, we give a proof of the hardness result, which follows from the next lemma.

Lemma E.7.3. *Let $\mathcal{M}_{\mathcal{A}}$ be a singly exponential-time bounded ATM making a polynomial bounded number of alternations. Moreover, let $c \geq 1$ and $c_a \geq 1$ be integer constants such that, for each input α, when started on α, $\mathcal{M}_{\mathcal{A}}$ makes at most $|\alpha|^{c_a}$ alternations and reaches a terminal configuration in at most $2^{|\alpha|^c}$ steps. Then, given an input α, one can construct in time polynomial in α and in the size of $\mathcal{M}_{\mathcal{A}}$ an instance $(2|\alpha|^{max\{c,c_a\}}, \mathcal{M})$ of the* TM Alternation problem *such that the instance is positive* if and only if *$\mathcal{M}_{\mathcal{A}}$ accepts α.*

Proof. Let $\mathcal{M}_{\mathcal{A}}$, c, and c_a be as in the above statement. Let $I_{\mathcal{A}}$ (resp., $A_{\mathcal{A}}$) be the input (resp., work) alphabet of $\mathcal{M}_{\mathcal{A}}$, where $I_{\mathcal{A}} \subset A_{\mathcal{A}}$, and Q be the set of $\mathcal{M}_{\mathcal{A}}$-states. Without loss of generality, we assume that the initial state of $\mathcal{M}_{\mathcal{A}}$ is existential. Fix an input $\alpha \in I_{\mathcal{A}}^*$. We define $k = \max\{c, c_a\}$ and $n = 2|\alpha|^k$.

An α-configuration is a word in $A_{\mathcal{A}}^* \cdot (Q \times A_{\mathcal{A}}) \cdot A_{\mathcal{A}}^*$ of length exactly $2^{|\alpha|^k}$. Note that any configuration of $\mathcal{M}_{\mathcal{A}}$ reachable from the input α can be encoded by an α-configuration. We denote by C_α the initial (existential) α-configuration associated with the input α. A *partial computation* of $\mathcal{M}_{\mathcal{A}}$ is a finite sequence $\pi = C_1, \ldots, C_p$ of α-configurations such that $p \leq 2^{|\alpha|^k}$ and for each $1 \leq i < p$, C_{i+1} is a $\mathcal{M}_{\mathcal{A}}$-successor of C_i (note that a computation of $\mathcal{M}_{\mathcal{A}}$ over α is a partial computation). We say that π is *uniform* if additionally one of the following holds:

- C_p is terminal and π visits only existential n-configurations;
- C_p is terminal and π visits only universal n-configurations;
- $p > 1$, C_p is existential and for each $1 \leq h < p$, C_h is universal;

- $p > 1$, C_p is universal and for each $1 \leq h < p$, C_h is existential.

Let $\diamond$ be a fresh symbol and $I = A_{\mathcal{A}} \cup \{\diamond\}$. The *code* of a partial computation $\pi = C_1, \ldots, C_p$ is the word over I of length exactly 2^n (recall that $n = 2|\alpha|^k$) given by $C_1, \ldots, C_p, C^0_{p+1}, \ldots, C^0_{2|\alpha|^k}$, where $C^0_i \in \{\diamond\}^{2^{|\alpha|^k}}$ for all $p+1 \leq i \leq 2^{|\alpha|^k}$. We construct a polynomial-time bounded n-ary deterministic Turing machine $\mathcal{M}$, which satisfies Lemma E.7.3 for the given input α of $\mathcal{M}_{\mathcal{A}}$. The input alphabet of $\mathcal{M}$ is I. Given a n-ary input $(w_1, \ldots, w_n) \in (I^*)^n$, $\mathcal{M}$ operates in n-steps (macro steps). At step i, for $i \in [1, n]$, the behavior of $\mathcal{M}$ is as follows, where for a partial computation $\pi = C_1, \ldots, C_p$, $\mathrm{first}(\pi) = C_1$ and $\mathrm{last}(\pi) = C_p$:

- $i = 1$.
  1. If $w_1 \in I^{2^n}$ and w_1 encodes a uniform partial computation π_1 of $\mathcal{M}_{\mathcal{A}}$ *from* C_α, then the behavior is as follows. If $\mathrm{last}(\pi_1)$ is accepting (resp., rejecting), then $\mathcal{M}$ accepts (resp., rejects) the input. Conversely, if $\mathrm{last}(\pi_1)$ is not a terminal configuration, then $\mathcal{M}$ goes to step $i+1$.
  2. Otherwise, $\mathcal{M}$ *rejects* the input.
- $i > 1$.
  1. If $w_i \in I^{2^n}$ and w_i encodes a uniform partial computation π_i of $\mathcal{M}_{\mathcal{A}}$ such that $\mathrm{first}(\pi_i) = \mathrm{last}(\pi_{i-1})$, where π_{i-1} is the uniform partial computation encoded by w_{i-1}, then the behavior is as follows. If $\mathrm{last}(\pi_i)$ is accepting (resp., rejecting), then $\mathcal{M}$ accepts (resp., rejects) the input. If instead $\mathrm{last}(\pi_i)$ is not a terminal configuration, then $\mathcal{M}$ goes to step $i+1$, if $i+1 \leq n$, and rejects the input otherwise.
  2. Otherwise, if i is odd (resp., even), then $\mathcal{M}$ *rejects* (resp., *accepts*) the input.

Note that (1.) in the steps above can be checked by $\mathcal{M}$ in polynomial time (in the size of the input) by using the transition function of $\mathcal{M}_{\mathcal{A}}$ and n-bit counters. Hence, $\mathcal{M}$ is a polynomial-time bounded n-ary deterministic Turing machine which can be built in time polynomial in n and in the size of $\mathcal{M}_{\mathcal{A}}$.

Now, we prove that the construction is correct, that is, $(n, \mathcal{M})$ is a positive instance of the TM Alternation problem if and only if $\mathcal{M}_{\mathcal{A}}$ accepts α. For each $\ell \in [1, n]$, let $\mathrm{Q}_\ell = \exists$ if ℓ is odd, and $\mathrm{Q}_\ell = \forall$ otherwise. Since C_α is existential, $\mathcal{M}_{\mathcal{A}}$ accepts α if and only if there is a uniform partial computation π_1 of $\mathcal{M}_{\mathcal{A}}$ from C_α such that $\mathrm{last}(\pi_1)$ leads to acceptance. Moreover, for each $w_1 \in I^{2^n}$, $\mathcal{M}$ accepts an input of the form $(w_1, w'_2, \ldots, w'_k)$ *only if* w_1 encodes a non-rejecting uniform partial computation of $\mathcal{M}_{\mathcal{A}}$ from C_α. Thus, since $\mathrm{Q}_1 = \exists$, correctness of the construction directly follows from the following claim.

Claim E.7.4. Let $\ell \in [1, n]$ and $\pi = \pi_1 \dots \pi_\ell$ be a partial computation of $\mathcal{M}_{\mathcal{A}}$ from C_α such that π_ℓ is uniform, and for each $1 \leq j < \ell$, π_j is non-empty and $\pi_j \cdot \mathrm{first}(\pi_{j+1})$ is uniform as well. Let $w_\ell \in I^{2^n}$ be the word encoding π_ℓ, and for each $1 \leq j < \ell$, $w_j \in I^{2^n}$ be the word encoding $\pi_j \cdot \mathrm{first}(\pi_{j+1})$. Then, $\mathrm{last}(\pi_\ell)$ leads to acceptance in $\mathcal{M}_{\mathcal{A}}$ if and only if

$$\mathrm{Q}_{\ell+1} x_{\ell+1} \in I^{2^n}. \dots \mathrm{Q}_n x_n \in I^{2^n}. \mathcal{M}(w_1, \dots, w_\ell, x_{\ell+1}, \dots, x_n). \tag{E.1}$$

Proof. The proof is by induction on $n - \ell$.

In the base case, we have $\ell = n$. Note that, in this case, $\mathrm{last}(\pi_n)$ is a terminal configuration of $\mathcal{M}_{\mathcal{A}}$ (otherwise, the number of alternations of existential and universal configurations along π would be greater than $n - 1 \geq |\alpha|^{c_a}$). Thus, we need to show that $\mathrm{last}(\pi_n)$ is accepting if and only if $\mathcal{M}(w_1, \dots, w_n)$. By construction, when started on the input $(w_1, \dots, w_n)$, $\mathcal{M}$ reaches the n-th step and (1.) in this step is satisfied. Moreover, either $\mathrm{last}(\pi_n)$ is accepting and $\mathcal{M}$ accepts the input $(w_1, \dots, w_n)$, or $\mathrm{last}(\pi_n)$ is rejecting and $\mathcal{M}$ rejects the input $(w_1, \dots, w_n)$. Hence, the result follows.

Let us now consider the inductive step, where $\ell < n$. First, assume that $\mathrm{last}(\pi_\ell)$ is a terminal configuration. By construction, on any input of the form $(w_1, \dots, w_\ell, w'_{\ell+1}, \dots, w'_n)$, $\mathcal{M}$ reaches the ℓ-th step and (1.) in this step is satisfied. Moreover, either $\mathrm{last}(\pi_\ell)$ is accepting and $\mathcal{M}$ accepts the input $(w_1, \dots, w_\ell, w'_{\ell+1}, \dots, w'_n)$, or $\mathrm{last}(\pi_\ell)$ is rejecting and $\mathcal{M}$ rejects the input $(w_1, \dots, w_\ell, w'_{\ell+1}, \dots, w'_n)$. Hence, in this case the result holds. Now, assume that $\mathrm{last}(\pi_\ell)$ is not terminal. We consider the case where $\ell + 1$ is even (the other case being similar). Then, $\mathrm{Q}_{\ell+1} = \forall$. Since C_α is existential and $\mathrm{last}(\pi_\ell)$ is not terminal, by hypothesis, $\mathrm{last}(\pi_\ell)$ must be a universal configuration. First, assume that $\mathrm{last}(\pi_\ell)$ leads to acceptance. Let $w_{\ell+1} \in I^{2^n}$. By construction, on any input of the form $(w_1, \dots, w_\ell, w_{\ell+1}, w'_{\ell+2} \dots, w'_n)$, $\mathcal{M}$ reaches the $(\ell + 1)$-th step. If $w_{\ell+1}$ satisfies (2.) in this step, then since $\ell + 1$ is even, $\mathcal{M}$ accepts the input. Hence, $\mathrm{Q}_{\ell+2} x_{\ell+2} \in I^{2^n}. \dots \mathrm{Q}_n x_n \in I^{2^n}. \mathcal{M}(w_1, \dots, w_\ell, w_{\ell+1}, x_{\ell+2}, \dots, x_n)$. Otherwise $w_{\ell+1}$ encodes a uniform partial computation $\pi_{\ell+1}$ of $\mathcal{M}_{\mathcal{A}}$ from $\mathrm{last}(\pi_\ell)$. Since $\mathrm{last}(\pi_\ell)$ leads to acceptance and $\mathrm{last}(\pi_\ell)$ is universal, $\mathrm{last}(\pi_{\ell+1})$ leads to acceptance as well. Thus, by applying the inductive hypothesis to the partial computation $\pi_1 \dots \pi_{\ell-1} \pi'_\ell \pi_{\ell+1}$ (where π'_ℓ is obtained from π_ℓ by removing $\mathrm{last}(\pi_\ell)$), it follows that $\mathrm{Q}_{\ell+2} x_{\ell+2} \in I^{2^n}. \dots \mathrm{Q}_n x_n \in I^{2^n}. \mathcal{M}(w_1, \dots, w_\ell, w_{\ell+1}, x_{\ell+2}, \dots, x_n)$. Thus the previous condition holds for each $w_{\ell+1} \in I^{2^n}$. Since $\mathrm{Q}_{\ell+1} = \forall$, it follows that (E.1) holds. For the converse direction, assume that (E.1) holds. Let $\pi_{\ell+1}$ be any uniform partial computation of $\mathcal{M}_{\mathcal{A}}$ from $\mathrm{last}(\pi_\ell)$. We need to show that $\mathrm{last}(\pi_{\ell+1})$ leads to acceptance. Since (E.1) holds and $\mathrm{Q}_{\ell+1} = \forall$, we can apply the inductive hypothesis to the partial computation $\pi_1 \dots \pi_{\ell-1} \pi'_\ell \pi_{\ell+1}$ (where π'_ℓ is obtained from π_ℓ by removing $\mathrm{last}(\pi_\ell)$). The result follows, concluding the proof of the claim. □

This concludes the proof of Lemma E.7.3, as well. □

E.7.2 $\mathbf{AEXP_{pol}}$-hardness of the alternating multi-tiling problem

We prove here the $\mathbf{AEXP_{pol}}$-hardness of the alternating multi-tiling problem by a polynomial time reduction from the TM alternation problem. Fix an instance $(n, \mathcal{M})$ of the TM alternation problem where $\mathcal{M} = (n, I, A, Q, \{q_{acc}, q_{rej}\}, q_0, \delta)$ is a polynomial-time bounded n-ary deterministic Turing machine.

Remark E.7.5 (Assumptions on $\mathcal{M}$). In order to simplify the reduction, w.l.o.g., we can assume that $\mathcal{M}$ satisfies the following constraints:

- n is even;
- for each n-ary input $(w_1, \ldots, w_n) \in I^{2^n} \times \ldots \times I^{2^n}$, $\mathcal{M}$ reaches a terminal configuration in exactly $2^n - 1$ steps, and when $\mathcal{M}$ halts, the read/write head points to a cell of the n-th tape;
- there is no move leading to the initial state q_0;
- for all $a \in A$, $\delta(q_0, a) \in Q \times A \times \{\rightarrow\}$;
- for all $(q, a), (q', a') \in Q \times A$, if $\delta(q, a) \in \{q'\} \times \{\mathsf{prev}, \mathsf{next}\}$, then $\delta(q', a') \notin Q \times \{\mathsf{prev}, \mathsf{next}\}$.

We construct, in polynomial time in the size of $(n, \mathcal{M})$, an instance $\mathcal{I}$ of the alternating multi-tiling problem such that $(n, \mathcal{M})$ is a positive instance of the TM alternation problem if and only if $\mathcal{I}$ is a positive instance of the alternating multi-tiling problem. By the definitions of the considered problems, it suffices to show the following.

Proposition E.7.6. *One can construct in time polynomial in the size of $(n, \mathcal{M})$, an instance $\mathcal{I} = (n, D, D_0, H, V, M, D_{acc})$ of the alternating multi-tiling problem such that $D_0 = I$ and the following holds: for each n-ary input $(w_1, \ldots, w_n) \in I^{2^n} \times \ldots \times I^{2^n}$, $\mathcal{M}(w_1, \ldots, w_n)$ if and only if there exists a multi-tiling $F = (f_1, \ldots, f_n)$ of $\mathcal{I}$ such that, for all $\ell \in [1, n]$, the initial condition Init(f_ℓ) of the tiling f_ℓ is w_ℓ.*

Proof. We adapt the well-known translation between time-space diagrams of computations (also known as computation tableaux) of a nondeterministic TM and tilings for a set of domino types entirely determined by the given TM. In such a translation, adjacent rows in the tiled region encode successive configurations in a computation of the machine.

Let $\lceil A \rceil$ be a fresh copy of the work alphabet A of $\mathcal{M}$. For a word w over A, we denote by $\lceil w \rceil$ the associated word over $\lceil A \rceil$. We define $U = A \times [1, n]$ and

$\lceil U \rceil = \lceil A \rceil \times [1,n]$. We adopt the following set D of domino types:

$$D = I \cup U \cup \lceil U \rceil \cup (Q \times U) \cup ((Q \times \{\leftarrow, \rightarrow, \text{prev}, \text{next}\}) \times U)$$

Intuitively, in the encoding we keep track of the tape-indexes of $\mathcal{M}$. Moreover, the domino types in $(Q \times \{\leftarrow, \rightarrow\}) \times U$ (resp., $(Q \times \{\text{prev}, \text{next}\}) \times U$) are used to encode the effects of the ordinary moves (resp., jump moves). For each $d \in D$, we denote by $\text{symb}(d)$ the associated letter in A (recall that $I \subset A$). Moreover, if $d \in D \setminus I$, we write $\text{tape}(d)$ to mean the associated tape index $\ell \in [1,n]$. Additionally, if $d \in (Q \times U) \cup ((Q \times \{\leftarrow, \rightarrow, \text{prev}, \text{next}\}) \times U)$, we denote by $\text{state}(d)$, the state $q \in Q$ associated with d. If instead $d \in I \cup U \cup \lceil U \rceil$, we set $\text{state}(d) = \perp$ ($\perp$ is for "undefined").

Given $\ell \in [1,n]$ and a word v over the alphabet

$$A \cup \lceil A \rceil \cup (Q \times A) \cup ((Q \times \{\leftarrow, \rightarrow, \text{prev}, \text{next}\}) \times A)$$

we write $v \oplus \ell$ to denote the word over alphabet $D \setminus I$ defined in the obvious way.

Fix an n-ary input $(w_1, \ldots, w_n) \in I^{2^n} \times \ldots \times I^{2^n}$ and a non-rejecting configuration C of $\mathcal{M}$ reachable from the input $(w_1, \ldots, w_n)$. Assume that in C, the read/write head points to the k-th cell of the ℓ-th tape for some $k \geq 1$ and $\ell \in [1,n]$, and let $(q,a) \in Q \times A$ be the pair state/scanned cell content associated with C. Since on input $(w_1, \ldots, w_n)$, $\mathcal{M}$ halts in $2^n - 1$ steps, it holds that $k \leq 2^n$ and C can be encoded by the tuples of words in D^{2^n} of the form $(w_1^C \oplus 1, \ldots, w_n^C \oplus n)$ defined as follows:

- *cases* $q = q_{acc}$ *or* $\delta(q,a) \in Q \times A \times \{\leftarrow, \rightarrow\}$:
  for each $j \in [1,n] \setminus \{\ell\}$, $w_j^C = w_j$ or $w_j^C = \lceil w_j \rceil$ where w_j is the content of the first 2^n cells of the j-th tape, and one of the following holds:
  - $q = q_{acc}$: w_ℓ^C is of the form $w' \cdot (q,a) \cdot \lceil w'' \rceil$, where $w' \cdot a \cdot w''$ is the content of the first 2^n cells of the ℓ-th tape and $|w'| = k-1$ (the read/write head points to the k-th cell of the ℓ-th tape);
  - $\delta(q,a) \in \{q'\} \times A \times \{\rightarrow\}$ for some $q' \in Q$: w_ℓ^C is of the form $w' \cdot (q,a) \cdot ((q', \rightarrow), a') \cdot \lceil w'' \rceil$, where $w' \cdot a \cdot a' \cdot w''$ is the content of the first 2^n cells of the ℓ-th tape and $|w'| = k-1$;
  - $\delta(q,a) \in \{q'\} \times A \times \{\leftarrow\}$ for some $q' \in Q$: w_ℓ^C is of the form $w' \cdot ((q', \leftarrow), a') \cdot (q,a) \cdot \lceil w'' \rceil$, where $w' \cdot a' \cdot a \cdot w''$ is the content of the first 2^n cells of the ℓ-th tape and $|w'| = k-1$;
- *case* $\delta(q,a) \in \{q'\} \times \{prev\}$ for some $q' \in Q$:
  - for each $j \in [1,n] \setminus \{\ell, \ell-1\}$, $w_j^C = w_j$ or $w_j^C = \lceil w_j \rceil$ where w_j is the content of the first 2^n cells of the j-th tape;
  - w_ℓ^C is of the form $w' \cdot (q,a) \cdot \lceil w'' \rceil$, where $w' \cdot a \cdot w''$ is the content of the first 2^n cells of the ℓ-th tape and $|w'| = k-1$;

  - if $\ell > 1$, $w^C_{\ell-1}$ is of the form $w' \cdot ((q', \text{prev}), a) \cdot \lceil w'' \rceil$, where $w' \cdot a \cdot w''$ is the content of the first 2^n cells of the $(\ell-1)$-th tape and $|w'| = k-1$;
- *case* $\delta(q, a) \in \{q'\} \times \{next\}$ for some $q' \in Q$:
  - for each $j \in [1, n] \setminus \{\ell, \ell+1\}$, $w^C_j = w_j$ or $w^C_j = \lceil w_j \rceil$ where w_j is the content of the first 2^n cells of the j-th tape;
  - w^C_ℓ is of the form $w' \cdot (q, a) \cdot \lceil w'' \rceil$, where $w' \cdot a \cdot w''$ is the content of the first 2^n cells of the ℓ-th tape and $|w'| = k-1$;
  - if $\ell < n$, $w^C_{\ell+1}$ is of the form $w' \cdot ((q', \text{next}), a) \cdot \lceil w'' \rceil$, where $w' \cdot a \cdot w''$ is the content of the first 2^n cells of the $(\ell+1)$-th tape and $|w'| = k-1$;

We construct in polynomial-time an instance $\mathcal{I} = (n, D, D_0, H, V, M, D_{acc})$ of the alternating multi-tiling problem with $D_0 = I$ and $D_{acc} = \{q_{acc}\} \times U$ such that for each n-ary input $(w_1, \ldots, w_n) \in I^{2^n} \times \ldots \times I^{2^n}$, $\mathcal{M}(w_1, \ldots, w_n)$ if and only if there exists a multi-tiling $F = (f_1, \ldots, f_n)$ of $\mathcal{I}$ such that for all $\ell \in [1, n]$, the initial condition of the tiling f_ℓ is w_ℓ. Moreover, if $\mathcal{M}(w_1, \ldots, w_n)$, then the following holds:

- let $\pi = C_1 \ldots C_{2^n-1}$ be the accepting computation of $\mathcal{M}$ over $(w_1, \ldots, w_n)$ (by our assumptions, the write/read head in the accepting configuration C_{2^n-1} points to the n-th tape). Then, there exist multi-tilings $F = (f_1, \ldots, f_n)$ of $\mathcal{I}$ associated with the input $(w_1, \ldots, w_n)$ such that, for all $j \in [1, 2^n - 1]$, there is an encoding $cod(C_j)$ of configuration C_j so that, for all $\ell \in [1, n]$, the row of index j of f_ℓ coincides with the ℓ-th component of $cod(C_j)$.

We define the matching relations H, V and M in order to ensure the above conditions. In particular, the horizontal matching relation H guarantees that the TM configurations are correctly encoded, while the vertical matching relation is used to encode the ordinary moves of $\mathcal{M}$. Finally, the multi-tiling matching relation M is used to encode the jump moves. Additionally, H, V and M also ensure that the rows of index 1 encode the initial configuration of $\mathcal{M}$ associated with the given n-ary input (the latter corresponds to the tuple of rows of index 0).

Formally H is the set of pairs $(d, d') \in D \times D$ satisfying the following constraints:

- $d \in I$ if and only if $d' \in I$;
- if $d \in D \setminus I$, then $d' \in D \setminus I$ and $\text{tape}(d) = \text{tape}(d')$;
- if $d, d' \in U \cup \lceil U \rceil$, then either $d, d' \in U$, or $d, d' \in \lceil U \rceil$;
- $\text{state}(d') \neq q_0$, and whenever $\text{state}(d) = q_0$, then $d \in \{q_0\} \times U$ and $\text{tape}(d) = 1$;
- if $d \in \lceil U \rceil$ or $d \in (Q \times \{\rightarrow, \text{prev}, \text{next}\}) \times U$, then $d' \in \lceil U \rceil$;
- if $d \in Q \times U$ and $\delta(\text{state}(d), \text{symb}(d)) \notin Q \times A \times \{\rightarrow\}$, then $d' \in \lceil U \rceil$;
- if $d' \in U$ or $d' \in (Q \times \{\leftarrow, \text{prev}, \text{next}\}) \times U$, then $d \in U$;
- if $d' \in Q \times U$ and $\delta(\text{state}(d), \text{symb}(d)) \notin Q \times A \times \{\leftarrow\}$, then $d \in U$;
- for $q' \in Q$, $d \in Q \times U$ and $\delta(\text{state}(d), \text{symb}(d)) \in \{q'\} \times A \times \{\rightarrow\}$ *if and only if* $d' \in \{(q', \rightarrow)\} \times U$;

- for $q' \in Q$, $d' \in Q \times U$ and $\delta(\text{state}(d), \text{symb}(d)) \in \{q'\} \times A \times \{\leftarrow\}$ *if and only if* $d \in \{(q', \leftarrow)\} \times U$.

By definition of H (independently of V and M), we deduce the following.

Claim E.7.7. Let f be a tiling of $\mathcal{I}$ and row be the content of any row of f. Then, either row $\in I^*$ or row $=$ row$' \oplus \ell$ for some $\ell \in [1, n]$, and row$'$ satisfies one of the following:

- row$' =$ row$\cdot(q, a)\cdot\lceil w' \rceil$ such that $w, w' \in A^*$ and $\delta(q, a) \notin Q \times A \times \{\leftarrow, \rightarrow\}$;
- row$' =$ row$\cdot d \cdot \lceil w' \rceil$ such that $w, w' \in A^*$, and $d \in (Q \times \{\text{prev}, \text{next}\}) \times A$;
- row$' =$ row such that row $\in A^*$;
- row$' = \lceil w \rceil$ such that $w \in A^*$;
- row$' =$ row$\cdot d \cdot ((q', \rightarrow), a') \cdot \lceil w' \rceil$ such that $w, w' \in A^*$, $d \in Q \times A$, and $\delta(\text{state}(d), \text{symb}(d)) \in \{q'\} \times A \times \{\rightarrow\}$;
- row$' =$ row$\cdot ((q', \leftarrow), a') \cdot d \cdot \lceil w' \rceil$ such that $w, w' \in A^*$, $d \in Q \times A$, and $\delta(\text{state}(d), \text{symb}(d)) \in \{q'\} \times A \times \{\leftarrow\}$;

Moreover, if for some i, $\text{state}(\text{row}(i)) = q_0$, then $i = 0$, $\text{row}(0) \in \{q_0\} \times U$, and $\text{tape}(\text{row}(0)) = 1$.

Now, let us define the vertical matching relation V. V is the set of pairs $(d, d') \in D \times D$ satisfying the following constraints:

- if $d \in I$ then $d' \in (\{q_0\} \times U) \cup U \cup \lceil U \rceil \cup (Q \times \{\rightarrow\}) \times U$ and $\text{symb}(d) = \text{symb}(d')$;
- if $d \in D \setminus I$, then $d' \in D \setminus I$ and $\text{tape}(d) = \text{tape}(d')$;
- if $d \in U \cup \lceil U \rceil$ then $d' \in U \cup \lceil U \rceil \cup ((Q \times \{\text{prev}, \text{next}\}) \times U)$ and $\text{symb}(d) = \text{symb}(d')$;
- if $d \in (Q \times \{\leftarrow, \rightarrow, \text{prev}, \text{next}\}) \times U$, then $d' = (\text{state}(d), \text{symb}(d), \text{tape}(d))$;
- if $d \in Q \times U$, then $\text{state}(d) \neq q_{acc}$ and one of the following holds:
  - $\delta(\text{state}(d), \text{symb}(d)) \in Q \times \{\text{symb}(d')\} \times \{\leftarrow, \rightarrow\}$ and $d' \in U \cup \lceil U \rceil$;
  - $\delta(\text{state}(d), \text{symb}(d)) \in Q \times \{\text{prev}, \text{next}\}$, $d' \in U$ and $\text{symb}(d') = \text{symb}(d)$.

Finally, we define the multi-tiling matching relation M. M is the set of pairs $(d, d') \in D \times D$ satisfying the following constraints:

- $d \in I$ if and only if $d' \in I$;
- if $d \in D \setminus I$, then $d' \in D \setminus I$ and $\text{tape}(d') = \text{tape}(d) + 1$;
- $\text{state}(d) \neq q_{acc}$ and $\text{state}(d') \neq q_0$;
- for each $q \in Q$, $d \in Q \times U$ and $\delta(\text{state}(d), \text{symb}(d)) = (q, \text{next})$ *if and only if* $d' \in \{(q, \text{next})\} \times U$;
- for each $q \in Q$, $d' \in Q \times U$ and $\delta(\text{state}(d), \text{symb}(d)) = (q, \text{prev})$ *if and only if* $d \in \{(q, \text{prev})\} \times U$;
- if $d \in (Q \times \{\text{next}\}) \times U$, then $\text{tape}(d) > 1$;

- if $d' \in (Q \times \{\text{prev}\}) \times U$, then $\text{tape}(d') < n$.

By Claim E.7.7 and the definitions of the relations V and M, one can prove that if F is a multi-tiling of $\mathcal{I}$ with initial conditions $(w_1, \ldots, w_n) \in I^{2^n} \times \ldots \times I^{2^n}$, then F encodes an accepting computation of $\mathcal{M}$ over the n-ary input $(w_1, \ldots, w_n) \in I^{2^n} \times \ldots \times I^{2^n}$. Vice versa, by Remark E.7.5, if $\mathcal{M}(w_1, \ldots, w_n)$, then it follows that there is a multi-tiling encoding the accepting computation of $\mathcal{M}$ over $(w_1, \ldots, w_n)$. □

E.8 Proof of Proposition 7.4.2

Proposition (7.4.2). *Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure, φ be an $\overline{\text{A}}\text{A}\overline{\text{B}}\text{B}$ formula with RE's $r_1, \ldots, r_k$ over $\mathcal{AP}$, $\rho \in \text{Trc}_{\mathcal{K}}$ be a trace, and $(q^1, \ldots, q^k) \in Q(\varphi)$. There exists a trace $\pi \in \text{Trc}_{\mathcal{K}}$, which is $(q^1, \ldots, q^k)$-well-formed w.r.t. ρ, such that $|\pi| \leq |S| \cdot 2^{2\sum_{\ell=1}^{k} |r_\ell|}$.*

Proof. Let $\rho \in \text{Trc}_{\mathcal{K}}$ with $|\rho| = n$. If $n \leq |S| \cdot 2^{2\sum_{\ell=1}^{k} |r_\ell|}$, the thesis trivially holds. Thus, let us assume $n > |S| \cdot 2^{2\sum_{\ell=1}^{k} |r_\ell|}$. We show that there exists a trace which is $(q^1, \ldots, q^k)$-well-formed w.r.t. ρ, whose length is less than n. The number of possible (joint) configurations of the DFAs $\mathcal{D}(\varphi)$ is (at most) $|Q(\varphi)| \leq 2^{2|r_1|} \ldots 2^{2|r_k|} = 2^{2\sum_{\ell=1}^{k} |r_\ell|}$. Since $n > |S| \cdot 2^{2\sum_{\ell=1}^{k} |r_\ell|}$, there exists some state $s \in S$ occurring in ρ at least twice in the ρ-positions, say $1 \leq l_1 < l_2 \leq |\rho|$, such that $\mathcal{D}^t_{q^t}(\mu(\rho^{l_1})) = \mathcal{D}^t_{q^t}(\mu(\rho^{l_2}))$, for all $t = 1, \ldots, k$. Let us consider $\pi = \rho(1, l_1) \star \rho(l_2, n)$ (see Figure E.1). It is easy to see that $\pi \in \text{Trc}_{\mathcal{K}}$, as $\rho(l_1) = \rho(l_2)$, and $|\pi| < n$. Moreover, π is $(q^1, \ldots, q^k)$-well-formed w.r.t. ρ (the corresponding positions are $i_j = j$ if $j \leq l_1$, and $i_j = j + (l_2 - l_1)$ otherwise). If $|\pi| \leq |S| \cdot 2^{2\sum_{\ell=1}^{k} |r_\ell|}$, the thesis holds. Otherwise, the same basic step can be iterated a finite number of times, and the thesis follows by transitivity of the $(q^1, \ldots, q^k)$-well-formedness relation. □

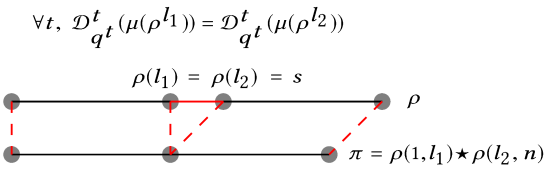


Figure E.1: The contraction step of Proposition 7.4.2.

E.9 Proof of Theorem 7.4.3

Theorem (7.4.3, Exponential small-model for $\mathrm{A\overline{A}B\overline{B}}$). *Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure, $\sigma, \rho \in \mathrm{Trc}_{\mathcal{K}}$, and φ be an $\mathrm{A\overline{A}B\overline{B}}$ formula in NNF, with* RE*'s $r_1, \ldots, r_u$ over $\mathcal{AP}$, such that $\mathcal{K}, \sigma\star\rho \models \varphi$. There exists $\pi \in \mathrm{Trc}_{\mathcal{K}}$, induced by ρ, such that $\mathcal{K}, \sigma\star\pi \models \varphi$ and $|\pi| \leq |S|\cdot(|\varphi|+1)\cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|}$.*

Proof. Let $Wt(\varphi, \sigma\star\rho)$ be the set of witness positions of $\sigma\star\rho$ for φ. Let $\{i_1, \ldots, i_k\}$ be the ordering of $Wt(\varphi, \sigma\star\rho)$ such that $i_1 < \ldots < i_k$. Let $i_0 = 1$ and $i_{k+1} = |\sigma\star\rho|$. Hence, $1 = i_0 \leq i_1 < \ldots < i_k < i_{k+1} = |\sigma\star\rho|$. If the length of ρ is at most $|S|\cdot(|\varphi|+1)\cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|}$, the thesis trivially holds. Let us assume that $|\rho| > |S|\cdot(|\varphi|+1)\cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|}$. We show that there exists a trace π induced by ρ, with $|\pi| < |\rho|$, such that $\mathcal{K}, \sigma\star\pi \models \varphi$.

W.l.o.g., we can assume that $i_0 \leq i_1 < \ldots < i_{j-1}$, for some $j \geq 1$, are σ-positions (while $i_j < \ldots < i_{k+1}$ are $(\sigma\star\rho)$-positions not in σ). We claim that either (i) there exists $t \in [j, k]$ such that $i_{t+1} - i_t > |S|\cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|}$ or (ii) $|(\sigma\star\rho)(|\sigma|, i_j)| > |S|\cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|}$. By way of contradiction, suppose that neither (i) nor (ii) holds. We need to distinguish two cases. If $\sigma\star\rho = \rho$, then $|\rho| = (i_{k+1} - i_0) + 1 \leq (k+1)\cdot|S|\cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|} + 1$ (a contradiction); otherwise ($|\rho| < |\sigma\star\rho|$), $|\rho| = (i_{k+1} - i_j) + |(\sigma\star\rho)(|\sigma|, i_j)| \leq k\cdot|S|\cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|} + |S|\cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|} \leq (k+1)\cdot|S|\cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|}$. The contradiction follows since $(k+1)\cdot|S|\cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|} + 1 \leq |\varphi|\cdot|S|\cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|} + 1 \leq |S|\cdot(|\varphi|+1)\cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|}$.

Let us define $(\alpha, \beta) = (i_t, i_{t+1})$ in case (i), and $(\alpha, \beta) = (|\sigma|, i_j)$ in case (ii). Moreover let $\rho' = \rho(\alpha, \beta)$. In both the cases, we have $|\rho'| > |S|\cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|}$. By Proposition 7.4.2, there exists a trace π' of $\mathcal{K}, (q^1, \ldots, q^u)$-well-formed with respect to ρ', such that $|\pi'| \leq |S|\cdot 2^{2\sum_{\ell=1}^{u}|r_\ell|} < |\rho'|$, where we choose $q^x = \mathcal{D}^x(\mu((\sigma\star\rho)^{\alpha-1}))$ for $x = 1, \ldots, u$ (as a particular case we set q^x as the initial state of $\mathcal{D}^x$ if $\alpha = 1$). Let π be the trace induced by ρ obtained by replacing the subtrace ρ' of ρ with π' (see Figure E.2). Since $|\pi| < |\rho|$, it remains to prove that $\mathcal{K}, \sigma\star\pi \models \varphi$.

Let us denote $\sigma\star\pi$ by $\overline{\pi}$ and $\sigma\star\rho$ by $\overline{\rho}$. Moreover, let $H : [1, |\overline{\pi}|] \to [1, |\overline{\rho}|]$ be the function mapping positions of $\overline{\pi}$ into positions of $\overline{\rho}$ in this way: positions "outside" π' (i.e., outside the interval $[\alpha, \alpha + |\pi'| - 1]$) are mapped into

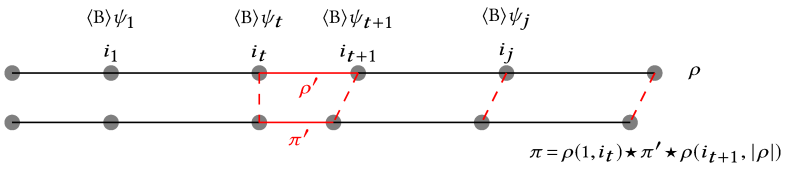


Figure E.2: Representation of the contraction step of Theorem 7.4.3—case (i)

their original position in $\overline{\rho}$; positions "inside" π' (i.e., in $[\alpha, \alpha + |\pi'| - 1]$) are mapped to the corresponding position in ρ' (exploiting well-formedness of π' w.r.t. ρ'). Formally, H is defined as:

$$H(m) = \begin{cases} m & \text{if} \quad m < \alpha \\ \alpha + \ell_{m-\alpha+1} - 1 & \text{if} \quad \alpha \leq m < \alpha + |\pi'| \\ m + (|\rho'| - |\pi'|) & \text{if} \quad m \geq \alpha + |\pi'| \end{cases}$$

where ℓ_m is the ρ'-position corresponding to the π'-position m. It is easy to check that H satisfies the following properties:

1. H is strictly monotonic, i.e., for all $j, j' \in [1, |\overline{\pi}|]$, $j < j' \iff H(j) < H(j')$;
2. for all $j \in [1, |\overline{\pi}|]$, $\overline{\pi}(j) = \overline{\rho}(H(j))$;
3. $H(1) = 1$ and $H(|\overline{\pi}|) = |\overline{\rho}|$;
4. $Wt(\varphi, \overline{\rho}) \subseteq \{H(j) \mid j \in [1, |\overline{\pi}|]\}$ i.e., all witness positions are preserved;
5. for each $j \in [1, |\overline{\pi}|]$ and $x = 1, \ldots, u$, $\mathcal{D}^x(\mu(\overline{\pi}^j)) = \mathcal{D}^x(\mu(\overline{\rho}^{H(j)}))$.

We only comment on Property 5. The property holds for $j \in [1, \alpha - 1]$, as $\overline{\pi}^j = \overline{\rho}^{H(j)} = \overline{\rho}^j$. For $j \in [\alpha, \alpha + |\pi'| - 1]$, $\mathcal{D}^x(\mu(\overline{\pi}^j)) = \mathcal{D}^x(\mu(\overline{\rho}^{H(j)}))$ follows from the well-formedness hypothesis. Finally, being $\overline{\rho}(\beta, |\overline{\rho}|) = \overline{\pi}(\alpha + |\pi'| - 1, |\overline{\pi}|)$ and $\mathcal{D}^x(\mu(\overline{\pi}^{\alpha+|\pi'|-1})) = \mathcal{D}^x(\mu(\overline{\rho}^{\beta}))$, the property holds also for $j \in [\alpha + |\pi'|, |\overline{\pi}|]$.

The fact that $\mathcal{K}, \overline{\pi} \models \varphi$ is an immediate consequence of the following claim, considering that $H(|\overline{\pi}|) = |\overline{\rho}|$, $\mathcal{K}, \overline{\rho} \models \varphi$, $\overline{\rho}^{|\overline{\rho}|} = \overline{\rho}$, and $\overline{\pi}^{|\overline{\pi}|} = \overline{\pi}$.

Claim E.9.1. For all $j \in [1, |\overline{\pi}|]$, all subformulas ψ of φ, and all $\xi \in \mathrm{Trc}_{\mathcal{K}}$, it holds that $\mathcal{K}, \overline{\rho}^{H(j)} \star \xi \models \psi \Longrightarrow \mathcal{K}, \overline{\pi}^j \star \xi \models \psi$.

Proof. Assume that $\mathcal{K}, \overline{\rho}^{H(j)} \star \xi \models \psi$. Note that $\overline{\rho}^{H(j)} \star \xi$ is defined if and only if $\overline{\pi}^j \star \xi$ is defined. We prove by induction on the structure of ψ that $\mathcal{K}, \overline{\pi}^j \star \xi \models \psi$. Since φ is in NNF, only the following cases can occur.

- $\psi = r_t$ or $\psi = \neg r_t$ where r_t is some RE over $\mathcal{AP}$. By Property 5 of H, $\mathcal{D}^t(\mu(\overline{\pi}^j)) = \mathcal{D}^t(\mu(\overline{\rho}^{H(j)}))$, thus $\mathcal{D}^t(\mu(\overline{\pi}^j \star \xi)) = \mathcal{D}^t(\mu(\overline{\rho}^{H(j)} \star \xi))$. It follows that $\mathcal{K}, \overline{\pi}^j \star \xi \models r_t$ if and only if $\mathcal{K}, \overline{\rho}^{H(j)} \star \xi \models r_t$, and the thesis holds.
- $\psi = \theta_1 \wedge \theta_2$ or $\psi = \theta_1 \vee \theta_2$ for some $\mathsf{A\overline{A}B\overline{B}}$ formulas θ_1 and θ_2. The result holds by the inductive hypothesis.
- $\psi = [\mathrm{B}]\,\theta$. We need to show that for each proper prefix η of $\overline{\pi}^j \star \xi$, we have $\mathcal{K}, \eta \models \theta$. We distinguish two cases:
  - η is *not* a proper prefix of $\overline{\pi}^j$. Hence, η has the form $\overline{\pi}^j \star \xi^h$ for some $h \in [1, |\xi| - 1]$. Since $\mathcal{K}, \overline{\rho}^{H(j)} \star \xi \models [\mathrm{B}]\,\theta$, then $\mathcal{K}, \overline{\rho}^{H(j)} \star \xi^h \models \theta$. By the inductive hypothesis, we have $\mathcal{K}, \overline{\pi}^j \star \xi^h \models \theta$.

– η is a proper prefix of $\overline{\pi}^j$. Hence, $\eta = \overline{\pi}^h$ for some $h \in [1, j-1]$. By Property 1 of H, $H(h) < H(j)$, and since $\mathcal{K}, \overline{\rho}^{H(j)} \star \xi \models [\mathrm{B}]\,\theta$, we have that $\mathcal{K}, \overline{\rho}^{H(h)} \models \theta$. By the inductive hypothesis, $\mathcal{K}, \overline{\pi}^h \models \theta$.

Therefore, $\mathcal{K}, \overline{\pi}^j \star \xi \models [\mathrm{B}]\,\theta$ holds.

- $\psi = \langle \mathrm{B} \rangle\,\theta$. We have to show that there exists a proper prefix of $\overline{\pi}^j \star \xi$ satisfying θ. Since $\mathcal{K}, \overline{\rho}^{H(j)} \star \xi \models \psi$, there exists a proper prefix η' of $\overline{\rho}^{H(j)} \star \xi$ such that $\mathcal{K}, \eta' \models \theta$. We distinguish two cases:
  – η' is *not* a proper prefix of $\overline{\rho}^{H(j)}$. Hence, η' is of the form $\overline{\rho}^{H(j)} \star \xi^h$ for some $h \in [1, |\xi| - 1]$. By the inductive hypothesis, $\mathcal{K}, \overline{\pi}^j \star \xi^h \models \theta$, hence $\mathcal{K}, \overline{\pi}^j \star \xi \models \langle \mathrm{B} \rangle\,\theta$.
  – η' is a proper prefix of $\overline{\rho}^{H(j)}$. Hence, $\eta' = \overline{\rho}^i$ for some $i \in [1, H(j) - 1]$, and $\mathcal{K}, \overline{\rho}^i \models \theta$. Let i' be the smallest position of $\overline{\rho}$ such that $\mathcal{K}, \overline{\rho}^{i'} \models \theta$. Hence $i' \leq i$ and $i' \in Wt(\varphi, \overline{\rho})$. By Property 4 of H, $i' = H(h)$ for some $\overline{\pi}$-position h. Since $H(h) < H(j)$, it holds that $h < j$ (Property 1). By the inductive hypothesis, $\mathcal{K}, \overline{\pi}^h \models \theta$, and thus $\mathcal{K}, \overline{\pi}^j \star \xi \models \langle \mathrm{B} \rangle\,\theta$.

  Therefore, in both cases, $\mathcal{K}, \overline{\pi}^j \star \xi \models \langle \mathrm{B} \rangle\,\theta$.
- $\psi = [\overline{\mathrm{B}}]\,\theta$ or $\psi = \langle \overline{\mathrm{B}} \rangle\,\theta$. The thesis directly follows from the inductive hypothesis.
- $\psi = [\mathrm{A}]\,\theta$, $\psi = \langle \mathrm{A} \rangle\,\theta$, $\psi = [\overline{\mathrm{A}}]\,\theta$ or $\psi = \langle \overline{\mathrm{A}} \rangle\,\theta$. Since $\overline{\pi}^j \star \xi$ and $\overline{\rho}^{H(j)} \star \xi$ start at the same state and lead to the same state (by Property 2 and 3 of H), the result trivially follows. This concludes the proof of the claim. □

We have shown that $\mathcal{K}, \overline{\pi} \models \varphi$, with $|\pi| < |\rho|$. If $|\pi| \leq |S| \cdot (|\varphi| + 1) \cdot 2^{2 \sum_{\ell=1}^{u} |r_\ell|}$, the thesis holds. Otherwise, we can iterate the above contraction a finite number of times until the bound is reached. □

E.10 Proof of Theorem 7.4.5

Theorem (7.4.5). *Let Φ be a $\mathrm{B}\overline{\mathrm{B}}$ formula, ψ be a subformula of Φ, and $\rho \in \mathrm{Trc}_{\mathcal{K}}$ be a trace with $s = \mathrm{lst}(\rho)$. Let G be the subset of formulas in $\mathrm{Subf}_{\langle \mathrm{B} \rangle}(\psi)$ that hold on some proper prefix of ρ. Let $\mathcal{D}(\Phi)$ be the current configuration of the DFAs associated with the regular expressions in Φ after reading $\mu(\rho(1, |\rho|-1))$. Then $\texttt{Check}(\mathcal{K}, \psi, s, G, \mathcal{D}(\Phi)) = \top \iff \mathcal{K}, \rho \models \psi$.*

Proof. The proof is by induction on the structure of ψ. The thesis trivially follows for the cases $\psi = r$ (regular expression), $\psi = \neg\psi'$, $\psi = \psi_1 \wedge \psi_2$, and $\psi = \langle \mathrm{B} \rangle\,\psi'$.

Let us now assume $\psi = \langle \overline{\mathrm{B}} \rangle\,\psi'$. $\texttt{Check}(\mathcal{K}, \psi, s, G, \mathcal{D}(\Phi)) = \top$ if and only if, for some $b'' \in \{1, \ldots, |S| \cdot (2|\psi'| + 1) \cdot 2^{2 \sum_{\ell=1}^{u} |r_\ell|} - 1\}$ and some $(G'', \mathcal{D}(\Phi)'', s'') \in \mathrm{Conf}(\mathcal{K}, \psi)$ $(= \mathrm{Conf}(\mathcal{K}, \psi'))$, we have $\texttt{Reach}(\mathcal{K}, \psi', (G, \mathcal{D}(\Phi), s), (G'', \mathcal{D}(\Phi)'', s''),$

$b'') = \top$ and $\texttt{Check}(\mathcal{K}, \psi', s'', G'', \mathcal{D}(\Phi)'') = \top$. We preliminary prove the following claim.

> *Claim* E.10.1. Let $b \in \mathbb{N}$, $b > 0$. Let $\tilde{\rho} \in \mathrm{Trc}_{\mathcal{K}}$ be a trace with $\tilde{s} = \mathrm{lst}(\tilde{\rho})$. Let $\tilde{G}$ be the subset of formulas in $\mathrm{Subf}_{\langle \mathrm{B} \rangle}(\psi')$ that hold on some proper prefix of $\tilde{\rho}$. Let $\tilde{\mathcal{D}}(\Phi)$ be the current configuration of states of the DFAs associated with the regular expressions in Φ, reached from the initial states after reading $\mu(\tilde{\rho}(1, |\tilde{\rho}| - 1))$.
> For $(\tilde{G}, \tilde{\mathcal{D}}(\Phi), \tilde{s}), (G', \mathcal{D}(\Phi)', s') \in \mathrm{Conf}(\mathcal{K}, \psi')$, we have $\texttt{Reach}(\mathcal{K}, \psi', (\tilde{G}, \tilde{\mathcal{D}}(\Phi), \tilde{s}), (G', \mathcal{D}(\Phi)', s'), b) = \top$ if and only if there exists $\rho' \in \mathrm{Trc}_{\mathcal{K}}$ such that $\tilde{\rho} \cdot \rho' \in \mathrm{Trc}_{\mathcal{K}}$, $|\rho'| = b$, $\mathrm{lst}(\rho') = s'$, G' is the subset of formulas in $\mathrm{Subf}_{\langle \mathrm{B} \rangle}(\psi')$ that hold on some proper prefix of $\tilde{\rho} \cdot \rho'$, and $\mathcal{D}(\Phi)'$ is the current configuration of the DFAs associated with the regular expressions of Φ, after reading $\mu(\tilde{\rho} \cdot \rho'(1, |\tilde{\rho} \cdot \rho'| - 1))$.

Proof. The proof is by induction on $b \geq 1$.

If $b = 1$ we have $\texttt{Reach}(\mathcal{K}, \psi', (\tilde{G}, \tilde{\mathcal{D}}(\Phi), \tilde{s}), (G', \mathcal{D}(\Phi)', s'), b) = \top$ if and only if $\texttt{Compatible}(\mathcal{K}, \psi', (\tilde{G}, \tilde{\mathcal{D}}(\Phi), \tilde{s}), (G', \mathcal{D}(\Phi)', s')) = \top$; this happens if and only if:

1. $(\tilde{s}, s') \in R$, i.e., $(\tilde{s}, s')$ is an edge of $\mathcal{K}$;
2. $\texttt{advance}(\tilde{\mathcal{D}}(\Phi), \mu(\tilde{s})) = \mathcal{D}(\Phi)'$;
3. $\tilde{G} \subseteq G'$;
4. for each $\varphi \in (G' \setminus \tilde{G})$, $\texttt{Check}(\mathcal{K}, \varphi, \tilde{s}, \tilde{G} \cap \mathrm{Subf}_{\langle \mathrm{B} \rangle}(\varphi), \tilde{\mathcal{D}}(\Phi)) = \top$;
5. for each $\varphi \in (\mathrm{Subf}_{\langle \mathrm{B} \rangle}(\psi') \setminus G')$, $\texttt{Check}(\mathcal{K}, \varphi, \tilde{s}, \tilde{G} \cap \mathrm{Subf}_{\langle \mathrm{B} \rangle}(\varphi), \tilde{\mathcal{D}}(\Phi)) = \bot$.

Let $\rho' = s'$. ($\Rightarrow$) By the inductive hypothesis (of the external theorem over $\tilde{\rho}$), by (4.) it follows that $\mathcal{K}, \tilde{\rho} \models \varphi$ for each $\varphi \in (G' \setminus \tilde{G})$. By (5.) it follows that $\mathcal{K}, \tilde{\rho} \not\models \varphi$ for each $\varphi \in (\mathrm{Subf}_{\langle \mathrm{B} \rangle}(\psi') \setminus G')$ and the claim follows.

($\Leftarrow$) Conversely (1.), (2.), and (3.) easily follow. Moreover, it must hold that $\mathcal{K}, \tilde{\rho} \models \varphi$ for each $\varphi \in (G' \setminus \tilde{G})$, and $\mathcal{K}, \tilde{\rho} \not\models \varphi$ for each $\varphi \in (\mathrm{Subf}_{\langle \mathrm{B} \rangle}(\psi') \setminus G')$ and, therefore, (4.) and (5.) follow by the inductive hypothesis (of the external theorem).

If $b \geq 2$, $\texttt{Reach}(\mathcal{K}, \psi', (\tilde{G}, \tilde{\mathcal{D}}(\Phi), \tilde{s}), (G', \mathcal{D}(\Phi)', s'), b) = \top$ if and only if, for some $(G_3, \mathcal{D}(\Phi)_3, s_3) \in \mathrm{Conf}(\mathcal{K}, \psi')$, $\texttt{Reach}(\mathcal{K}, \psi', (\tilde{G}, \tilde{\mathcal{D}}(\Phi), \tilde{s}), (G_3, \mathcal{D}(\Phi)_3, s_3), \lfloor b/2 \rfloor) = \top$ and $\texttt{Reach}(\mathcal{K}, \psi', (G_3, \mathcal{D}(\Phi)_3, s_3), (G', \mathcal{D}(\Phi)', s'), b - \lfloor b/2 \rfloor) = \top$.

($\Rightarrow$) By the inductive hypothesis (over b), there exists $\rho_3 \in \mathrm{Trc}_{\mathcal{K}}$ such that $\tilde{\rho} \cdot \rho_3 \in \mathrm{Trc}_{\mathcal{K}}$, $|\rho_3| = \lfloor b/2 \rfloor$, $\mathrm{lst}(\rho_3) = s_3$, G_3 is the subset of subformulas in $\mathrm{Subf}_{\langle \mathrm{B} \rangle}(\psi')$ that hold on some proper prefix of $\tilde{\rho} \cdot \rho_3$, and $\mathcal{D}(\Phi)_3$ is the current configuration of the DFAs associated with the regular expressions in Φ, after reading $\mu(\tilde{\rho} \cdot \rho_3(1, |\tilde{\rho} \cdot \rho_3| - 1))$.

By the inductive hypothesis (over b, applied to the trace $\tilde{\rho} \cdot \rho_3$), there exists $\rho' \in \mathrm{Trc}_{\mathcal{K}}$ such that $\tilde{\rho} \cdot \rho_3 \cdot \rho' \in \mathrm{Trc}_{\mathcal{K}}$, $|\rho'| = b - \lfloor b/2 \rfloor$, $\mathrm{lst}(\rho') = s'$, G' is the subset of subformulas in $\mathrm{Subf}_{\langle \mathrm{B} \rangle}(\psi')$ that hold on some proper prefix of $\tilde{\rho} \cdot \rho_3 \cdot \rho'$,

and $\mathcal{D}(\Phi)'$ is the current configuration of the DFAs associated with the regular expressions in Φ, after reading $\mu(\tilde{\rho} \cdot \rho_3 \cdot \rho'(1, |\tilde{\rho} \cdot \rho_3 \cdot \rho'| - 1))$. The claim follows, as $\rho_3 \cdot \rho' \in \mathrm{Trc}_{\mathcal{K}}$ and $|\rho_3 \cdot \rho'| = b$.

($\Leftarrow$) Conversely, there exists $\rho' \in \mathrm{Trc}_{\mathcal{K}}$ such that $\tilde{\rho} \cdot \rho' \in \mathrm{Trc}_{\mathcal{K}}$, $|\rho'| = b \geq 2$, $\mathrm{lst}(\rho') = s'$, G' is the subset of subformulas in $\mathrm{Subf}_{\langle \mathrm{B} \rangle}(\psi')$ that hold on some proper prefix of $\tilde{\rho} \cdot \rho'$, and $\mathcal{D}(\Phi)'$ is the current configuration of the DFAs associated with the regular expressions in Φ, after reading $\mu(\tilde{\rho} \cdot \rho'(1, |\tilde{\rho} \cdot \rho'| - 1))$. Let us split $\rho' = \rho_3 \cdot \rho_4$, where $|\rho_3| = \lfloor b/2 \rfloor$ and $|\rho_4| = b - \lfloor b/2 \rfloor$. Let $(G_3, \mathcal{D}(\Phi)_3, s_3) \in \mathrm{Conf}(\mathcal{K}, \psi')$ be such that $\mathcal{D}(\Phi)_3$ is the current configuration of the DFAs associated with the regular expressions in Φ, after reading $\mu(\tilde{\rho} \cdot \rho_3(1, |\tilde{\rho} \cdot \rho_3| - 1))$, $s_3 = \mathrm{lst}(\rho_3)$, G_3 is the subset of subformulas in $\mathrm{Subf}_{\langle \mathrm{B} \rangle}(\psi')$ that hold on some proper prefix of $\tilde{\rho} \cdot \rho_3$. By the inductive hypothesis (on b over $\tilde{\rho} \cdot \rho_3$), $\mathtt{Reach}(\mathcal{K}, \psi', (G_3, \mathcal{D}(\Phi)_3, s_3), (G', \mathcal{D}(\Phi)', s'), b - \lfloor b/2 \rfloor) = \top$. Moreover, by the inductive hypothesis (on b over $\tilde{\rho}$), we have $\mathtt{Reach}(\mathcal{K}, \psi', (\tilde{G}, \tilde{\mathcal{D}}(\Phi), \tilde{s}), (G_3, \mathcal{D}(\Phi)_3, s_3), \lfloor b/2 \rfloor) = \top$.

Hence, both the recursive calls at line 6 return $\top$, when at line 5 $(G_3, \mathcal{D}(\Phi)_3, s_3)$ is considered by the loop. Thus, $\mathtt{Reach}(\mathcal{K}, \psi', (\tilde{G}, \tilde{\mathcal{D}}(\Phi), \tilde{s}), (G', \mathcal{D}(\Phi)', s'), b)$ returns $\top$ concluding the proof of the claim. □

($\Rightarrow$) Let us now assume that in the execution of `Check`, at lines 15–19, for some $b'' \in \{1, \ldots, |S| \cdot (2|\psi'| + 1) \cdot 2^{2 \sum_{\ell=1}^{u} |r_\ell|} - 1\}$ and some $(G'', \mathcal{D}(\Phi)'', s'') \in \mathrm{Conf}(\mathcal{K}, \psi)$ $(= \mathrm{Conf}(\mathcal{K}, \psi'))$, we have $\mathtt{Reach}(\mathcal{K}, \psi', (G, \mathcal{D}(\Phi), s), (G'', \mathcal{D}(\Phi)'', s''), b'') = \top$ and $\mathtt{Check}(\mathcal{K}, \psi', s'', G'', \mathcal{D}(\Phi)'') = \top$. By the claim above, there exists $\rho'' \in \mathrm{Trc}_{\mathcal{K}}$ such that $\rho \cdot \rho'' \in \mathrm{Trc}_{\mathcal{K}}$, $\mathrm{lst}(\rho'') = s''$, G'' is the subset of subformulas in $\mathrm{Subf}_{\langle \mathrm{B} \rangle}(\psi')$ that hold on some proper prefix of $\rho \cdot \rho''$, and $\mathcal{D}(\Phi)''$ is the current configuration of the DFAs associated with the regular expressions of Φ, after reading $\mu(\rho \cdot \rho''(1, |\rho \cdot \rho''| - 1))$. By the inductive hypothesis, since $\mathtt{Check}(\mathcal{K}, \psi', s'', G'', \mathcal{D}(\Phi)'') = \top$, we have $\mathcal{K}, \rho \cdot \rho'' \models \psi'$ implying that $\mathcal{K}, \rho \models \langle \overline{\mathrm{B}} \rangle \psi'$.

($\Leftarrow$) Conversely, if $\mathcal{K}, \rho \models \langle \overline{\mathrm{B}} \rangle \psi'$, we have $\mathcal{K}, \rho \cdot \rho'' \models \psi'$ for some $\rho'' \in \mathrm{Trc}_{\mathcal{K}}$, with $\rho \cdot \rho'' \in \mathrm{Trc}_{\mathcal{K}}$. By the exponential small-model property (Theorem 7.4.3), there exists $\rho' \in \mathrm{Trc}_{\mathcal{K}}$ such that $\mathrm{lst}(\rho'') = \mathrm{lst}(\rho')$, $|\rho'| \leq |S| \cdot (2|\psi'| + 1) \cdot 2^{2 \sum_{\ell=1}^{u} |r_\ell|} - 1$ (recall that the factor 2 in front of $|\psi'|$ is due to the fact that a formula in NNF is required), $\rho \cdot \rho' \in \mathrm{Trc}_{\mathcal{K}}$ and $\mathcal{K}, \rho \cdot \rho' \models \psi'$. Let G' be the subset of subformulas in $\mathrm{Subf}_{\langle \mathrm{B} \rangle}(\psi') = \mathrm{Subf}_{\langle \mathrm{B} \rangle}(\psi)$ that hold on some proper prefix of $\rho \cdot \rho'$, and $\mathcal{D}(\Phi)'$ be the current configuration of the DFAs associated with the regular expressions in Φ, after reading $\mu(\rho \cdot \rho'(1, |\rho \cdot \rho'| - 1))$. By the inductive hypothesis (over $\rho \cdot \rho'$), $\mathtt{Check}(\mathcal{K}, \psi', \mathrm{lst}(\rho'), G', \mathcal{D}(\Phi)') = \top$. By the claim above, $\mathtt{Reach}(\mathcal{K}, \psi', (G, \mathcal{D}(\Phi), s), (G', \mathcal{D}(\Phi)', \mathrm{lst}(\rho')), |\rho'|) = \top$, hence $\mathtt{Check}(\mathcal{K}, \psi, s, G, \mathcal{D}(\Phi)) = \top$.

This concludes the proof of the theorem. □

E.11 Proof of Theorem 7.4.6

Theorem (7.4.6). *Let $\mathcal{K} = (\mathcal{AP}, S, R, \mu, s_0)$ be a finite Kripke structure, and Φ be a* B$\overline{\text{B}}$ *formula. Then,* `CheckAux`$(\mathcal{K}, \Phi) = \top \iff \mathcal{K} \models \Phi$.

Proof. If $\mathcal{K} \models \Phi$, then for all $\rho \in \mathrm{Trc}_{\mathcal{K}}$ with $\mathrm{fst}(\rho) = s_0$, we have $\mathcal{K}, \rho \models \Phi$. Hence, $\mathcal{K}, s_0 \models \Phi$, and $\mathcal{K}, s_0 \cdot \rho' \models \Phi$ for all $s_0 \cdot \rho' \in \mathrm{Trc}_{\mathcal{K}}$, implying that $\mathcal{K}, s_0 \models [\overline{\mathrm{B}}]\,\Phi$ and $\mathcal{K}, s_0 \not\models \langle\overline{\mathrm{B}}\rangle\,\neg\Phi$. By Theorem 7.4.5, $\mathtt{Check}(\mathcal{K}, \neg\Phi, s_0, \emptyset, \mathcal{D}(\Phi)_0) = \bot$ and $\mathtt{Check}(\mathcal{K}, \langle\overline{\mathrm{B}}\rangle\,\neg\Phi, s_0, \emptyset, \mathcal{D}(\Phi)_0) = \bot$ implying that $\mathtt{CheckAux}(\mathcal{K}, \Phi) = \top$.

Conversely, if $\mathtt{CheckAux}(\mathcal{K}, \Phi) = \top$, it must be $\mathtt{Check}(\mathcal{K}, \neg\Phi, s_0, \emptyset, \mathcal{D}(\Phi)_0) = \bot$ and $\mathtt{Check}(\mathcal{K}, \langle\overline{\mathrm{B}}\rangle\,\neg\Phi, s_0, \emptyset, \mathcal{D}(\Phi)_0) = \bot$. By Theorem 7.4.5 applied to the trace $\rho = s_0$, we have $\mathcal{K}, s_0 \not\models \neg\Phi$ and $\mathcal{K}, s_0 \not\models \langle\overline{\mathrm{B}}\rangle\,\neg\Phi$, and thus $\mathcal{K} \models \Phi$. □

E.12 Proof of Theorem 7.4.9

Theorem (7.4.9). *The MC problem for* Prop *formulas extended with regular expressions over finite Kripke structures is* ***PSPACE****-hard (under polynomial-time reductions).*

Proof. Given a regular expression r with $\mathcal{L}(r) \subseteq \Sigma^*$, let us define the finite Kripke structure $\mathcal{K} = (\Sigma, \{s_0\} \cup \Sigma, R, \mu, s_0)$, where $s_0 \notin \Sigma$, $\mu(s_0) = \emptyset$, for $c \in \Sigma$, $\mu(c) = \{c\}$, and $R = \{(s_0, c) \mid c \in \Sigma\} \cup \{(c, c') \mid c, c' \in \Sigma\}$.

It is easy to see that

$$\mathcal{L}(r) = \Sigma^* \iff \mathcal{K} \models \top \cdot \overline{r},$$

where $\overline{r}$ is a RE over Σ, *syntactically* equal to r (i.e., having exactly the same structure as r). Note that even though if r and $\overline{r}$ are syntactically equal, r is a regular expression defining a finitary language over Σ, whereas $\overline{r}$ defines a finitary language *over* 2^Σ (see Section 7.1). The different notations r and $\overline{r}$ are kept to avoid confusion between the two different semantics.

We show by induction on the structure of r that, for all $w \in \Sigma^*$, $w \in \mathcal{L}(r) \iff \mathcal{K}, w \models \overline{r}$. The thesis follows as $\mathcal{K}, w \models \overline{r}$ if and only if $\mathcal{K}, s_0 \cdot w \models \top \cdot \overline{r}$.

- $r = \varepsilon$. We have $w \in \mathcal{L}(\varepsilon)$ if and only if $w = \varepsilon$, if and only if $\mu(w) \in \mathcal{L}(\overline{\varepsilon}) = \{\varepsilon\}$, if and only if $\mathcal{K}, w \models \overline{\varepsilon}$.
- $r = c \in \Sigma$. We have $w \in \mathcal{L}(c)$ if and only if $w = c$, thus $\mu(w) = \{c\} \in \mathcal{L}(\overline{c})$, and $\mathcal{K}, w \models \overline{c}$. Conversely, if $\mathcal{K}, w \models \overline{c}$, we have $\mu(w) \in \mathcal{L}(\overline{c}) = \{A \in 2^\Sigma \mid c \in A\}$. In particular $|w| = 1$. Moreover, by definition of μ, $\mu(w)$ is a singleton, hence $\mu(w) = \{c\}$. By definition of $\mathcal{K}$, we get $w = c$, thus $w \in \mathcal{L}(c)$.

- $r = r_1 \cdot r_2$. We have $w \in \mathcal{L}(r_1 \cdot r_2)$ if and only if $w = w_1 \cdot w_2$, with $w_1 \in \mathcal{L}(r_1)$ and $w_2 \in \mathcal{L}(r_2)$. By applying the inductive hypothesis, $\mathcal{K}, w_1 \models \overline{r_1}$ and $\mathcal{K}, w_2 \models \overline{r_2}$, thus $\mu(w_1) \in \mathcal{L}(\overline{r_1})$ and $\mu(w_2) \in \mathcal{L}(\overline{r_2})$. It follows that $\mu(w) = \mu(w_1) \cdot \mu(w_2) \in \mathcal{L}(\overline{r_1}) \cdot \mathcal{L}(\overline{r_2}) = \mathcal{L}(\overline{r_1 \cdot r_2})$, namely, $\mathcal{K}, w \models \overline{r_1 \cdot r_2}$. Conversely, $\mu(w) \in \mathcal{L}(\overline{r_1 \cdot r_2}) = \mathcal{L}(\overline{r_1}) \cdot \mathcal{L}(\overline{r_2})$. Hence $\mu(w_1) \in \mathcal{L}(\overline{r_1})$ and $\mu(w_2) \in \mathcal{L}(\overline{r_2})$, for some $w_1 \cdot w_2 = w$. By the inductive hypothesis, $w_1 \in \mathcal{L}(r_1)$ and $w_2 \in \mathcal{L}(r_2)$, hence $w \in \mathcal{L}(r_1 \cdot r_2)$.
- $r = r_1 \cup r_2$. We have $w \in \mathcal{L}(r_1 \cup r_2)$ if and only if $w \in \mathcal{L}(r_i)$ for some $i = 1, 2$. By the inductive hypothesis this is true if and only if $\mathcal{K}, w \models \overline{r_i}$, if and only if $\mu(w) \in \mathcal{L}(\overline{r_i})$, if and only if $\mu(w) \in \mathcal{L}(\overline{r_1 \cup r_2})$, if and only if $\mathcal{K}, w \models \overline{r_1 \cup r_2}$.
- $r = r_1^*$. The thesis trivially holds if $w = \varepsilon$. Let us now assume that $w \neq \varepsilon$. We have $w \in \mathcal{L}(r_1^*)$ if and only if, for some $t \geq 1$, $w = w_1 \cdots w_t$ and $w_\ell \in \mathcal{L}(r_1)$ for all $1 \leq \ell \leq t$. By the inductive hypothesis, $\mathcal{K}, w_\ell \models \overline{r_1}$, thus $\mu(w_\ell) \in \mathcal{L}(\overline{r_1})$, and $\mu(w) \in \mathcal{L}(\overline{r_1^*})$. We conclude that $\mathcal{K}, w \models \overline{r_1^*}$. Conversely, $\mu(w) \in \mathcal{L}(\overline{r_1^*}) = (\mathcal{L}(\overline{r_1}))^*$, hence it must be the case that, for some $t \geq 1$, $w = w_1 \cdots w_t$ and $\mu(w_\ell) \in \mathcal{L}(\overline{r_1})$ for all $1 \leq \ell \leq t$. By the inductive hypothesis, $w_\ell \in \mathcal{L}(r_1)$, hence $w \in \mathcal{L}(r_1^*)$.

Finally, by observing that $\mathcal{K}$ can be built in polynomial time, the thesis follows. □



Proofs and complements of Chapter 8

Contents

F.1 Definition of the value transition function T in the proof of Theorem 8.2.1 307
F.2 Non-primitive recursive-hardness of future TP 309
F.3 Future TP, simple trigger rules, non-sing. intervals: EXPSPACE-hardness 314

F.1 Definition of the value transition function T in the proof of Theorem 8.2.1

The value transition function T of x_M is defined as follows.

- For each instruction label $\ell \in \mathsf{Inc} \cup \{\ell_{halt}\}$, let $P_\ell = \emptyset$ if $\ell = \ell_{halt}$, and $P_\ell = \{(succ(\ell), \mathrm{inc}_h)\}$ otherwise, where $c_h = c(\ell)$. Then, $T(\ell)$, $T((\ell, c_i))$, and $T((\ell, (c_i, \#))$, for $i = 1, 2$, are defined as follows:

$$T(\ell) = \{(\ell, c_1), (\ell, c_2)\} \cup P_\ell$$

$$T((\ell, c_1)) = \{(\ell, c_1), (\ell, c_2)\} \cup P_\ell$$

$$T((\ell, c_2)) = \{(\ell, c_2)\} \cup P_\ell$$

- For each instruction label $\ell \in \text{Dec}$ and for each $\ell' \in \{\text{zero}(\ell), \text{dec}(\ell)\}$, $T((\ell, \ell'))$, $T((\ell, \ell', c_i))$, and $T((\ell, \ell', (c_i, \#))$, for $i = 1, 2$, are defined as:

$$T((\ell,\ell')) = \begin{cases} \{(\ell,\ell',c_2),(\ell',\text{zero}_1)\} & \text{if } c(\ell)=c_1, \ell'=\text{zero}(\ell) \\ \{(\ell,\ell',c_1),(\ell',\text{zero}_2)\} & \text{if } c(\ell)=c_2, \ell'=\text{zero}(\ell) \\ \{(\ell,\ell',(c_1,\#))\} & \text{if } c(\ell)=c_1, \ell'=\text{dec}(\ell) \\ \{(\ell,\ell',c_1),(\ell,\ell',(c_2,\#))\} & \text{otherwise} \end{cases}$$

$$T((\ell,\ell',c_1)) = \begin{cases} \emptyset & \text{if } c(\ell)=c_1, \ell'=\text{zero}(\ell) \\ \{(\ell,\ell',c_1),(\ell',\text{zero}_2)\} & \text{if } c(\ell)=c_2, \ell'=\text{zero}(\ell) \\ \{(\ell,\ell',c_1),(\ell,\ell',c_2),(\ell',\text{dec}_1)\} & \text{if } c(\ell)=c_1, \ell'=\text{dec}(\ell) \\ \{(\ell,\ell',c_1),(\ell,\ell',(c_2,\#))\} & \text{otherwise} \end{cases}$$

$$T((\ell,\ell',c_2)) = \begin{cases} \{(\ell,\ell',c_2),(\ell',\text{zero}_1)\} & \text{if } c(\ell)=c_1, \ell'=\text{zero}(\ell) \\ \emptyset & \text{if } c(\ell)=c_2, \ell'=\text{zero}(\ell) \\ \{(\ell,\ell',c_2),(\ell',\text{dec}_1)\} & \text{if } c(\ell)=c_1, \ell'=\text{dec}(\ell) \\ \{(\ell,\ell',c_2),(\ell',\text{dec}_2)\} & \text{otherwise} \end{cases}$$

$$T((\ell,\ell',(c_1,\#))) = \begin{cases} \{(\ell,\ell',c_1),(\ell,\ell',c_2),(\ell',\text{dec}_1)\} & \text{if } c(\ell)=c_1, \ell'=\text{dec}(\ell) \\ \emptyset & \text{otherwise} \end{cases}$$

$$T((\ell,\ell',(c_2,\#))) = \begin{cases} \{(\ell,\ell',c_2),(\ell',\text{dec}_2)\} & \text{if } c(\ell)=c_2, \ell'=\text{dec}(\ell) \\ \emptyset & \text{otherwise} \end{cases}$$

- For each label $\ell \in \text{InstLab}$ and operation $op \in \{\text{inc}_1, \text{inc}_2, \text{zero}_1, \text{zero}_2, \text{dec}_1, \text{dec}_2\}$, $T((\ell, op))$, $T((\ell, op, c_i))$, and $T((\ell, op, (c_i, \#))$, for $i = 1, 2$, are defined as follows, where $S_\ell = \{(\ell, \text{zero}(\ell)), (\ell, \text{dec}(\ell))\}$ if $\ell \in \text{Dec}$, and $S_\ell = \{\ell\}$ otherwise:

$$T((\ell,op)) = \begin{cases} \{(\ell,op,c_2)\} \cup S_\ell & \text{if } op=\text{zero}_1, \ell \neq \ell_{init} \\ \{(\ell,op,c_1)\} \cup S_\ell & \text{if } op=\text{zero}_2, \ell \neq \ell_{init} \\ \{(\ell,op,c_1),(\ell,op,c_2)\} \cup S_\ell & \text{if } op \in \{\text{dec}_1,\text{dec}_2\}, \ell \neq \ell_{init} \\ \{(\ell,op,(c_1,\#))\} & \text{if } op=\text{inc}_1, \ell \neq \ell_{init} \\ \{(\ell,op,c_1),(\ell,op,(c_2,\#))\} & \text{if } op=\text{inc}_2, \ell \neq \ell_{init} \\ \{\ell_{init}\} & \text{if } op=\text{zero}_1, \ell = \ell_{init} \\ \emptyset & \text{otherwise} \end{cases}$$

$$T((\ell,op,c_1)) = \begin{cases} \emptyset & \text{if } op=\text{zero}_1 \text{ or } \ell=\ell_{init} \\ \{(\ell,op,c_1)\} \cup S_\ell & \text{if } op=\text{zero}_2, \ell \neq \ell_{init} \\ \{(\ell,op,c_1),(\ell,op,c_2)\} \cup S_\ell & \text{if } op \in \{\text{dec}_1,\text{dec}_2,\text{inc}_1\}, \ell \neq \ell_{init} \\ \{(\ell,op,c_1),(\ell,op,(c_2,\#))\} & \text{if } op=\text{inc}_2, \ell \neq \ell_{init} \end{cases}$$

$$T((\ell,op,c_2)) = \begin{cases} \emptyset & \text{if } op=\text{zero}_2 \text{ or } \ell=\ell_{init} \\ \{(\ell,op,c_2)\} \cup S_\ell & \text{otherwise} \end{cases}$$

$$T((\ell, op, (c_1, \#))) = \begin{cases} \emptyset & \text{if } op \neq \mathsf{inc}_1 \text{ or } \ell = \ell_{init} \\ \{(\ell, op, c_1), (\ell, op, c_2)\} \cup S_\ell & \text{otherwise} \end{cases}$$

$$T((\ell, op, (c_2, \#))) = \begin{cases} \emptyset & \text{if } op \neq \mathsf{inc}_2 \text{ or } \ell = \ell_{init} \\ \{(\ell, op, c_2)\} \cup S_\ell & \text{otherwise} \end{cases}$$

This concludes the definition of T of x_M.

F.2 Non-primitive recursive-hardness of future TP

In this section, we establish the following result.

Theorem (8.2.3). *The future TP problem, even with* one state variable, *is non-primitive recursive-hard also under one of the following two assumptions:* either (1) *the trigger rules are simple,* or (2) *the intervals are in* $Intv_{(0,\infty)}$.

Theorem 8.2.3 is proved by a polynomial-time reduction from the halting problem for *gainy counter machines* [DL09], a variant of standard Minsky machines, whose counters may erroneously increase. Such a machine is a tuple $M = (Q, q_{init}, q_{halt}, n, \Delta)$, where:

- Q is a finite set of (control) locations/states, $q_{init} \in Q$ is the initial location, and $q_{halt} \in Q$ is the halting location,
- $n \in \mathbb{N} \setminus \{0\}$ is the number of counters of M, and
- $\Delta \subseteq Q \times \mathsf{L} \times Q$ is a transition relation over the instruction set $\mathsf{L} = \{\mathsf{inc}, \mathsf{dec}, \mathsf{zero}\} \times \{1, \ldots, n\}$.

We adopt the following notational conventions. For an instruction $op \in \mathsf{L}$, let $c(op) \in \{1, \ldots, n\}$ be the counter associated with op. For a transition $\delta \in \Delta$ of the form $\delta = (q, op, q')$, we define $from(\delta) = q$, $op(\delta) = op$, $c(\delta) = c(op)$, and $to(\delta) = q'$. We denote by op_{init} the instruction (zero, 1). W.l.o.g., we make these assumptions:

- for each transition $\delta \in \Delta$, $from(\delta) \neq q_{halt}$ and $to(\delta) \neq q_{init}$, and
- there is exactly one transition in Δ, denoted δ_{init}, having as source the initial location q_{init}.

An *M-configuration* is a pair (q, ν) consisting of a location $q \in Q$ and a counter valuation $\nu : \{1, \ldots, n\} \to \mathbb{N}$. Given two valuations ν and ν', we write $\nu \geq \nu'$ if and only if $\nu(c) \geq \nu'(c)$ for all $c \in \{1, \ldots, n\}$.

Under the *exact semantics* (with no errors), M induces a transition relation, denoted by $\longrightarrow$, over pairs of M-configurations and instructions, defined as follows: for configurations (q, ν) and (q', ν'), and instructions $op \in \mathsf{L}$, we have $(q, \nu) \xrightarrow{op} (q', \nu')$ if the following holds, where $c \in \{1, \ldots, n\}$ is the counter associated with the instruction op:

- $(q, op, q') \in \Delta$ and $\nu'(c') = \nu(c')$ for all $c' \in \{1, \ldots, n\} \setminus \{c\}$;
- $\nu'(c) = \nu(c) + 1$ if $op = (\mathsf{inc}, c)$;
- $\nu'(c) = \nu(c) - 1$ if $op = (\mathsf{dec}, c)$ (in particular, it has to be $\nu(c) > 0$);
- $\nu'(c) = \nu(c) = 0$ if $op = (\mathsf{zero}, c)$.

The *gainy semantics* is obtained from the exact one by allowing *increment* errors. Formally, M induces a transition relation, denoted by $\longrightarrow_{gainy}$, defined as follows: for configurations (q, ν) and (q', ν'), and instructions $op \in \mathsf{L}$, we have $(q, \nu) \xrightarrow{op}_{gainy} (q', \nu')$ if the following holds, where $c = c(op)$ is the counter associated with the instruction op: $(q, \nu) \xrightarrow{op}_{gainy} (q', \nu')$ iff there are valuations ν_+ and ν'_+ such that $\nu_+ \geq \nu$, $(q, \nu_+) \xrightarrow{op} (q', \nu'_+)$, and $\nu' \geq \nu'_+$. Equivalently, $(q, \nu) \xrightarrow{op}_{gainy} (q', \nu')$ iff the following conditions hold:

- $(q, op, q') \in \Delta$ and $\nu'(c') \geq \nu(c')$ for all $c' \in \{1, \ldots, n\} \setminus \{c\}$;
- $\nu'(c) \geq \nu(c) + 1$ if $op = (\mathsf{inc}, c)$;
- $\nu'(c) \geq \nu(c) - 1$ if $op = (\mathsf{dec}, c)$;
- $\nu(c) = 0$ if $op = (\mathsf{zero}, c)$.

A (gainy) *M-computation* is a finite sequence of the form:

$$(q_0, \nu_0) \xrightarrow{op_0}_{gainy} (q_1, \nu_1) \xrightarrow{op_1}_{gainy} \cdots \xrightarrow{op_{k-1}}_{gainy} (q_k, \nu_k).$$

M halts if there exists an M-computation starting at the *initial* configuration (q_{init}, ν_{init}), where $\nu_{init}(c) = 0$ for all $c \in \{1, \ldots, n\}$, and leading to some halting configuration (q_{halt}, ν). Given a gainy counter machine M, *the halting problem for M* is to decide whether M halts, and it was shown to be decidable and non-primitive recursive [DL09].

We now prove the following result, from which Theorem 8.2.3 directly follows.

Proposition F.2.1. *One can construct in polynomial time a TP domain $P = (\{x_M\}, R_M)$ where the trigger rules in R_M are simple (resp., the intervals in P are in $Intv_{(0,\infty)}$) such that M halts iff there is a future plan for P.*

Proof. We focus on the reduction where the intervals in P are in $Intv_{(0,\infty)}$. At the end of the proof, we show how to adapt the construction for the case of simple trigger rules with arbitrary intervals.

Encoding of M-computations. First, we define a suitable encoding of a computation of M as a timeline for x_M. For this, we exploit the finite set of symbols $V = V_{main} \cup V_{sec} \cup V_{dummy}$ corresponding to the finite domain of the state variable x_M. The set of *main* values V_{main} is given by

$$V_{main} = \{(\delta, op) \in \Delta \times \mathsf{L} \mid op \neq (\mathsf{inc}, c) \text{ if } op(\delta) = (\mathsf{zero}, c)\}.$$

Intuitively, in the encoding of an M-computation, a main value (δ, op) keeps track of the transition δ used in the current step of the computation, while op represents the instruction exploited in the previous computation step (if any).

The set of *secondary* values V_{sec} is defined as

$$V_{sec} = V_{main} \times \{1, \ldots, n\} \times 2^{\{\#_{\mathsf{inc}}, \#_{\mathsf{dec}}\}},$$

where $\#_{\mathsf{inc}}$ and $\#_{\mathsf{dec}}$ are two special symbols used as markers. V_{sec} is used for encoding counter values, as shown later. Finally, the set of *dummy values* is $V_{dummy} = (V_{main} \cup V_{sec}) \times \{dummy\}$; their use will be clear when we introduce synchronization rules: they are used to specify punctual time constraints by means of non-simple trigger rules over intervals in $Intv_{(0,\infty)}$.

Given a word $w \in V^*$, we denote by $||w||$ the length of the word obtained from w by removing dummy symbols.

For $c \in \{1, \ldots, n\}$ and $v_{main} = (\delta, op) \in V_{main}$, the *set* $Tag(c, v_{main})$ *of markers of counter* c *for the main value* v_{main} is the subset of $\{\#_{\mathsf{inc}}, \#_{\mathsf{dec}}\}$ defined as follows:

- $\#_{\mathsf{inc}} \in Tag(c, v_{main})$ iff $op = (\mathsf{inc}, c)$;
- $\#_{\mathsf{dec}} \in Tag(c, v_{main})$ iff $op(\delta) = (\mathsf{dec}, c)$;

A *c-code for the main value* $v_{main} = (\delta, op)$ is a finite word w_c over V such that *either* (i) w_c is empty and $\#_{\mathsf{inc}} \notin Tag(c, v_{main})$, *or* (ii) $op(\delta) \neq (\mathsf{zero}, c)$ and $w_c = (v_{main}, c, Tag(c, v_{main}))(v_{main}, c, \emptyset, dummy)^{h_0} \cdot (v_{main}, c, \emptyset) \cdot (v_{main}, c, \emptyset, dummy)^{h_1} \cdots (v_{main}, c, \emptyset) \cdot (v_{main}, c, \emptyset, dummy)^{h_n}$ for some $n \geq 0$ and $h_0, h_1, \ldots, h_n \geq 0$. The c-code w_c encodes the value for the counter c given by $||w_c||$. Intuitively, w_c can be seen as an interleaving of secondary values with dummy ones, the latter being present only for technical aspects, but not encoding any counter value.

A *configuration-code* w *for a main value* $v_{main} = (\delta, op) \in V_{main}$ is a finite word over V of the form $w = v_{main} \cdot (v_{main}, dummy)^h \cdot w_1 \cdots w_n$, where $h \geq 0$ and for each counter $c \in \{1, \ldots, n\}$, w_c is a c-code for the main value v_{main}. The configuration-code w encodes the M-configuration $(from(\delta), \nu)$, where $\nu(c) = ||w_c||$ for all $c \in \{1, \ldots, n\}$. Note that if $op(\delta) = (\mathsf{zero}, c)$, then $\nu(c) = 0$ and $op \neq (\mathsf{inc}, c)$.

The marker $\#_{\mathsf{inc}}$ occurs in w iff op is an increment instruction, and in such a case $\#_{\mathsf{inc}}$ marks the *first* symbol of the encoding $w_{c(op)}$ of counter $c(op)$. Intuitively, if the operation performed in the previous step of the computation increments counter c, then the tag $\#_{\mathsf{inc}}$ "marks" the unit of the counter c in the current configuration which has been added by the increment.

The marker $\#_{\mathsf{dec}}$ occurs in w iff δ is a decrement instruction and the value of counter $c(\delta)$ in w is non-zero; in such a case, $\#_{\mathsf{dec}}$ marks the *first* symbol of the encoding $w_{c(\delta)}$ of counter $c(\delta)$. Intuitively, if the operation to be performed in the current step decrements counter c and the current value of c is non-zero, then the tag $\#_{\mathsf{dec}}$ marks the unit of the counter c in the current configuration which has to be removed by the decrement.

A *computation*-code is a sequence of configuration-codes $\pi = w_{(\delta_0, op_0)} \cdots w_{(\delta_k, op_k)}$, where, for all $0 \leq i \leq k$, $w_{(\delta_i, op_i)}$ is a configuration-code with main value (δ_i, op_i), and whenever $i < k$, it holds that $to(\delta_i) = from(\delta_{i+1})$ and $op(\delta_i) = op_{i+1}$. Note that by our assumptions $to(\delta_i) \neq q_{halt}$ for all $0 \leq i < k$, and $\delta_j \neq \delta_{init}$ for all $0 < j \leq k$. The computation-code π is *initial* if the first configuration-code $w_{(\delta_0, op_0)}$ is $(\delta_{init}, op_{init})$ (which encodes the initial configuration), and it is *halting* if for the last configuration-code $w_{(\delta_k, op_k)}$ in π, it holds that $to(\delta_k) = q_{halt}$. For all $0 \leq i \leq k$, let (q_i, v_i) be the M-configuration encoded by the configuration-code $w_{(\delta_i, op_i)}$ and $c_i = c(\delta_i)$. The computation-code π is *well-formed* if, additionally, for all $0 \leq j \leq k-1$, the following conditions hold:

- $v_{j+1}(c) \geq v_j(c)$ for all $c \in \{1, \ldots, n\} \setminus \{c_j\}$ (*gainy monotonicity*);
- $v_{j+1}(c_j) \geq v_j(c_j) + 1$ if $op(\delta_j) = (\mathsf{inc}, c_j)$ (*increment requirement*);
- $v_{j+1}(c_j) \geq v_j(c_j) - 1$ if $op(\delta_j) = (\mathsf{dec}, c_j)$ (*decrement requirement*).

Clearly, M halts *iff* there is an initial and halting well-formed computation-code.

Definition of x_M and R_M. We now define a state variable x_M and a set R_M of synchronization rules for x_M with intervals in $Intv_{(0,\infty)}$ such that the untimed part of any *future plan* for $P = (\{x_M\}, R_M)$ is an initial and halting well-formed computation-code. Thus, M halts if and only if there is a future plan of P.

Formally, the state variable x_M is given by $x_M = (V, T, D)$ where, for each $v \in V$, $D(v) =]0, \infty[$ if $v \notin V_{dummy}$, and $D(v) = [0, \infty[$ otherwise: we require that the duration of a non-dummy token is always greater than zero (*strict time monotonicity*).

The value transition function T of x_M ensures the following requirement.

Claim F.2.2. The untimed part of any timeline for x_M whose first token has value $(\delta_{init}, op_{init})$ corresponds to a prefix of some initial computation-code. Moreover, $(\delta_{init}, op_{init}) \notin T(v)$ for all $v \in V$.

T can be built by adapting the construction of Appendix F.1.

Let $V_{halt} = \{(\delta, op) \in V_{main} \mid to(\delta) = q_{halt}\}$. By Claim F.2.2 and the assumption that $from(\delta) \neq q_{halt}$ for each transition $\delta \in \Delta$, to ensure the initialization and halting requirements, it suffices to enforce the timeline to feature a token with value $(\delta_{init}, op_{init})$ and a token with value in V_{halt}. This is captured by the trigger-less rules

$$\top \longrightarrow \exists o[x_M = (\delta_{init}, op_{init})].\, \top$$

and

$$\top \longrightarrow \bigvee_{v \in V_{halt}} \exists o[x_M = v].\, \top\,.$$

The crucial well-formedness requirement is captured by the trigger rules in R_M which express the following punctual time constraints. Note that we take advantage of the dense temporal domain to allow for the encoding of arbitrarily large values of counters in two time units.

- *2-Time distance between consecutive main values:* the overall duration of the sequence of tokens corresponding to a configuration-code amounts exactly to 2 time units. By Claim F.2.2, strict time monotonicity, and the halting requirement, it suffices to ensure that each token tk having a main value in $V_{main} \setminus V_{halt}$ is eventually followed by a token tk' such that tk' has a main value and $\mathrm{s}(tk') - \mathrm{s}(tk) = 2$. To this aim, for each $v \in V_{main} \setminus V_{halt}$, we have the following non-simple trigger rule with intervals in $Intv_{(0,\infty)}$ which uses a dummy token for capturing the punctual time constraint:

$$o[x_M = v] \rightarrow \bigvee_{u \in V_{main}} \bigvee_{u_d \in V_{dummy}} \exists o'[x_M = u] \exists o_d[x_M = u_d].o \leq^{\mathrm{s,s}}_{[1,+\infty[} o_d \wedge o_d \leq^{\mathrm{s,s}}_{[1,+\infty[} o' \wedge o \leq^{\mathrm{s,s}}_{[0,2]} o'.$$

- For a counter $c \in \{1, \ldots, n\}$, let us denote as $V_c \subseteq V_{sec}$ the set of secondary values given by $V_{main} \times \{c\} \times 2^{\{\#_{\mathrm{inc}}, \#_{\mathrm{dec}}\}}$. We require that each token tk with a V_c-value of the form $((\delta, op), c, Tag)$ such that $c \neq c(\delta)$ and $to(\delta) \neq q_{halt}$ is eventually followed by a token tk' with a V_c-value such that $\mathrm{s}(tk') - \mathrm{s}(tk) = 2$. Note that our encoding, Claim F.2.2, strict time monotonicity, and 2-Time distance between consecutive main values guarantee that the previous requirement captures *gainy monotonicity*. Thus, for each counter c and $v \in V_c$ such that v is of the form $((\delta, op), c, Tag)$, where $c \neq c(\delta)$ and $to(\delta) \neq q_{halt}$, we have the following non-simple trigger rule over $Intv_{(0,\infty)}$:

$$o[x_M = v] \rightarrow \bigvee_{u \in V_c} \bigvee_{u_d \in V_{dummy}} \exists o'[x_M = u] \exists o_d[x_M = u_d].o \leq^{\mathrm{s,s}}_{[1,+\infty[} o_d \wedge o_d \leq^{\mathrm{s,s}}_{[1,+\infty[} o' \wedge o \leq^{\mathrm{s,s}}_{[0,2]} o'.$$

- For capturing the increment and decrement requirements, by construction, it suffices to enforce that:
  1. each token tk with a V_c-value of the form $((\delta, op), c, Tag)$ such that $to(\delta) \neq q_{halt}$ and $\delta = (\mathrm{inc}, c)$ is eventually followed by a token tk' with a V_c-value which is *not* marked by $\#_{\mathrm{inc}}$ such that $\mathrm{s}(tk') - \mathrm{s}(tk) = 2$;
  2. each token tk with a V_c-value of the form $((\delta, op), c, Tag)$ such that $to(\delta) \neq q_{halt}$, $\delta = (\mathrm{dec}, c)$, and $\#_{\mathrm{dec}} \notin Tag$ is eventually followed by a token tk' with a V_c-value such that $\mathrm{s}(tk') - \mathrm{s}(tk) = 2$. These requirements can be expressed by non-simple trigger rules with intervals in $Intv_{(0,\infty)}$ similar to the previous ones.

Finally, to prove Proposition F.2.1 for the case of simple trigger rules with arbitrary intervals, it suffices to remove the dummy values and replace the conjunction $o \leq^{s,s}_{[1,+\infty[} o_d \wedge o_d \leq^{s,s}_{[1,+\infty[} o' \wedge o \leq^{s,s}_{[0,2]} o'$ in the previous trigger rules with the "punctual" atom $o \leq^{s,s}_{[2,2]} o'$, whose interval at the subscript is singular.

This concludes the proof of Proposition F.2.1. □

F.3 Future TP with simple trigger rules and non-singular intervals: EXPSPACE-hardness

Here we show that the future TP problem with simple trigger rules and non-singular intervals is **EXPSPACE**-hard. The claim is proved by a polynomial-time reduction from the domino-tiling problem for grids with rows of single exponential length [Har12] (see Section 2.4 for the definition and notation). Hardness holds also when only a single state variable is involved.

Theorem (8.4.1). *The future TP problem, even with* one state variable*, with simple trigger rules and non-singular intervals is **EXPSPACE**-hard (under polynomial-time reductions).*

Proof. For the sake of the reduction, we define the state variable $y = (V, T, D)$ where (we recall that Δ represents the set of domino-types and $d, d', d_{final} \in \Delta$):

- $V = \{\$, \$'\} \cup \Delta$ (with $\$, \$' \notin \Delta$),
- $T(\$) = \Delta$ and $T(\$') = \{\$'\}$,
- for $d \in \Delta \setminus \{d_{final}\}$, $T(d) = \{\$\} \cup \{d' \in \Delta \mid [d]_{right} = [d']_{left}\}$,
- $T(d_{final}) = \{\$, \$'\} \cup \{d' \in \Delta \mid [d_{final}]_{right} = [d']_{left}\}$,
- for all $v \in V$, $D(v) = [2, +\infty[$.

Basically, the domain of the state variable y contains all domino-types, as well as two auxiliary symbols \$ and \$′. The idea is encoding a tiling by the concatenation of its rows, separated by an occurrence of \$. The last row is terminated by \$′.

More precisely, each cell of the grid is encoded by (the value of) a token having duration 2. A row of the grid is then represented by the sequence of tokens of its cells, ordered by increasing column index. Finally, a full tiling is just given by the timeline for y obtained by concatenating the sequences of tokens of all rows, ordered by increasing row index. See Figure F.1 for an example.

We observe that T guarantees the horizontal constraint among domino-types, and that it allows only occurrences of \$′ after the first \$′.

We start with the next simple trigger rules, one for each $v \in V$:

$$o[y = v] \rightarrow o \leq^{s,e}_{[0,2]} o.$$

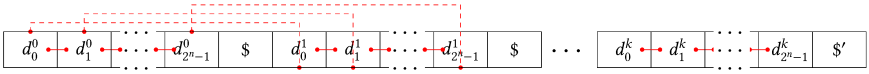


Figure F.1: A timeline encoding the ordered concatenation of the rows of a tiling. Red lines represent the horizontal and vertical constraints among domino-types.

These, paired with the constraint function D, enforce all tokens' durations to be exactly 2. This is done for technical convenience: intuitively, since we exclude singular intervals, requiring, for instance, that a token o' starts t instants of time after the end of o, with $t \in [\ell, \ell+1]$ and even $\ell \in \mathbb{N}$, boils down to o' starting *exactly* ℓ instants after the end of o. We also observe that, given the constant token duration, in this proof the density of the time domain does not play any role.

We now define the following synchronization rules (of which all trigger ones are simple and future). The next ones state (together) that the *first occurrence* of (a token having value) \$ starts exactly at $2 \cdot 2^n$:

$$\top \longrightarrow \exists o[y = \$].o \geq^{\mathsf{s}}_{[0,1]} 2 \cdot 2^n, \tag{F.1}$$

and

$$o[y = \$] \longrightarrow o \geq^{\mathsf{s}}_{[0,+\infty[} 2 \cdot 2^n. \tag{F.2}$$

Thus, all tokens before such a first occurrence of \$ have a value in Δ.

Every occurrence of \$ must be followed, after exactly $2 \cdot 2^n$ instants of time (namely, after 2^n tokens), by another occurrence of \$ or of \$′.

$$o[y = \$] \longrightarrow (\exists o'[y = \$].o \leq^{\mathsf{e,s}}_{[2\cdot 2^n, 2\cdot 2^n+1]} o') \vee (\exists o''[y = \$'].o \leq^{\mathsf{e,s}}_{[2\cdot 2^n, 2\cdot 2^n+1]} o''). \tag{F.3}$$

Now we force every token with value $d \in \Delta$ either (i) to be followed, after $2 \cdot 2^n$ instants, by another token with value $d' \in \Delta$, in particular, satisfying the vertical requirement, i.e., $[d]_{up} = [d']_{down}$, or (ii) to be in the last row (which is terminated by \$′). For each $d \in \Delta$,

$$o[y = d] \longrightarrow \left(\bigvee_{d' \in \Delta,\, [d]_{up} = [d']_{down}} \exists o'[y = d'].o \leq^{\mathsf{e,s}}_{[2\cdot 2^n, 2\cdot 2^n+1]} o' \right) \vee (\exists o''[y = \$'].o \leq^{\mathsf{e,s}}_{[0, 2\cdot 2^n-2]} o''). \tag{F.4}$$

It is straightforward to check that rules (F.1), (F.2), (F.3), and (F.4), along with the horizontal constraint guaranteed by the function T, enforce the following property.

Proposition F.3.1. *There exists $k' \in \mathbb{N}^+$ such that all tokens with value \$ end at all and only times $k \cdot 2(2^n + 1)$, for $1 \leq k < k'$. Moreover the first token with value \$′ ends at time $k' \cdot 2(2^n + 1)$. Finally, all other tokens having end time less than $k' \cdot 2(2^n + 1)$ have value in Δ and satisfy the horizontal and vertical constraints.*

Finally, we settle the initialization and acceptance requirements by means of the following pair of trigger-less rules:

$$\top \rightarrow \exists o[y = d_{Init}].o \geq^{\mathrm{s}}_{[0,1]} 0,$$

$$\top \rightarrow \exists o[y = d_{final}]\exists o'[y = \$'].o \leq^{\mathrm{e,s}}_{[0,1]} o'.$$

The former rule states that a token with value d_{Init} must start at $t = 0$, the latter that a token with value d_{final} must occur just before the terminator of the last row \$′.

To conclude the proof, we observe that the state variable $y = (V, T, D)$ as well as all synchronization rules can be generated in polynomial time in the size of the instance $\mathcal{I}$ of the domino-tiling problem (in particular, note that all interval bounds and time constants of time-point atoms have a value, encoded in binary, which is in $O(2^n)$). □

Bibliography

[All83] J. F. Allen. “Maintaining Knowledge about Temporal Intervals”. In: *Communications of the ACM* 26.11 (1983), pp. 832–843. DOI: 10.1145/182.358434 (cit. on p. 11).

[ACZ06] R. Alur, P. Cerný, and S. Zdancewic. “Preserving Secrecy Under Refinement”. In: *Proceedings of ICALP*. Springer, 2006, pp. 107–118. DOI: 10.1007/11787006_10 (cit. on p. 140).

[AD94] R. Alur and D. L. Dill. “A theory of timed automata”. In: *Theoretical Computer Science* 126.2 (1994), pp. 183–235. DOI: 10.1016/0304-3975(94)90010-8 (cit. on pp. 210, 212, 219, 221, 222, 227, 235).

[AFH96] R. Alur, T. Feder, and T. A. Henzinger. “The Benefits of Relaxing Punctuality”. In: *Journal of the ACM* 43.1 (1996), pp. 116–146. DOI: 10.1145/227595.227602 (cit. on pp. 212, 222, 235).

[AH93] R. Alur and T. A. Henzinger. “Real-Time Logics: Complexity and Expressiveness”. In: *Information and Computation* 104.1 (1993), pp. 35–77. DOI: 10.1006/inco.1993.1025 (cit. on p. 202).

[Alu+07] R. Alur, M. Arenas, P. Barceló, K. Etessami, N. Immerman, and L. Libkin. “First-Order and Temporal Logics for Nested Words”. In: *Proceedings of LICS*. IEEE Computer Society, 2007, pp. 151–160. DOI: 10.1109/LICS.2007.19 (cit. on p. 239).

[AEM04] R. Alur, K. Etessami, and P. Madhusudan. “A Temporal Logic of Nested Calls and Returns”. In: *Proceedings of TACAS*. Springer, 2004, pp. 467–481. DOI: 10.1007/978-3-540-24730-2_35 (cit. on p. 239).

[ACC07] A. Armando, R. Carbone, and L. Compagna. “LTL Model Checking for Security Protocols”. In: *Proceedings of CSF*. IEEE Computer Society, 2007, pp. 385–396. DOI: 10.1109/CSF.2007.24 (cit. on p. 6).

[Aro+05] T. Arons, E. Elster, L. Fix, S. Mador-Haim, M. Mishaeli, J. Shalev, E. Singerman, A. Tiemeyer, M. Y. Vardi, and L. D. Zuck. “Formal Verification of Backward Compatibility of Microcode”. In: *Proceedings of CAV*. Springer, 2005, pp. 185–198. DOI: 10.1007/11513988_20 (cit. on p. 6).

[BK08] C. Baier and J. P. Katoen. *Principles of Model Checking*. Representation and Mind Series. The MIT Press, 2008. ISBN: 978-0262026499 (cit. on pp. 4, 5, 7, 147).

[BPR01] T. Ball, A. Podelski, and S. K. Rajamani. “Boolean and Cartesian Abstraction for Model Checking C Programs”. In: *Proceedings of TACAS*. Springer, 2001, pp. 268–283. DOI: 10.1007/3-540-45319-9_19 (cit. on p. 8).

[Bar+12] J. Barreiro, M. Boyce, M. Do, J. Frank, M. Iatauro, T. Kichkaylo, P. Morris, J. Ong, E. Remolina, T. Smith, and D. Smith. “EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization”. In: *Proceedings of ICKEPS*. 2012 (cit. on p. 196).

[BCM15] D. Basin, C. Cremers, and C. Meadows. *Model Checking Security Protocols*. 2015. URL: http://www-oldurls.inf.ethz.ch/personal/basin/pubs/security-modelchecking.pdf (cit. on p. 6).

[BMP09] T. Batzold, G. Morin, and J. Pecoraro. *Completeness in the Σ_2^p Hierarchy. A Compendium*. 2009 (cit. on p. 97).

[Ben+17] M. Benerecetti, R. D. Guglielmo, U. Gentile, S. Marrone, N. Mazzocca, R. Nardone, A. Peron, L. Velardi, and V. Vittorini. “Dynamic state machines for modelling railway control systems”. In: *Science of Computer Programming* 133 (2017), pp. 116–153. DOI: 10.1016/j.scico.2016.09.002 (cit. on p. 6).

[Bie+99] A. Biere, A. Cimatti, E. M. Clarke, M. Fujita, and Y. Zhu. “Symbolic model checking using SAT procedures instead of BDDs”. In: *Proceedings of DAC*. ACM. 1999, pp. 317–320. DOI: 10.1145/309847.309942 (cit. on p. 8).

[Bie+03] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu. "Bounded model checking". In: *Advances in computers* 58 (2003), pp. 117–148. DOI: 10.1016/S0065-2458(03)58003-2 (cit. on p. 8).

[BS96] P. Blackburn and J. Seligman. "What are Hybrid Languages?" In: *Proceedings of AiML*. CSLI Publications, 1996, pp. 41–62 (cit. on p. 139).

[BT03] H. Bowman and S. J. Thompson. "A Decision Procedure and Complete Axiomatization of Finite Interval Temporal Logic with Projection". In: *Journal of Logic and Computation* 13.2 (2003), pp. 195–239. DOI: 10.1093/logcom/13.2.195 (cit. on p. 11).

[BDP15] L. Bozzelli, H. van Ditmarsch, and S. Pinchinat. "The Complexity of One-agent Refinement Modal Logic". In: *Theoretical Computer Science* 603.C (2015), pp. 58–83. DOI: 10.1016/j.tcs.2015.07.015 (cit. on p. 155).

[Boz+18a] L. Bozzelli, A. Molinari, A. Montanari, and A. Peron. "Complexity of timeline-based planning over dense temporal domains: exploring the middle ground". In: *Proceedings of GandALF*. EPTCS, 2018, pp. 191–205. DOI: 10.4204/EPTCS.277.14 (cit. on p. 195).

[Boz+18b] L. Bozzelli, A. Molinari, A. Montanari, and A. Peron. "Decidability and Complexity of Timeline-based Planning over Dense Temporal Domains". In: *Proceedings of KR*. Full version at https://www.uniud.it/it/ateneo-uniud/ateneo-uniud-organizzazione/dipartimenti/dmif/assets/preprints/1-2018-molinari. AAAI Press, 2018 (cit. on p. 195).

[Boz+18c] L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and G. Woeginger. "Timeline-Based Planning over Dense Temporal Domains with Trigger-less Rules is NP-Complete". In: *Proceedings of ICTCS*. CEUR Workshop Proceedings, 2018 (cit. on p. 195).

[Boz+17a] L. Bozzelli, A. Molinari, A. Montanari, and A. Peron. "An In-Depth Investigation of Interval Temporal Logic Model Checking with Regular Expressions". In: *Proceedings of SEFM*. Springer, 2017, pp. 104–119. DOI: 10.1007/978-3-319-66197-1_7 (cit. on p. 153).

[Boz+17b] L. Bozzelli, A. Molinari, A. Montanari, and A. Peron. "On the Complexity of Model Checking for Syntactically Maximal Fragments of the Interval Temporal Logic HS with Regular Expressions". In: *Proceedings of GandALF*. EPTCS, 2017, pp. 31–45. DOI: 10.4204/EPTCS.256.3 (cit. on p. 153).

[Boz+18d] L. Bozzelli, A. Molinari, A. Montanari, and A. Peron. "Model Checking Interval Temporal Logics with Regular Expressions". In: *Information and Computation* (2018) (cit. on p. 153).

[Boz+16a] L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. "Interval Temporal Logic Model Checking Based on Track Bisimilarity and Prefix Sampling". In: *Proceedings of ICTCS*. CEUR Workshop Proceedings, 2016, pp. 49–61 (cit. on p. 73).

[Boz+16b] L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. "Interval Temporal Logic Model Checking: The Border Between Good and Bad HS Fragments". In: *Proceedings of IJCAR*. Springer, 2016, pp. 389–405. DOI: 10.1007/978-3-319-40229-1_27 (cit. on pp. 21, 73).

[Boz+16c] L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. "Interval vs. Point Temporal Logic Model Checking: an Expressiveness Comparison". In: *Proceedings of FSTTCS*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, 26:1–26:14. DOI: 10.4230/LIPIcs.FSTTCS.2016.26 (cit. on p. 121).

[Boz+16d] L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. "Model Checking the Logic of Allen's Relations Meets and Started-by is P^{NP}-Complete". In: *Proceedings of GandALF*. EPTCS, 2016, pp. 76–90. DOI: 10.4204/EPTCS.226.6 (cit. on p. 93).

[Boz+17c] L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. "Satisfiability and Model Checking for the Logic of Sub-Intervals under the Homogeneity Assumption". In: *Proceedings of ICALP*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 120:1–120:14. DOI: 10.4230/LIPIcs.ICALP.2017.120 (cit. on p. 51).

[Boz+18e] L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. "Interval vs. Point Temporal Logic Model Checking: an Expressiveness Comparison". In: *Transactions on Computational Logic* (2018). DOI: 10.1145/3281028 (cit. on p. 121).

[Boz+18f] L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. "Model Checking for Fragments of the Interval Temporal Logic HS at the Low Levels of the Polynomial Time Hierarchy". In: *Information and Computation* 262 (2018), pp. 241–264. DOI: 10.1016/j.ic.2018.09.006 (cit. on p. 93).

[Boz+18g] L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. "Which fragments of the interval temporal logic HS are tractable in model checking?" In: *Theoretical Computer Science* (2018). DOI: 10.1016/j.tcs.2018.04.011 (cit. on pp. 21, 73).

[Bra11] A. R. Bradley. "SAT-Based Model Checking without Unrolling". In: *Proceedings of VMCAI*. Springer, 2011, pp. 70–87. DOI: 10.1007/978-3-642-18275-4_7 (cit. on p. 8).

[Bra12] A. R. Bradley. "Understanding IC3". In: *Proceedings of SAT*. Springer, 2012, pp. 1–14. DOI: 10.1007/978-3-642-31612-8_1 (cit. on p. 8).

[Bre+08] D. Bresolin, D. Della Monica, V. Goranko, A. Montanari, and G. Sciavicco. "Decidable and Undecidable Fragments of Halpern and Shoham's Interval Temporal Logic: Towards a Complete Classification". In: *Proceedings of LPAR*. Springer, 2008, pp. 590–604. DOI: 10.1007/978-3-540-89439-1_41 (cit. on p. 15).

[Bre+09a] D. Bresolin, D. Della Monica, V. Goranko, A. Montanari, and G. Sciavicco. "Undecidability of Interval Temporal Logics with the Overlap Modality". In: *Proceedings of TIME*. IEEE Computer Society, 2009, pp. 88–95. DOI: 10.1109/TIME.2009.24 (cit. on p. 15).

[Bre+10a] D. Bresolin, D. Della Monica, V. Goranko, A. Montanari, and G. Sciavicco. "Metric Propositional Neighborhood Logics: Expressiveness, Decidability, and Undecidability". In: *Proceedings of ECAI*. ACM, 2010, pp. 695–700. DOI: 10.3233/978-1-60750-606-5-695 (cit. on p. 15).

[Bre+13] D. Bresolin, D. Della Monica, V. Goranko, A. Montanari, and G. Sciavicco. "Metric propositional neighborhood logics on natural numbers". In: *Software and System Modeling* 12.2 (2013), pp. 245–264. DOI: 10.1007/s10270-011-0195-y (cit. on pp. 15, 198, 239).

[Bre+14] D. Bresolin, D. Della Monica, V. Goranko, A. Montanari, and G. Sciavicco. "The dark side of interval temporal logic: marking the undecidability border". In: *Annals of Mathematics and Artificial Intelligence* 71.1–3 (2014), pp. 41–83. DOI: 10.1007/s10472-013-9376-4 (cit. on p. 15).

[Bre+10b] D. Bresolin, V. Goranko, A. Montanari, and P. Sala. "Tableaux for Logics of Subinterval Structures over Dense Orderings". In: *Journal of Logic and Computation* 20.1 (2010), pp. 133–166. DOI: 10.1093/logcom/exn063 (cit. on pp. 15, 51).

[Bre+09b] D. Bresolin, V. Goranko, A. Montanari, and G. Sciavicco. "Propositional interval neighborhood logics: Expressiveness, decidability, and undecidable extensions". In: *Annals of Pure and Applied Logic* 161.3 (2009), pp. 289–304. DOI: 10.1016/j.apal.2009.07.003 (cit. on pp. 15, 52).

[Bre+11a] D. Bresolin, A. Montanari, P. Sala, and G. Sciavicco. "An Optimal Decision Procedure for MPNL over the Integers". In: *Proceedings of GandALF*. EPTCS, 2011, pp. 192–206. DOI: 10.4204/EPTCS.54.14 (cit. on p. 15).

[Bre+11b] D. Bresolin, A. Montanari, P. Sala, and G. Sciavicco. "Optimal Tableau Systems for Propositional Neighborhood Logic over All, Dense, and Discrete Linear Orders". In: *Proceedings of TABLEAUX*. Springer, 2011, pp. 73–87. DOI: 10.1007/978-3-642-22119-4_8 (cit. on p. 15).

[Bre+11c] D. Bresolin, A. Montanari, P. Sala, and G. Sciavicco. "What's decidable about Halpern and Shoham's interval logic? The maximal fragment ABB̄L̄". In: *Proceedings of LICS*. IEEE Computer Society, 2011, pp. 387–396. DOI: 10.1109/LICS.2011.35 (cit. on p. 15).

[Bur+90a] J. R. Burch, E. M. Clarke, K. L. McMillan, and D. L. Dill. "Sequential Circuit Verification Using Symbolic Model Checking". In: *Proceedings of DAC*. ACM, 1990, pp. 46–51. DOI: 10.1145/123186.123223 (cit. on p. 6).

[Bur+90b] J. Burch, E. Clarke, K. McMillan, D. Dill, and L. Hwang. "Symbolic model checking: 10^{20} states and beyond". In: *Proceedings of LICS*. IEEE Computer Society, 1990, pp. 428–439. DOI: 10.1109/LICS.1990.113767 (cit. on p. 8).

[BH91] S. Buss and L. Hay. "On truth-table reducibility to SAT". In: *Information and Computation* 102 (1991), pp. 86–102. DOI: 10.1109/SCT.1988.5282 (cit. on p. 98).

[CS92] J. Castro and C. Seara. "Characterizations of some complexity classes between Θ_2^p and Δ_2^p". In: *Proceedings of STACS*. Springer, 1992, pp. 303–317 (cit. on p. 98).

[Ces+07] A. Cesta, G. Cortellessa, S. Fratini, A. Oddi, and N. Policella. "An Innovative Product for Space Mission Planning: An A Posteriori Evaluation". In: *Proceedings of ICAPS*. AAAI Press, 2007, pp. 57–64 (cit. on p. 196).

[Cha+98] W. Chan, R. J. Anderson, P. Beame, S. Burns, F. Modugno, D. Notkin, and J. D. Reese. "Model checking large software specifications". In: *Transactions on Software Engineering* 24.7 (1998), pp. 498–520. DOI: 10.1109/32.708566 (cit. on p. 6).

[CKS81] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. "Alternation". In: *Journal of the ACM* 28.1 (1981), pp. 114–133. DOI: 10.1145/322234.322243 (cit. on p. 169).

[CH04] Z. Chaochen and M. R. Hansen. *Duration Calculus—A Formal Approach to Real-Time Systems*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2004. ISBN: 978-3-662-06784-0 (cit. on p. 11).

[CHR91] Z. Chaochen, C. A. R. Hoare, and A. P. Ravn. "A calculus of durations". In: *Information Processing Letters* 40.5 (1991), pp. 269–276. DOI: 10.1016/0020-0190(91)90122-X (cit. on p. 11).

[Chi+10] S. Chien, D. Tran, G. Rabideau, S. Schaffer, D. Mandl, and S. Frye. "Timeline-Based Space Operations Scheduling with External Constraints". In: *Proceedings of ICAPS*. AAAI Press, 2010, pp. 34–41 (cit. on p. 196).

[COU16] M. Cialdea Mayer, A. Orlandini, and A. Umbrico. "Planning and Execution with Flexible Timelines: a Formal Account". In: *Acta Informatica* 53.6–8 (2016), pp. 649–680. DOI: 10.1007/s00236-015-0252-z (cit. on p. 198).

[Cim01] A. Cimatti. "Industrial Applications of Model Checking". In: *Proceedings of MOVEP*. Springer, 2001, pp. 153–168. DOI: 10.1007/3-540-45510-8_6 (cit. on p. 6).

[CG12] A. Cimatti and A. Griggio. "Software Model Checking via IC3". In: *Proceedings of CAV*. Springer, 2012, pp. 277–293. DOI: 10.1007/978-3-642-31424-7_23 (cit. on p. 10).

[Cim+14] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta. "IC3 Modulo Theories via Implicit Predicate Abstraction". In: *Proceedings of TACAS*. Springer, 2014, pp. 46–61. DOI: 10.1007/978-3-642-54862-8_4 (cit. on p. 10).

[Cim+16] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta. "Infinite-state invariant checking with IC3 and predicate abstraction". In: *Formal Methods in System Design* 49.3 (2016), pp. 190–218. DOI: 10.1007/s10703-016-0257-4 (cit. on p. 10).

[Cim+00] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. "NuSMV: a new symbolic model checker". In: *International Journal on Software Tools for Technology Transfer* 2.4 (2000), pp. 410–425. DOI: 10.1007/s100090050046 (cit. on p. 8).

[CE81] E. M. Clarke and E. A. Emerson. "Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic". In: *Proceedings of LP*. Springer, 1981, pp. 52–71. DOI: 10.1007/BFb0025774 (cit. on pp. 5, 10).

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. "Automatic Verification of Finite-state Concurrent Systems Using Temporal Logic Specifications". In: *Transactions on Programming Languages and Systems* 8.2 (1986), pp. 244–263. DOI: 10.1145/5397.5399 (cit. on p. 10).

[CGP02] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. Cyber Physical Systems Series. MIT Press, 2002. ISBN: 978-0262032704 (cit. on pp. 5, 10).

[CM90] O. Coudert and J. Madre. "A unified framework for the formal verification of sequential circuits". In: *Proceedings of ICCAD*. IEEE Computer Society, 1990, pp. 126–129. DOI: 10.1109/ICCAD.1990.129859 (cit. on p. 8).

[DP14] J. Daniel and P. Parízek. "Predicate Abstraction in Program Verification: Survey and Current Trends". In: *Imperial College Computing Student Workshop*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014, pp. 27–35. DOI: 10.4230/OASIcs.ICCSW.2014.27 (cit. on p. 10).

[DV13] G. De Giacomo and M. Y. Vardi. "Linear Temporal Logic and Linear Dynamic Logic on Finite Traces". In: *Proceedings of IJCAI*. AAAI Press, 2013, pp. 854–860 (cit. on p. 126).

[Del+11] D. Della Monica, V. Goranko, A. Montanari, and G. Sciavicco. "Interval Temporal Logics: a Journey". In: *Bulletin of the EATCS* 105 (2011), pp. 73–99 (cit. on pp. 11, 15, 16).

[Del11] D. Della Monica. "Expressiveness, decidability, and undecidability of Interval Temporal Logic". PhD thesis. University of Udine, 2011 (cit. on p. 16).

[DGL16] S. Demri, V. Goranko, and M. Lange. *Temporal Logics in Computer Science: Finite-State Systems*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2016. ISBN: 978-1139236119. DOI: 10.1017/CBO9781139236119 (cit. on p. 135).

[DL09] S. Demri and R. Lazic. "LTL with the freeze quantifier and register automata". In: *Transactions on Computational Logic* 10.3 (2009), 16:1–16:30. DOI: 10.1145/1507244.1507246 (cit. on pp. 210, 309, 310).

[Don+06] F. Donini, M. Mongiello, M. Ruta, and R. Totaro. "A Model Checking-based Method for Verifying Web Application Design". In: *Electronic Notes in Theoretical Computer Science* 151.2 (2006), pp. 19–32. DOI: 10.1016/j.entcs.2005.07.034 (cit. on p. 6).

[EL85] E. A. Emerson and C. Lei. "Modalities for Model Checking: Branching Time Strikes Back". In: *Proceedings of PoPL*. Elsevier, 1985, pp. 84–96. DOI: 10.1145/318593.318620 (cit. on p. 126).

[EH86] E. Emerson and J. Halpern. ""Sometimes" and "Not Never" Revisited: On Branching Versus Linear Time". In: *Journal of the ACM* 33.1 (1986), pp. 151–178. DOI: 10.1145/4904.4999 (cit. on pp. 10, 121, 125, 149).

[FR75] J. Ferrante and C. Rackoff. "A Decision Procedure for the First Order Theory of Real Addition with Order". In: *SIAM Journal of Computation* 4.1 (1975), pp. 69–76. DOI: 10.1137/0204006 (cit. on p. 155).

[FL79] M. Fischer and R. Ladner. "Propositional Dynamic Logic of Regular Programs". In: *Journal of Computer and System Sciences* 18.2 (1979), pp. 194–211. DOI: 10.1016/0022-0000(79)90046-1 (cit. on p. 10).

[Fis85] P. C. Fishburn. *Interval Orders and Interval Graphs: A Study of Partially Ordered Sets*. Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons, 1985. ISBN: 978-0471812845 (cit. on p. 196).

[Fix08] L. Fix. "Fifteen years of formal property verification in Intel". In: *25 Years of Model Checking* (2008), pp. 139–144. DOI: 10.1007/978-3-540-69850-0_8 (cit. on p. 6).

[FJ03] J. Frank and A. Jónsson. "Constraint-based Attribute and Interval Planning". In: *Constraints* 8.4 (2003), pp. 339–364. DOI: 10.1023/A:1025842019552 (cit. on p. 196).

[Gab87] D. M. Gabbay. "The Declarative Past and Imperative Future: Executable Temporal Logic for Interactive Systems". In: *Temporal Logic in Specification*. Springer, 1987, pp. 409–448. DOI: 10.1007/3-540-51803-7_36 (cit. on pp. 38, 141, 142, 145).

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Series of Books in the Mathematical Sciences. W. H. Freeman and Company, 1979. ISBN: 978-0716710455 (cit. on pp. 188, 192).

[Gig+16] N. Gigante, A. Montanari, M. Cialdea Mayer, and A. Orlandini. "Timelines are Expressive Enough to Capture Action-Based Temporal Planning". In: *Proceedings of TIME*. IEEE Computer Society, 2016, pp. 100–109. DOI: 10.1109/TIME.2016.18 (cit. on pp. 196, 198).

[Gig+17] N. Gigante, A. Montanari, M. Cialdea Mayer, and A. Orlandini. "Complexity of Timeline-Based Planning". In: *Proceedings of ICAPS*. AAAI Press, 2017, pp. 116–124 (cit. on p. 196).

[Gio+06] L. Giordano, P. Terenziani, A. Bottrighi, S. Montani, and L. Donzella. "Model Checking for Clinical Guidelines: an Agent-based Approach". In: *Proceedings of AMIA*. 2006, pp. 289–293 (cit. on p. 6).

[GT99] F. Giunchiglia and P. Traverso. "Planning as Model Checking". In: *Proceedings of ECP*. Springer, 1999, pp. 1–20. DOI: 10.1007/10720246_1 (cit. on pp. 6, 11).

[GM13] M. Gligoric and R. Majumdar. "Model Checking Database Applications". In: *Proceedings of TACAS*. Springer, 2013, pp. 549–564. DOI: 10.1007/978-3-642-36742-7_40 (cit. on p. 6).

[God97] P. Godefroid. “Model Checking for Programming Languages Using VeriSoft”. In: *Proceedings of POPL*. ACM, 1997, pp. 174–186. DOI: 10.1145/263699.263717 (cit. on p. 8).

[GMS04] V. Goranko, A. Montanari, and G. Sciavicco. “A Road Map of Interval Temporal Logics and Duration Calculi”. In: *Journal of Applied Non-Classical Logics* 14.1–2 (2004), pp. 9–54. DOI: 10.3166/jancl.14.9-54 (cit. on pp. 11, 16).

[Got95] G. Gottlob. “NP Trees and Carnap’s Modal Logic”. In: *Journal of the ACM* 42.2 (1995), pp. 421–457. DOI: 10.1145/201019.201031 (cit. on pp. 97, 105).

[HS91] J. Y. Halpern and Y. Shoham. “A Propositional Modal Logic of Time Intervals”. In: *Journal of the ACM* 38.4 (1991), pp. 935–962. DOI: 10.1145/115234.115351 (cit. on pp. 11, 13–15).

[HV07] M. R. Hansen and D. Van Hung. “A Theory of Duration Calculus with Application”. In: *Domain Modeling and the Duration Calculus: International Training School*. Springer, 2007, pp. 119–176. DOI: 10.1007/978-3-540-74964-6_3 (cit. on p. 11).

[Har12] D. Harel. *Algorithmics: The spirit of computing*. 3rd. Springer, 2012. ISBN: 978-3642272653 (cit. on pp. 39, 40, 223, 314).

[HD01] J. Hatcliff and M. Dwyer. “Using the Bandera Tool Set to Model-Check Properties of Concurrent Java Software”. In: *Proceedings of CONCUR*. Springer, 2001, pp. 39–58. DOI: 10.1007/3-540-44685-0_5 (cit. on p. 8).

[HP00] K. Havelund and T. Pressburger. “Model checking JAVA programs using JAVA PathFinder”. In: *International Journal on Software Tools for Technology Transfer* 2.4 (2000), pp. 366–381. DOI: 10.1007/s100090050043 (cit. on p. 8).

[Hen+02] T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. “Lazy Abstraction”. In: *Proceedings of POPL*. ACM, 2002, pp. 58–70. DOI: 10.1145/503272.503279 (cit. on p. 9).

[HK11] M. Holzer and M. Kutrib. “Descriptional and computational complexity of finite automata—A survey”. In: *Information and Computation* 209.3 (2011), pp. 456–470. DOI: 10.1016/j.ic.2010.11.013 (cit. on pp. 71, 247).

[Hol94] G. J. Holzmann. “The theory and practice of a formal method: NewCoRe”. In: *Proceedings of the IFIP World Computer Congress*. 1994 (cit. on pp. 4, 6).

[JM97] J. M. Jazequel and B. Meyer. "Design by contract: The lessons of Ariane". In: *Computer* 30.1 (1997), pp. 129–130. DOI: 10.1109/2.562936 (cit. on p. 4).

[Jón+00] A. K. Jónsson, P. H. Morris, N. Muscettola, K. Rajan, and B. D. Smith. "Planning in Interplanetary Space: Theory and Practice". In: *Proceedings of ICAPS*. AAAI Press, 2000, pp. 177–186 (cit. on p. 196).

[Jun13] D. Jungnickel. *Graphs, Networks and Algorithms*. Algorithms and Computation in Mathematics. Springer, 2013. ISBN: 978-3540637608. DOI: 10.1007/978-3-642-32278-5 (cit. on p. 230).

[Kam68] H. Kamp. "Tense Logic and the Theory of Linear Order". PhD thesis. UCLA, 1968 (cit. on p. 133).

[Koy90] R. Koymans. "Specifying Real-Time Properties with Metric Temporal Logic". In: *Real-Time Systems* 2.4 (1990), pp. 255–299. DOI: 10.1007/BF01995674 (cit. on pp. 210, 212).

[KPV12] O. Kupferman, A. Pnueli, and M. Y. Vardi. "Once and for all". In: *Journal of Computer and System Sciences* 78.3 (2012), pp. 981–996. DOI: 10.1016/j.jcss.2011.08.006 (cit. on p. 140).

[KPV09] O. Kupferman, N. Piterman, and M. Y. Vardi. "From liveness to promptness". In: *Formal Methods in System Design* 34.2 (2009), pp. 83–103. DOI: 10.1007/s10703-009-0067-z (cit. on pp. 10, 153).

[Lan06] M. Lange. "Model checking propositional dynamic logic with all extras". In: *Journal of Applied Logic* 4.1 (2006), pp. 39–49. DOI: 10.1016/j.jal.2005.08.002 (cit. on p. 38).

[LMS01] F. Laroussinie, N. Markey, and P. Schnoebelen. "Model Checking CTL+ and FCTL is Hard". In: *Proceedings of FOSSACS*. Springer, 2001, pp. 318–331. DOI: 10.1007/3-540-45315-6_21 (cit. on pp. 97, 110, 111, 119).

[LMS02] F. Laroussinie, N. Markey, and P. Schnoebelen. "Temporal Logic with Forgettable Past". In: *Proceedings of LICS*. IEEE Computer Society, 2002, pp. 383–392. DOI: 10.1109/LICS.2002.1029846 (cit. on p. 97).

[LMP10] F. Laroussinie, A. Meyer, and E. Petonnet. "Counting CTL". In: *Proceedings of FOSSACS*. Springer, 2010, pp. 206–220. DOI: 10.1007/978-3-642-12032-9_15 (cit. on p. 97).

[LS95] F. Laroussinie and P. Schnoebelen. "A Hierarchy of Temporal Logics with Past". In: *Theoretical Computer Science* 148.2 (1995), pp. 303–324. DOI: 10.1016/0304-3975(95)00035-U (cit. on p. 124).

[LPY97] G. K. Larsen, P. Pettersson, and W. Yi. "UPPAAL in a Nutshell". In: *International Journal on Software Tools for Technology Transfer* 1 (1997), pp. 134–152. DOI: 10.1007/s100090050010 (cit. on p. 235).

[LS07] M. Leucker and C. Sánchez. "Regular Linear Temporal Logic". In: *Proceedings of ICTAC*. Springer, 2007, pp. 291–305. DOI: 10.1007/978-3-540-75292-9_20 (cit. on p. 153).

[LP00] O. Lichtenstein and A. Pnueli. "Propositional Temporal Logics: Decidability and Completeness". In: *Logic Journal of the IGPL* 8.1 (2000), pp. 55–85. DOI: 10.1093/jigpal/8.1.55 (cit. on p. 135).

[Lod00] K. Lodaya. "Sharpening the Undecidability of Interval Temporal Logic". In: *Proceedings of ASIAN*. Springer, 2000, pp. 290–298. DOI: 10.1007/3-540-44464-5_21 (cit. on pp. 15, 51).

[LM13] A. Lomuscio and J. Michaliszyn. "An Epistemic Halpern-Shoham Logic". In: *Proceedings of IJCAI*. AAAI Press, 2013 (cit. on pp. 48, 49, 122, 154, 192).

[LM14] A. Lomuscio and J. Michaliszyn. "Decidability of model checking multi-agent systems against a class of EHS specifications". In: *Proceedings of ECAI*. IOS, 2014, pp. 543–548. DOI: 10.3233/978-1-61499-419-0-543 (cit. on pp. 48, 49, 122, 154, 192).

[LM16] A. Lomuscio and J. Michaliszyn. "Model Checking Multi-Agent Systems against Epistemic HS Specifications with Regular Expressions". In: *Proceedings of KR*. AAAI Press, 2016, pp. 298–308 (cit. on pp. 48, 49, 122, 154, 159, 192).

[LR06] A. Lomuscio and F. Raimondi. "MCMAS: A Model Checker for Multi-agent Systems". In: *Proceedings of TACAS*. Springer, 2006, pp. 450–454. DOI: 10.1007/11691372_31 (cit. on p. 11).

[Low96] G. Lowe. "Breaking and fixing the Needham-Schroeder Public-Key Protocol using FDR". In: *Proceedings of TACAS*. Springer, 1996, pp. 147–166. DOI: 10.1007/3-540-61042-1_43 (cit. on pp. 4, 6).

[MM14] J. Marcinkowski and J. Michaliszyn. "The Undecidability of the Logic of Subintervals". In: *Fundamenta Informaticae* 131.2 (2014), pp. 217–240. DOI: 10.3233/FI-2014-1011 (cit. on pp. 15, 47, 52, 53, 71, 253, 254).

[Mat+11] R. Mateescu, P. T. Monteiro, E. Dumas, and H. de Jong. "CTRL: Extension of CTL with regular expressions and fairness operators to verify genetic regulatory networks". In: *Theoretical Computer Science* 412.26 (2011), pp. 2854–2883. DOI: 10.1016/j.tcs.2010.05.009 (cit. on p. 154).

[McM93] K. L. McMillan. *Symbolic Model Checking*. Kluwer, 1993. ISBN: 978-0-7923-9380-1 (cit. on p. 8).

[McM06] K. L. McMillan. "Lazy Abstraction with Interpolants". In: *Proceedings of CAV*. Springer, 2006, pp. 123–136. DOI: 10.1007/11817963_14 (cit. on p. 9).

[Mei+08] A. Meier, M. Mundhenk, M. Thomas, and H. Vollmer. "The Complexity of Satisfiability for Fragments of CTL and CTL*". In: *Electronic Notes in Theoretical Computer Science* 223 (2008), pp. 201–213. DOI: 10.1016/j.entcs.2008.12.040 (cit. on p. 10).

[MK12] A. Mentis and P. Katsaros. "Model checking and code generation for transaction processing software". In: *Concurrency and Computation: Practice and Experience* 24.7 (2012), pp. 711–722. DOI: 10.1002/cpe.1876 (cit. on p. 6).

[Min67] M. L. Minsky. *Computation: Finite and Infinite Machines*. Automatic Computation. Prentice-Hall, Inc., 1967. ISBN: 978-0131655638 (cit. on pp. 202, 203).

[MBT00] A. A. Mir, S. Balakrishnan, and S. Tahar. "Modeling and verification of embedded systems using Cadence SMV". In: *Proceedings of CCECE*. IEEE Computer Society, 2000, pp. 179–1831. DOI: 10.1109/CCECE.2000.849694 (cit. on p. 8).

[MMP15a] A. Molinari, A. Montanari, and A. Peron. "Complexity of ITL model checking: some well-behaved fragments of the interval logic HS". In: *Proceedings of TIME*. IEEE Computer Society, 2015, pp. 90–100. DOI: 10.1109/TIME.2015.12 (cit. on pp. 46, 82, 84, 258).

[Mol+16a] A. Molinari, A. Montanari, A. Murano, G. Perelli, and A. Peron. "Checking interval properties of computations". In: *Acta Informatica* 53.6 (2016), pp. 587–619. DOI: 10.1007/s00236-015-0250-1 (cit. on pp. 21, 37, 38).

[MMP15b] A. Molinari, A. Montanari, and A. Peron. "A Model Checking Procedure for Interval Temporal Logics based on Track Representatives". In: *Proceedings of CSL*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015, pp. 193–210. DOI: 10.4230/LIPIcs.CSL.2015.193 (cit. on p. 73).

[MMP17] A. Molinari, A. Montanari, and A. Peron. "Model checking for fragments of Halpern and Shoham's interval temporal logic based on track representatives". In: *Information and Computation* 259 (2017), pp. 412–443. DOI: 10.1016/j.ic.2017.08.011 (cit. on pp. 44, 46, 73, 74, 84, 85, 91, 264).

[Mol+16b] A. Molinari, A. Montanari, A. Peron, and P. Sala. “Model Checking Well-Behaved Fragments of HS: The (Almost) Final Picture”. In: *Proceedings of KR*. AAAI Press, 2016, pp. 473–483 (cit. on p. 93).

[MPS10a] A. Montanari, I. Pratt-Hartmann, and P. Sala. “Decidability of the Logics of the Reflexive Sub-interval and Super-interval Relations over Finite Linear Orders”. In: *Proceedings of TIME*. IEEE Computer Society, 2010, pp. 27–34. DOI: 10.1109/TIME.2010.18 (cit. on p. 53).

[MPS10b] A. Montanari, G. Puppis, and P. Sala. “Maximal Decidable Fragments of Halpern and Shoham’s Modal Logic of Intervals”. In: *Proceedings of ICALP*. Springer, 2010, pp. 345–356. DOI: 10.1007/978-3-642-14162-1_29 (cit. on p. 15).

[MPS14] A. Montanari, G. Puppis, and P. Sala. “Decidability of the Interval Temporal Logic A, $\overline{\text{A}}$, B, $\overline{\text{B}}$ over the Rationals”. In: *Proceedings of MFCS*. Springer, 2014, pp. 451–463. DOI: 10.1007/978-3-662-44522-8_38 (cit. on p. 16).

[MS12] A. Montanari and P. Sala. “An Optimal Tableau System for the Logic of Temporal Neighborhood over the Reals”. In: *Proceedings of TIME*. IEEE Computer Society, 2012, pp. 39–46. DOI: 10.1109/TIME.2012.18 (cit. on p. 15).

[Mon+10] A. Montanari, G. Puppis, P. Sala, and G. Sciavicco. “Decidability of the Interval Temporal Logic $\overline{\text{A}}$, B, $\overline{\text{B}}$ over the Natural Numbers”. In: *Proceedings of STACS*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010, pp. 597–608. DOI: 10.4230/LIPIcs.STACS.2010.2488 (cit. on p. 15).

[MS14] A. Montanari and P. Sala. “Interval-based Synthesis”. In: *Proceedings of GandALF*. EPTCS, 2014, pp. 102–115. DOI: 10.4204/EPTCS.161.11 (cit. on p. 239).

[Mos83] B. Moszkowski. “Reasoning About Digital Circuits”. PhD thesis. Stanford University, 1983 (cit. on pp. 11, 15).

[Mus94] N. Muscettola. “HSTS: Integrating Planning and Scheduling”. In: *Intelligent Scheduling*. Morgan Kaufmann, 1994, pp. 169–212 (cit. on p. 196).

[Nar+16] R. Nardone, U. Gentile, M. Benerecetti, A. Peron, V. Vittorini, S. Marrone, and N. Mazzocca. “Modeling Railway Control Systems in Promela”. In: *Proceedings of FTSCS*. Springer, 2016, pp. 121–136. DOI: 10.1007/978-3-319-29510-7_7 (cit. on p. 6).

[Ott01] M. Otto. “Two Variable First-Order Logic over Ordered Domains”. In: *Journal of Symbolic Logic* 66.2 (2001), pp. 685–702. DOI: 10.2307/2695037 (cit. on p. 52).

[OW07] J. Ouaknine and J. Worrell. “On the decidability and complexity of Metric Temporal Logic over finite words”. In: *Logical Methods in Computer Science* 3.1 (2007). DOI: 10.2168/LMCS-3(1:8)2007 (cit. on pp. 210, 221, 222).

[OW08] J. Ouaknine and J. Worrell. “Some Recent Results in Metric Temporal Logic”. In: *Proceedings of FORMATS*. Springer, 2008, pp. 1–13. DOI: 10.1007/978-3-540-85778-5_1 (cit. on p. 10).

[Pap94] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994. ISBN: 978-0-201-53082-7 (cit. on p. 223).

[PZ82] C. H. Papadimitriou and S. K. Zachos. “Two remarks on the power of counting”. In: *Theoretical Computer Science* (1982), pp. 269–275. DOI: 10.1007/BFb0036487 (cit. on p. 97).

[Pel93] D. Peled. “All from one, one for all: on model checking using representatives”. In: *Proceedings of CAV*. Springer, 1993, pp. 409–423. DOI: 10.1007/3-540-56922-7_34 (cit. on p. 8).

[Pnu77] A. Pnueli. “The Temporal Logic of Programs”. In: *Proceedings of FOCS*. IEEE Computer Society, 1977, pp. 46–57. DOI: 10.1109/SFCS.1977.32 (cit. on pp. 10, 121, 125).

[Pra05] I. Pratt-Hartmann. “Temporal prepositions and their logic”. In: *Artificial Intelligence* 166.1–2 (2005), pp. 1–36. DOI: 10.1016/j.artint.2005.04.003 (cit. on pp. 11, 51).

[Pra95] V. Pratt. “Anatomy of the Pentium bug”. In: *Proceedings of TAPSOFT*. Springer, 1995, pp. 97–107. DOI: 10.1007/3-540-59293-8_189 (cit. on p. 4).

[QS82] J. P. Queille and J. Sifakis. “Specification and verification of concurrent systems in CESAR”. In: *Proceedings of SP*. Springer, 1982, pp. 337–351. DOI: 10.1007/3-540-11494-7_22 (cit. on p. 5).

[Roe80] P. Roeper. “Intervals and Tenses”. In: *Journal of Philosophical Logic* 9 (1980), pp. 451–469. DOI: 10.1007/BF00262866 (cit. on pp. 16, 24).

[Sch16] S. Schmitz. “Complexity Hierarchies Beyond Elementary”. In: *Transactions on Computation Theory* 8.1 (2016), 3:1–3:36. DOI: 10.1145/2858784 (cit. on p. 43).

[Sch03] P. Schnoebelen. “Oracle Circuits for Branching-Time Model Checking”. In: *Proceedings of ICALP*. Springer, 2003, pp. 790–801. DOI: 10.1007/3-540-45061-0_62 (cit. on pp. 47, 94, 98, 103–105).

[Sha04] I. Shapirovsky. “On PSPACE-decidability in Transitive Modal Logic”. In: *Proceedings of AiML*. CSLI Publications, 2004, pp. 269–287 (cit. on p. 51).

[SS05] I. Shapirovsky and V. B. Shehtman. “Modal Logics of Regions and Minkowski Spacetime”. In: *Journal of Logic and Computation* 15.4 (2005), pp. 559–574. DOI: 10.1093/logcom/exi039 (cit. on p. 51).

[Sip12] M. Sipser. *Introduction to the Theory of Computation*. 3rd. International Thomson Publishing, 2012. ISBN: 978-1133187790 (cit. on p. 257).

[SC85] A. P. Sistla and E. M. Clarke. “The Complexity of Propositional Linear Temporal Logics”. In: *Journal of the ACM* 32.3 (1985), pp. 733–749. DOI: 10.1145/3828.3837 (cit. on pp. 10, 46, 126).

[Sto76] L. J. Stockmeyer. “The polynomial-time hierarchy”. In: *Theoretical Computer Science* 3.1 (1976), pp. 1–22. DOI: 10.1016/0304-3975(76)90061-X (cit. on p. 96).

[SM73] L. J. Stockmeyer and A. R. Meyer. “Word Problems Requiring Exponential Time (Preliminary Report)”. In: *Proceedings of STOC*. ACM, 1973, pp. 1–9. DOI: 10.1145/800125.804029 (cit. on pp. 43, 44).

[Sto74] L. J. Stockmeyer. “The complexity of decision problems in automata theory and logic”. PhD thesis. Massachusetts Institute of Technology, 1974 (cit. on pp. 43, 44).

[Tho94] M. Thomas. “A proof of incorrectness using the LP theorem prover: the editing problem in the Therac-25”. In: *High Integrity Systems Journal* 1.1 (1994), pp. 35–48 (cit. on p. 4).

[van91] J. van Benthem. “Language in action”. In: *Journal of Philosophical Logic* 20.3 (1991), pp. 225–263. DOI: 10.1007/BF00250539 (cit. on pp. 51, 53).

[Var96] M. Y. Vardi. “An automata-theoretic approach to linear temporal logic”. In: *Logics for concurrency*. Springer, 1996, pp. 238–266. DOI: 10.1007/3-540-60915-6_6 (cit. on p. 136).

[VW86] M. Vardi and P. Wolper. “An Automata-Theoretic Approach to Automatic Program Verification”. In: *Proceedings of LICS*. IEEE Computer Society, 1986, pp. 332–344 (cit. on p. 5).

[Ven90] Y. Venema. “Expressiveness and completeness of an interval tense logic”. In: *Notre Dame Journal of Formal Logic* 31.4 (1990), pp. 529–547. DOI: 10.1305/ndjfl/1093635589 (cit. on pp. 11, 55).

[Ven91] Y. Venema. “A Modal Logic for Chopping Intervals”. In: *Journal of Logic and Computation* 1.4 (1991), pp. 453–476. DOI: 10.1093/logcom/1.4.453 (cit. on p. 11).

[Wag87] K. W. Wagner. “More complicated questions about maxima and minima, and some closures of NP”. In: *Theoretical Computer Science* 51.1 (1987), pp. 53–80. DOI: 10.1016/0304-3975(87)90049-1 (cit. on pp. 98, 115).

[Wag90] K. W. Wagner. “Bounded Query Classes”. In: *SIAM Journal of Computation* 19.5 (1990), pp. 833–846. DOI: 10.1137/0219058 (cit. on p. 98).

[Wil99] T. Wilke. “Classifying Discrete Temporal Properties”. In: *Proceedings of STACS*. Springer, 1999, pp. 32–46. DOI: 10.1007/3-540-49116-3_3 (cit. on p. 136).

[Wit+07] T. Witkowski, N. Blanc, D. Kroening, and G. Weissenbacher. “Model Checking Concurrent Linux Device Drivers”. In: *Proceedings of ASE*. ACM, 2007, pp. 501–504. DOI: 10.1145/1321631.1321719 (cit. on p. 6).