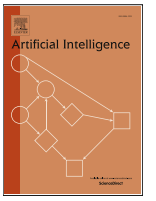
Artificial Intelligence 357 (2026) 104553

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



An optimal pastification algorithm for LTL[X, F] and LTL[X, G]

Alessandro Artale [a], Luca Geatti [b,*], Nicola Gigante [a], Alessio Mansutti [c], Andrea Mazzullo [a], Angelo Montanari [b]

[a] Free University of Bozen-Bolzano, Piazza Università, 1, Bolzano, 39100, Italy
[b] University of Udine, Via delle Scienze, 206, Udine, 33100, Italy
[c] IMDEA Software Institute, Pozuelo de Alarcón, Madrid, 28223, Spain

ARTICLE INFO

Keywords:
Linear temporal logics (LTLs)
Pure past LTL
Safety and cosafety languages

ABSTRACT

We investigate a fragment of Linear Temporal Logic (LTL) comprising the *tomorrow* (X) and *eventually* (F) modalities, and present a singly exponential time algorithm for the *pastification* problem within this fragment. The pastification problem consists of constructing, for a given LTL formula, an equivalent formula that exclusively employs past temporal operators. While the best known algorithms for this task in full LTL–and in the fragment under consideration–exhibit triply exponential time complexity, our approach achieves optimal complexity for this fragment.

The proposed algorithm proceeds in two main stages: (i) the input formula is first translated into a tailored normal form, and then (ii) a pure past formula is synthesized from a tree-like structure derived from the normalized formula. With minor adaptations, the algorithm extends to handle the fragment of LTL featuring the *tomorrow* and *globally* modalities.

We provide an implementation of the algorithm in the temporal reasoning tool BLACK, and report on an experimental evaluation of its performance.[1]

1. Introduction

Linear Temporal Logic (LTL) [2] extends propositional logic with *future* temporal modalities to reason on *infinite traces*, that is, infinite discrete linear structures with a minimal element. Together with its extension with *past* modalities LTL+P [3] and its finite variant LTL_f [4] (where formulae are interpreted over *finite* traces), Linear Temporal Logic proved itself to be an essential tool in many areas including formal verification, automated reasoning, and knowledge representation. In this paper, we focus on the *pure past* fragment of LTL+P, which we denote by pLTL [3,5]. This fragment is defined as the subset of formulae of LTL+P devoid of future modalities. By featuring only past modalities, formulae of pLTL are naturally interpreted on finite traces.

The logic pLTL has received a renewed attention in the last years, due to its good theoretical and algorithmic properties.

In terms of expressive power, pLTL is equivalent to LTL_f [3], although one can be exponentially more succinct than the other, and *vice versa* [6,7]. In addition, pLTL is closely related to *safety* and *cosafety fragments* of LTL [8]. The former, denoted by SafetyLTL, is the fragment of LTL that characterizes properties stating that "*something bad never happens*". Structurally, SafetyLTL contains all formulae of LTL that are in Negation Normal Form and whose temporal modalities are universal (*tomorrow* (X), *globally* (G), and *release* (R)). The cosafety fragment of LTL, denoted by coSafetyLTL, characterizes instead all properties stating that "*something good will eventually happen*" and it contains all formulae of LTL that are in Negation Normal Form and whose temporal modalities are all

* Corresponding author.
E-mail address: luca.geatti@uniud.it (L. Geatti).

[1] A preliminary version of this paper appeared in the proceedings of KR 2023 [1]

https://doi.org/10.1016/j.artint.2026.104553
Received 6 August 2025; Received in revised form 9 February 2026; Accepted 30 April 2026
Available online 9 May 2026

existential (*tomorrow* (X), *eventually* (F), and *until* (U)). The crucial feature of SafetyLTL (resp., coSafetyLTL) is that a *finite prefix* of an infinite trace suffices to establish whether the whole trace is not (resp., is) a model of a formula. This makes it possible the translation of the two fragments into the framework of pLTL, as detailed in the following.

Let F(pLTL) be the logic defined as the set of formulae of the form F(α), where F is the *eventually* modality (*"there exists a time point in the future"*) and α is a pLTL formula. Formulas in F(pLTL) specify that there exists some point in the future such that the prefix of the trace up to that point satisfies α. It has been shown that F(pLTL) is expressively equivalent to coSafetyLTL [8]. Similarly, the logic G(pLTL), featuring formulae of the form G(α), with α in pLTL, is expressively equivalent to SafetyLTL. In these formulae, the *globally* modality G forces *every* prefix of an infinite trace to satisfy α.

In terms of algorithmic properties, formulae of pLTL can be compiled into *Deterministic Finite Automata* (DFAs) of *singly exponential size* [5,9]. This result is in contrast with the case of full LTL_f, where DFAs may have doubly exponential size. This "singly exponential gap" persists through *monitoring*, *planning*, and *reactive synthesis* problems [10–12]. For instance, it has been recently shown that the *reactive synthesis* problem for pLTL specifications is EXPTIME-complete [13], whereas in the case of LTL and LTL_f the problem is EXPTIME [2]-complete [14,15].

Given the expressiveness and the algorithmic features of pLTL, the study of procedures to transform fragments of coSafetyLTL, SafetyLTL, and LTL_f into equivalent ones of, respectively, F(pLTL), G(pLTL), and pLTL, is receiving an increasing attention. Transformations of this form are instances of a process known as *pastification*. Planning patterns which can be expressed by LTL_f formulae with *at most two* nested temporal modalities can be easily "pastified" into formulae whose size is polynomial in that of the original ones. These patterns include all those expressible in DECLARE [16,17], as well as the trajectory constraints of PDDL [11]. A similar polynomial-time translation applies for the fragment of LTL with *tomorrow* as its only temporal modality [18,19], denoted by LTL[X] below.

With the exception of the two examples given above, devising optimal pastification procedures is, in general, a very challenging task. The best known algorithms for the pastification of coSafetyLTL, SafetyLTL, and LTL_f formulae produce a pure past formula of *triply exponential* size with respect to the input one [5]. This result is at odds with the aforementioned "singly exponential gap" found in the complexity of the various computational problems; from this gap, it is reasonable to conjecture the existence of a pastification procedure producing formulae of exponential size. This also matches the best-known lower bound for the pastification of coSafetyLTL and SafetyLTL, which is singly exponential [7].

At present, no alternative pastification algorithms are known for natural fragments of these logics. Consequently, as soon as the input formula falls outside the limited fragments for which polynomial-size pastifications are known, one must resort to the general triply exponential procedure.

Our contribution. In this paper, we focus primarily on the logics LTL[X, F], the fragment of LTL featuring only the temporal modalities *tomorrow* X and *eventually* F, and F(pLTL[Y, $\widetilde{\text{Y}}$, O]), which is the fragment of F(pLTL) obtained by allowing only the past modalities *yesterday* (Y), *weak yesterday* ($\widetilde{\text{Y}}$), and *once* (O). The fragments LTL[X, F] and F(pLTL[Y, $\widetilde{\text{Y}}$, O]) are among the so-called *cosafety fragments* of LTL or LTL+P, because they can express only cosafety languages. In addition, we investigate the 'universal' counterparts of these fragments, that is, LTL[X, G], which is the fragment of LTL with the tomorrow (X) and *always* (G) modalities only, and G(pLTL[Y, $\widetilde{\text{Y}}$, H]), which consists of those formulas of G(pLTL) allowing only the past modalities *yesterday* (Y), *weak yesterday* ($\widetilde{\text{Y}}$), and *historically* (H). By 'counterpart' we mean that a language is expressible in LTL[X, F] (resp., F(pLTL[Y, $\widetilde{\text{Y}}$, O])) if and only if its complement is expressible in LTL[X, G] (resp., G(pLTL[Y, $\widetilde{\text{Y}}$, H])). The logics LTL[X, G] and G(pLTL[Y, $\widetilde{\text{Y}}$, H]) are included in the *safety fragments*, as they only allow one to express safety properties.

We devise a *singly exponential* pastification algorithm for LTL[X, F] into F(pLTL[Y, $\widetilde{\text{Y}}$, O]), that transforms formulas of the former fragment into equivalent formulas of the latter.

At a high level, the proposed procedure consists of two main phases:

1. A transformation of the input LTL[X, F] formula ϕ into a suitably defined *normal form* whose size is, in the worst case, exponential in the size of ϕ. This step involves the bottom-up application of a set of rewriting rules based on tautologies of the logic.
2. The construction, for any formula in normal form, of a *dependency tree* representing the temporal relations between subformulae in the scope of an F modality. From such a tree, we extract an F(pLTL) formula which is equivalent to the original one and whose size is at most quadratic in the size of the formula obtained from the previous step.

The F(pLTL[Y, $\widetilde{\text{Y}}$, O]) formula returned as output is in Negation Normal Form and it does *not* contain the *since* modality S (the past counterpart of the *until* U). The absence of S is in line with the results in [7], showing that F(pLTL[Y, $\widetilde{\text{Y}}$, O]) is expressively equivalent to LTL[X, F]. Thanks to the duality of modalities F and G, a singly exponential pastification procedure for the LTL[X, G] into G(pLTL[Y, $\widetilde{\text{Y}}$, H]) can be easily obtained from the proposed algorithm.

The algorithm runs in time $2^{\mathcal{O}(n)}$, where n is the size of the input formula. In [20], it is shown that LTL[X, F] can be exponentially more succinct than F(pLTL[Y, $\widetilde{\text{Y}}$, O]). More precisely, there is an infinite family $\phi_1, \phi_2, \ldots$ of formulae in LTL[X, F] such that, for every $k \geq 1$, the formula ϕ_k has size k and can only be expressed in F(pLTL[Y, $\widetilde{\text{Y}}$, O]) using formulae of size $2^{\Omega(k)}$. This results implies a $2^{\Omega(n)}$ lower bound for the pastification problem, and thus the optimality of the proposed algorithm.

We implemented the developed algorithm in the temporal reasoning tool BLACK [21,22]. The implementation consists of about 500 lines of code only, thanks largely to the tautology-driven nature of the algorithm. We provide an experimental evaluation comparing input formulae and the corresponding F(pLTL) formulae produced by BLACK, with respect to different parameters such as the size of the formula and the number of temporal modalities.

This paper is a considerably revised and extended version of [1], that appeared in the proceedings of KR 2023. Besides providing full proofs of all results (not present in the conference contribution), this paper presents some simplifications to the original algorithm, together with a refined analysis of its complexity, showing the aforementioned $2^{\mathcal{O}(n)}$ runtime upper bound (the conference paper only established an $2^{\mathcal{O}(n^2)}$ upper bound). In view of the recent $2^{\Omega(n)}$ lower bound given in [20], which appeared only after the conference paper was published, the refinement of the complexity analysis is important to prove the optimality of the algorithm.

Outline of the paper. Section 2 reviews relevant previous work on the considered fragment and on the pastification problem. Section 3 introduces the necessary background on LTL. The core of the paper begins with Section 4, which presents the transformation of LTL formulae into a normal form, along with an analysis of its computational complexity. Section 5 introduces the concept of dependency tree and details the translation into purely past temporal formulae. Section 6 completes the complexity analysis of the overall procedure, establishing its optimality. Section 7 reports on the experimental evaluation of the implementation. Finally, Section 8 concludes the paper with a discussion of possible directions for future research.

2. Related work

2.1. LTL[X, F] and related fragments

In [23, Theorem 3.7], Sistla and Clarke show that the satisfiability problem, that is, the problem of checking whether the language of a formula is not empty, for LTL[X, F] is NP-complete, whereas Markey proves in [24, Theorem 28] that the same problem for LTL[X, G] is PSPACE-complete.

As for expressiveness, the exact power of LTL[X, F] and F(pLTL[Y, $\widetilde{\text{Y}}$, O]), as well as of the counterpart logics LTL[X, G] and G(pLTL[Y, $\widetilde{\text{Y}}$, H]), is investigated in [7]. In particular, LTL[X, F] and F(pLTL[Y, $\widetilde{\text{Y}}$, O])) are shown to be *expressively equivalent*, meaning that a language is definable by a formula of the former fragment if and only if it is definable by a formula of the latter, and the same holds for the fragments LTL[X, G] and G(pLTL[Y, $\widetilde{\text{Y}}$, H]). Moreover, it has been shown that the languages expressible in LTL[X, F] are precisely those that are *closed under pseudo-stuttering*, that is, languages where every word contains a bounded number of positions such that, if one inserts an arbitrary symbol anywhere outside those positions, the resulting word remains in the language. Finally, the set of languages definable in LTL[X, F] (resp., LTL[X, G]) is strictly included in the set of cosafety (resp., safety) languages definable in LTL: there are LTL-definable cosafety (resp., safety) properties that are not definable in LTL[X, F] (resp., LTL[X, G]) [7].

As for the succinctness of these fragments, as already mentioned in the introduction, in [7] it has been proved that LTL[X, F] can be exponentially more succinct than F(pLTL[Y, $\widetilde{\text{Y}}$, O]), and in [20] it has been shown that the converse holds as well, implying that LTL[X, F] and F(pLTL[Y, $\widetilde{\text{Y}}$, O]) are expressively incomparable with respect to succinctness (a formal definition of succinctness will be given in Section 6). Moreover, the same incomparability result applies to the fragments LTL[X, G] and G(pLTL[Y, $\widetilde{\text{Y}}$, H]). As we pointed out in the introduction, the above-discussed results, together with the singly exponential pastification of LTL[X, F] into F(pLTL[Y, $\widetilde{\text{Y}}$, O]) established in this paper, show that the exponential bound on the succinctness of LTL[X, F] with respect to F(pLTL[Y, $\widetilde{\text{Y}}$, O]) is *tight*, that is, there is at least one language such that

(i) it is definable with a formula of LTL[X, F];
(ii) any formula of F(pLTL[Y, $\widetilde{\text{Y}}$, O]) that defines it is *at most* of exponential size.

The same applies to the logics LTL[X, G] and G(pLTL[Y, $\widetilde{\text{Y}}$, H]).

2.2. Pastification, separability, and back to the future

The expressive equivalence of LTL+P and LTL proved in [25, Theorem 2.1], thanks to which past temporal modalities are shown to be eliminable in favour of a future-only logic, can be viewed as a result that goes in the opposite direction of the pastification process considered here. Moreover, the succinctness result provided by [26, Theorem 3.1] gives a (single) exponential lower bound for such a procedure. However, for such a "futurization" direction, the best procedure in the literature is only triply exponential in the size of the formula [5], leaving open the problem of a tight complexity result. This is in contrast with the pastification process for the fragments considered in this paper, for which a tight singly exponential bound is provided.

The expressive equivalence of LTL+P and LTL is also related to the notion of *separability* as formulated in Gabbay's celebrated Separation Theorem [27], stating that an LTL+P formula can be expressed as a Boolean combination of pure past, pure present, and pure future formulas. However, despite being related in spirit, the outcome of pastification is quite different from the one guaranteed by separability. Indeed, rather than imposing a clear-cut separation between pure past and pure future components, pastification only requires a pure past component to be enclosed in a pure future envelope.

Moreover, Gabbay's Separation Theorem [27,28] links the separability of a formula (in a given temporal logic) to *expressive completeness*, namely the ability to capture exactly the properties definable in the first-order theory of the successor function. Indeed, the theorem states that a temporal logic is expressively complete if and only if every formula in it is separable. In contrast, pastification does not yield any correspondence with expressive equivalence. Finally, it is known that algorithms for effectively computing a separated temporal formula from an LTL+P formula are non-elementary, whereas pastification procedures for LTL+P (over infinite or finite traces) are at most triply exponential.

2.3. Pastification compared to DFA construction

One might wonder whether transforming a temporal logic formula, e.g., an LTL[X, F] formula, into a DFA, rather than performing pastification, may lead to a better approach in practice. Indeed, it can be argued, any problem that involves the pastification of an LTL[X, F] formula and then the construction of the corresponding DFA—which will be doubly exponential in the size of the original formula—could likewise be addressed by first generating an NFA from the LTL[X, F] formula and subsequently determinizing it into a DFA.

First, we point out that, while the study of the compilation complexity of LTL[X, F] formulas into equivalent deterministic automata—both over finite words (DFA) and over infinite ones (deterministic Büchi automata, DBA)—remains an open problem, it is unlikely that LTL[X, F] formulas can be compiled into DFA (or DBA) in singly-exponential time and space. Consider, for instance, the language defined by the formula $\bigwedge_{i=1}^{n}(p_i \rightarrow \mathsf{X}\mathsf{F}p_i)$. Any equivalent DFA must keep track of all possibilities concerning both the proposition letters that hold at the initial time point and the orders in which they may reappear in subsequent time points. As an example, if p_1, p_2, and p_3 are true at the initial time point, the automaton must account for the possibility of seeing first p_1, then p_2, and finally p_3 in the future, or first p_2, then p_1, and finally p_3; and so on. This is equivalent to requiring that, for every possible subset $S \subseteq \{p_1, \dots, p_n\}$ of proposition letters true at the initial state (2^n choices), the DFA maintains a number of possibilities proportional to the subsets of S ($2^{|S|}$), leading to a doubly-exponential complexity.

Second, we wouild like to emphasize that even if DFA corresponding to LTL[X, F] formulas were doubly exponential—matching the complexity of pastifying LTL[X, F] and then generating a DFA from the resulting pure-past formula—pastification as a *tool* would remain valuable in its own right. Indeed, compiling into a DFA is not always a desirable strategy: although DFA provide an executable (albeit non-declarative) formalism that can be implemented directly, in many applications one never intends to construct the DFA explicitly, e.g.,

(i) in the context of declarative programming, and
(ii) when generating executable code for a runtime monitor for a temporal formula (or in symbolic model checking).

Below, we provide further support to our claim in these two contexts.

In Gabbay's contributions [27,29], a programming language (MetaTeM) is introduced where only the future-time component is imperative (with all the usual drawbacks), whereas the past-time component is declarative. The crucial remark is that a past-time temporal formalism is *both declarative and executable* [27], because the past has already occurred. Thus, the programming code basically corresponds one-to-one to a pure-past temporal formula. An algorithm for pastification is therefore essential when one wishes to preserve the declarative nature of the language while still compiling into an "executable" or "implementable" fragment (declarative past). If such a pastification algorithm runs in singly exponential time—as ours does—then the resulting declarative and implementable code is only singly exponential, rather than a non-declarative, doubly-exponential DFA. Even if LTL[X, F] formulas admitted singly exponential automata, going from an imperative formalism like DFA into a declarative and executable one is likely to incur into another exponential blowup, like in the case of translating automata into temporal logic (see Section 2.4).

In the context of monitoring (and symbolic model checking), symbolic automata are represented via Boolean formulas over a set X of state variables, where $I(X)$ specifies the set of initial states, $T(X, X')$ the transitions, and $F(X)$ the set of accepting states (we refer the reader to [30,31] for details). They offer three important advantages: (i) they can be exponentially more succinct than DFA; (ii) they correspond essentially to executable code implementing the DFA (a form of monitor that reads the input trace and updates its internal variables accordingly), and are therefore directly implementable; (iii) given a pure-past LTL formula, there exists an equivalent symbolic automaton of *linear* size.

To summarize, by applying the proposed pastification algorithm, we can translate a future-time fragment such as LTL[X, F] into a past-time one, obtaining symbolic automata of singly-exponential size, that can be used directly as executable code for monitoring or model checking. Using DFA in this setting would instead either incur into doubly-exponential complexity (if the conjecture of the doubly exponential lower bound on DFA corresponding to LTL[X, F] formulas is correct) or require a procedure to go from explicit-state automata (DFA) to symbolic ones, while typically one looks for the opposite translation.

2.4. A closer look to existing pastification algorithms

Pastification techniques were firstly introduced in relation to a fragment of *Metric Temporal Logic*, known as *bounded response Metric Temporal Logic* (MTL-B, for short) [19]. Such a logic is interpreted over dense linear orders, and temporal modalities have bounds that constrain their interval of application. As an example, the formula $\varphi \mathsf{U}_{[a,b]} \psi$, where $a, b \in \mathbb{N}$ are given in binary, when interpreted at time t, states that ψ must happen at a time t' in between $t + a$ and $t + b$ and ϕ must be true at all times in the interval between t and t'. In [18,19], Maler *et al.* developed a pastification algorithm for MTL-B producing polynomial-size formulae. The procedure exploits the fact that, in every model of an MTL-B formula ϕ, there is a furthermost time point t after which ϕ does not impose any constraint on the model. The algorithm returns a formula that

(i) uses only past modalities,
(ii) is polynomial in $|\phi|$, and

(iii) is equivalent to ϕ when interpreted at time point t instead of at the origin.

When interpreting the logic over discrete linear orders, and representing all time bounds, e.g., a and b above, in unary instead of binary, MTL-B is equivalent to LTL[X]. The algorithm in [18,19] is therefore a pastification procedure for LTL[X] as well.

As for the more expressive logics coSafetyLTL and SafetyLTL, the best-known upper bound on their pastification into G(pLTL) and F(pLTL), respectively, is *triply exponential* [32,33], as already pointed out in the introduction. Such a triply exponential procedure behaves as follows:

1. Starting from the input coSafetyLTL formula ϕ, build the minimal DFA $\mathcal{A}$ recognizing the set of *good prefixes* of ϕ [34]. These are those *finite* traces that, no matter how extended, generate a model of ϕ. In the worst case, the size of $\mathcal{A}$ is doubly exponential in the size of ϕ.
2. Decompose $\mathcal{A}$ into a *cascade product* $\mathcal{C}$ of *reset automata*.[2] This decomposition is based on the celebrated Krohn-Rhodes cascaded decomposition theorem [35], which decomposes any DFA into a cascade of automata of only two types: permutations and resets. In the case of *counter-free automata* [36]—like $\mathcal{A}$—reset automata suffice [32]. In the worst case, the size of $\mathcal{C}$ is exponential in the size of $\mathcal{A}$.
3. Construct a pure past LTL formula ψ that defines the language of $\mathcal{C}$. This step uses the algorithm from [33].[3] The size of ψ is polynomial in the size of $\mathcal{C}$, and by construction the input ϕ is equivalent to F(ψ).

The case of SafetyLTL is analogous, with the roles of good and bad prefixes reversed. In this context, a bad prefix is a finite trace such that any of its extensions necessarily leads to a violation of the formula ϕ.

It is worth observing that, despite being decades old, to the best of our knowledge there is only one implementation of the Krohn-Rhodes cascaded decomposition procedure required in the second step of the algorithm[4], and no implementation of the pastification algorithm for coSafetyLTL and SafetyLTL. This is maybe partially due to the highly technical nature of this algorithm. Our choice of designing a purely tautology-based procedure was motivated by the goal of obtaining a solution that is simple to implement and to maintain.

3. Preliminaries

In this section, we formally describe syntax and semantics of the logics we consider, and formalize the notion of pastification.

Let $\mathcal{AP}$ be a (countable) set of proposition letters. Formulae $\phi, \psi, \ldots$ of *Linear Temporal Logic with Past* (LTL+P) are built from the grammar:

$$
\begin{aligned}
\phi ::= \top \mid \bot \mid p \mid \neg p & \qquad \text{true, false, and literals} \\
\mid \phi \vee \phi \mid \phi \wedge \phi & \qquad \text{Boolean connectives} \\
\mid \mathsf{X}\phi \mid \mathsf{F}\phi \mid \phi \mathsf{U} \phi \mid \mathsf{G}\phi \mid \phi \mathsf{R} \phi & \qquad \text{future modalities} \\
\mid \mathsf{Y}\phi \mid \widetilde{\mathsf{Y}}\phi \mid \mathsf{O}\phi \mid \phi \mathsf{S} \phi \mid \mathsf{H}\phi \mid \phi \mathsf{T} \phi & \qquad \text{past modalities}
\end{aligned}
$$

where $p \in \mathcal{AP}$. The future modalities X, F, U, G, and R are called *tomorrow*, *eventually*, *until*, *globally*, and *release*, respectively. The past modalities are, instead, *yesterday* (Y), *weak yesterday* ($\widetilde{\text{Y}}$), *once* (O), *since* (S), *historically* (H), and *trigger* (T). Notice that formulae from the above grammar are in Negation Normal Form (NNF), that is, negation is applied to proposition letters only. This is without loss of generality, as the grammar is closed under taking duals of temporal operators. Indeed, based on the forthcoming semantics, the following equivalences hold: $\neg \mathsf{F}\phi \equiv \mathsf{G}\neg\phi$; $\neg(\phi \mathsf{U} \psi) \equiv (\neg\phi)\mathsf{R}\neg\psi$; $\neg \mathsf{O}\phi \equiv \mathsf{H}\neg\phi$; $\neg(\phi \mathsf{S} \psi) \equiv (\neg\phi)\mathsf{T}\neg\psi$.

An *infinite trace* σ is infinite sequence $\sigma_0\sigma_1 \ldots$ of *states* σ_i from $2^{\mathcal{AP}}$, that is, σ is an element of $(2^{\mathcal{AP}})^\omega$. The *satisfaction* of a formula ϕ of LTL+P with respect to a trace σ and time $i \in \mathbb{N}$, denoted by $\sigma, i \vDash \phi$, is defined as follows:

[2] The term "cascade product" is used to refer to a particular product of two deterministic automata where the first one reads symbols belonging to an alphabet, say, Γ and the second one reads symbols that belong to $\Gamma \times Q$, where Q is the set of state of the first automaton. A reset automaton with alphabet Γ and set of states Q is a DFA such that, for each letter $\tau \in \Gamma$, the function from Q to Q induced by τ either is the identity or has a range of cardinality 1. We refer the reader to [33] for details.

[3] The rationale behind such an algorithm is that each reset automaton can be mapped directly into a pure past LTL formula, and the cascade product corresponds to the nesting of pure past LTL formulae.

[4] https://github.com/gap-packages/sgpdec.

$\sigma, i \vDash \top$	is	always true;
$\sigma, i \vDash \bot$	is	always false;
$\sigma, i \vDash p$	iff	$p \in \sigma_i$;
$\sigma, i \vDash \neg \mathsf{p}$	iff	$p \notin \sigma_i$;
$\sigma, i \vDash \phi \vee \psi$	iff	$\sigma, i \vDash \phi$ or $\sigma, i \vDash \psi$;
$\sigma, i \vDash \phi \wedge \psi$	iff	$\sigma, i \vDash \phi$ and $\sigma, i \vDash \psi$;
$\sigma, i \vDash \mathsf{X}\phi$	iff	$\sigma, i+1 \vDash \phi$;
$\sigma, i \vDash \mathsf{F}\phi$	iff	there is $j \geq i$ such that $\sigma, j \vDash \phi$;
$\sigma, i \vDash \phi \mathsf{U} \psi$	iff	there is $j \geq i$ such that $\sigma, j \vDash \psi$, and $\sigma, k \vDash \phi$ for every k satisfying $i \leq k < j$;
$\sigma, i \vDash \mathsf{G}\phi$	iff	for all $j \geq i$ it holds that $\sigma, j \vDash \phi$;
$\sigma, i \vDash \phi \mathsf{R} \psi$	iff	either $\sigma, j \vDash \psi$ for every $j \geq i$, or there is $k \geq i$ such that $\sigma, k \vDash \phi$ and $\sigma, j \vDash \psi$ for every $i \leq j \leq k$;
$\sigma, i \vDash \mathsf{Y}\phi$	iff	$i > 0$ and $\sigma, i-1 \vDash \phi$;
$\sigma, i \vDash \widetilde{\mathsf{Y}}\phi$	iff	either $i = 0$ or $\sigma, i-1 \vDash \phi$;
$\sigma, i \vDash \mathsf{O}\phi$	iff	there is $0 \leq j \leq i$ such that $\sigma, j \vDash \phi$;
$\sigma, i \vDash \phi \mathsf{S} \psi$	iff	there is $0 \leq j \leq i$ such that $\sigma, j \vDash \psi$, and $\sigma, k \vDash \phi$ for every k satisfying $j < k \leq i$;
$\sigma, i \vDash \mathsf{H}\phi$	iff	for all $0 \leq j \leq i$ it holds that $\sigma, j \vDash \phi$;
$\sigma, i \vDash \phi \mathsf{T} \psi$	iff	either $\sigma, j \vDash \psi$ for every $0 \leq j \leq i$, or there is $0 \leq k \leq i$ such that $\sigma, k \vDash \phi$ and $\sigma, j \vDash \psi$ for every $k \leq j \leq i$.

We say that σ is a *model* of ϕ, denoted by $\sigma \vDash \phi$, whenever $\sigma, 0 \vDash \phi$. We write $\mathcal{L}(\phi)$ for the *language* of ϕ, that is, the set of all models of ϕ. We say that two formulae ϕ and ψ of LTL+P are *equivalent*, written $\phi \equiv \psi$, whenever $\mathcal{L}(\phi) = \mathcal{L}(\psi)$. In Section 4.2, we will make use of the notion of strong equivalence, which requires that, for every trace σ and every time point i belonging to it, the two formulae are either both satisfied or both violated. It is formally defined as follows.

Definition 1. Two formulas ϕ and ψ are *strongly equivalent*, denoted by $\phi \doteq \psi$, if (and only if) for every $\sigma \in (2^{\mathcal{AP}})^\omega$ and every $k \geq 0$, $\sigma, k \vDash \phi$ if and only if $\sigma, k \vDash \psi$.

Notice that, in the case of pure future fragments of LTL+P, the notions of equivalence and strong equivalence coincide. On the contrary, the LTL+P formulas $\mathsf{p} \vee \mathsf{Y}\mathsf{q}$ and p are examples of formulae which are equivalent (since $\sigma, 0 \vDash \mathsf{Y}\mathsf{q}$ iff $\sigma, 0 \vDash \bot$), but not strongly equivalent.

Throughout the paper, given $n \in \mathbb{N}$, we write $\mathsf{X}^n\phi$ for the formula inductively defined as $\mathsf{X}^0\phi := \phi$ and $\mathsf{X}^{n+1}\phi := \mathsf{X}\mathsf{X}^n\phi$, and write $\mathsf{Y}^n\phi$ and $\widetilde{\mathsf{Y}}^n\phi$ for similarly defined formulae featuring Y and $\widetilde{\mathsf{Y}}$ instead of X, respectively. The *size* of a formula $\phi \in$ LTL+P, denoted by $\mathrm{size}(\phi)$, is inductively defined as follows: $\mathrm{size}(\top) = \mathrm{size}(\bot) = \mathrm{size}(p) = \mathrm{size}(\neg p) := 1$, $\mathrm{size}(\otimes\phi) := \mathrm{size}(\phi) + 1$ for $\otimes \in \{\mathsf{X}, \mathsf{F}, \mathsf{G}, \mathsf{Y}, \widetilde{\mathsf{Y}}, \mathsf{O}, \mathsf{H}\}$, and $\mathrm{size}(\phi_1 \oplus \phi_2) := \mathrm{size}(\phi_1) + \mathrm{size}(\phi_2) + 1$ for $\oplus \in \{\vee, \wedge, \mathsf{U}, \mathsf{R}, \mathsf{S}, \mathsf{T}\}$.

A *pure future* (resp., *pure past*) *formula* is an LTL+P formula devoid of occurrences of past (resp., future) modalities. We denote by LTL (resp., pLTL) the set of pure future (resp., pure past) formulae. Given a list $\mathcal{OP}$ of temporal modalities, we denote by LTL[$\mathcal{OP}$] (resp., pLTL[$\mathcal{OP}$]) the set of LTL (resp., pLTL) formulae only using temporal modalities from $\mathcal{OP}$. As an example, LTL[X, F] stands for the fragment of LTL only featuring modalities X and F. Furthermore, we write F(pLTL[$\mathcal{OP}$]) (resp., G(pLTL[$\mathcal{OP}$])) for the set of formulae of the form $\mathsf{F}(\alpha)$ (resp., $\mathsf{G}(\alpha)$), where α belongs to pLTL[$\mathcal{OP}$].

Finally, for all $k \in \mathbb{N}$, we define the notion of *k-exponential pastification*. It refers to an arbitrary procedure that transforms formulae of a given fragment of LTL+P into equivalent ones belonging to either F(pLTL) or G(pLTL). We say it to be k-exponential if it always outputs a formula whose size is bounded by a tower of k exponentials in the size of the input formula.

Definition 2 (k-exponential pastification)**.** Given a set $\mathbb{L}$ of formulae in LTL+P and a set of formulae $\mathbb{L}'$ either in F(pLTL) or in G(pLTL), a *pastification procedure from $\mathbb{L}$ into $\mathbb{L}'$ of k-exponential size* is an algorithm that, for any $\phi \in \mathbb{L}$, returns a formula $\psi \in \mathbb{L}'$, equivalent to ϕ, such that $\mathrm{size}(\psi) \in \exp(k, \mathcal{O}(\mathrm{size}(\phi)))$, where $\exp(0, n) := n$ and $\exp(i+1, n) := 2^{\exp(i,n)}$.

Definition 2 allows us to summarize the picture outlined in Section 2 as follows: there exist a 0-exponential pastification procedure from LTL[X] into F(pLTL) and a 3-exponential one from coSafetyLTL (resp., SafetyLTL) into F(pLTL) (resp., G(pLTL)). Our main result is a 1-exponential pastification procedure from LTL[X, F] into F(pLTL[Y, $\widetilde{\mathsf{Y}}$, O]).

4. Normalization

Let us now describe the proposed pastification procedure for LTL[X, F]. In this section, we illustrate the first step of the procedure, which brings the input LTL[X, F] formula into a suitably defined normal form. The transformation is guided by rewriting rules that are based on tautologies of LTL[X, F].

4.1. A normal form for LTL[X, F]

Before providing the formal definitions, let us explain the rational behind the normal form we are going to use. As an example, take the formula (for readability, we use $\rightarrow$ instead of $\vee$)

$$\phi := \mathsf{F}\big((p_1 \rightarrow \mathsf{F}q_1) \wedge (p_2 \rightarrow \mathsf{F}q_2)\big),$$

where, for $i = 1, 2$, both p_i and q_i belong to $\mathcal{AP}$. Imagine constructing a model for ϕ. A first choice to be made is selecting *which* (if any) of the atomic letters p_1 and p_2 are true in the state of the trace satisfying the formula $(p_1 \rightarrow \mathsf{F}q_1) \wedge (p_2 \rightarrow \mathsf{F}q_2)$. Let us assume

both p_1 and p_2 are set to true. Afterwards, a second choice to be made is *when* (and so, in what order) the letters q_1 and q_2 are to be satisfied. The formula does not impose any temporal dependency on this, so having q_1 followed by q_2, or the *vice versa*, or having q_1 and q_2 true at the same state of the trace are all equally acceptable solutions. Four examples of possible models are:

$$\dots\{p_1,p_2\}\dots\{q_1\}\dots\{q_2\}\dots \qquad \dots\{p_1,p_2\}\dots\{q_1,q_2\}\dots$$
$$\dots\{p_2\}\dots\{q_2\}\dots \qquad \dots\{p_1,q_1\}\dots$$

Our normal form is designed to "move" the choice of *which* subformulae to satisfy at the top-level of the formula, while preserving the temporal dependencies regarding *when* subformulae are to be satisfied. In our example, this is simple to do: seeing $\mathsf{F}q_i$ as atomic propositions, we first bring the formula $(p_1 \to \mathsf{F}q_1) \wedge (p_2 \to \mathsf{F}q_2)$ in *disjunctive normal form* (DNF), to then distribute the top-most F under the scope of disjunction. The result is

$$\mathsf{F}(\neg p_1 \wedge \neg p_2) \vee \mathsf{F}(\neg p_2 \wedge \mathsf{F}q_1) \vee \mathsf{F}(\neg p_1 \wedge \mathsf{F}q_2) \vee \mathsf{F}(\mathsf{F}q_1 \wedge \mathsf{F}q_2).$$

We see that each disjunct of this formula clarifies *which* of the subformulae p_1, p_2, $\mathsf{F}q_1$ and $\mathsf{F}q_2$ must be satisfied, while preserving the (lack of) temporal dependencies between q_1 and q_2.

Moving to the formal definition, our is a "bilevel" normal form. The inner level asserts the temporal dependencies given by the modality F:

Definition 3 (Inner level of the normal form)**.** We write $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL[Y]})$ for the set of formulae ϕ generated by the grammar $\phi ::= \psi \mid \phi \wedge \phi \mid \mathsf{F}\phi$, where ψ is a formula from the logic $\mathsf{pLTL[Y]}$.

The outer level guesses *which* subformulae are to be satisfied:

Definition 4 (The normal form of $\mathsf{LTL[X,F]}$)**.** We write $\mathsf{nf}(\mathsf{LTL[X,F]})$ for the set of all formulae of the form $\mathsf{X}^k \bigvee_{i=1}^{c} \phi_i$, where $k, c \in \mathbb{N}$, and every ϕ_i belongs to $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL[Y]})$.

4.2. *Translating* $\mathsf{LTL[X,F]}$ *into* $\mathsf{nf}(\mathsf{LTL[X,F]})$

The transformation of $\mathsf{LTL[X,F]}$ into $\mathsf{nf}(\mathsf{LTL[X,F]})$ consists of the application of six rewriting rules $\phi \mapsto \psi$ ($\mathbf{R}_1$–$\mathbf{R}_6$). It is worth pointing out that every rule $\phi \mapsto \psi$ will be applied to arbitrary subformulae of the input formula. Because of this, it is not sufficient for ϕ and ψ to be equivalent, they should rather be *strongly equivalent* (Definition 1), that is, they should be equivalent when interpreted at any point along a trace. An exception to this is $\mathbf{R}_4$ dealing with the yesterday modality Y introduced by rule $\mathbf{R}_1$: the formulae in this rule are equivalent only under the assumption that $k \geq 1$ (the normalization procedure—Algorithm 1—will only use this rule by respecting this constraint). Moreover, recall that, in the case of pure future fragments of $\mathsf{LTL{+}P}$ the notions of equivalence and strong equivalence coincide. However, formulae in $\mathsf{nf}(\mathsf{LTL[X,F]})$ contain the yesterday modality Y, making the distinction necessary.

Here is the list of rules $\mathbf{R}_1$–$\mathbf{R}_6$, together with an informal description of their role in the normalization procedure:

$\mathbf{R}_1$: $\mathsf{X}^i\phi_1 \otimes \mathsf{X}^j\phi_2 \;\mapsto\; \mathsf{X}^j(\mathsf{Y}^{j-i}\phi_1 \otimes \phi_2)$, where $i \leq j$, $j \geq 1$, $\otimes \in \{\wedge, \vee\}$;

$\mathbf{R}_2$: $\mathsf{FX}\phi \;\mapsto\; \mathsf{XF}\phi$

[The role of $\mathbf{R}_1$ and $\mathbf{R}_2$ is to pull modality X at the top-most level of the formula, following the "outer level" of the normal form (see Definition 4)];

$\mathbf{R}_3$: $\mathsf{Y}(\phi_1 \otimes \phi_2) \;\mapsto\; \mathsf{Y}\phi_1 \otimes \mathsf{Y}\phi_2$, where $\otimes \in \{\wedge, \vee\}$;

$\mathbf{R}_4$: $\mathsf{YF}\phi \;\mapsto\; \mathsf{FY}\phi$;

[The role of $\mathbf{R}_3$ and $\mathbf{R}_4$ is to push modality Y so that no occurrence of F is in the scope of this modality (as in Definition 3)];

$\mathbf{R}_5$: $\mathsf{F}(\phi_1 \vee \phi_2) \;\mapsto\; (\mathsf{F}\phi_1) \vee (\mathsf{F}\phi_2)$;

$\mathbf{R}_6$: $(\phi_1 \vee \phi_2) \wedge \phi_3 \;\mapsto\; (\phi_1 \wedge \phi_3) \vee (\phi_2 \wedge \phi_3)$

[The role of $\mathbf{R}_5$ and $\mathbf{R}_6$ is to push F and $\wedge$ in the scope of $\vee$, creating the disjunction in the "outer level" of the normal form (see Definition 4)],

where $\phi, \phi_1, \phi_2, \phi_3 \in \mathsf{LTL[X,F,Y]}$.

We tacitly assume to have a rule to commute both conjuncts and disjuncts. This rule is only used to apply rule $\mathbf{R}_1$ also on formulae of the form $\mathsf{X}^i\phi_1 \otimes \mathsf{X}^j\phi_2$, with $i > j$, and rule $\mathbf{R}_6$ on formulae of the form $\phi_3 \wedge (\phi_1 \vee \phi_2)$.

The lemmas below state the correctness of the six rules.

Lemma 1. *For every rule $\phi \mapsto \psi$ among $\mathbf{R}_1$–$\mathbf{R}_6$, except $\mathbf{R}_4$, we have $\phi \doteq \psi$.*

Algorithm 1 $\mathtt{NF}(\phi)$: algorithm to bring $\mathsf{LTL[X,F]}$ formulae in normal form.

▷ below, all formulae ψ_i and $\psi_{i,j}$ are in $\mathsf{conj(F, pLTL[Y])}$

1: **if** $\phi \in \mathsf{conj(F, pLTL[Y])}$ **then return** ϕ
2: **if** $\phi = \mathsf{X}(\psi)$ **then return** $\mathsf{X}(\mathtt{NF}(\psi))$
3: **if** $\phi = \mathsf{F}(\psi)$ **then**
4: $\quad \left(\mathsf{X}^k \bigvee_{i=1}^{d} \psi_i\right) := \mathtt{NF}(\psi)$
5: $\quad$ **return** $\mathsf{X}^k \bigvee_{i=1}^{d} \mathsf{F}\psi_i$ ▷ $\mathbf{R}_2, \mathbf{R}_5$
6: **if** $\phi = (\phi_1 \otimes \phi_2)$, where $\otimes \in \{\vee, \wedge\}$ **then**
7: $\quad$ **for** $i \in \{1,2\}$ **let** $\left(\mathsf{X}^{k_i} \bigvee_{j=1}^{d_i} \psi_{i,j}\right) := \mathtt{NF}(\phi_i)$
8: $\quad$ **and for** $j \in [1, d_i]$ **let** $\gamma_{i,j} := \mathtt{pushY}(\mathsf{Y}^{\max(k_1,k_2)-k_i}\, \psi_{i,j})$ ▷ $\mathbf{R}_3, \mathbf{R}_4$
9: $\quad$ **if** $\otimes$ is $\vee$ **then return** $\mathsf{X}^{\max(k_1,k_2)}\left(\left(\bigvee_{j=1}^{d_1} \gamma_{1,j}\right) \vee \left(\bigvee_{k=1}^{d_2} \gamma_{2,k}\right)\right)$ ▷ $\mathbf{R}_1$
10: $\quad$ **if** $\otimes$ is $\wedge$ **then return** $\mathsf{X}^{\max(k_1,k_2)} \bigvee_{j=1}^{d_1} \bigvee_{k=1}^{d_2} \left(\gamma_{1,j} \wedge \gamma_{2,k}\right)$ ▷ $\mathbf{R}_1, \mathbf{R}_6$

Proof. The proofs for Rules $\mathbf{R}_5$ and $\mathbf{R}_6$ are straightforward (in fact, the latter is a standard tautology of propositional logic). We only prove the strong equivalence of the remaining rules. Let $\sigma \in (2^{\mathcal{AP}})^\omega$ be an arbitrary trace and $k \geq 0$. Below, with some abuse of notation, we will interchangeably use the symbols $\wedge$ and $\vee$ both as constructs of LTL+P and as symbols in the meta-language that we use to formalize the proof.

Proof of $\mathbf{R}_1$*:* For $i \leq j$ and $\otimes \in \{\wedge, \vee\}$, we have:

$$\begin{aligned}
\sigma, k \vDash \mathsf{X}^i\phi_1 \otimes \mathsf{X}^j\phi_2 \;\; \textit{iff} \;\; & (\sigma, k+i \vDash \phi_1) \otimes (\sigma, k+j \vDash \phi_2) \\
\textit{iff} \;\; & (\sigma, k+j-(j-i) \vDash \phi_1) \otimes (\sigma, k+j \vDash \phi_2) \\
\textit{iff} \;\; & \sigma, k+j \vDash \mathsf{Y}^{j-i}\phi_1 \otimes \phi_2 \\
\textit{iff} \;\; & \sigma, k \vDash \mathsf{X}^j(\mathsf{Y}^{j-i}\phi_1 \otimes \phi_2).
\end{aligned}$$

Proof of $\mathbf{R}_2$*:*

$$\begin{aligned}
\sigma, k \vDash \mathsf{FX}\phi \;\; \textit{iff} \;\; & \sigma, h \vDash \mathsf{X}\phi \text{ for some } h \geq k \\
\textit{iff} \;\; & \sigma, h+1 \vDash \phi \text{ for some } h \geq k \\
\textit{iff} \;\; & \sigma, h \vDash \phi \text{ for some } h \geq k+1 \\
\textit{iff} \;\; & \sigma, k+1 \vDash \mathsf{F}\phi \\
\textit{iff} \;\; & \sigma, k \vDash \mathsf{XF}\phi.
\end{aligned}$$

Proof of $\mathbf{R}_3$*:* For $\otimes \in \{\wedge, \vee\}$, we have:

$$\begin{aligned}
\sigma, k \vDash \mathsf{Y}(\phi_1 \otimes \phi_2) \;\; \textit{iff} \;\; & \sigma, k-1 \vDash \phi_1 \otimes \phi_2 \text{ and } k > 0 \\
\textit{iff} \;\; & (\sigma, k-1 \vDash \phi_1) \otimes (\sigma, k-1 \vDash \phi_2), \text{ and } k > 0 \\
\textit{iff} \;\; & \sigma, k \vDash \mathsf{Y}\phi_1 \otimes \mathsf{Y}\phi_2.
\end{aligned}$$

□

Lemma 2. *The formulae in* $\mathbf{R}_4$ *are equivalent at all positions of a trace except the first one, that is,* $\sigma, k+1 \vDash \mathsf{YF}\phi$ *if and only if* $\sigma, k+1 \vDash \mathsf{FY}\phi$*, for every trace* $\sigma \in (2^{\mathcal{AP}})^\omega$ *and* $k \geq 0$*.*

Proof.

$$\begin{aligned}
\sigma, k+1 \vDash \mathsf{YF}\phi \;\; \textit{iff} \;\; & \sigma, k \vDash \mathsf{F}\phi \\
\textit{iff} \;\; & \sigma, h \vDash \phi \text{ for some } h \geq k \\
\textit{iff} \;\; & \sigma, h+1 \vDash \mathsf{Y}\phi \text{ for some } h \geq k \\
\textit{iff} \;\; & \sigma, h' \vDash \mathsf{Y}\phi \text{ for some } h' \geq k+1 \\
\textit{iff} \;\; & \sigma, k+1 \vDash \mathsf{FY}\phi.
\end{aligned}$$

□

Algorithm 1 specifies the enforcement strategy of rules $\mathbf{R}_1$–$\mathbf{R}_6$ in order to put the input $\mathsf{LTL[X,F]}$ formula in normal form. Comments in the pseudocode highlight where each rule is used. The algorithm calls the procedure $\mathtt{pushY}(\mathsf{Y}^j\psi)$, with ψ a formula in $\mathsf{conj(F, pLTL[Y])}$. When $j \geq 1$, the procedure $\mathtt{pushY}(\mathsf{Y}^j\psi)$ returns the formula in $\mathsf{conj(F, pLTL[Y])}$ obtained from $\mathsf{Y}^j\psi$ by iterating rules $\mathbf{R}_3$ and $\mathbf{R}_4$; when $j = 0$, it returns the input formula ψ. Observe that when $j \geq 1$, the formulae $\mathtt{pushY}(\mathsf{Y}^j\psi)$ built by the algorithm are later put in the scope of at least $j+1$ nested modalities X (lines 9 and 10). By Lemma 2, this justifies the use of rule $\mathbf{R}_4$.

Proposition 1. *Correctness of Algorithm 1 For every input* $\mathsf{LTL[X,F]}$ *formula* ϕ*, Algorithm 1 returns a formula* γ *in* $\mathsf{nf(LTL[X,F])}$*, such that* $\gamma \equiv \phi$*.*

Proof. The termination of the algorithm is straightforward: the input of every recursive call of $\mathtt{NF}(\cdot)$ is a proper subformula of the input. The proof that the output formula γ belongs to $\mathsf{nf(LTL[X,F])}$, and that $\phi \equiv \gamma$, is by induction on the structure of ϕ:

base case: $\phi \in \mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$. We have $\gamma = \phi$ (line 1). Trivially, $\gamma \equiv \phi$, and γ belongs to $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$, which is a subset of $\mathsf{nf}(\mathsf{LTL}[\mathsf{X}, \mathsf{F}])$.

induction step: $\phi = \mathsf{X}(\psi)$. We have $\gamma = \mathsf{X}(\mathtt{NF}(\psi))$ (line 2). By induction hypothesis, $\mathtt{NF}(\psi) \equiv \psi$, and so $\gamma \equiv \phi$. Moreover, $\mathtt{NF}(\psi)$ belongs to $\mathsf{nf}(\mathsf{LTL}[\mathsf{X}, \mathsf{F}])$. Hence, following Definition 4, the same holds for γ.

induction step: $\phi = \mathsf{F}(\psi)$. By induction hypothesis, $\mathtt{NF}(\psi) = \mathsf{X}^k \bigvee_{i=1}^{d} \psi_i$, where each ψ_i belongs to $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$. The output of the algorithm is a formula $\gamma = \mathsf{X}^k \bigvee_{i=1}^{d} \mathsf{F}\psi_i$ (line 5) which, by Definition 4, is in $\mathsf{nf}(\mathsf{LTL}[\mathsf{X}, \mathsf{F}])$. By induction, $\mathtt{NF}(\psi) \equiv \psi$ and thus $\phi \equiv \mathsf{F}(\mathtt{NF}(\psi))$. Since γ is obtained by applying rules $\mathbf{R}_2$ and $\mathbf{R}_5$ on $\mathsf{F}(\mathtt{NF}(\psi))$, then, by Lemma 1, $\mathsf{F}(\mathtt{NF}(\psi)) \equiv \gamma$. So, $\phi \equiv \gamma$.

induction step: $\phi = \phi_1 \otimes \phi_2$, **with** $\otimes \in \{\vee, \wedge\}$. By induction hypothesis, $\mathtt{NF}(\phi_i) = \mathsf{X}^{k_i} \bigvee_{j=1}^{d_i} \psi_{i,j}$, where each $\psi_{i,j}$ belongs to $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$ and $i \in \{1, 2\}$. Assume $k_1 \leq k_2$ (the case $k_2 < k_1$ is analogous). We have:

$$
\begin{aligned}
\phi &\equiv \left(\mathsf{X}^{k_1} \bigvee_{j=1}^{d_1} \psi_{1,j}\right) \otimes \left(\mathsf{X}^{k_2} \bigvee_{j=1}^{d_2} \psi_{2,j}\right) && \text{(by induction hypothesis)} \\
&\equiv \mathsf{X}^{k_2}\left(\left(\mathsf{Y}^{k_2-k_1} \bigvee_{j=1}^{d_1} \psi_{1,j}\right) \otimes \left(\bigvee_{j=1}^{d_2} \psi_{2,j}\right)\right) && \text{(by rule } \mathbf{R}_1\text{)} \\
&\equiv \mathsf{X}^{k_2}\left(\left(\bigvee_{j=1}^{d_1} \mathsf{Y}^{k_2-k_1} \psi_{1,j}\right) \otimes \left(\bigvee_{j=1}^{d_2} \psi_{2,j}\right)\right) && \text{(by rule } \mathbf{R}_5\text{)} \\
&\equiv \mathsf{X}^{k_2}\left(\left(\bigvee_{j=1}^{d_1} \mathtt{pushY}(\mathsf{Y}^{k_2-k_1} \psi_{1,j})\right) \otimes \left(\bigvee_{j=1}^{d_2} \psi_{2,j}\right)\right).
\end{aligned}
$$

where the last equivalence is trivial if $k_1 = k_2$, and otherwise follows from Lemma 2 by iteratively applying rules $\mathbf{R}_3, \mathbf{R}_4$. Now, if $\otimes$ stands for the symbol $\vee$, the last formula we obtained is equal to the output formula γ (see line 9). When instead $\otimes$ stands for $\wedge$, this formula is still equivalent to γ (line 10), by iterated applications of rule $\mathbf{R}_6$. Finally, by observing that $\mathtt{pushY}(\mathsf{Y}^{k_2-k_1} \psi_{1,j})$ generates a formula in $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$, we conclude that $\mathtt{NF}(\phi)$ is in $\mathsf{nf}(\mathsf{LTL}[\mathsf{X}, \mathsf{F}])$. □

4.3. Analysis of the complexity of Algorithm 1

We show that Algorithm 1 returns a formula of size $2^{\mathcal{O}(\mathrm{size}(\phi))}$, where ϕ is the input $\mathsf{LTL}[\mathsf{X}, \mathsf{F}]$ formula. For the analysis, we rely on three parameters:

1. $\mathrm{size}_{\mathsf{Y}}(\psi)$, where ψ is from $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$. This is the maximum size of any subformula of ψ in which the modality F does not occur.
2. $\mathrm{size}_{\mathsf{F}}(\psi)$, where ψ is from $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$. This is the size of the formula obtained from ψ by replacing with $\top$ all maximal subformulae in which the modality F does not occur.
3. $\mathcal{R}(\phi)$, where ϕ is from $\mathsf{LTL}[\mathsf{X}, \mathsf{F}]$. This quantity, measuring the number of clauses of a DNF-converted expression, is defined as follows:

$$
\begin{aligned}
\mathcal{R}(\chi) &:= 1 && \text{for } \chi \in \{\top, \bot, p, \neg p | p \in \mathcal{AP}\} \\
\mathcal{R}(\phi_1 \wedge \phi_2) &:= \mathcal{R}(\phi_1) \cdot \mathcal{R}(\phi_2) \\
\mathcal{R}(\phi_1 \vee \phi_2) &:= \mathcal{R}(\phi_1) + \mathcal{R}(\phi_2) \\
\mathcal{R}(\oplus \phi') &:= \mathcal{R}(\phi') && \text{for } \oplus \in \{\mathsf{X}, \mathsf{F}\}.
\end{aligned}
$$

By definition, we have $\mathrm{size}(\psi) \leq \mathrm{size}_{\mathsf{F}}(\psi) \cdot \mathrm{size}_{\mathsf{Y}}(\psi)$, for every formula ψ in $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$, and $\mathcal{R}(\phi) \leq 2^{\mathrm{size}(\phi)}$, for every formula ϕ in $\mathsf{LTL}[\mathsf{X}, \mathsf{F}]$.

Lemma 3. *Consider a formula ϕ from $\mathsf{LTL}[\mathsf{X}, \mathsf{F}]$, and let $\mathtt{NF}(\phi) = \mathsf{X}^j \bigvee_{i=1}^{c} \psi_i$. Then, $j \leq \mathrm{size}(\phi)$, $c \leq \mathcal{R}(\phi) \leq 2^{\mathrm{size}(\phi)}$, $\mathrm{size}_{\mathsf{F}}(\psi_i) \leq \mathrm{size}(\phi)$ and $\mathrm{size}_{\mathsf{Y}}(\psi_i) \leq \mathrm{size}(\phi) + j$.*

Proof. We proceed by induction on the structure of ϕ. Let $\gamma := \mathtt{NF}(\phi)$.

base case: ϕ **has no temporal modalities.** Following line 1, $\gamma = \phi$ and the statement trivially follows.

induction step: $\phi = \mathsf{X}(\psi)$. Let $\mathtt{NF}(\psi) = \mathsf{X}^k \bigvee_{i=1}^{d} \psi_i$. By induction hypothesis, $k \leq \mathrm{size}(\psi)$, $d \leq \mathcal{R}(\psi)$, $\mathrm{size}_{\mathsf{F}}(\psi_i) \leq \mathrm{size}(\psi)$ and $\mathrm{size}_{\mathsf{Y}}(\psi_i) \leq \mathrm{size}(\psi) + k$. We have $\gamma = \mathsf{X}^{k+1} \bigvee_{i=1}^{d} \psi_i$ (line 2), and the statement follows from $\mathrm{size}(\phi) = \mathrm{size}(\psi) + 1$.

induction step: $\phi = \mathsf{F}(\psi)$. Let $\mathtt{NF}(\psi) = \mathsf{X}^k \bigvee_{i=1}^{d} \psi_i$. By induction hypothesis, $k \leq \mathrm{size}(\psi)$, $d \leq \mathcal{R}(\psi)$, $\mathrm{size}_{\mathsf{F}}(\psi_i) \leq \mathrm{size}(\psi)$ and $\mathrm{size}_{\mathsf{Y}}(\psi_i) \leq \mathrm{size}(\psi) + k$. We have $\gamma = \mathsf{X}^k \bigvee_{i=1}^{d} \mathsf{F}\psi_i$ (line 5), and the statement follows from $\mathrm{size}_{\mathsf{Y}}(\mathsf{F}\psi_i) = \mathrm{size}_{\mathsf{Y}}(\psi_i) \leq \mathrm{size}(\psi) + k \leq \mathrm{size}(\phi) + k$ and $\mathrm{size}_{\mathsf{F}}(\mathsf{F}\psi_i) = \mathrm{size}_{\mathsf{F}}(\psi_i) + 1 \leq \mathrm{size}(\psi) + 1 = \mathrm{size}(\phi)$.

induction step: $\phi = \phi_1 \otimes \phi_2$, **for** $\otimes \in \{\wedge, \vee\}$. Let $\mathtt{NF}(\phi_i) = \mathsf{X}^{k_i} \bigvee_{j=1}^{d_i} \psi_{i,j}$, for $i \in \{1, 2\}$. By induction hypothesis, $k_i \leq \mathrm{size}(\phi_i)$, $d_i \leq \mathcal{R}(\phi_i)$, $\mathrm{size}_{\mathsf{F}}(\psi_{i,j}) \leq \mathrm{size}(\phi_i)$ and $\mathrm{size}_{\mathsf{Y}}(\psi_{i,j}) \leq \mathrm{size}(\phi_i) + k_i$. Without loss of generality, suppose $k_1 \leq k_2$. Then, given $j \in [1, d_2]$, note that $\gamma_{2,j}$ defined in line 8 is equal to $\psi_{i,j}$. For $j \in [1, d_1]$ we have instead $\gamma_{1,j} := \mathtt{pushY}(\mathsf{Y}^{k_2-k_1} \psi_{1,j})$. By definition of $\mathtt{pushY}$, $\mathrm{size}_{\mathsf{F}}(\gamma_{1,j}) = \mathrm{size}_{\mathsf{F}}(\psi_{1,j})$ and $\mathrm{size}_{\mathsf{Y}}(\gamma_{1,j}) = \mathrm{size}_{\mathsf{Y}}(\psi_{1,j}) + k_2 - k_1 \leq \mathrm{size}(\phi_1) + k_2$. Hence, for $i \in \{1, 2\}$ and $j \in [1, d_i]$, $\mathrm{size}_{\mathsf{F}}(\gamma_{i,j}) \leq \mathrm{size}(\phi_i)$ and $\mathrm{size}_{\mathsf{Y}}(\gamma_{i,j}) \leq \mathrm{size}(\phi_i) + k_2$. We split the proof depending on $\otimes$.

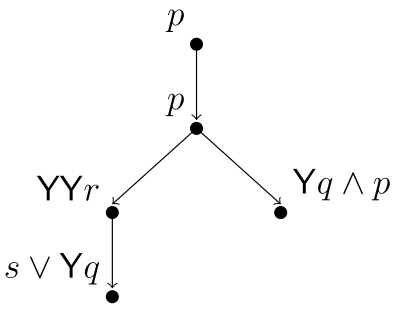


Fig. 1. Dependency tree $T(\phi)$ of $\phi := p \wedge \mathsf{F}\big(p \wedge \mathsf{F}(\mathsf{YY}r \wedge \mathsf{F}(s \vee \mathsf{Y}q)) \wedge \mathsf{F}(\mathsf{Y}q \wedge p)\big)$. For sake of clarity, only the labeling function ν is depicted.

If $\otimes$ is $\vee$, then the output γ is $\mathsf{X}^{k_2}\big(\big(\bigvee_{j=1}^{d_1}\gamma_{1,j}\big) \vee \big(\bigvee_{k=1}^{d_2}\gamma_{2,k}\big)\big)$ (line 9). Note that $k_2 \leq \mathrm{size}(\phi_2) \leq \mathrm{size}(\phi)$, $\mathrm{size}_{\mathsf{F}}(\gamma_{i,j}) \leq \mathrm{size}(\phi)$ and $\mathrm{size}_{\mathsf{Y}}(\gamma_{i,j}) \leq \mathrm{size}(\phi) + k_2$. Lastly, $d_1 + d_2 \leq \mathcal{R}(\phi_1) + \mathcal{R}(\phi_2) = \mathcal{R}(\phi)$.

If $\otimes$ is $\wedge$, then the output is $\mathsf{X}^{k_2}\bigvee_{j=1}^{d_1}\bigvee_{k=1}^{d_2}(\gamma_{1,j} \wedge \gamma_{2,k})$ (line 10). Again, $k_2 \leq \mathrm{size}(\phi_2) \leq \mathrm{size}(\phi)$ and $\mathrm{size}_{\mathsf{Y}}(\gamma_{1,j} \wedge \gamma_{2,k}) \leq \mathrm{size}(\phi) + k_2$, for every $j \in [1, d_1]$ and $k \in [1, d_2]$. Moreover, $\mathrm{size}_{\mathsf{F}}(\gamma_{1,j} \wedge \gamma_{2,k}) \leq \mathrm{size}_{\mathsf{F}}(\gamma_{1,j}) + \mathrm{size}_{\mathsf{F}}(\gamma_{2,k}) + 1 \leq \mathrm{size}(\phi_1) + \mathrm{size}(\phi_2) + 1 = \mathrm{size}(\phi)$. Lastly, $d_1 \cdot d_2 \leq \mathcal{R}(\phi_1) \cdot \mathcal{R}(\phi_2) = \mathcal{R}(\phi)$. □

The $2^{\mathcal{O}(\mathrm{size}(\phi))}$ upper bound on the size of $\mathtt{NF}(\phi)$ follows:

Proposition 2 (Size of the output of Algorithm 1). *For every formula ϕ from* LTL[X, F]*, the formula* $\mathtt{NF}(\phi)$ *is of size* $2^{\mathcal{O}(\mathrm{size}(\phi))}$.

Proof. Let $\mathtt{NF}(\phi) = \mathsf{X}^j \bigvee_{i=1}^{c} \psi_i$.

By Lemma 3, $j \leq \mathrm{size}(\phi)$, $c \leq \mathcal{R}(\phi) \leq 2^{\mathrm{size}(\phi)}$, and $\mathrm{size}_{\mathsf{F}}(\psi_i) \leq \mathrm{size}(\phi)$ and $\mathrm{size}_{\mathsf{Y}}(\psi_i) \leq \mathrm{size}(\phi) + j$, for every $i \in [1, c]$. Then,

$$\begin{aligned}
\mathrm{size}(\mathtt{NF}(\phi)) &= j + \sum\nolimits_{i=1}^{c} \mathrm{size}(\psi_i) \\
&\leq j + c \cdot \max\{\mathrm{size}(\psi_i) \,:\, 1 \leq i \leq c\} \\
&\leq j + c \cdot \max\{\mathrm{size}_{\mathsf{F}}(\psi_i) \cdot \mathrm{size}_{\mathsf{Y}}(\psi_i) \,:\, 1 \leq i \leq c\} \\
&\leq j + c \cdot \mathrm{size}(\phi) \cdot (\mathrm{size}(\phi) + j) \\
&\leq \mathrm{size}(\phi) + 2^{\mathrm{size}(\phi)+1} \cdot \mathrm{size}(\phi)^2 \\
&\in 2^{\mathcal{O}(\mathrm{size}(\phi))}.
\end{aligned}$$

□

Remark 1 (Complexity of Algorithm 1). Proposition 2 implies that Algorithm 1 runs in time $2^{\mathcal{O}(\mathrm{size}(\phi))}$. In fact, the procedure NF is recursively applied on all subformulae of the input formula ϕ; resulting in $\mathrm{size}(\phi)$ calls overall. By Proposition 2, all outputs of recursive calls have size $2^{\mathcal{O}(\mathrm{size}(\phi))}$, and the procedure only applies polynomial-time operations to this output. Since $\mathrm{poly}(2^{\mathcal{O}(n)}) = 2^{\mathcal{O}(n)}$, this implies a $2^{\mathcal{O}(\mathrm{size}(\phi))}$ runtime overall.

5. Translation into F(pLTL[Y, $\widetilde{\mathsf{Y}}$, O])

The second step of the pastification algorithm translates (in polynomial time) $\mathsf{nf}(\mathsf{LTL}[\mathsf{X},\mathsf{F}])$ formulae into formulae from $\mathsf{F}(\mathsf{pLTL}[\mathsf{Y}, \widetilde{\mathsf{Y}}, \mathsf{O}])$. To better explain the ideas behind the translation, we first define the notion of *dependency trees*, which are simple representations of $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$ formulae whose role is to highlight the interplay between the various occurrences of the modality F. Next, we show how to perform the pastification of $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$ formulae starting from these trees, and then extend it to arbitrary $\mathsf{nf}(\mathsf{LTL}[\mathsf{X},\mathsf{F}])$ formulae.

5.1. Dependency trees

Consider a formula $\phi \in \mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$. Recall that such a formula is of the form $\psi \wedge \mathsf{F}(\phi_1) \wedge \cdots \wedge \mathsf{F}(\phi_n)$, where ψ belongs to $\mathsf{pLTL}[\mathsf{Y}]$, and each ϕ_i is from $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$ (and $n \in \mathbb{N}$). Then, the formula ϕ can be represented in a straightforward way as a labeled tree: in such a tree, the root is labeled with the formula ψ, and has n children, where the subtree rooted at the ith child (inductively) represents the formula ϕ_i. Here is the formal definition:

Definition 5 (Dependency tree). The *dependency tree* $T(\phi)$ of a formula $\phi \in \mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$ is a tuple (V, E, r, μ, ν), where (V, E, r) is a rooted tree having *nodes* V, *edges* E, and *root* $r \in V$, and $\mu : V \to \mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$ and $\nu : V \to \mathsf{pLTL}[\mathsf{Y}]$ are *labeling functions* satisfying the following two properties: *(i)* $\mu(r) = \phi$ and *(ii)* for every $v \in V$, $\mu(v) = \psi \wedge \mathsf{F}(\phi_1) \wedge \cdots \wedge \mathsf{F}(\phi_n)$ (where $\psi \in \mathsf{pLTL}[\mathsf{Y}]$, $n \in \mathbb{N}$, and every ϕ_i belongs to $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$) if and only if $\nu(v) = \psi$ and the set of children of v is $E(v) = \{v_1, \ldots, v_n\}$, where $\mu(v_i) = \phi_i$ for every $i \in [1, n]$.

Fig. 1 shows an example of a dependency tree. Note that $T(\phi)$ (abbreviated as T when ϕ is clear from the context) has size linear in $\mathrm{size}(\phi)$. We denote with $\Pi(T)$ the set of *paths from the root to a leaf of* T, that is, sequences $\pi = \langle \pi_1, \ldots, \pi_n \rangle$ of nodes in V such that *(i)* $\pi_1 = r$, *(ii)* $\pi_{i+1} \in E(\pi_i)$ for every $i \in [1, n-1]$, and *(iii)* $E(\pi_n) = \emptyset$.

5.2. From dependency trees to $\mathsf{F}(\mathsf{pLTL}[\mathsf{Y}, \widetilde{\mathsf{Y}}, \mathsf{O}])$

Consider a formula $\mathsf{X}^k \bigvee_{i=1}^{c} \phi_i$ from $\mathsf{nf}(\mathsf{LTL}[\mathsf{X}, \mathsf{F}])$, where $k, c \in \mathbb{N}$, and each ϕ_i belongs to $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$. Given $i \in [1, c]$, our goal is now to use the dependency tree $T(\phi_i)$ to construct a pure past formula ψ_i^k that is equivalent to ϕ_i with respect to the kth time point of any trace (this is the case relevant to us, due to the k modalities X preceding $\bigvee_{i=1}^{c} \phi_i$). In a nutshell, the idea is to consider separately each path in $\Pi(T(\phi_i))$, and "rewrite it upside-down" (*i.e.*, going from the leaf to the root), by means of a formula that uses only the past modalities *once* (O), *yesterday* (Y), and *weak yesterday* ($\widetilde{\mathsf{Y}}$). In particular, the weak yesterday modality is used to characterize properties of the root of $T(\phi_i)$, as the formula $\mathsf{at}_k := \mathsf{Z}^{k+1}\bot \wedge \mathsf{Y}^k\top$ is only true at the kth time point of a trace. Here is the formal translation:

Definition 6 (Path formulae). Let ϕ be a formula of $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$, and $k \in \mathbb{N}$. Furthermore, let $\pi = \langle \pi_1, \dots, \pi_n \rangle \in \Pi(T)$. Let $\langle\langle \pi \rangle\rangle_k^i$ be the formula inductively defined, for every $i \in [1, n]$, as:

$$\langle\langle \pi \rangle\rangle_k^i = \begin{cases} \mathsf{O}(\nu(\pi_1) \wedge \mathsf{at}_k) & \text{if } i = 1, \\ \mathsf{O}(\nu(\pi_i) \wedge \langle\langle \pi \rangle\rangle_k^{i-1}) & \text{otherwise.} \end{cases}$$

We define the *path formula* of π as $\langle\langle \pi \rangle\rangle_k := \langle\langle \pi \rangle\rangle_k^n$.

Given ϕ in $\mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$ and $k \in \mathbb{N}$, by relying on the above path formulae, we define $\mathsf{past}(\phi, k) := \mathsf{F}(\bigwedge_{\pi \in \Pi(T)} \langle\langle \pi \rangle\rangle_k)$. Observe that this is a formula from $\mathsf{F}(\mathsf{pLTL}[\mathsf{Y}, \widetilde{\mathsf{Y}}, \mathsf{O}])$, which we will show to be equivalent to ϕ when interpreted at the kth time point of a trace.

Example 1. Consider the formula ϕ from Fig. 1, and let $k \in \mathbb{N}$. Then, $\mathsf{past}(\phi, k) = \mathsf{F}(\langle\langle \pi_1 \rangle\rangle_k \wedge \langle\langle \pi_2 \rangle\rangle_k)$, where:

$$\langle\langle \pi_1 \rangle\rangle_k := \mathsf{O}\big((\mathsf{s} \vee \mathsf{Yq}) \wedge \mathsf{O}(\mathsf{YYr} \wedge \mathsf{O}(\mathsf{p} \wedge \mathsf{O}(\mathsf{p} \wedge \mathsf{at}_k)))\big),$$
$$\langle\langle \pi_2 \rangle\rangle_k := \mathsf{O}\big((\mathsf{p} \wedge \mathsf{Yq}) \wedge \mathsf{O}(\mathsf{p} \wedge \mathsf{O}(\mathsf{p} \wedge \mathsf{at}_k)))\big).$$

What may seem surprising in the example above is that, comparatively to the dependency tree of Fig. 1, the formula $\mathsf{past}(\phi, k)$ seems to have lost all information about the nodes that π_1 and π_2 share (that is, the first two nodes in these paths). The forthcoming Lemma 4 shows that this is unproblematic: the formula $\mathsf{past}(\phi, k)$ is equivalent to $\mathsf{X}^k\phi$. Observe that from this equivalence, we can at last complete the translation from $\mathsf{X}^k \bigvee_{i=1}^{c} \phi_i$ of $\mathsf{nf}(\mathsf{LTL}[\mathsf{X}, \mathsf{F}])$ to $\mathsf{F}(\mathsf{pLTL}[\mathsf{Y}, \widetilde{\mathsf{Y}}, \mathsf{O}])$. It suffices for **the algorithm to output**:

$$\mathsf{F} \bigvee_{i=1}^{c} \psi_i,$$

where each ψ_i is such that $\mathsf{past}(\phi_i, k) = \mathsf{F}\psi_i$. Indeed:

$$\begin{aligned} \mathsf{X}^k \bigvee_{i=1}^{c} \phi_i \;&\equiv\; \bigvee_{i=1}^{c} \mathsf{X}^k \phi_i && \text{(see Lemma 1 and } \mathbf{R}_1\text{)} \\ &\equiv\; \bigvee_{i=1}^{c} \mathsf{past}(\phi_i, k) && \text{(Lemma 4, below)} \\ &\equiv\; \mathsf{F} \bigvee_{i=1}^{c} \psi_i && \text{(see Lemma 1 and } \mathbf{R}_5\text{)}. \end{aligned}$$

Lemma 4. *Let $\phi \in \mathsf{conj}(\mathsf{F}, \mathsf{pLTL}[\mathsf{Y}])$, and $k \in \mathbb{N}$. Then, $\mathsf{X}^k\phi \equiv \mathsf{past}(\phi, k)$.*

Proof. As a preliminary step, let us consider the following rewriting rule:

$$\mathbf{R}_7: \quad \mathsf{F}\big(\psi_1 \wedge \mathsf{F}(\psi_2) \wedge \mathsf{F}(\psi_3)\big) \;\mapsto\; \mathsf{F}\big(\psi_1 \wedge \mathsf{F}(\psi_2)\big) \wedge \mathsf{F}\big(\psi_1 \wedge \mathsf{F}(\psi_3)\big).$$

Let us show that the two sides of this rule are strongly equivalent. The left-to-right direction is straightforward to prove, so let us focus on the right-to-left direction. Consider $\sigma \in (2^{AP})^\omega$ and $i \geq 0$, and assume that

$$\sigma, i \vDash \mathsf{F}\big(\psi_1 \wedge \mathsf{F}(\psi_2)\big) \wedge \mathsf{F}\big(\psi_1 \wedge \mathsf{F}(\psi_3)\big).$$

Then, there are positions j_1, j_2, k_1, k_2 such that $i \leq j_1 \leq j_2$, $i \leq k_1 \leq k_2$, and $\sigma, j_1 \vDash \psi_1$, $\sigma, k_1 \vDash \psi_1$, $\sigma, j_2 \vDash \psi_2$ and $\sigma, k_2 \vDash \psi_3$. Let us assume $j_1 \leq k_1$ (the case $k_1 < j_1$ is analogous). From $\sigma, j_2 \vDash \psi_2$ and $\sigma, k_2 \vDash \psi_3$ we conclude that $\sigma, j_1 \vDash \mathsf{F}(\psi_2)$ and $\sigma, j_1 \vDash \mathsf{F}(\psi_2)$. This implies $\sigma, i \vDash \mathsf{F}\big(\psi_1 \wedge \mathsf{F}(\psi_2) \wedge \mathsf{F}(\psi_3)\big)$; that is, the two sides of $\mathbf{R}_7$ are strongly equivalent. Note that, more generally, applying $\mathbf{R}_7$ to a formula $\mathsf{F}\big(\psi_1 \wedge \mathsf{F}(\psi_2) \wedge \ldots \wedge \mathsf{F}(\psi_n)\big)$ produces the formula $\mathsf{F}\big(\psi_1 \wedge \mathsf{F}(\psi_2)\big) \wedge \ldots \wedge \mathsf{F}\big(\psi_1 \wedge \mathsf{F}(\psi_n)\big)$.

In order to prove that $\mathsf{X}^k\phi \equiv \mathsf{past}(\phi, k)$, let us first apply the rule $\mathbf{R}_7$ to ϕ, in a bottom-up fashion. Fig. 2 shows the effect of applying this rule (once) to the dependency tree of a formula. In general, by definition of the dependency tree T of ϕ, the result of this rewriting step is the formula

$$\gamma := \bigwedge_{\substack{\pi \in \Pi(T) \text{ s.t.} \\ \pi = \langle \pi_1, \dots, \pi_n \rangle}} \big(\nu(\pi_1) \wedge \mathsf{F}\big(\nu(\pi_2) \wedge \mathsf{F}(\dots \mathsf{F}(\nu(\pi_n)) \dots)\big)\big),$$

and we have $\mathsf{X}^k\phi \equiv \mathsf{X}^k\gamma$. Recall moreover that $\mathsf{past}(\phi, k) = \mathsf{F}(\bigwedge_{\pi \in \Pi(T)} \langle\langle \pi \rangle\rangle_k)$, where $\langle\langle \pi \rangle\rangle_k$ is a formula of the form $\mathsf{O}(\psi)$. The fact that $\mathsf{past}(\phi, k)$ is equivalent to $\bigwedge_{\pi \in \Pi(T)} \mathsf{F}(\langle\langle \pi \rangle\rangle_k)$ follows from the equivalence $\mathsf{F}\big(\mathsf{O}(\psi_1) \wedge \mathsf{O}(\psi_2)\big) \equiv \mathsf{F}\big(\mathsf{O}(\psi_1)\big) \wedge \mathsf{F}\big(\mathsf{O}(\psi_2)\big)$, whose proof is similar to the one given for rule $\mathbf{R}_7$. Then, the lemma follows as soon as we show that, for every path $\pi = \langle \pi_1, \dots, \pi_n \rangle$ of $\Pi(T)$, we have

$$\mathsf{F}(\langle\langle \pi \rangle\rangle_k) \;\equiv\; \mathsf{X}^k\big(\nu(\pi_1) \wedge \mathsf{F}\big(\nu(\pi_2) \wedge \mathsf{F}(\dots \mathsf{F}(\nu(\pi_n)) \dots)\big)\big).$$

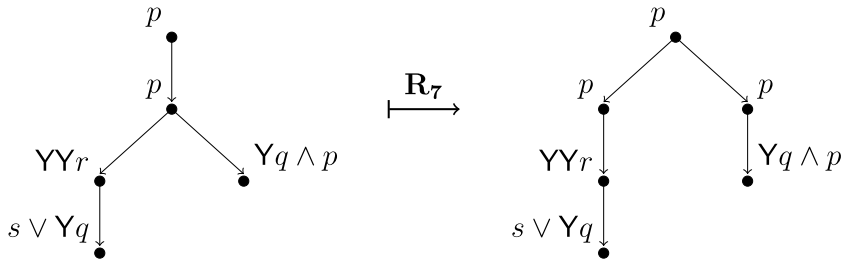


Fig. 2. Example of application of rule $\mathbf{R}_7$ to the dependency tree of the formula $\phi := p \wedge \mathsf{F}\big(p \wedge \mathsf{F}(\mathsf{YY}r \wedge \mathsf{F}(s \vee \mathsf{Y}q)) \wedge \mathsf{F}(\mathsf{Y}q \wedge p)\big)$. The rightmost dependency tree corresponds to the formula $p \wedge \mathsf{F}\big(p \wedge \mathsf{F}(\mathsf{YY}r \wedge \mathsf{F}(s \vee \mathsf{Y}q))\big) \wedge \mathsf{F}\big(p \wedge \mathsf{F}(\mathsf{Y}q \wedge p)\big)$.

Let us show the right-to-left direction of this equivalence. The proof of the other direction is analogous. Consider a trace $\sigma \in (2^{\mathcal{AP}})^\omega$, and assume $\sigma, 0 \vDash \mathsf{X}^k\big(\nu(\pi_1) \wedge \mathsf{F}\big(\nu(\pi_2) \wedge \mathsf{F}(\dots \mathsf{F}(\nu(\pi_n)) \dots)\big)\big)$. By definition, there are $j_1, j_2, \dots, j_n \in \mathbb{N}$ such that $k = j_1 \leq j_2 \leq \cdots \leq j_n$ and $\sigma, j_i \vDash \nu(\pi_i)$, for every $i \in [1, n]$. Directly from Definition 6, for every $i \in [1, n]$, we have $\sigma, j_i \vDash \langle\!\langle \pi \rangle\!\rangle_k^i$; and in particular $\sigma, j_n \vDash \langle\!\langle \pi \rangle\!\rangle_k$. Hence, $\sigma, 0 \vDash \mathsf{F}(\langle\!\langle \pi \rangle\!\rangle_k)$. □

5.3. Analysis on the size of the output of the algorithm

We now prove that our procedure outputs an exponential-size formula, thus showing the main result of the paper:

Theorem 1. *There exists a pastification procedure from* $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ *into the logic* $\mathsf{F}(\mathsf{pLTL}[\mathsf{Y},\widetilde{\mathsf{Y}},\mathsf{O}])$ *of* 1*-exponential size.*

Proof. Let ϕ be a formula from $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$. From Propositions 1 and 2, there is a formula $\mathsf{X}^k \bigvee_{i=1}^c \phi_i$ in $\mathsf{nf}(\mathsf{LTL}[\mathsf{X},\mathsf{F}])$ that is equivalent to ϕ and has size $2^{\mathcal{O}(\mathrm{size}(\phi))}$. This formula is computed by Algorithm 1. The output of our pastification procedure is then the equivalent (by Lemma 4) formula $\mathsf{F}\bigvee_{i=1}^c \psi_i$, where $\mathsf{past}(\phi_i, k) = \mathsf{F}\psi_i$, for every $i \in [1, c]$. Therefore, in order to prove the theorem it suffices to show that, for a given formula γ in $\mathsf{conj}(\mathsf{F},\mathsf{pLTL}[\mathsf{Y}])$, and $k \in \mathbb{N}$, the formula $\mathsf{past}(\gamma, k)$ has size polynomial in $\mathrm{size}(\gamma)$ and k. In fact, we show that this formula has size $\mathcal{O}(n \cdot (k+n))$, where $n := \mathrm{size}(\gamma)$. Let $T := T(\gamma)$. By definition of $\mathsf{past}(\gamma, k)$, we have

$$\mathrm{size}(\mathsf{past}(\gamma, k)) = \sum\nolimits_{\pi \in \Pi(T)} \mathrm{size}(\langle\!\langle \pi \rangle\!\rangle_k),$$

where, for a generic path π of length $j \leq n$, we set

$$\mathrm{size}(\langle\!\langle (\pi_1, \dots, \pi_j) \rangle\!\rangle_k) = \mathrm{size}(\langle\!\langle (\pi_1, \dots, \pi_j) \rangle\!\rangle_k^j)$$

and

$$\mathrm{size}(\langle\!\langle (\pi_1, \dots, \pi_j) \rangle\!\rangle_k^1) = \mathrm{size}(\mathsf{at}_k) + \mathrm{size}(\nu(\pi_1)) + 2,$$
$$\mathrm{size}(\langle\!\langle (\pi_1, \dots, \pi_j) \rangle\!\rangle_k^i) = \mathrm{size}(\langle\!\langle (\pi_1, \dots, \pi_j) \rangle\!\rangle_k^{i-1}) + \mathrm{size}(\nu(\pi_i)) + 2,$$

where $i \in [2, j]$. A simple induction then shows that $\mathrm{size}(\langle\!\langle (\pi_1, \dots, \pi_j) \rangle\!\rangle_k) = \mathrm{size}(\mathsf{at}_k) + 2 \cdot j + \sum_{i=1}^j \mathrm{size}(\nu(\pi_i))$. Note that both j and $\sum_{i=1}^j \mathrm{size}(\nu(\pi_i))$ are bounded by n, and that $\mathrm{size}(\mathsf{at}_k)$ is in $\mathcal{O}(k)$, and therefore $\mathrm{size}(\mathsf{past}(\gamma, k))$ is in $\mathcal{O}(|\Pi(T)| \cdot (k+n))$. Lastly, $|\Pi(T)| \leq n$, as each path in $\Pi(T)$ is identified by a distinct leaf of the dependency tree T, and every such leaf is labeled, through the map ν, by distinct occurrences of subformulae of γ. □

Remark 2 (Complexity of the procedure)**.** Following Remark 1, let us also point out that our algorithm does not only produce an exponential-size formula, but also runs in $2^{\mathcal{O}(n)}$, where n is the size of the input $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ formula.

5.4. A pastification procedure for $\mathsf{LTL}[\mathsf{X},\mathsf{G}]$

From the duality between the *eventually* and *always* modalities, that is, $\neg\mathsf{G}\phi \equiv \mathsf{F}\neg\phi$, it is simple to obtain a 1-exponential pastification algorithm for *safety* fragment $\mathsf{LTL}[\mathsf{X},\mathsf{G}]$:

Corollary 1. *There exists a pastification procedure from* $\mathsf{LTL}[\mathsf{X},\mathsf{G}]$ *into the logic* $\mathsf{G}(\mathsf{pLTL}[\mathsf{Y},\widetilde{\mathsf{Y}},\mathsf{H}])$ *of* 1*-exponential size.*

Proof. Let ϕ belong to $\mathsf{LTL}[\mathsf{X},\mathsf{G}]$. Here is the pastification procedure:

1. Bring $\neg\phi$ in negation normal form, and let γ be the resulting formula. Since $\neg\mathsf{X}\psi \equiv \mathsf{X}\neg\psi$ and $\neg\mathsf{G}\psi \equiv \mathsf{F}\neg\psi$ this formula is in $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$.
2. Apply the pastification procedure from Theorem 1 on γ. The resulting formula is of the form $\mathsf{F}\chi$, with χ in $\mathsf{pLTL}[Y,\widetilde{\mathsf{Y}},O]$, and $\phi \equiv \neg\mathsf{F}\chi$.
3. Bring $\neg\chi$ in negation normal form, and let χ' be the resulting formula. Since $\neg\mathsf{O}\psi \equiv \mathsf{H}\neg\psi$, $\neg\mathsf{Y}\psi \equiv \widetilde{\mathsf{Y}}\neg\psi$ (and $\neg\widetilde{\mathsf{Y}}\psi \equiv \mathsf{Y}\neg\psi$), this formula belongs to $\mathsf{pLTL}[Y,\widetilde{\mathsf{Y}},H]$.
4. Output $\mathsf{G}\chi'$. From $\neg\mathsf{F}\chi \equiv \mathsf{G}\neg\chi$ and $\neg\chi \equiv \chi'$,we have $\phi \equiv \mathsf{G}\chi'$. □

6. Discussion on the optimality of the algorithm

Let us now discuss the optimality of the proposed algorithm for the pastification of LTL[X, F] formulae into F(pLTL[Y, $\widetilde{\text{Y}}$, O]) formulae. We begin by providing the definition of succinctness.

Definition 7. Given two temporal logics $\mathbb{L}$ and $\mathbb{L}'$, we say that $\mathbb{L}$ *can be exponentially more succinct than* $\mathbb{L}'$ whenever there is an (infinite) family of languages $(\mathcal{L}_n)_{n\in\mathbb{N}}$ such that, for every $n \in \mathbb{N}$,

- there is a formula in $\mathbb{L}$ of size in $\mathcal{O}(n)$ and with language $\mathcal{L}_n \subseteq (2^{\mathcal{AP}})^\omega$;
- all formulae in $\mathbb{L}'$ with language $\mathcal{L}_n$ have size in $2^{\Omega(n)}$.

Clearly, our algorithm is optimal as soon as one shows that LTL[X, F] can be exponentially more succinct than F(pLTL[Y, $\widetilde{\text{Y}}$, O]). The recent paper [20] shows a stronger result: the logic obtained from LTL[X, F] by removing modality X is already exponentially more succinct than the logic extending F(pLTL[Y, $\widetilde{\text{Y}}$, O]) by adding the modality H.

Proposition 3 ([20]). *The logic* LTL[F] *is exponentially more succinct than the logic* F(LTL[Y, $\widetilde{\text{Y}}$, O, H]). *The vice versa holds as well.*

Remark 3. The definition of "being exponentially more succinct" in [20] is slightly weaker than the one given above. In particular, $\mathbb{L}$ is only required to have a formula of size *polynomial* in n (instead of *linear*) characterizing $\mathcal{L}_n$. However, the formula characterizing $\mathcal{L}_n$ used in [20] to show Proposition 3 *does* have linear size. In our case, we need this stronger result (and definition) in order to conclude that our algorithm is optimal.

The difficult part in establishing Proposition 3 is showing that, for the family of languages considered in [20], no formula of F(LTL[Y, $\widetilde{\text{Y}}$, O, H]) of size less than 2^n can express $\mathcal{L}_n$. In [20] this is done by defining a *combinatorial proof system*: a set of rules that, given two sets of traces A and B, can be used to establish whether there is a formula ϕ of F(LTL[Y, $\widetilde{\text{Y}}$, O, H]) *separating* A from B, that is, ϕ is satisfied by all traces in A and violated by all traces in B. Crucially, a proof obtained by applying k rules of the proof system corresponds to the existence of one such separating formula ϕ of size k. It suffices then to define, parametrically in n, two sets of traces A_n and B_n that are separated by the formula characterizing $\mathcal{L}_n$, to then show that no "proof of separation" of A_n and B_n in the proof system can have size less than 2^n.

7. Implementation

We implemented our pastification procedure in a tool called Pastello,[5] which uses the APIs of the temporal reasoning tool BLACK [21,22] for manipulating the formulae (and testing the equivalence between input and output formulae as a way of debugging our code). The implementation has only about 500 lines of C++ code.

We evaluated Pastello on the following set of benchmarks. For each $i \in \{5, \dots, 50\}$, we randomly generated 20 formulae of LTL[X, F] of size $i \pm 10$. The formulae are generated with BLACK's random formulae generator also used in the experimental evaluations described in [21]. The abstract syntax tree of the formulae is generated, starting from a single node being the root, by choosing from a uniform distribution whether the current node has to be a proposition letter or a temporal modality. In the former case, the proposition letter gets chosen from a uniform distribution among the alphabet's proposition letters, while, in the second case, the temporal modality is chosen again from a uniform distribution.

For each input formula, we measured four different metrics with respects to its corresponding output formula:

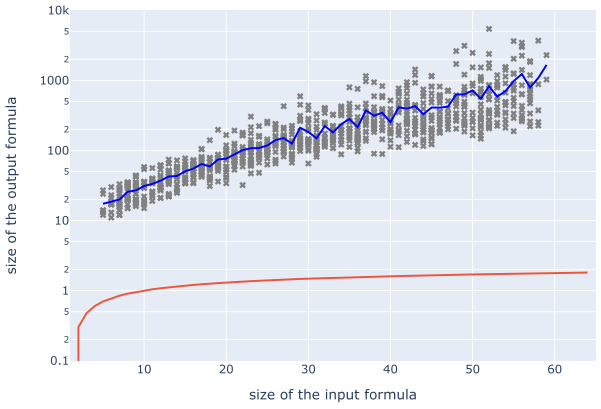
(i) *size*, see Fig. 3(a);
(ii) *number of occurrences of temporal modalities*, see Fig. 3(b);
(iii) *number of occurrences of Boolean operators*, see Fig. 3(c);
(iv) *depth*, defined as the maximum number of nested temporal modalities, see Fig. 3(d).

Before discussing the results of the experimental evaluation, we describe how the plots in Fig. 3 are organized. The x-axis refers to the input formula, whereas the y-axis refers to the output formula. In all the plots, the x-axis is in linear scale, while the y-axis is in logarithmic scale, except for Fig. 3(d) in which both axes are in linear scale. The orange lines represent the diagonals of the plots. The blue lines, instead, are piecewise functions that interpolate the mean value for the output formulae corresponding to an input formula of a given size. The more the blue line converges (resp., diverges) from the orange line, the closer to a polynomial (resp. exponential) runtime the algorithm has.

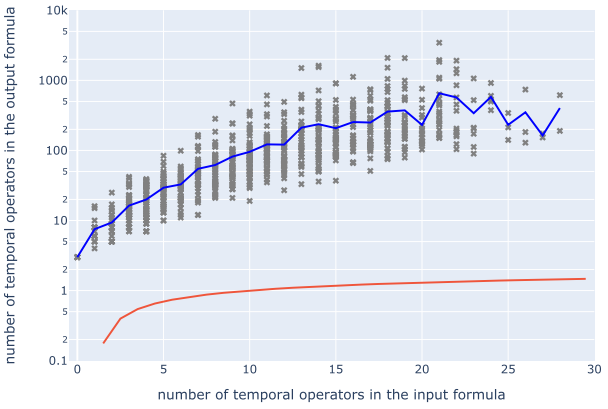
Consider Fig. 3(a), which plots the size of the output formula (on the y-axis, in logarithmic scale) with respect to the size of the input formula (on the x-axis, in linear scale). The exponential growth is quite evident from the trend of the blue line, which diverges from the orange line (the diagonal). Hence, at least on random inputs, the exponential blowup of our pastification algorithm from LTL[X, F] into F(pLTL[Y, $\widetilde{\text{Y}}$, O]) is not only a matter of worst-case scenario; it occurs in the majority of the cases.

We refined the previous analysis of the size of the output formula by plotting the number of *temporal modalities* (Fig. 2(b)) and *Boolean operators* (Fig. 3(c)) in the output formulae with respect to the input ones. Both plots show the same exponential growth and they indicate that the exponential trend in Fig. 3(a) is due to the growth of the number of temporal modalities as much as to the growth of the number of Boolean operators.

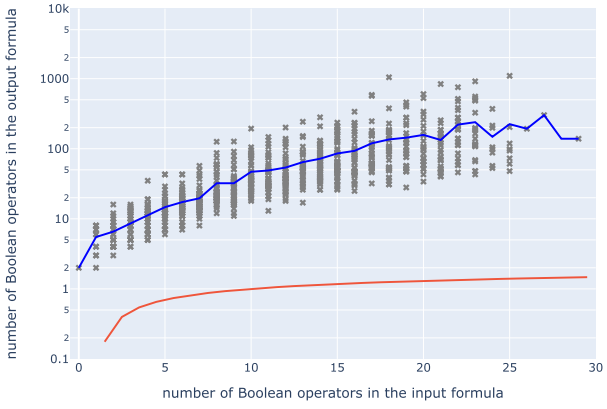
[5] http://users.dimi.uniud.it/~luca.geatti/tools/pastello.html.



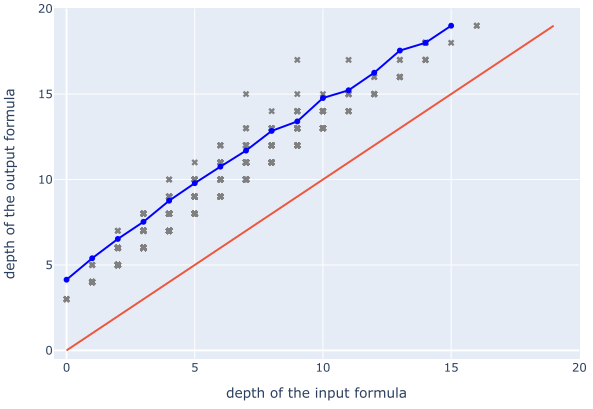
(a) Size of the formulae.



(b) Occurrences of temporal operators.



(c) Occurrences of Boolean operators.



(d) Depth of the formulae.

Fig. 3. Results of the evaluation of Pastello on randomly generated formulae.

Finally, the only one of the four metric that does not grow exponentially is the *depth*. In fact, the depth of the output formulae, plotted in Fig. 3(d), follows a linear trend with respect to the input ones. This comes with no surprise, since:

(i) during the transformation into normal form, all rewriting rules increase the depth of the formula only by a constant factor,
(ii) for any formula ϕ in normal form, the *height* of its dependency tree T is exactly the depth of ϕ, and
(iii) the *depth* of the formula built starting from a dependency tree T is exactly the *height* of T.

8. Conclusions and open problems

We designed (and implemented) a pastification algorithm for LTL[X, F] formulae that produces F(pLTL) formulae of size singly exponential with respect to the size of the input formula. Following a matching lower bound from [20], the algorithm turns out to be optimal.

The main open problem related to our work is the complexity of pastification procedures for the logic coSafetyLTL, that is, the extension of LTL[X, F] featuring the modality U. The best available pastification procedure for this logic is the one based on the Krohn-Rhodes cascaded decomposition [5], which has been sketched in Section 2. whose output may be of triply exponential size. This procedure passes through several intermediate steps that only ultimately build the cascaded decomposition. In principle, it could be the case that this procedure can be improved studying efficient algorithms for implementing closure properties of cascaded decompositions of reset automata [33] under temporal operations. These algorithms take as input(s) cascaded decompositions, and *directly* construct cascaded decompositions for all modalities and Boolean connectives in the grammar of coSafetyLTL. For instance, one such algorithm takes as input the cascaded decomposition representing a formula ϕ of coSafetyLTL, and returns a cascaded decomposition for Fϕ. From a complete suite of these algorithms, one can construct the cascaded decomposition of the whole input formula without passing through intermediate automata representations; hopefully obtaining a substantial improvement in the runtime of the pastification procedure. Preliminary results in this direction recently appeared in [37].

The approach of directly investigating the closure properties of cascaded decompositions appears more promising than attempting to extend the technique proposed in this paper to the entirety of coSafetyLTL. To explain a current limitation of our technique, consider the formula

$$\mathsf{F}\big((\psi_1 \mathsf{U} \gamma_1) \wedge (\psi_2 \mathsf{U} \gamma_2)\big).$$

When $\psi_1 = \psi_2 = \top$, this formula is equivalent to $\mathsf{F}(\mathsf{F}\gamma_1 \wedge \mathsf{F}\gamma_2)$ which, by rule $\mathbf{R}_7$, is in turn equivalent to $(\mathsf{F}\gamma_1) \wedge (\mathsf{F}\gamma_2)$, essentially stating that the two occurrences of γ_1 and γ_2 are independent. This equivalence has a fundamental role in the proof of Lemma 4. Replacing $\mathsf{F}\gamma_i$ with $\psi_i \mathsf{U} \gamma_i$ results in the formula $\mathsf{F}(\psi_1 \mathsf{U} \gamma_1) \wedge \mathsf{F}(\psi_2 \mathsf{U} \gamma_2)$, which is however not equivalent to the initial formula. Intuitively, this is because the original formula was constraining a single point in the trace to satisfy *both* occurrences of the until modality U, whereas the latter formula allows for distinct points in the trace to satisfy each one occurrence of this modality. Because of this, it is unclear how to define a suitable notion of dependency tree.

The above formula seem to be an interesting starting point also for obtaining stronger lower bounds for the pastification of coSafetyLTL. In particular, we see no way to produce a pastification for a formula of the form $\mathsf{F}(\bigwedge_{i=1}^{n}(p_i \mathsf{U} q_i))$ that does not enumerate essentially all possible orders over $q_1, \ldots, q_n$. Because of this, it is natural to conjecture that there is no pastification algorithm for coSafetyLTL producing formulae of size less than $\mathcal{O}(n!)$, where n is the size of the input formula; in other words, we conjecture a singly exponential pastification procedure for coSafetyLTL, as the one proposed in this paper, to be unlikely to exist.

CRediT authorship contribution statement

Alessandro Artale: Writing – review & editing, Writing – original draft, Validation, Supervision, Formal analysis; **Luca Geatti:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Project administration, Methodology, Investigation, Formal analysis, Conceptualization; **Nicola Gigante:** Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Conceptualization; **Alessio Mansutti:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Formal analysis, Conceptualization; **Andrea Mazzullo:** Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Angelo Montanari:** Writing – review & editing, Writing – original draft, Validation, Resources, Methodology, Investigation, Formal analysis, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Luca Geatti and Angelo Montanari acknowledge the support from the Interconnected Nord-Est Innovation Ecosystem (iNEST), which received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) - MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.5 - D.D. 1058 23/06/2022, ECS00000043).

Alessio Mansutti acknowledge the support from the César Nombela grant 2023-T1/COM-29001, funded by the Madrid Regional Government, and the support from the grant PID2022-138072OB-I00, funded by MCIN/AEI/10.13039/501100011033 (FEDER, EU).

References

[1] A. Artale, L. Geatti, N. Gigante, A. Mazzullo, A. Montanari, A singly exponential transformation of LTL[X,F] into pure past LTL, in: KR, 2023. https://doi.org/10.24963/KR.2023/7
[2] A. Pnueli, The temporal logic of programs, in: SFCS, 1977, pp. 46–57. https://doi.org/10.1109/SFCS.1977.32
[3] O. Lichtenstein, A. Pnueli, L. Zuck, The glory of the past, in: Workshop on Logic of Programs, 1985. https://doi.org/10.1007/3-540-15648-8_16
[4] G. De Giacomo, M.Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: IJCAI, 2013. isbn: 9781577356332.
[5] G. De Giacomo, A. Di Stasio, F. Fuggitti, S. Rubin, Pure-past linear temporal and dynamic logic on finite traces, in: IJCAI, 2021. https://doi.org/10.24963/ijcai.2020/690
[6] A. Artale, L. Geatti, N. Gigante, A. Mazzullo, A. Montanari, LTL over finite words can be exponentially more succinct than pure-past LTL, and vice versa, in: TIME, 2023. https://doi.org/10.4230/LIPICS.TIME.2023.2
[7] A. Artale, L. Geatti, N. Gigante, A. Mazzullo, A. Montanari, Succinctness issues for LTL and safety and cosafety fragments of LTL, Inf. Comput. 302 (2025) 105262. https://doi.org/10.1016/J.IC.2024.105262
[8] E.Y. Chang, Z. Manna, A. Pnueli, Characterization of temporal property classes, in: ICALP, 1992. https://doi.org/10.1007/3-540-55719-9_97
[9] A. Cimatti, L. Geatti, N. Gigante, A. Montanari, S. Tonetta, Extended bounded response LTL: a new safety fragment for efficient reactive synthesis, Form. Methods Syst. Des. (2021). https://doi.org/10.1007/s10703-021-00383-3
[10] G. De Giacomo, R. De Masellis, F.M. Maggi, M. Montali, Monitoring constraints and metaconstraints with temporal logics on finite traces, ACM Trans. Softw. Eng. Methodol. (2022). https://doi.org/10.1145/3506799
[11] G. De Giacomo, M. Favorito, F. Fuggitti, Planning for temporally extended goals in pure-past linear temporal logic: a polynomial reduction to standard planning, in: ICAPS, 2022. https://doi.org/10.1609/icaps.v33i1.27179

[12] A. Pnueli, R. Rosner, On the synthesis of a reactive module, in: Proceedings of POPL'89, ACM Press, 1989, pp. 179–190. https://doi.org/10.1145/75277.75293
[13] A. Artale, L. Geatti, N. Gigante, A. Mazzullo, A. Montanari, Complexity of safety and cosafety fragments of linear temporal logic, in: AAAI, 2023. https://doi.org/10.1609/aaai.v37i5.25768
[14] R. Rosner, Modular Synthesis of Reactive Systems, Ph.D. thesis, Weizmann Institute of Science, 1992.
[15] G. De Giacomo, M.Y. Vardi, Synthesis for LTL and LDL on finite traces, in: IJCAI, 2015. isbn: 9781577357384.
[16] L. Geatti, M. Montali, A. Rivkin, Foundations of reactive synthesis for declarative process specifications, in: AAAI, 2024. https://doi.org/10.1609/AAAI.V38I16.29690
[17] L. Geatti, M. Montali, A. Rivkin, Foundations of collaborative sfDECLARE, in: BPM, 2023. https://doi.org/10.1007/978-3-031-41623-1_4
[18] O. Maler, D. Nickovic, A. Pnueli, Real time temporal logic: past, present, future, in: FORMATS, 2005. https://doi.org/10.1007/3-540-45739-9_14
[19] O. Maler, D. Nickovic, A. Pnueli, On synthesizing controllers from bounded-response properties, in: CAV, 2007. https://doi.org/10.1023/A:1008734703554
[20] L. Geatti, A. Mansutti, A. Montanari, Succinctness of cosafety fragments of LTL via combinatorial proof systems, in: FOSSACS, 2024. https://doi.org/10.1007/978-3-031-57231-9_5
[21] L. Geatti, N. Gigante, A. Montanari, G. Venturato, SAT meets tableaux for linear temporal logic satisfiability, J. Autom. Reason. (2024). https://doi.org/10.1007/S10817-023-09691-1
[22] L. Geatti, N. Gigante, A. Montanari, BLACK: A fast, flexible and reliable LTL satisfiability checker, in: OVERLAY, 2021. Available athttp://ceur-ws.org/Vol-2987/paper2.pdf accessed May 13, 2026.
[23] A.P. Sistla, E.M. Clarke, The complexity of propositional linear temporal logics, J. ACM 32 (3) (1985) 733–749. https://doi.org/10.1145/3828.3837
[24] N. Markey, Past is for free: on the complexity of verifying linear temporal properties with past, Acta Inform. 40 (6–7) (2004) 431–458. https://doi.org/10.1007/S00236-003-0136-5
[25] D.M. Gabbay, A. Pnueli, S. Shelah, J. Stavi, On the temporal analysis of fairness, in: P.W. Abrahams, R.J. Lipton, S.R. Bourne (Eds.), Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages, Las Vegas, Nevada, USA, January 1980, ACM Press, 1980, pp. 163–173.
[26] F. Laroussinie, N. Markey, P. Schnoebelen, Temporal logic with forgettable past, in: Proceedings 17th Annual IEEE Symposium on Logic in Computer Science, IEEE, 2002, pp. 383–392.
[27] D.M. Gabbay, The declarative past and imperative future: executable temporal logic for interactive systems, in: B. Banieqbal, H. Barringer, A. Pnueli (Eds.), Temporal Logic in Specification, Altrincham, UK, April 8–10, 1987, Proceedings, Vol. 398 of *Lecture Notes in Computer Science*, Springer, 1987, pp. 409–448. https://doi.org/10.1007/3-540-51803-7_36
[28] I.M. Hodkinson, M. Reynolds, Separation–past, present, and future, in: S.N. Artëmov, H. Barringer, A.S.d. Garcez, L.C. Lamb, J. Woods (Eds.), We Will Show Them! Essays in Honour of Dov Gabbay, Vol. 2, College Publications, 2005, pp. 117–142.
[29] H. Barringer, M. Fisher, D.M. Gabbay, G. Gough, R. Owens, METATEM: a framework for programming in temporal logic, in: J.W. de Bakker, W.P. de Roever, G. Rozenberg (Eds.), Stepwise Refinement of Distributed Systems, Models, Formalisms, Correctness, REX Workshop, Mook, the Netherlands, May 29–June 2, 1989, Proceedings, Vol. 430 of *Lecture Notes in Computer Science*, Springer, 1989, pp. 94–129. https://doi.org/10.1007/3-540-52559-9_62
[30] L. Geatti, M. Montali, A. Rivkin, Foundations of reactive synthesis for declarative process specifications, in: M.J. Wooldridge, J.G. Dy, S. Natarajan (Eds.), Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20–27, 2024, Vancouver, Canada, AAAI Press, 2024, pp. 17416–17425. https://doi.org/10.1609/AAAI.V38I16.29690
[31] L. Geatti, A. Gianola, N. Gigante, First-order automata, in: T. Walsh, J. Shah, Z. Kolter (Eds.), AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 2,–March 4, 2025, Philadelphia, PA, USA, AAAI Press, 2025, pp. 14940–14948. https://doi.org/10.1609/AAAI.V39I14.33638
[32] O. Maler, On the Krohn-Rhodes cascaded decomposition theorem, in: Time for Verification, Essays in Memory of Amir Pnueli, 2010. https://doi.org/10.1007/978-3-642-13754-9_12
[33] O. Maler, A. Pnueli, Tight bounds on the complexity of cascaded decomposition of automata, in: FOCS, 1990. https://doi.org/10.1109/FSCS.1990.89589
[34] O. Kupferman, M.Y. Vardi, Model checking of safety properties, Form. Methods Syst. Des. (2001). https://doi.org/10.1023/A:1011254632723
[35] K. Krohn, J. Rhodes, Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines, Trans. Am. Math. Soc. (1965). https://doi.org/0.2307/1994127
[36] R. McNaughton, S.A. Papert, Counter-Free Automata (MIT Research Monograph no. 65), The MIT Press, 1971. isbn: 9780262130769.
[37] R. Borelli, L. Geatti, M. Montali, A. Montanari, On cascades of reset automata, in: STACS, 2025. https://doi.org/10.4230/LIPICS.STACS.2025.20