An Algebraic Approach to Mso-Definability on Countable Linear Orderings

AN ALGEBRAIC APPROACH TO MSO-DEFINABILITY ON COUNTABLE LINEAR ORDERINGS

OLIVIER CARTON [1], THOMAS COLCOMBET [2], AND GABRIELE PUPPIS [3]

Abstract. We develop an algebraic notion of recognizability for languages of words indexed by countable linear orderings. We prove that this notion is effectively equivalent to definability in monadic second-order (MSO) logic. We also provide three logical applications. First, we establish the first known collapse result for the quantifier alternation of MSO logic over countable linear orderings. Second, we solve an open problem posed by Gurevich and Rabinovich, concerning the MSO-definability of sets of rational numbers using the reals in the background. Third, we establish the MSO-definability of the set of yields induced by an MSO-definable set of trees, confirming a conjecture posed by Bruyère, Carton, and Sénizergues.

Contents

§1. Introduction. The paper continues a long line of research aiming at understanding the notions of regularity for languages of infinite objects, e.g., infinite words and trees. The central objects in this paper are *words* indexed by countable linear orderings, i.e., total orders over finite or countable sets paired with functions mapping elements to letters in some finite alphabet. Accordingly, languages here are just sets of countable words. We use *monadic second-order (MSO) logic* as a formalism for describing such languages. In particular, an MSO formula may involve quantifications over positions of a word, as well as quantifications over sets of positions. A sentence naturally defines the language of all words that make the sentence true.

This paper provides a fine comprehension of the expressive power of MSO logic over countable linear orderings by proving a correspondence between definability in MSO and recognizability by suitable algebraic structures. More precisely, we introduce a generalization of the classical notion of finite monoid (i.e., a finite set equipped with an associative product), that we call $\circledast$-monoid, and we extend accordingly the notion of recognizability by monoid morphism to capture a large class of languages of countable words. Differently from the classical setting, $\circledast$-monoids are not finite objects, as the product mapping is defined over countable sequences of elements and a priori it is not clear how to represent this mapping by a finite table. To obtain finite presentations of the recognized languages, we follow an approach similar to [28], namely, we associate with each $\circledast$-monoid a finite number of operators with finite domain. We prove that, under natural conditions, the associated algebraic structure, called $\circledast$-algebra, uniquely determines a $\circledast$-monoid. The correspondence between $\circledast$-monoids and $\circledast$-algebras, together with the proposed notion of recognizability, gives a natural framework where languages of countable words can be represented and manipulated algorithmically. Our main contribution consists in proving that recognizability by $\circledast$-monoids/algebras corresponds effectively to definability in MSO logic, exactly as it happens for regular languages of finite words and ω-words:

> *The languages recognized by $\circledast$-monoids are the same as the languages definable in MSO logic.*

Prior results (see related work below) also focused on MSO logic over countable linear orderings and similar correspondences with algebraic structures, but mostly from the point of view of decidability of the logical theory. Our study gives a deeper insight on the expressive power of MSO logic on these structures. For example, as a by-product of our results we obtain that the quantifier hierarchy of MSO logic collapses to its second level:

> *Every language of countable words defined in MSO logic can be equally defined in the $\exists\forall$-fragment.*

The above result is reminiscent of the collapse of MSO to its existential fragment when interpreted over ω, as shown by Büchi in [6]. We also show that our collapse result is optimal, in the sense that the first level of the quantifier hierarchy does not capture the full expressive power of MSO logic on countable linear orderings. This situation is also very similar to the setting of regular languages of infinite trees, where a collapse of MSO at the second level holds [20]. Despite this

similarity and the fact that recognizable languages of countable words are MSO-interpretable from regular languages of infinite trees, our collapse result does not follow immediately from Rabin's result. Indeed, an MSO-interpretation may exploit second-order quantifications to define linear orderings inside infinite trees.

Our investigation on recognizability by $\circledast$-monoids provides also new insights on the type of properties that can be expressed in MSO logic over *uncountable* linear orderings. For example, we consider the following question that was raised and left open by Gurevich and Rabinovich in [14]:

> *Given a property for sets of rational numbers that is MSO-definable in the real line, is it possible to define it directly in the rational line? In other words, is it true that the presence of reals 'at the background' does not increase the expressive power of MSO logic?*

We answer positively the above question by building up on the correspondence between MSO-definability and recognizability by $\circledast$-monoids. The latter expressiveness result is inherently non-effective since the MSO theory of the real line is undecidable [24], while that of the rational line is decidable.

Finally, we establish an interesting correspondence between MSO-definability of languages of (possibly infinite) trees and MSO-definability of their yields:

> *Define the yield of a tree as the set of leaves ordered by the infix relation. Consider an MSO-definable tree language L that is yield-invariant, namely, such that for all trees t, t' with the same yield, $t \in L$ iff $t' \in L$. The set of yields of trees in L is effectively MSO-definable.*

In [25] a similar result was shown in the restricted setting of *finite* trees.

Related work. Büchi initiated the study of MSO logic using the tools of language theory. He established that every language of ω-words (i.e., the particular case of words indexed by the ordinal ω) definable in MSO logic is effectively recognized by a suitable form of automaton [6]. A major advance was obtained by Rabin, who extended this result to infinite trees [20]. One consequence of Rabin's result is that MSO logic is decidable over the class of all countable linear orderings. Indeed, every linear ordering can be seen as a set of nodes of the infinite tree, with the order corresponding to the infix ordering on nodes. Another proof of the decidability of the MSO theory of countable linear orderings has been given by Shelah using the composition method [24]. This automaton-free approach to logic is based on syntactic operations on formulas and is inspired from Feferman and Vaught [12]. The same paper of Shelah is also important for another result it contains: the undecidability of the MSO theory of the real line (the reals with order). However, for infinite words as for infinite trees, the theory is much richer than simply the decidability of MSO logic. In particular, MSO logic is known to be equivalent to a number of different formalisms, such as automata, some forms of algebras, and, in the ω-word case, regular expressions. MSO logic is also known to collapse to its existential fragment when interpreted on the linear order ω, that is, every formula is equivalent to a formula consisting of a block of existential quantifiers followed by a first-order formula.

Another branch of research has been pursued to raise the equivalence between logic, automata, and algebra to infinite words beyond ω-words. In [5], Büchi introduced ω_1-automata on transfinite words to prove the decidability of MSO

logic for ordinals less than ω_1. Besides the usual transitions, ω_1-automata are equipped with limit transitions of the form $P \to q$, with P set of states, which are used in a Muller-like way to process words indexed over ordinals. Büchi proved that these automata have the same expressive power as MSO logic over ordinals less than ω_1. The key ingredient is the closure under complementation of ω_1-automata. In [2], ω_1-automata have been extended to $\diamond$-automata by introducing limit transitions of the form $q \to P$ to process words over linear orderings. In [22], $\diamond$-automata are proven to be closed under complementation with respect to countable and scattered linear orderings (a linear ordering is scattered if it is nowhere dense, namely, if none of its suborders is isomorphic to the rational line). More precisely, $\diamond$-automata have the same expressive power as MSO logic over countable and scattered linear orderings [1]. However, it was already noticed in [1] that $\diamond$-automata are strictly weaker than MSO logic over countable (possibly non-scattered) linear orderings: indeed, the closure under complementation fails as there is an automaton that accepts all words with non-scattered domains, whereas there is none for scattered words.

Some of the results presented here appeared in preliminary form in the conference papers [7] and [9].

Structure of the paper. After the preliminaries in Section 2, we introduce in Section 3 the notions of $\circledast$-monoids and $\circledast$-algebras, and present the corresponding tools and results. In Section 4 we translate MSO formulas to $\circledast$-algebras and in Section 5 we establish the converse. In Section 6 we exploit the developed algebraic framework to solve three open problems that we discussed earlier, namely: (i) the collapse of the quantifier hierarchy of MSO logic, (ii) the correspondence between classical MSO-definability and definability with the reals 'at the background', and (iii) the MSO-definability of the set of yields induced by a regular yield-invariant tree language.

Acknowledgements. We are grateful to Achim Blumensath for his numerous comments on this work and to Alexander Rabinovich for the discussions on the subject and for introducing us to the question of definability with the reals at the background.

§2. Preliminaries.

In this section we recall some definitions for linear orderings, condensations, words, and languages.

2.1. Linear orderings. A *linear ordering* $\alpha = (X, <)$ is a set X equipped with a total order $<$. By a slight abuse of terminology, we call a linear ordering *countable* when its domain is finite or countable. We write α^* to denote the reverse linear ordering $(X, >)$. Two linear orderings have same *order type* if there is an order-preserving bijection between their domains. We denote by ω, ω^*, ζ, η the order types of $(\mathbb{N}, <)$, $(-\mathbb{N}, <)$, $(\mathbb{Z}, <)$, $(\mathbb{Q}, <)$, respectively. Unless strictly necessary, we do not distinguish between a linear ordering and its order type.

Given a subset I of a linear ordering α, we denote by $\alpha|_I$ the *induced subordering*. Given two subsets I, J of α, we write $I < J$ iff $x < y$ for all $x \in I$ and all

$y \in J$. A subset I of α is said to be *convex* if for all $x, y \in I$ and all $z \in \alpha$, $x < z < y$ implies $z \in I$.

The *sum* $\alpha_1 + \alpha_2$ of two linear orderings $\alpha_1 = (X_1, <_1)$ and $\alpha_2 = (X_2, <_2)$ (up to renaming, assume that X_1 and X_2 are disjoint) is the linear ordering $(X_1 \uplus X_2, <)$, where $<$ coincides with $<_1$ on X_1, with $<_2$ on X_2, and, furthermore, it satisfies $X_1 < X_2$. More generally, given a linear ordering $\alpha = (X, <)$ and, for each $i \in X$, a linear ordering $\beta_i = (Y_i, <_i)$ (assume that the sets Y_i are pairwise disjoint), we define the sum $\sum_{i \in \alpha} \beta_i$ to be the linear ordering $(Y, <')$, where $Y = \biguplus_{i \in X} Y_i$ and, for every $i, j \in X$, every $x \in Y_i$, and every $y \in Y_j$, $x <' y$ iff either $i = j$ and $x <_i y$ hold or $i < j$ holds.

A subset I of a linear ordering α is *dense in* α if for every $x < y \in \alpha$, there exists $z \in I$ such that $x < z < y$. For example, $(\mathbb{Q}, <)$ is dense in $(\mathbb{R}, <)$ and $(\mathbb{R}, <)$ is dense in itself. If a linear ordering α is dense in itself, then we simply say that α is *dense*. A linear ordering α is *scattered* if all its dense suborderings are empty or singletons. For example, $(\mathbb{N}, <)$, $(\mathbb{Z}, <)$, and all the ordinals are scattered. Being scattered is preserved under taking a subordering. A scattered sum of scattered linear orderings also yields a scattered linear ordering.

Additional material on linear orderings can be found in [23].

2.2. Condensations. A standard way to prove properties of linear orderings is to decompose them into basic objects (e.g., finite sequences, ω-sequences, ω^*-sequences, and η-orderings). This can be done by exploiting the notion of condensation.

Precisely, a *condensation* of a linear ordering α is an equivalence relation $\sim$ over α such that for all $x < y < z$, $x \sim z$ implies $x \sim y \sim z$. Equivalently, a condensation of α can be seen as a partition of α into convex subsets.

The order on α induces a corresponding order on the quotient $\alpha/_\sim$, which is called the *condensed ordering*. This condensed ordering $\alpha/_\sim$ inherits some properties from α: if α is countable (resp., scattered), then $\alpha/_\sim$ is countable (resp., scattered).

2.3. Words and languages. We use a generalized notion of word, which coincides with the notion of labelled linear ordering. Given a linear ordering α and a finite alphabet A, a *word over A with domain α* is a mapping of the form $w : \alpha \to A$. The *domain* of a word w is denoted $\mathsf{dom}(w)$. Unless specifically required, we shall always consider words of countable domain, and up to isomorphism. The set of all words (of countable domain) over an alphabet A is denoted $A^{\circledast}$. The set of all words of non-empty (countable) domain over an alphabet A is denoted $A^{\oplus}$. Given a word w and a subset I of $\mathsf{dom}(w)$, we denote by $w|_I$ the *subword* resulting from restricting the domain of w to I. If in addition I is convex, then $w|_I$ is said to be a *factor* of w.

Certain words will play a crucial role in the sequel, so we introduce specific notation for them. For example, we denote the *empty word* by ε. A word w is said to be an *η-shuffle* of set A of letters if (i) the domain $\mathsf{dom}(w)$ has order type η and (ii) for every symbol $a \in A$, the set $w^{-1}(a) = \{x \in \mathsf{dom}(w) \mid w(x) = a\}$ is dense in $\mathsf{dom}(w)$. Recall that η is – up to isomorphism – the unique countable dense linear ordering with no end-points. Likewise, for every finite set A, there is a unique, up to isomorphism, η-shuffle of A.

Given two words u, v, we denote by uv the *concatenation* of u and v, namely, the word with domain $\mathsf{dom}(u) + \mathsf{dom}(v)$, where each position $x \in \mathsf{dom}(u)$ (resp., $x \in \mathsf{dom}(v)$) is labelled by $u(x)$ (resp., $v(x)$). This is readily generalized to infinite concatenations of the form $\prod_{i\in\alpha} w_i$, for any linear ordering α and any sequences of words $(w_i)_{i\in\alpha}$, the resulting word having domain $\sum_{i\in\alpha} \mathsf{dom}(w_i)$. The *$\omega$-power* of a word w is defined as $w^\omega = \prod_{i\in\omega} w$. Similarly, we define the *ω^*-power* $w^{\omega^*} = \prod_{i\in\omega^*} w$. By a slight abuse of terminology, we also define the *η-shuffle* of a tuple of words $w_1, \dots, w_k$ as the word

$$\{w_1, \dots, w_k\}^\eta \ =^{\text{def}} \ \prod_{i\in\eta} w_{f(i)}$$

where f is the unique η-shuffle of the set of letters $I = \{1, \dots, k\}$.

A *$\circledast$-language* (resp., *$\oplus$-language*) is any set of words (resp., non-empty words) over a fixed finite alphabet. The operations of concatenation, ω-power, ω^*-power, η-shuffle, etc. are extended to languages in the obvious way.

§3. Algebras for countable words. In this section we present the algebraic objects that are suited for deriving a notion of recognizability for languages of countable words. As it was already the case for words with domain ω, [19, 28], our definitions come in two flavors, $\circledast$-monoids (corresponding to ω-monoids) and $\circledast$-algebras (corresponding to Wilke's algebras). We prove the equivalence of the two notions when the supports are finite.

3.1. Countable products. We introduce below a notion of product indexed by countable linear orderings that satisfies a generalized associativity property.

DEFINITION 3.1. *A* (generalized) product *over a set S is a function π from $S^\oplus$ to S such that, for every $a \in S$, $\pi(a) = a$ and, for every family of words $(u_i)_{i\in\alpha} \in (S^\oplus)^\oplus$,*

$$\text{(generalized associativity)} \qquad \pi\Big(\prod\nolimits_{i\in\alpha} \pi(u_i)\Big) \ = \ \pi\Big(\prod\nolimits_{i\in\alpha} u_i\Big) \ .$$

The pair (S, π) is called a $\oplus$-semigroup.
If the same definition holds, with π function from $S^\circledast$ to S and $(u_i)_{i\in\alpha} \in (S^\circledast)^\circledast$, then (S, π) is called a $\circledast$-monoid.

As an example, the function $\prod$ that maps any countable sequence of non-empty words to their concatenation is a generalized product over $A^\oplus$. Hence, $(A^\oplus, \prod)$ is a $\oplus$-semigroup; it is indeed the *free* $\oplus$-semigroup generated by A. Similarly, $(A^\circledast, \prod)$ is the free $\circledast$-monoid generated by A.

Given a $\oplus$-semigroup (S, π), we call *neutral element* an element $1 \in S$ such that, for every word $w \in S^\oplus$, if $w|_{\neq 1}$ is the subword of w obtained by removing every occurrence of the element 1 and $w|_{\neq 1}$ is non-empty, then $\pi(w) = \pi(w|_{\neq 1})$. Note that the neutral element, if exists, is unique: given two neutral elements $1, 1' \in S$, we have $1 = \pi(1) = \pi(11'|_{\neq 1'}) = \pi(11') = \pi(11'|_{\neq 1}) = \pi(1') = 1'$.

At some places in the proofs it will be necessary to use $\oplus$-semigroups rather than $\circledast$-monoids. The two notions are however very close. On the one hand, any $\circledast$-monoid (S, π) can be seen as a $\oplus$-semigroup by simply restricting its generalized product π to $S^\oplus$. On the other hand, any $\oplus$-semigroup (S, π) can be

extended to a $\circledast$-monoid either by letting $\pi(\varepsilon) = 1$, where ε is the empty word and 1 is the (unique) neutral element of (S, π), or, if (S, π) has no neutral element, by introducing a fresh element $1 \notin S$ and by letting $\pi(\varepsilon) = 1$ and $\pi(w) = \pi(w|_{\neq 1})$ for all words w over $S \uplus \{1\}$.

A *morphism* from a $\oplus$-semigroup (S, π) to another $\oplus$-semigroup (S', π') is a mapping $h : S \to S'$ such that, for every word $(w_i)_{i \in \alpha} \in S^{\oplus}$,

$$h(\pi(w)) = \pi'(\bar{h}(w)) \ ,$$

where $\bar{h}$ is the pointwise extension of h to words. A *morphism of $\circledast$-monoids* is defined similarly, this time with $(w_i)_{i \in \alpha} \in S^{\circledast}$.

A $\oplus$-language $L \subseteq A^{\oplus}$ is *recognizable by a $\oplus$-semigroup* if there exists a morphism h from $(A^{\oplus}, \prod)$ to some finite $\oplus$-semigroup (S, π) (here finite means that S is finite) such that $L = h^{-1}(F)$ for some $F \subseteq S$ (equivalently, $h^{-1}(h(L)) = L$). Similarly, a $\circledast$-language $L \subseteq A^{\circledast}$ is *recognizable by a $\circledast$-monoid* if there exists a morphism h from $(A^{\circledast}, \prod)$ to some finite $\circledast$-monoid (M, π) (here finite means that M is finite) such that $L = h^{-1}(F)$ for some $F \subseteq M$ (equivalently, $h^{-1}(h(L)) = L$).

We are mainly interested in languages recognizable by finite $\circledast$-monoids. However, it is worth noticing that, with respect to membership of non-empty words, this notion is the same as recognizability by finite $\oplus$-semigroups: indeed, a language L is recognizable by finite $\circledast$-monoids iff $L \setminus \{\varepsilon\}$ is recognizable by finite $\oplus$-semigroups.

3.2. From countable products to algebras. The notion of recognizability for $\circledast$-languages makes use of a product function π that needs to be represented, a priori, by an infinite table. This is a not usable as it stands for finite presentations of languages, nor for decision procedures. That is why, given a finite $\oplus$-semigroup (S, π), we define the following (finitely presentable) algebraic operators:

- the *binary product* $\cdot : S^2 \to S$, mapping any pair of elements $a, b \in S$ to the element $\pi(ab)$,
- the *τ-iteration* $\tau : S \to S$, mapping any element $a \in S$ to the element $\pi(a^{\omega})$ (thus, τ is the analogous of the ω-power inside S),
- the *τ^*-iteration* $\tau^* : S \to S$, mapping any element $a \in S$ to the element $\pi(a^{\omega^*})$ (thus, τ^* is the analogous of the ω^*-power inside S),
- the *κ-iteration* $\kappa : \mathscr{P}(S) \setminus \{\varnothing\} \to S$, mapping any non-empty subset $\{a_1, \ldots, a_k\}$ of S to the element $\pi\big(\{a_1, \ldots, a_k\}^{\eta}\big)$ (κ is the analogous of the η-shuffle inside S).

Furthermore, if (S, π) is a $\circledast$-monoid, we see the neutral element as a nullary operator induced by π, namely, as $1 = \pi(\varepsilon)$. One says that $\cdot$, τ, τ^*, κ (and possibly 1) are *induced by* π. From now on, we shall use the operator $\cdot$ with infix notation (e.g., $a \cdot b$) and the operators τ, τ^*, and κ with superscript notation (e.g., a^{τ}, $\{a_1, \ldots, a_k\}^{\kappa}$). As shown below, the resulting structures $(S, \cdot, \tau, \tau^*, \kappa)$ $(S, 1, \cdot, \tau, \tau^*, \kappa)$ have the property of being, respectively, a $\oplus$-algebra and a $\circledast$-algebra.

Definition 3.2. *A structure $(S, \cdot, \tau, \tau^*, \kappa)$, with $\cdot : S^2 \to S$, $\tau, \tau^* : S \to S$, and $\kappa : \mathscr{P}(S) \setminus \{\varnothing\} \to S$, is called a* $\oplus$-algebra *if:*

(A1) $(S,\cdot)$ *is a semigroup, namely, for every* $a,b,c \in S$, $a\cdot(b\cdot c) = (a\cdot b)\cdot c$,

(A2) τ *is* compatible to the right*, namely, for every* $a,b \in S$ *and every* $n>0$, $(a\cdot b)^\tau = a\cdot(b\cdot a)^\tau$ *and* $(a^n)^\tau = a^\tau$,

(A3) τ^* *is* compatible to the left*, namely, for every* $a,b \in S$ *and every* $n>0$, $(b\cdot a)^{\tau^*} = (a\cdot b)^{\tau^*}\cdot a$ *and* $(a^n)^{\tau^*} = a^{\tau^*}$,

(A4) κ *is* compatible with shuffles*, namely, for every non-empty subset* P *of* S*, every element* c *in* P*, every subset* P' *of* P*, and every non-empty subset* P'' *of* $\{P^\kappa,\ a\cdot P^\kappa,\ P^\kappa\cdot b,\ a\cdot P^\kappa\cdot b \mid a,b\in P\}$*, we have*

$$\begin{aligned} P^\kappa &= P^\kappa\cdot P^\kappa = P^\kappa\cdot c\cdot P^\kappa \\ &= (P^\kappa)^\tau = (P^\kappa\cdot c)^\tau \\ &= (P^\kappa)^{\tau^*} = (c\cdot P^\kappa)^{\tau^*} \\ &= (P'\cup P'')^\kappa\ . \end{aligned}$$

A $\circledast$-algebra $(M,1,\cdot,\tau,\tau^*,\kappa)$ *is a* $\oplus$*-algebra* $(M,\cdot,\tau,\tau^*,\kappa)$ *with a distinguished element* $1\in M$ *such that*

(A5) $x\cdot 1 = 1\cdot x = x$, $1^\tau = 1^{\tau^*} = \{1\}^\kappa = 1$, *and* $P^\kappa = (P\cup\{1\})^\kappa$, *for all* $x\in M$ *and all non-empty* $P\subseteq M$.

The typical $\oplus$-algebras and $\circledast$-algebras are:

LEMMA 3.3. *For every alphabet* A, $(A^\oplus,\cdot,\omega,\omega^*,\eta)$ *is a* $\oplus$*-algebra and* $(A^\circledast,\varepsilon,\cdot,\omega,\omega^*,\eta)$ *is a* $\circledast$*-algebra*[1].

PROOF. By a systematic analysis of Axioms A1-A5. ⊣

Furthermore, as we mentioned above, every $\oplus$-semigroup induces a $\oplus$-algebra and every $\circledast$-monoid induces a $\circledast$-algebra:

LEMMA 3.4. *For every* $\oplus$*-semigroup* (S,π), $(S,\cdot,\tau,\tau^*,\kappa)$ *is a* $\oplus$*-algebra, where the operators* $\cdot$, τ, τ^*, *and* κ *are those induced by* π*. Similarly every* $\circledast$*-monoid* (S,π), $(S,1,\cdot,\tau,\tau^*,\kappa)$ *is a* $\circledast$*-algebra, where the operators* $\cdot$, τ, τ^*, κ, *and* 1 *are those induced by* π.

PROOF. The results are simply inherited from Lemma 3.3 by morphism. Let (S,π) be a $\oplus$-semigroup inducing the operators $\cdot,\tau,\tau^*,\kappa$. The structure $(S^\oplus,\prod)$ is also a $\oplus$-semigroup, which induces the operations of concatenation, ω-power, ω^*-power, and η-shuffle. Furthermore, the product π can be seen as a surjective morphism from $(S^\oplus,\prod)$ to (S,π) (just a morphism of abstract algebras, not of $\oplus$-algebras). By definition of $\cdot,\tau,\tau^*,\kappa$, this morphism maps concatenation to $\cdot$, ω-power to τ-iteration, ω^*-power to τ^*-iteration, and η-shuffle to κ-iteration. It follows that any equality involving concatenation, ω-power, ω^*-power, and η-shuffle is also satisfied by the analogous operations $\cdot$, τ, τ^*, and κ. In particular, the axioms that, thanks to Lemma 3.3, are satisfied by the $\oplus$-algebra $(S^\oplus,\cdot,\omega,\omega^*,\eta)$ are directly transferred to $(S,\cdot,\tau,\tau^*,\kappa)$. The case of a $\circledast$-monoid is similar. ⊣

[1] Similarly to what happens for Wilke's algebras [28], $(A^\oplus,\cdot,\omega,\omega^*,\eta)$ is not the free $\oplus$-algebra generated by A, as the free algebra generated by a finite set is by definition countable, while $A^\oplus$ has the cardinality of the continuum.

3.3. From algebras to countable products. Here, we aim at proving a converse to Lemma 3.4, namely, that every *finite* $\oplus$-algebra $(S, \cdot, \tau, \tau^*, \kappa)$ can be uniquely extended to a $\oplus$-semigroup (S, π), and similarly for $\circledast$-algebras and $\circledast$-monoids (Theorem 3.11 and Corollary 3.12).

Let us fix a finite $\oplus$-algebra $(S, \cdot, \tau, \tau^*, \kappa)$. In this section, we assume that all words are over the alphabet S. The objective of the construction is to attach to each word u (over the alphabet S) a 'value' in S. Furthermore, this value needs to be shown unique.

The key ingredient for associating a unique value in S to each word $u \in S^\oplus$ is the notion of *evaluation tree*. Intuitively, this is an infinite tree describing a strategy for evaluating larger and larger factors of the word u. To define these objects, we need to first introduce the concept of *condensation tree*, which is a convenient representation of nested condensations of a linear ordering. This will provide the underlying structure of an evaluation tree. The nodes of a condensation tree are convex subsets of the linear ordering and the descendant relation is given by inclusion. The set of children of each node defines a condensation. Furthermore, in order to provide an induction parameter, we require that the branches of a condensation tree are finite (but their length may not be uniformly bounded).

DEFINITION 3.5. *A* condensation tree *over a linear ordering α is a set T of non-empty convex subsets of α such that:*

- $\alpha \in T$,
- *for all I, J in T, either $I \subseteq J$ or $J \subseteq I$ or $I \cap J = \varnothing$,*
- *for all $I \in T$, the union of all $J \in T$ such that $J \subsetneq I$ is either I or $\varnothing$,*
- *every subset of T totally ordered by inclusion is finite.*

Elements in T are called *nodes*. The node α is called the *root* of the tree. Nodes minimal for $\subseteq$ are called *leaves*; the other nodes, including the root, are called *internal nodes*. A node $I \in T$ is a *descendant* of a node $J \in T$ (and accordingly J is an *ancestor* of I) if $I \subseteq J$. If in addition we have $I \neq J$, then we say that I is a *proper descendant* of J. Similarly, I is a *child* of a node J (and accordingly J is the *parent* of I) if $I \subsetneq J$ and, for all $K \in T$, $I \subsetneq K$ implies $J \subseteq K$. According to the definition, if I is an internal node of a condensation tree T over α, then it has a set of children that forms a partition of I into convex subsets. We denote this partition by $\mathsf{children}_T(I)$, and we observe that it naturally corresponds to a condensation of $\alpha|_I$. When the tree T is clear from the context, we will denote by $\mathsf{children}(I)$ the set of all children of I in T and, by extension, the corresponding condensation and the corresponding condensed ordering. Finally, we define the *subtree* of T rooted at some of node I of it as the condensation tree obtained by restricting T to the descendants of I (including I itself).

We now introduce evaluation trees. Intuitively, these are condensation trees where each internal node has an associated value in S that can be 'easily computed' from the values of its children. Here it comes natural to consider a word u 'easy to compute' if it is isomorphic to either ab, a^ω, a^{ω^*}, or P^η, for some elements $a, b \in S$ and some non-empty set $P \subseteq S$. Indeed, in each of these cases, the value of u can be computed by a single application of the operations of the

$\oplus$-algebra $(S, \cdot, \tau, \tau^*, \kappa)$. Formally, the words that are easily computable are precisely those that belong to the domain of the partial function π_0, defined just below:

DEFINITION 3.6. *Let π_0 be the partial function from $S^\oplus$ to S such that:*

- $\pi_0(ab) = a \cdot b$ *for all* $a, b \in S$,
- $\pi_0(e^\omega) = e^\tau$ *for all* idempotents $e \in S$ *(i.e., all $e \in S$ such that $e \cdot e = e$),*
- $\pi_0(e^{\omega^*}) = e^{\tau^*}$ *for all idempotents $e \in S$,*
- $\pi_0(P^\eta) = P^\kappa$ *for all non-empty sets $P \subseteq S$,*
- *in all remaining cases, π_0 is undefined.*

DEFINITION 3.7. *An* evaluation tree *over a word u is a pair $\mathcal{T} = (T, \gamma)$, where T is a condensation tree over the domain of u and γ is a function from T to S such that:*

- *every leaf of T is a singleton of the form $\{x\}$ and $\gamma(\{x\}) = u(x)$,*
- *for every internal node I of T, the partial function π_0 is defined on the word $\gamma(\mathsf{children}(I))$ that has domain $\mathsf{children}(I)$ and labels each position $J \in \mathsf{children}(I)$ with $\gamma(J)$; in addition, we have $\gamma(I) = \pi_0(\gamma(\mathsf{children}(I)))$.*

The value *of (T, γ) is defined to be $\gamma(\alpha)$, i.e., the value of the root.*

Let us turn back to the problem of associating a unique value in S to each word $u \in S^\circledast$. Based on the previous definitions, we can solve this problem in two steps. First, we show that every word u has an evaluation tree, and thus a possible value that can be associated with it. Then, we show that the associated value in fact does not depend on the choice of the evaluation tree over u, namely, that evaluation trees over the same word induce the same value. The next two propositions formalize precisely these two steps.

PROPOSITION 3.8. *For every word u, there exists an evaluation tree over u.*

PROPOSITION 3.9. *Evaluation trees over the same word have the same value.*

The proofs of the two propositions are quite technical and deferred to Sections 3.4 and 3.5, respectively. Before seeing those proofs in detail, we discuss the basic ingredients here. We then conclude the section by mentioning a few important consequences of the developed framework.

The proof of Proposition 3.8 resembles the construction used by Shelah in his proof of decidability of the monadic second-order theory of countable linear orderings [24]. In particular, it uses a theorem of Ramsey [21] and a lemma stating that every non-trivial word indexed by a countable dense linear ordering has an η-shuffle as a factor. Note that this latter lemma, and hence also Proposition 3.8, relies on the fact that the domain of the word is countable. The proof of the proposition also makes use of Zorn's Lemma (or equally, the Axiom of Choice), so it is a proof in ZFC. On the other hand, we observe that it does not make any use of Axioms A1-A4.

Proposition 3.9 can be regarded as the core contribution of the paper, and its proof technique is quite original. For example. as opposed to Proposition 3.8, one cannot find any ingredient of the proof of Proposition 3.9 in [24]. The proof heavily relies on the use of Axioms A1-A4. As a matter of fact, each axiom

can be seen as an instance of Proposition 3.9 in some special cases of evaluation trees of small height. The proof also depends on Proposition 3.8, in the sense that is exploits in several places the existence of evaluation trees over arbitrary (countable) words.

Another key ingredient for the proof of Proposition 3.9, which is also reused in other proofs, is the formalization of a suitable induction principle on condensation and evaluation trees. More precisely, by exploiting the fact that all branches of a condensation tree are finite, one can associate with any condensation tree T a countable ordinal $\mathsf{rank}(T)$, called the *rank* of T. Intuitively, this is the smallest ordinal β that enables a labelling of the nodes of T by ordinals less than or equal to β in such a way that the label of each node is strictly greater than the labels of its children.

LEMMA 3.10. *It is possible to associate with each condensation tree T a countable ordinal $\mathsf{rank}(T)$ in such a way that $\mathsf{rank}(T') < \mathsf{rank}(T)$ for all subtrees T' of T rooted at proper descendants of the root.*

PROOF. We associate with each node $I \in T$ a countable ordinal β_I as follows. For every leaf I of T, let $\beta_I = 0$. Then, given an internal node I of T, we assume that β_J is defined for every child J of I, and we define β_I as the ordinal $\sup\{\beta_J + 1 \mid J \in \mathsf{children}(I)\}$ (note that this is either a successor ordinal or a limit ordinal, depending on whether the set $\{\beta_J + 1 \mid J \in \mathsf{children}(I)\}$ has a maximum element or not). Since T has no infinite branch, it follows that β_I is defined for every node of T. We thus let $\mathsf{rank}(T) = \beta_I$, where I is the root of T. By construction, the function rank that maps any condensation tree T to its rank $\mathsf{rank}(T)$ satisfies the properties stated in the lemma. ⊣

Now, assuming that Propositions 3.8 and 3.9 hold, we can prove the desired correspondence between $\oplus$-semigroups and $\oplus$-algebras:

THEOREM 3.11. *Every finite $\oplus$-algebra $(S, \cdot, \tau, \tau^*, \kappa)$ is induced by a unique product π from $S^\oplus$ to S.*

PROOF. Given a word w with domain α, one defines $\pi(w)$ to be the value of some evaluation tree over w (the evaluation tree exists by Proposition 3.8 and the value $\pi(w)$ is unique by Proposition 3.9).

We prove that π satisfies the generalized associativity property. Let $\sim$ be a condensation of the domain α. For all classes $I \in \alpha/_\sim$, let $\mathcal{T}_I$ be some evaluation tree over $w|_I$. Let also $\mathcal{T}'$ be some evaluation tree over the word $w' = \prod_{I \in \alpha/_\sim} \pi(w|_I)$. One constructs an evaluation tree $\mathcal{T}$ over w by first lifting $\mathcal{T}'$ from the linear ordering $\alpha/_\sim$ to α (this is done by replacing each node J in $\mathcal{T}'$ by $\bigcup J$) and then substituting each leaf of $\mathcal{T}'$ corresponding to some class $I \in \alpha/_\sim$ with the evaluation tree $\mathcal{T}_I$. The last step is possible (namely, respects the definition of evaluation tree) because the value of each evaluation tree $\mathcal{T}_I$ is $\pi(w|_I)$, which coincides with the value $w'(I)$ at the leaf I of $\mathcal{T}'$. By Proposition 3.9, the resulting evaluation tree $\mathcal{T}$ has the same value as $\mathcal{T}'$ and this proves that $\pi(w) = \pi\left(\prod_{I \in \alpha/_\sim} \pi(w|_I)\right)$.

It remains to prove that the above choice of π indeed induces the operators $\cdot, \tau, \tau^*, \kappa$. This is done by a straightforward case analysis. ⊣

The result that we just proved immediately implies an analogous correspondence between $\circledast$-monoids and $\circledast$-algebras:

COROLLARY 3.12. *Every finite $\circledast$-algebra $(M, 1, \cdot, \tau, \tau^*, \kappa)$ is induced by a unique product π from $M^\circledast$ to M.*

Finally, we discuss the algorithmic implications of the above results. In the same way as we talked of languages recognized by finite $\circledast$-monoids, we can equally talk of languages *recognized by finite $\circledast$-algebras*. Moreover, because finite $\circledast$-algebras are finite objects, this enables the possibility of manipulating and reasoning on recognized languages by means of algorithms. An example of such a possibility is given just below, in a theorem that shows the decidability of the emptiness problem for languages recognized by finite $\circledast$-algebras. The theorem also gives effective witnesses of non-empty languages, in the same spirit as some results of Laüchli and Leonard for models of first-order logic and weak monadic second-order logic [17, 18]. Other examples of algorithmic manipulation of languages can be found in Section 4, where we will prove some closure properties of languages recognized by finite $\circledast$-algebras.

THEOREM 3.13. *The problem of testing whether $L \neq \varnothing$ for any language $L \subseteq A^\circledast$ recognized by a given finite $\circledast$-algebra is decidable. Moreover, if $L \neq \varnothing$, then a finite expression can be effectively constructed that represents some word in L and is generated by the following grammar:*

$$\mathtt{w} \ ::= \ \varepsilon \mid a \mid \mathtt{w} \cdot \mathtt{w} \mid \mathtt{w}^\omega \mid \mathtt{w}^{\omega^*} \mid \{\mathtt{w}, \dots, \mathtt{w}\}^\eta \qquad \text{for } a \in A.$$

PROOF. Recall that a language $L \subseteq A^\circledast$ is recognized by a $\circledast$-algebra $(M, 1, \cdot, \tau, \tau^*, \kappa)$ if there is $F \subseteq M$ and a morphism $h : (A^\circledast, \prod) \to (M, \pi)$ such that $L = h^{-1}(F)$, where (M, π) is the $\circledast$-monoid induced by $(M, 1, \cdot, \tau, \tau^*, \kappa)$ (Corollary 3.12). To decide the emptiness problem, it is sufficient to describe an algorithm that, given $(M, 1, \cdot, \tau, \tau^*, \kappa)$ and $h : A \to M$ (which uniquely extends to a function from $(A^\circledast, \prod)$ to (M, π)), computes the set

$$h(A^\circledast) = \big\{1\big\} \cup \big\{h(u) \;\big|\; u \in A^\circledast\big\}$$

(note that $L = h^{-1}(F) \neq \varnothing$ iff $h(A^\circledast) \cap F \neq \varnothing$).

To compute the set $h(A^\circledast)$, one can simply saturate the subset $\{1\} \cup h(A)$ of M under the operations $\cdot, \tau, \tau^*, \kappa$. Formally, given $S \subseteq M$, we define the set *generated* by S in $(M, 1, \cdot, \tau, \tau^*, \kappa)$ as the least set $\langle S \rangle$ that contains S and satisfies the following closure properties:

- if $a, b \in \langle S \rangle$, then $a \cdot b \in \langle S \rangle$,
- if $a \in \langle S \rangle$, then $a^\tau \in \langle S \rangle$,
- if $a \in \langle S \rangle$, then $a^{\tau^*} \in \langle S \rangle$,
- if $\varnothing \neq P \subseteq \langle S \rangle$, then $P^\kappa \in \langle S \rangle$.

Clearly, the set $\langle S \rangle$ can be easily computed from S.

Below we prove that the set generated by $\{1\} \cup h(A)$, denoted $\big\langle \{1\} \cup h(A) \big\rangle$, coincide with $h(A^\circledast)$. First, it is easy to see that $\big\langle \{1\} \cup h(A) \big\rangle \subseteq h(A^\circledast)$, since $\{1\} \cup h(A) \subseteq h(A^\circledast)$ and containments in $h(A^\circledast)$ are preserved under all operations of the saturation. The opposite containment $h(A^\circledast) \subseteq \big\langle \{1\} \cup h(A) \big\rangle$ follows by Proposition 3.8 and some inductive argument. More precisely, one first observes

that the value $h(w)$ of any word $w \in A^{\circledast}$ is the same as witnessed by some evaluation tree $\mathcal{T}_w$. Then, one exploits a simple induction on $\mathcal{T}_w$ – in fact, on the rank of the underlying condensation tree – to verify that the set $\langle\{1\} \cup h(A)\rangle$ contains the value of $\mathcal{T}_w$.

The above arguments show that the set $h(A^{\circledast}) = \langle\{1\} \cup h(A)\rangle$ can be effectively constructed by a saturation procedure. To conclude, we observe that this procedure implicitly associates with each element of $\langle\{1\} \cup h(A)\rangle$ a corresponding finite expression, as generated by the grammar of the claim. ⊣

3.4. Existence of evaluation trees. We introduce a few additional ingredients for the proof of Proposition 3.8, namely, for showing the existence of evaluation trees over any word. We begin with a variant of Ramsey's theorem for additive labellings. Recall that $(S, \cdot, \tau, \tau^*, \kappa)$ is a finite $\oplus$-algebra and, in particular, $(S, \cdot)$ is a finite semigroup.

DEFINITION 3.14. *Let $(S, \cdot)$ be a semigroup. An* additive labelling *is a function f that maps any two of points $x < y$ in a linear ordering α to an element $f(x, y)$ in S in such a way that, for all $x < y < z$, $f(x, y) \cdot f(y, z) = f(x, z)$.*

LEMMA 3.15 (Ramsey [21]). *Given a linear ordering α with a minimum element $\bot$ and no maximum element, and given an additive labelling $f : \alpha \times \alpha \to (S, \cdot)$, there exist an ω-sequence $\bot < x_1 < x_2 < \dots$ of points in α and two elements $a, e \in S$ such that:*

- *for all $y \in \alpha$, there is $x_i > y$,*
- *for all $i > 0$, $f(\bot, x_i) = a$,*
- *for all $j > i > 0$, $f(x_i, x_j) = e$.*

Note that the conditions in the above lemma imply that e is an idempotent: indeed, we have $e \cdot e = f(x_i, x_{i+1}) \cdot f(x_{i+1}, x_{i+2}) = f(x_i, x_{i+2}) = e$.

In the same spirit of Lemma 3.15, the following lemma shows that every countable dense word contains an η-shuffle as a factor. Even though this result appears already in [24], we give a proof of it for the sake of self-containment.

LEMMA 3.16 (Shelah [24]). *Every word indexed by a non-empty non-singleton countable dense linear ordering contains a factor that is an η-shuffle.*

PROOF. Let α be a non-empty non-singleton countable dense linear ordering, let $A = \{a_1, \dots, a_n\}$ be a generic alphabet, and let w be a word over A with domain α. For the sake of brevity, given a symbol $a \in A$, we denote by $w^{-1}(a)$ the set of all points $x \in \alpha$ such that $w(x) = a$. We then define $w_0 = w$ and $A_0 = \varnothing$, and we recursively apply the following construction for each index $1 \leq i \leq n$:

$$A_i = \begin{cases} A_{i-1} \cup \{a_i\} & \text{if } w_i^{-1}(a_i) \text{ is dense in } \mathsf{dom}(w_i), \\ A_{i-1} & \text{otherwise,} \end{cases}$$

$$w_i = \begin{cases} w_{i-1} & \text{if } w_i^{-1}(a_i) \text{ is dense in } \mathsf{dom}(w_i), \\ w_{i-1}|_I & \text{otherwise, where } I \text{ is any open non-empty} \\ & \text{convex subset of } \alpha \text{ such that } w^{-1}(a_i) \cap I = \varnothing. \end{cases}$$

By construction, the domain of the factor w_n is non-empty, non-singleton, countable, and dense. Moreover, for all symbols $a \in A$, either $w_n^{-1}(a_j)$ is dense in $\mathsf{dom}(w_n)$ or empty, depending on whether $a \in A_n$ or not.This shows that w_n is an η-shuffle of the set A_n. ⊣

We are now ready to prove Proposition 3.8:

PROOF OF PROPOSITION 3.8. Let u be a word with countable domain α. We say that a convex subset I of α is *definable* if there is an evaluation tree over the factor $u|_I$. Similarly, we say that I is *strongly definable* if every non-empty convex subset J of I is definable. We first establish the following claim:

CLAIM. *For every ascending chain $I_0 \subseteq I_1 \subseteq \dots$ of strongly definable convex subsets of α, the limit $I = \bigcup_{i\in\mathbb{N}} I_i$ is strongly definable.*

PROOF OF CLAIM. Let J be a non-empty convex subset of I and let $J_i = I_i \cap J$ for all $i \in \mathbb{N}$. We prove that $J = \bigcup_{i\in\mathbb{N}} J_i$ is definable, namely, we show how to construct an evaluation tree over the factor $u|_J$. Without loss of generality we assume that the J_i's are non-empty. Note that all the J_i's are strongly definable. Of course, if the sequence of the J_i's is ultimately constant, then $J = J_i$ for a sufficiently large $i \in \mathbb{N}$ and the existence of an evaluation tree over $u|_J$ follows trivially from the fact that J_i is strongly definable. We now consider the case when all the J_i's coincide on the left. We can partition J into a sequence of convex subsets $K_0 < K_1 < \dots$, where $K_0 = J_0$ and $K_{i+1} = J_{i+1} \setminus J_i$ for all $i \geq 1$. The convex subsets K_i form a condensation of J such that $J_i = K_0 \cup \dots \cup K_i$ for all $i \in \mathbb{N}$. For every $i < j$ in $\mathbb{N}$, we define $K_{i,j} = K_i \cup \dots \cup K_{j-1}$. We recall that every convex J_j, as well as every convex subset $K_{i,j}$ of it, is strongly definable. We can thus associate with each $K_{i,j}$ an evaluation tree $\mathcal{T}_{i,j}$ over $u|_{K_{i,j}}$. We denote by $c_{i,j}$ the value of $\mathcal{T}_{i,j}$. Using Lemma 3.15 (i.e., Ramsey's Theorem), one can extract a sequence $0 < i_1 < i_2 < \dots$ in ω such that $c_{i_1,i_2} = c_{i_2,i_3} = \dots$ (and moreover, this element is an idempotent). We can then construct an evaluation tree over $u|_J$ that has root J and the convex subsets K_{0,i_1}, K_{i_1,i_2}, $\dots$ for children, with the associated evaluation subtrees $\mathcal{T}_{0,i_1}$, $\mathcal{T}_{i_1,i_2}$, $\dots$. This allows us to conclude that J is a definable convex when the J_i's coincide on the left. The case where the J_i's coincide on the right is symmetric. Finally, in the general case, we can partition each set J_i into two subsets J_i' and J_i'' such that (i) $J_i' < J_i''$, (ii) the sequence of the J_i''s coincide on the right, and (iii) the sequence of the J_i'''s coincide on the left. Let $J' = \bigcup_{i\in\mathbb{N}} J_i'$ and $J'' = \bigcup_{i\in\mathbb{N}} I_i''$. One knows by the cases above that there exist evaluation trees over $u|_{J'}$ and over $u|_{J''}$. Finally, one can easily construct an evaluation tree over $u|_J = u|_{J'\cup J''}$ out of the evaluation trees for J' and J''. This proves that J is definable and hence I is strongly definable. ⊣

Turning back to the main proof, let us now consider the set $\mathcal{C}$ of all condensations C of α such that every class is strongly definable. Condensations in $\mathcal{C}$ are naturally ordered by the 'finer than' relation. Let us consider a chain $(C_i)_{i\in\beta}$ of condensations in $\mathcal{C}$ ordered by the finer than relation, i.e., for all $j < i$ in β, C_j is finer than C_j. Since α is countable, one can assume that β is countable, or even better that $\beta = \omega$. Let us consider the *limit condensation* C, i.e., the finest condensation that is coarser than every C_i. Each class $I \in C$ is the union of a sequence of convex subsets I_i, with $I_i \in C_i$ for all $i \in \mathbb{N}$. From the assumption

that every condensation C_i belongs to $\mathcal{C}$, we get that I_i is strongly definable and from the claim above, we conclude that I is strongly definable as well. This shows that the limit condensation C belongs to $\mathcal{C}$ and hence every chain of $\mathcal{C}$ has an upper bound in $\mathcal{C}$.

It follows that we can apply Zorn's Lemma and deduce that $\mathcal{C}$ contains a maximal element, say C. If C is a condensation with single class, this means that there exists an evaluation tree over u and the proposition is established. Otherwise, we shall head toward a contradiction. Consider the condensed ordering induced by C (by a slight abuse of notation, we denote it also by C). Two cases can happen: either C contains two consecutive classes or C is a dense linear order.

In the former case, we fix two consecutive classes $I, I' \in C$, with $I < I'$. We observe that each class of C is a limit of strongly definable convexes and hence, by the previous claim, it is also strongly definable. It is then easy to see that the union $I \cup I'$ of the two consecutive strongly definable convexes I and I' is also strongly definable, which contradicts the definition of C.

In the second case we have that the linear ordering C is dense in itself. As before, we recall that each class of C is strongly definable and we prove that there exist non-trivial unions of classes of C that are strongly definable (a contradiction). We begin by associating with each convex subset J of a class I of C an evaluation tree $\mathcal{T}_J$ over $u|_J$ and we denote by c_J the value induced by it. We then consider the word $v = \prod_{I \in C} c_I$. We know from Lemma 3.16 that that v contains a factor that is an η-shuffle, say, $v' = v|_{C'}$ for some convex $C' \subseteq C$. Let $J = \bigcup_{I \in C'} I$. To prove that J is strongly definable we consider a convex $K \subseteq J$ and we construct an evaluation tree $\mathcal{T}_K$ over $u|_K$ as follows. First we observe that K is the union of all non-empty convexes of the form $I \cap K$, for $I \in C'$, and that each set $I \cap K$ is contained in a class of C, hence it is definable and has value $c_{I \cap K}$. Now, one needs to distinguish some cases depending on whether C' contains minimal/maximal convexes I intersecting K. For the sake of simplicity, we only consider the case where C' contains a minimal convex I_0 such that $I_0 \cap K \neq \varnothing$, but no maximal convex I such that $I \cap K \neq \varnothing$. In this case, we recall that v' is an η-shuffle and that its restriction to the non-empty convexes $I \cap K$, with $I \in C'$, is the juxtaposition of the singleton $c_{I_0 \cap K}$ and the η-shuffle $\prod_{I \in C''}(c_{I \cap K})$, where $C'' = \{I \in C' \mid I \cap K \neq \varnothing,\ I \neq I_0\}$. An evaluation tree $\mathcal{T}_K$ over $u|_K$ can be constructed by appending to the root K two subtrees: the evaluation tree $\mathcal{T}_{I_0 \cap K}$ associated with the definable convex $I_0 \cap K$, and the evaluation tree $\mathcal{T}_{K \setminus I_0}$ that consists of the node $K \setminus I_0$ and the direct subtrees $\mathcal{T}_{I \cap K}$, for all $I \in C''$. This shows that there is a non-trivial union J of classes of C that is strongly definable, which contradicts the definition of C. ⊣

3.5. Equivalence of evaluation trees. We now turn towards proving Proposition 3.9, namely, the equivalence of evaluation trees with respect to the induced values. As we already mentioned, the proof is rather long and requires a series of technical lemmas.

For reasons that will be clear in the sequel, it is convenient to extend slightly the domain of the partial function π_0 that computes values of 'simple words'

(cf. Definition 3.6). Intuitively, such an extension adds prefixes and suffixes of finite length to the elements of the original domain of π_0.

DEFINITION 3.6bis. *We extend the partial function π_0 in such a way that:*

- $\pi_0(a_1 \dots a_n) = a_1 \cdot \ldots \cdot a_n$ *for all* $n \geq 1$ *and all* $a_1, \dots, a_n \in S$,
- $\pi_0(a\, b^\omega) = a \cdot b^\tau$ *for all* $a, b \in S$,
- $\pi_0(a^{\omega^*}\, b) = a^{\tau^*} \cdot b$ *for all* $a, b \in S$,
- $\pi_0(a\, P^\eta\, b) = a \cdot P^\kappa \cdot b$ *for all* $a, b \in S \uplus \{\varepsilon\}$
  (by a slight abuse of notation, we let $\varepsilon \cdot s = s \cdot \varepsilon = s$ *for all* $s \in S$*),*
- *in all remaining cases,* π_0 *remains undefined.*

The new definition of π_0 results in a more general notion of evaluation tree. Note that the definition of rank of an evaluation tree still applies to this generalized notion, since the rank was in fact defined on condensation trees independently of π_0. The generalized notion of evaluation tree, together with the associated rank, will give a strong enough invariant for having a proof by induction of the equivalence of evaluation trees.[2]

The lemma below basically shows that if the (extended) partial mapping π_0 is defined over a word, then it is also defined over all its factors. It is convenient here to allow also some change of letters at the extremities of the word and make some case distinctions for dealing with the empty word ε. This makes the statement of the following lemma a bit more technical.

LEMMA 3.17. *If π_0 is defined over a non-empty word of the form $u\, c\, v$, with $u, v \in S^\oplus \uplus \{\varepsilon\}$ and $c \in S \uplus \{\varepsilon\}$, then it is also defined over the words $u\, a$ and $b\, v$, for all $a, b \in S \uplus \{\varepsilon\}$ such that $a = b = \varepsilon$ implies $c = \varepsilon$. In addition, if $a = b = c = \varepsilon$ or $a \cdot b = c$, then $\pi_0(u\, c\, v) = \pi_0(u\, a) \cdot \pi_0(b\, v)$.*

The proof of the lemma is straightforward by a case distinction, and thus omitted.

The next step consists in showing how to restrict a condensation tree to an arbitrary convex subset (further along, we will lift this operation to the generalized notion of evaluation tree):

DEFINITION 3.18. *Given a condensation tree T over a linear ordering α and a convex subset I of α (not necessarily an element of T), define the* generalized subtree *of T rooted at I as follows:*

$$T|_I =^{def} \{I \cap J \mid J \in T,\ I \cap J \neq \varnothing\}.$$

The above operation can be seen as a further generalization of the notion of subtree that was given just after Definition 3.5. Below we prove that, not only $T|_I$ is a valid condensation tree, but also that this operation does not increase the rank.

LEMMA 3.19. *If T is a condensation tree over α and I is a convex subset of α, then $T|_I$ is a condensation tree over $\alpha \cap I$. Furthermore, we have* $\mathsf{rank}(T|_I) \leq \mathsf{rank}(T)$ *and $(T|_I)|_J = T|_J$ for all convex subsets J of I.*

[2]The extended definition of π_0 could have been introduced straight at the beginning, in place of Definition 3.6. Of course, all the results in the paper would still hold, but some proofs would become slightly more involved (in particular, those that show the correspondence between recognizability and MSO definability). This explains why we prefer to adopt a more restrictive definition of evaluation tree, and use the extended version only here for convenient.

PROOF. We only prove that $T|_I$ is a condensation tree. The remaining claims follow easily from our definitions. The property stated in the first item of Definition 3.5 follows from the fact that $\alpha \in T$ and $\alpha \cap I = I \in T|_I$. To prove the property in the second item, consider two convexes J, K in T. We have that either $J \subseteq K$ or $K \subseteq J$ or $J \cap K = \varnothing$. As a consequence, either $J \cap I \subseteq K \cap I$ or $K \cap I \subset J \cap I$ or $(J \cap I) \cap (K \cap I) = \varnothing$. Now, for the third item, consider two convexes J, K in T such that $K \cap I \in T|_I$ (or, equally, $K \cap I \neq \varnothing$) and $(K \cap I) \subsetneq (J \cap I)$. Since $K \cap I$ is non-empty, this means that $J \cap K$ is non-empty too. Thus, either $K \subseteq J$ or $J \subseteq K$. If $J \subseteq K$ held, then we would have $(J \cap I) \subseteq (K \cap I)$, which would contradict $(K \cap I) \subsetneq (J \cap I)$. We thus conclude that $K \subseteq J$. It remains to verify the property in the fourth item, namely, the fact that any subset of $T|_I$ that is totally ordered by inclusion is finite. Consider such a subset C. For each $J \in C$, define $T_{\supseteq J} = \{K \in T \mid K \cap I \supseteq J\}$. By construction, $T_{\supseteq J}$ is a subset of T that is totally ordered by inclusion. In particular, $T_{\supseteq J}$ is finite and has a minimal element, denoted $\mathsf{min}(T_{\supseteq J})$. We define C' as the set of all convexes of the form $\mathsf{min}(T_{\supseteq J})$, with $J \in C$. Since $J \subseteq J'$ implies $\mathsf{min}(T_{\supseteq J}) \subseteq \mathsf{min}(T_{\supseteq J'})$, we have that C' is a subset of T totally ordered by inclusion, and hence C' is finite. Moreover, since each convex $J \in C$ can be written as $\mathsf{min}(T_{\supseteq J}) \cap I$, we have that, for all $J, J' \in C$, $J \neq J'$ iff $\mathsf{min}(T_{\supseteq J}) \neq \mathsf{min}(T_{\supseteq J'})$. Since C' is finite, we conclude that C is finite too. ⊣

Putting together all the previous definitions and lemmas, we can show that an evaluation tree $\mathcal{T} = (T, \gamma)$ over a word u provides not only a value for u, but also, via restrictions to generalized subtrees, values for all the factors of u. Intuitively, this means that the mapping γ of $\mathcal{T}$ can be extended to all convex subsets I of α:

LEMMA 3.20. *For every evaluation tree $\mathcal{T} = (T, \gamma)$ over a word u with domain α and every convex subset I of α, there is an evaluation tree $\mathcal{T}|_I = (T|_I, \gamma_I)$ such that γ_I and γ coincide over $(T|_I) \cap T = \{J \in T \mid J \subseteq I\}$.*

In particular, by a slight abuse of notation, one can denote by $\gamma(I)$ the value associated with the convex I in the evaluation tree $\mathcal{T}|_I$ (this notation is consistent with the value associated with I in the evaluation tree $\mathcal{T}$, when $I \in T$).

PROOF. Let us first assume that I is an initial segment of α, namely, for every $y \in I$ and every $x \leq y$, $x \in I$. The proof is by induction on T, namely, on the rank of the underlying condensation tree. Let $C = \mathsf{children}(\alpha)$ be the top-level condensation of T. We distinguish between two subcases.

If the condensation $\{I,\ \alpha \setminus I\}$ is coarser than C, then for all $K \in T|_I$, with $K \neq I$, we have $K \in T$. Hence it makes sense to define $\gamma_I(K) = \gamma(K)$. We complete the definition by letting $\gamma_I(I) = \pi_0(\gamma(C|_I))$, where $\gamma(C|_I)$ is the word with domain $C|_I = \{K \in C \mid K \subseteq I\}$ and with each position J labelled by the value $\gamma(J)$ (thanks to Lemma 3.17 the function π_0 is defined on the word $\gamma(C|_I)$). It is easy to check that the $(T|_I, \gamma_I)$ thus defined is an evaluation tree over the factor $u|_I$ and that γ_I and γ coincide over $(T|_I) \cap T = \{K \in T \mid K \subseteq I\}$.

Otherwise, if the condensation $\{I,\ \alpha \setminus I\}$ is not coarser than C, then there exist three convex subsets $J_1 < J_2 < J_3$ of α such that (i) $\{J_1, J_2, J_3\}$ forms a partition coarser than C, (ii) $J_1 \subseteq I$, (iii) $J_3 \subseteq \alpha \setminus I$, and (iv) $J_2 \in C$ with $J_2 \cap I \neq \varnothing$ and $J_2 \setminus I \neq \varnothing$. In particular, we have that the convex $J_2 \cap I$ is included in a proper

descendant of the root of T, and hence by Lemmas 3.10 and 3.19 $\mathsf{rank}(T|_{J_2\cap I}) < \mathsf{rank}(T)$. We can thus apply the induction hypothesis to construct the evaluation tree $\mathcal{T}|_{J_2\cap I} = (T|_{J_2\cap I}, \gamma_{J_2\cap I})$. Note that for every $K \in T|_{J_1}$ with $K \neq J_1$, we have $K \in T$. Hence it makes sense to define $\gamma_I(K) = \gamma(K)$. For every $K \in T|_{J_2}$, we define $\gamma_I(K) = \gamma_{J_2\cap I}(K)$. Finally, we let $\gamma_I(I) = \pi_0\big(\gamma_I(C|_{J_1})\ \gamma_{J_2\cap I}(J_2 \cap I)\big)$ (again this is well defined thanks to Lemma 3.17). It is easy to check that the $(T|_I, \gamma_I)$ thus defined is an evaluation tree over $u|_I$ and that γ_I and γ coincide over $(T|_I) \cap T = \{K \in T \mid K \subseteq I\}$.

The proof for the symmetric case, where I is a final segment of α, is analogous.

Finally, we consider the case where I is not an initial segment, nor a final segment of α. In this case it is possible to write I as $I_1 \cap I_2$, where I_1 is an initial segment and I_2 is a final segment of α. By Lemma 3.19 we have $T|_I = (T|_{I_1})|_{I_2}$, and hence it suffices to apply twice the cases for the initial/final segment discussed above. ⊣

Now that we have set up the basic tools for reasoning on evaluation trees and their restrictions, we begin to exploit the axioms of $\oplus$-algebras to prove a series of equivalence results. The first of these results can be seen as a form of associativity rule for the function π_0, but for which equality is required to hold only when every expression is defined:

LEMMA 3.21. *For every word u of the form $\prod_{i\in\alpha} u_i$, with α countable linear ordering and $u_i \in S^\oplus$ for all $i \in \alpha$, if both $\pi_0(u)$ and $\pi_0\big(\prod_{i\in\alpha} \pi_0(u_i)\big)$ are defined, then the two values are equal.*

PROOF. We prove the lemma by a case analysis, namely, by distinguishing the order type of u (recall that, since $\pi_0(u)$ is defined, the order type of u must be either finite, ω, ω^*, η, $1+\eta$, $\eta+1$, or $1+\eta+1$). For the sake of brevity, we let $v = \prod_{i\in\alpha} \pi_0(u_i)$.

If $u = a_1 \dots a_n$ for some $a_1, \dots, a_n \in S$, then v has to be of the form $b_1 \dots b_m$, for some $m \geq 1$ and some $b_1, \dots, b_n \in S$. Since $\cdot$ is associative (see Axiom A1), we obtain $\pi_0(u) = a_1 \cdot \ldots \cdot a_n = b_1 \cdot \ldots \cdot b_m = \pi_0(v)$.

If $u = a\, e^\omega$ for some $a, e \in S$, with e idempotent, then v can be either of the form $c_1 \dots c_m$, for some $m \geq 1$ and some $c_1, \dots, c_m \in S$, or of the form $b\, f^\omega$, for some $b, f \in S$, with f idempotent. If $v = c_1 \dots c_m$, say with $m \geq 2$ (the case $m = 1$ is trivial), then we necessarily have $c_1 = a \cdot e^{n_1} = a \cdot e$ for some $n_1 \geq 0$, $c_i = e^{n_i} = e$ for all $2 \leq i < m-1$ and some $n_2, \dots, n_{m-1} \geq 1$, and $c_m = e^\tau$. Axioms A1 and A2 together imply $e \cdot e^\tau = e \cdot (e \cdot e)^\tau = (e \cdot e)^\tau = e^\tau$. We thus have $\pi_0(u) = a \cdot e^\tau = c_1 \cdot \ldots \cdot c_m = \pi_0(v)$. Otherwise, if $v = b\, d^\omega$, then, as above, we get $b = a \cdot e^{n_1} = a \cdot e$, for some $n_1 \geq 0$, and $f = e^{n_2} = e^{n_3} = \ldots = e$, for some $n_2, n_3, \ldots \geq 1$. Using Axioms A1 and A2 we finally derive $\pi_0(u) = a \cdot e^\tau = b \cdot f^\tau = \pi_0(v)$.

The case $u = e^{\omega^*}\, a$ is just symmetric to the previous case and uses Axiom A3 instead of Axiom A2.

Finally, the most interesting case is when $u = a\, P^\eta\, b$ for some non-empty set $P \subseteq S$ and some empty or singleton words $a, b \in S \uplus \{\varepsilon\}$. We further distinguish some cases depending on the form of v:

- If $v = c_1 \dots c_m$, then the proof goes by induction on m. The interesting base case is $m = 2$ (for $m = 1$ the claim holds trivially). We further distinguish

between five subcases. If the first factor u_1 has no last letter and the last factor u_2 has no first letter, then we have $c_1 = \pi_0(u_1) = a \cdot P^\kappa$ and $c_2 = \pi_0(u_2) = P^\kappa \cdot b$. Using Axiom A4, we get $\pi_0(u) = a \cdot P^\kappa \cdot b = (a \cdot P^\kappa) \cdot (P^\kappa \cdot b) = c_1 \cdot c_2 = \pi_0(v)$. If u_1 consists of a single letter, then this letter must be $a \neq \varepsilon$. Moreover, u_2 cannot have a first letter and hence, as above, we have $\pi_0(u_2) = P^\kappa \cdot b$. We thus derive $\pi_0(u) = a \cdot P^\kappa \cdot b = c_1 \cdot c_2 = \pi_0(v)$. If u_1 has a last letter, say p, but length greater than 1, then p must belong to P and u_2 has no first letter. We thus have $\pi_0(u) = a \cdot P^\kappa \cdot b = (a \cdot P^\kappa \cdot p) \cdot (P^\kappa \cdot b) = \pi_0(v)$. The cases where u_2 has length 1 and where u_2 has a first letter and length greater than 1 are symmetric. Finally, the induction for $m > 2$ is straightforward.

- If $v = ce^\omega$, then, by distinguishing some subcases as above, one verifies that $c = \pi_0(u_1)$ is either a or $a \cdot P^\kappa \cdot p$, for some $p \in P \uplus \{\varepsilon\}$, and that $e = \pi_0(u_2) = \pi_0(u_3) = \dots$ is either $P^\kappa \cdot q$ or $q \cdot P^\kappa$, for some $q \in P \uplus \{\varepsilon\}$, depending on whether u_1 has a first letter or not. Depending on the various subcases, and using Axiom A4, we derive either $\pi_0(u) = a \cdot P^\kappa = a \cdot (P^\kappa \cdot q)^\tau = \pi_0(v)$, or $\pi_0(u) = a \cdot P^\kappa = (a \cdot P^\kappa \cdot p) \cdot (P^\kappa \cdot q)^\tau = \pi_0(v)$, or $\pi_0(u) = a \cdot P^\kappa = (a \cdot P^\kappa) \cdot (q \cdot P^\kappa) = \pi_0(v)$.
- If $v = e^{\omega^*} c$, then the claim holds by symmetry with the previous case.
- If $v = c\, R^\eta\, d$ for some non-empty set $R \subseteq S$ and some empty or singleton words $c, d \in S \uplus \{\varepsilon\}$, then we prove that R is included in $P \cup (P \uplus \{\varepsilon\}) \cdot P^\kappa \cdot (P \uplus \{\varepsilon\})$. Let us treat first the case $c = d = \varepsilon$. Since v has no first nor final letter, this implies $a = b = \varepsilon$. Let us consider an element $r \in R$ and a corresponding factor u_i of u, with $i \in \alpha$, such that $\pi_0(u_i) = r$. If u_i consists of the single letter r, then we clearly have $r \in P$. Otherwise u_i has more than one letter and, depending on the existence of a first/last letter in u_i, we get one of the four possibilities $r = P^\kappa$, $r = p \cdot P^\kappa$, $r = P^\kappa \cdot q$ and $r = p \cdot P^\kappa \cdot q$, for suitable $p, q \in P$. This proves that R is included in $P \cup (P \uplus \{\varepsilon\}) \cdot P^\kappa \cdot (P \uplus \{\varepsilon\})$. Using Axiom A4 we immediately obtain $\pi_0(u) = P^\kappa = R^\kappa = \pi_0(v)$. The general case where $c, d \in S \uplus \{\varepsilon\}$, can be dealt with by using similar arguments plus Axiom A1.

⊣

Corollary 3.22. *Let u be a word with domain α such that $\pi_0(u)$ is defined and let $\mathcal{T} = (T, \gamma)$ be an evaluation tree over u. Then $\pi_0(u) = \gamma(\alpha)$.*

Proof. We prove the claim by induction on $\mathcal{T}$. If $\mathcal{T}$ consists of a single node, then this node must be a leaf and α must be a singleton leaf, and hence the claim follows immediately by definition of evaluation tree. Otherwise, let $C = \mathsf{children}(\alpha)$ be the top-level condensation. By Lemma 3.17, we know that $\pi_0(u|_I)$ is defined for all $I \in C$. We can then use the induction hypothesis on the evaluation tree $\mathcal{T}|_I$ and obtain $\pi_0(u|_I) = \gamma(I)$. Finally, using Lemma 3.21, we get $\pi_0(u) = \pi_0(\gamma(C)) = \gamma(\alpha)$. ⊣

The following series of lemmas prove equalities between the value at the root of an evaluation tree and the values induced by π_0 under different condensations of the root. We first consider finite condensations, then ω-condensations (and, by symmetry, ω^*-condensations), and finally η-condensations. The gathering of

those results will naturally entail that two evaluation trees over the same word have the same value (see Corollary 3.27).

LEMMA 3.23. *Given a word u with domain α, an evaluation tree $\mathcal{T} = (T, \gamma)$ over u, and a finite condensation $I_1 < \ldots < I_n$ of α, we have $\gamma(\alpha) = \gamma(I_1) \cdot \ldots \cdot \gamma(I_n)$.*

PROOF. The proof is by induction on $\mathcal{T}$. If $\mathcal{T}$ consists of a single leaf, then α must be a singleton and hence $n = 1$ and the claim follows trivially. Let us now consider the case where $\mathcal{T}$ has more than one node. We only prove the claim for $n = 2$ (for $n = 1$ it is obvious and for $n > 2$ it follows from a simple induction). Let $C = \mathsf{children}(\alpha)$ be the top-level condensation and let J be the unique convex subset in C that intersects both I_1 and I_2 (if C does not contain such an element, then we let $J = \varnothing$). For the sake of brevity, we define, for both $i = 1$ and $i = 2$, $C_i = \{K \in C \mid K \subseteq I_i\}$, $u_i = \prod_{K \in C_i} \gamma(K)$, and $a_i = \gamma(J \cap I_i)$ (with the convention that $\gamma(J \cap I_i) = \varepsilon$ if $J = \varnothing$). Note that $C = C_1 \cup \{J\} \cup C_2$ if $J \neq \varnothing$ (resp., $C = C_1 \cup C_2$ if $J = \varnothing$) and hence $\gamma(\alpha) = \pi_0\big(u_1 \gamma(J) u_2\big)$ (we assume that $\gamma(J) = \varepsilon$ if $J = \varnothing$). Let us consider the case where J is not empty (the case $J = \varnothing$ is similar). Since $J \in C$ and $C = \mathsf{children}(\alpha)$, we have $\mathsf{rank}(\mathcal{T}|_J) < \mathsf{rank}(\mathcal{T})$ and hence we can apply the induction hypothesis to the evaluation tree $\mathcal{T}|_J$ and the condensation $\{J \cap I_1, J \cap I_2\}$ of $\alpha|_J$. We thus obtain $\gamma(J) = \gamma(J \cap I_1) \cdot \gamma(J \cap I_2) = a_1 \cdot a_2$ and hence $\gamma(\alpha) = \pi_0\big(u_1 \, (a_1 \cdot a_2) \, u_2\big)$. Lemma 3.17 then implies $\pi_0\big(u_1 \, (a_1 \cdot a_2) \, u_2\big) = \pi_0(u_1 \, a_1) \cdot \pi_0(u_2 \, a_2)$. Similarly, Lemma 3.21 implies $\pi_0(u_1 \, a_1) = \gamma(I_1)$ and $\pi_0(u_2 \, a_2) = \gamma(I_2)$. Overall, we get $\pi_0(\alpha) = \gamma(I_1) \cdot \gamma(I_2)$. ⊣

LEMMA 3.24. *Given a word u with domain α, an evaluation tree $\mathcal{T} = (T, \gamma)$ over u, and an ω-condensation $I_0 < I_1 < I_2 < \ldots$ of α such that $\gamma(I_1) = \gamma(I_2) = \ldots$ is an idempotent, we have $\gamma(\alpha) = \gamma(I_0) \cdot \gamma(I_1)^\tau$.*

PROOF. The proof is again by induction on $\mathcal{T}$. Note that the case of $\mathcal{T}$ consisting of a single leaf cannot happen. Let $C = \mathsf{children}(\alpha)$ be the top-level condensation. We distinguish two cases depending on whether C has a maximal element or not.

Suppose that C has a maximal element, say J_{max}, and $C \neq \{J_{\mathsf{max}}\}$ (the case where $C = \{J_{\mathsf{max}}\}$ can be considered as a degenerate case, which can be dealt with by similar arguments). We can find a condensation $K_1 < K_2$ of α that is coarser than $I_0 < I_1 < I_2 < \ldots$ and such that $K_2 \subseteq J_{\mathsf{max}}$. By Lemma 3.23, we have $\gamma(\alpha) = \gamma(K_1) \cdot \gamma(K_2)$. Moreover, since K_1 is the union of a finite sequence of convex subsets $I_0, I_1, \ldots, I_k$, by repeatedly applying Lemma 3.21, we obtain $\gamma(K_1) = \gamma(I_0) \cdot \gamma(I_1) \cdot \ldots \cdot \gamma(I_k) = \gamma(I_0) \cdot \gamma(I_1)$ (the last equality follows from the fact that $\gamma(I_1) = \gamma(I_2) = \ldots$ is an idempotent). Finally, from the induction hypothesis (note that $\mathsf{rank}(\mathcal{T}|_{K_2}) < \mathsf{rank}(\mathcal{T})$), we get $\gamma(K_2) = \gamma(I_1)^\tau$. We thus conclude that $\gamma(\alpha) = \big(\gamma(I_0) \cdot \gamma(I_1)\big) \cdot \big(\gamma(I_1)^\tau\big) = \gamma(I_0) \cdot \gamma(I_1)^\tau$.

If C has no maximal element, then, using standard techniques and Ramsey's Theorem (Lemma 3.15), one can construct an ω-condensation $J_0 < K_1 < J_1 < K_2 < J_2 < \ldots$ of α such that:

- $\{J_0 \cup K_1, J_1 \cup K_2, \ldots\}$ is coarser than $\{I_0, I_1, I_2, \ldots\}$,

- $\{J_0, K_1 \cup J_1, K_2 \cup J_2, \dots\}$ is coarser than C,
- $\gamma(K_1 \cup J_1) = \gamma(K_2 \cup J_2) = \dots$ is an idempotent.

Let $\gamma(C)$ be the word with domain C where each position $H \in C$ is labelled by the value $\gamma(H)$. By construction, we have $\gamma(\alpha) = \pi_0(\gamma(C))$. Moreover, since the condensation $\{J_0, K_1 \cup J_1, K_2 \cup J_2, \dots\}$ is coarser than C, by repeatedly applying Lemma 3.21, we obtain $\pi_0(\gamma(C)) = \pi_0\big(\gamma(J_0)\ \gamma(K_1 \cup J_1)\ \gamma(K_2 \cup J_2)\ \dots\big) = \gamma(J_0) \cdot \gamma(K_1 \cup J_1)^\tau$. Similarly, since $\{J_0 \cup K_1, J_1 \cup K_2, \dots\}$ is coarser than $\{I_0, I_1, I_2, \dots\}$ and $\gamma(I_1) = \gamma(I_2) = \dots$ is an idempotent, we have $\gamma(J_0 \cup K_1) = \gamma(I_0) \cdot \gamma(I_1)$ and $\gamma(J_1 \cup K_2) = \gamma(J_2 \cup K_3) = \dots = \gamma(I_1)$. Thus, by Axioms A1 and A2, we obtain $\gamma(J_0) \cdot \gamma(K_1 \cup J_1)^\tau = \gamma(I_0) \cdot \gamma(I_1)^\tau$. ⊣

We can gather all the results seen so far and prove the following corollary (recall that an ordering is scattered if all dense suborderings of it are empty or singletons):

COROLLARY 3.25. *Given a word u with domain α, an evaluation tree $\mathcal{T} = (T, \gamma)$ over u, a scattered condensation C of α, and an evaluation tree $\mathcal{T}' = (T', \gamma')$ over the word $\gamma(C) = \prod_{I \in C} \gamma(I)$ with domain C, we have $\gamma(\alpha) = \gamma'(C)$.*

PROOF. As a preliminary remark, note that since the condensation C is scattered, we have that, for every node J in the evaluation tree $\mathcal{T}' = (T', \gamma')$, the condensation of J induced by $\mathcal{T}'$ is scattered as well. The proof is by induction on $\mathcal{T}'$. If $\mathcal{T}'$ consists of a single node, then $\gamma(C)$ is a singleton word of value $\gamma(\alpha)$ and hence the statement boils down to $\gamma(\alpha) = \gamma(\alpha)$. Otherwise, let D be the childhood of the root C of $\mathcal{T}'$. From the induction hypothesis, we know that for every $J \in D$, $\gamma'(J) = \gamma\big(\bigcup J\big)$, where $\bigcup J$ denotes the union of all convex subsets of J (recall that $J \subseteq C$). Moreover, if we denote by $\bigcup D$ the condensation of α obtained from the substitution of each element $J \in D$ by $\bigcup J$, we have

$$\gamma'(C) = \pi_0\left(\prod_{J \in D} \gamma'(J)\right) = \pi_0\left(\prod_{J \in D} \gamma\big(\bigcup J\big)\right) = \pi_0\Big(\gamma\big(\bigcup D\big)\Big).$$

Note that the condensation $\bigcup D$ of α has the same order type of the condensation D of C, namely, it is either a finite condensation, an ω-condensation, or an ω^*-condensation. Therefore, using either Lemma 3.23 or Lemma 3.24 (or its symmetric variant), we obtain $\pi_0\big(\gamma(\bigcup D)\big) = \gamma(\alpha)$. ⊣

It remains to consider the case of dense condensations, which give rise to η-shuffles:

LEMMA 3.26. *Given a word u with domain α, an evaluation tree $\mathcal{T} = (T, \gamma)$ over u, and a dense condensation C of α such that $\gamma(C) = \prod_{I \in C} \gamma(I)$ is isomorphic to a word of the form $a\, P^\eta\, b$, for some elements $a, b \in S \uplus \{\varepsilon\}$ and some non-empty set $P \subseteq S$, we have $\gamma(\alpha) = a \cdot P^\kappa \cdot b$.*

PROOF. We remark here that the proof works for any condensation C, independently of the form of the word $\gamma(C)$. However, the use of the following technical arguments does only make sense when C is a dense condensation. We prove the lemma by induction on $\mathcal{T}$. As in the proof of Lemma 3.24, the case of $\mathcal{T}$ consisting of a single node cannot happen. Let $D = \mathsf{children}(\alpha)$ be the top-level condensation and let E be the finest condensation that is coarser than

or equal to both C and D (note that E exists since condensations form a lattice structure with respect to the 'coarser than' relation). Moreover, let $\sim$ be the condensation over the condensed ordering C such that, for every $I, I' \in C$, $I \sim I'$ holds iff either $I = I'$ or there is $J \in D$ with $I \subseteq J$ and $I' \subseteq J$. This can naturally be seen as a condensation C' over α which is at least as coarse as C: the classes of C' are either the single classes of C that are not contained in any class of D, or the unions of the classes of C that are contained in the same class of D. Furthermore, it is easy to see that E is at least as coarse as C'. Below, we disclose further properties of the condensations C, D, E, and C'.

Let us consider a class $I \in C'$. Two cases can happen: either I is included in some $J \in D$, and in this case $\gamma(I) = \pi_0\big(\gamma(C|_I)\big)$ holds thanks to the induction hypothesis, or I belongs to C, and hence $\gamma(I) = \pi_0\big(\gamma(C|_I)\big)$ follows trivially. We have just proved that

$$\forall I \in C' \quad \gamma(I) \;=\; \pi_0\big(\gamma(C|_I)\big). \tag{1}$$

Now, let I, I' be two distinct classes in C'. We claim that there exist $x \in I$ and $x' \in I'$ that are not equivalent for D, namely,

$$\exists x \in I \exists x' \in I' \forall J \in D \quad x \notin J \vee x' \notin J. \tag{2}$$

The proof of this property is by case distinction. If I is contained in some $J \in D$ and I' is contained in some $J' \in D$, then we necessarily have $J \neq J'$ (otherwise, we would have $I = I'$ by definition of C') and hence Property (2) holds. Otherwise, either I is not contained in any class $J \in D$, or I' is not contained in any class $J \in D$. Without loss of generality, we assume that I is not contained in any class $J \in D$. This means that there exists $J \in D$ such that $I \cap J \neq \varnothing$ and $I \setminus J \neq \varnothing$. Let us pick some $x' \in I'$. Clearly, x' belongs to some $J' \in D$. Then either $J \cap J' = \varnothing$ or $J = J'$. In the first case, one chooses $x \in I \cap J$, while in the second case one chooses $x \in I \setminus J$. This completes the proof of Property (2).

From the above property, we can deduce the following:

$$\begin{aligned}&\textit{If } I, I' \in C',\ I < I',\ \textit{and } I, I' \subseteq K \textit{ for some } K \in E, \textit{ then}\\ &\textit{there are only finitely many classes } I'' \in C' \textit{ between } I \textit{ and } I'.\end{aligned} \tag{3}$$

Indeed, suppose that the above property does not hold, namely, that there are infinitely many classes $I'' \in C'$ between I and I'. In particular, we can find an ω-sequence of classes $I_1, I_2, \ldots$ such that $I = I_1 < I_2 < \ldots < I'$ or $I < \ldots < I_2 < I_1 = I'$. We only consider the first case (the second case is symmetric). By applying Property (2) to the classes $I_1, I_2, \ldots$, we can find some points $x_1 \in I_1$, $x'_1 \in I_2$, $x_2 \in I_3$, $x'_2 \in I_4$, ... such that, for all $i \geq 1$, x_i and x'_i are not equivalent for D (i.e., for all $J \in D$, $x_i \notin J$ or $x'_i \notin J$). Let X be the set of all points $x \in \alpha$, with $x < I_i$ for some $i \geq 1$, and let X' be the set of all points $x' \in \alpha$, with $x' > I_j$ for all $j \geq 1$. Since D is a condensation, we have that for all $x \in X$ and all $x' \in X'$, x and x' are not equivalent for D. Moreover, by construction, all such points x and x' are not equivalent for C', and hence neither for C (recall that C is finer than C'). Since E is the defined as the finest condensation that is coarser than or equal to both C and D and since $X \cup X' = \alpha$, it follows that there is no class $K \in E$ that intersects both X and X'. In particular, since $I \subseteq X$ and $I' \subseteq X'$,

it follows that there is no class $K \in E$ such that $I \subseteq K$ and $I' \subseteq K$, which is a contradiction. This completes the proof of Property (3).

We prove the following last property:

$$\forall K \in E \quad \gamma(K) = \pi_0\big(\gamma(C|_K)\big). \tag{4}$$

Let $K \in E$ and let $\mathcal{T}' = (T', \gamma')$ be an evaluation tree over the word $\gamma(C'|_K)$ (such a tree exists according to Proposition 3.8). From Property (3) we know that the condensation of $C'|_K$ induced by the evaluation tree $\mathcal{T}'$ is scattered. We can thus apply Corollary 3.25 and obtain $\gamma(K) = \gamma'(C'|_K)$. Moreover, the value $\pi_0\big(\gamma(C'|_K)\big)$ is defined and hence, by Corollary 3.22, $\gamma'(C'|_K) = \pi_0\big(\gamma(C'|_K)\big)$. By Property (1), we obtain $\gamma(C'|_K) = \prod_{I \in C'|_K} \gamma(I) = \prod_{I \in C'|_K} \pi_0\big(\gamma(C|_I)\big)$. Finally, from the properties of condensation trees, we derive $\pi_0\big(\gamma(C'|_K)\big) = \pi_0\big(\prod_{I \in C'|_K} \pi_0(\gamma(C|_I))\big) = \gamma(C|_K)$. This completes the proof of Property (4).

Towards a conclusion, we consider an evaluation tree $\mathcal{T}'' = (T'', \gamma'')$ over the word $\gamma(E)$ (such a tree exists thanks to Proposition 3.8). From Property (4) we know that $\gamma(E) = \prod_{K \in E} \gamma(K) = \prod_{K \in E} \pi_0\big(\gamma(C|_K)\big)$. Moreover, By Corollary 3.22, we know that $\pi_0\big(\gamma(C|_K)\big) = \gamma''(K)$ and hence $\prod_{K \in E} \pi_0\big(\gamma(C|_K)\big) = \gamma''(E)$. Similarly, since E is at least as coarse as D, Corollary 3.22 implies $\gamma''(E) = \pi_0\big(\gamma(D)\big) = \gamma(\alpha)$. This completes the proof of the lemma. ⊣

COROLLARY 3.27. *Given a word u with domain α, an evaluation tree $\mathcal{T} = (T, \gamma)$ over u, a condensation C of α, and an evaluation tree $\mathcal{T}' = (T', \gamma')$ over the word $\gamma(C) = \prod_{I \in C} \gamma(I)$ with domain C, we have $\gamma(\alpha) = \gamma'(C)$.*

PROOF. The proof is exactly the same as for Corollary 3.25, with the only difference that we do not use the assumption that the condensation C is scattered and we use Lemma 3.26 for treating the nodes I' of $\mathcal{T}'$ for which the condensation $\mathsf{children}(I')$ is dense. ⊣

Finally, Proposition 3.9 follows easily from the previous corollary.

PROOF OF PROPOSITION 3.9. Let $\mathcal{T} = (T, \gamma)$ and $\mathcal{T}' = (T', \gamma')$ be two evaluation trees over the same word u with domain α and let C be the finest condensation of α, whose classes are the singleton sets. Clearly, the evaluation tree $\mathcal{T}'$ is isomorphic to an evaluation tree $\mathcal{T}'' = (T'', \gamma'')$ over the word $\gamma(C) = \prod_{I \in C} \gamma(I)$ with domain C. Using Corollary 3.27 we immediately obtain that $\gamma(\alpha) = \gamma''(C) = \gamma'(\alpha)$. ⊣

§4. From monadic second-order logic to $\circledast$-algebras. Let us recall that *monadic second-order* (MSO) logic is the extension of first-order logic with set quantifiers. We assume the reader to have some familiarity with this logic, as well as with the technique used by Büchi to translate MSO formulas into equivalent automata. A good survey can be found in [27].

Here, we show a relatively direct consequence of the results obtained in the previous section, namely, that MSO formulas can be effectively translated to $\circledast$-algebras:

THEOREM 4.1. *The MSO definable $\circledast$-languages are effectively recognizable.*

Before turning to the proof of the above result, let us remark that we could have equally well used the composition method of Shelah for establishing Theorem 4.1. Indeed, given any MSO sentence ψ, one can construct effectively a $\circledast$-algebra recognizing the language defined by ψ [24].

Our proof of Theorem 4.1 follows Büchi's approach, namely, we establish a number of closure properties for recognizable $\circledast$-languages. Then, each construction of the logic will be translated into an operation on languages. To disjunction corresponds union, to conjunction corresponds intersection, to negation corresponds complementation, etc. We assume the reader to be familiar with this approach (in particular the coding of the valuations of free variables).

The $\circledast$-languages corresponding to the atomic predicates are easily shown to be recognizable. Similarly, the language operations of intersection, union, and complementation can be implemented easily by means of classical algebraic operations:

LEMMA 4.2. *The recognizable $\circledast$-languages are effectively closed under intersection, union, and complementation.*

PROOF. Given two $\circledast$-monoids (M_1, π_1) and (M_2, π_2) recognizing the languages $L_1 = h_1^{-1}(F_1)$ and $L_2 = h_2^{-1}(F_2)$, respectively, with $F_1 \subseteq M_1$, $F_2 \subseteq M_2$, and h_1 and h_2 morphisms to (M_1, π_1) and (M_2, π_2), respectively, we have that $A^{\circledast} \setminus L_1 = h_1^{-1}(M_1 \setminus F_1)$, $L_1 \cap L_2 = (h_1 \times h_2)^{-1}(F_1 \times F_2)$, and $L_1 \cup L_2 = (h_1 \times h_2)^{-1}\big((M_1 \times M_2) \setminus (F_1 \times F_2)\big)$. In particular, the complement of L_1 is recognized by (M_1, π_1), while the union and the intersection of L_1 and L_2 are recognized by the product $\circledast$-monoid $(M_1 \times M_2, \pi_1 \times \pi_2)$. Moreover, the latter product can be easily implemented at the level of $\circledast$-algebras: the operators of a $\circledast$-algebra that corresponds to $(M_1 \times M_2, \pi_1 \times \pi_2)$ can be obtained by applying component-wise the operators of some $\circledast$-algebras that correspond to (M_1, π_1) and (M_2, π_2). ⊣

What remains to be proved is the closure under projection. Formally, given a language L over some alphabet A, and a mapping f from A to another alphabet B, the *projection* of L via f is the language $f(L)$, where f is extended in a pointwise manner to words and languages. The logical operation of existential quantification corresponds, at the level of the defined languages, to a projection. Hence, it remains to prove the following:

LEMMA 4.3. *The recognizable $\circledast$-languages are effectively closed under projections.*

PROOF. We first describe the construction for a given $\circledast$-monoid (M, π), and then show how to adapt the construction at the level of $\circledast$-algebras. The projection is implemented, as usual, by a powerset construction, namely, by providing the definition of a generalized product over $\mathscr{P}(M)$. Given two words u and U over M and $\mathscr{P}(M)$, respectively, we write $u \in U$ if $\mathsf{dom}(u) = \mathsf{dom}(U)$ and $u(x) \in U(x)$ for all $x \in \mathsf{dom}(U)$. We then define the mapping $\tilde{\pi}$ from $\mathscr{P}(M)^{\circledast}$ to $\mathscr{P}(M)$ by letting

$$\tilde{\pi}(U) \ =^{\text{def}} \ \big\{\pi(u) \,\big|\, u \in U\big\} \qquad \text{for all } U \in \mathscr{P}(M)^{\circledast} \ .$$

Let us show that $\tilde{\pi}$ is associative. Consider a word U over $\mathscr{P}(M)$ and a condensation $\sim$ of its domain. Then,

$$\begin{aligned}\tilde{\pi}(U) &= \big\{\pi(u) \,\big|\, u \in U\big\}\\ &= \big\{\pi\big(\textstyle\prod_{I\in\alpha/_\sim} \pi(u|_I)\big) \,\big|\, u \in U\big\}\\ &= \big\{\pi\big(\textstyle\prod_{I\in\alpha/_\sim} a_I\big) \,\big|\, a_I \in \tilde{\pi}(U|_I) \text{ for all } I \in \alpha/_\sim\big\}\\ &= \tilde{\pi}\big(\textstyle\prod_{I\in\alpha/_\sim} \tilde{\pi}(U|_I)\big)\ ,\end{aligned}$$

where the second equality is derived from the associativity of π. Hence $(\mathscr{P}(M),\tilde{\pi})$ is a $\circledast$-monoid.

Next, we show that $(\mathscr{P}(M),\tilde{\pi})$ recognizes any projection of a language recognized by (M,π). Let let $L \subseteq A^{\circledast}$ be a language recognized by (M,π) via some morphism $h : (A,\prod) \to (M,\pi),$, namely, $L = h^{-1}(h(L))$, and let $f : A \to B$ be a projection. We claim that the projected language $L' = f(L)$ is recognized by $(\mathscr{P}(M),\tilde{\pi})$ via the morphism $g = h \circ f^{-1} : (B,\prod) \to (\mathscr{P}(M),\tilde{\pi})$. Clearly, we have $g^{-1}(g(L')) \supseteq L'$. For the opposite containment, consider a word $v \in g^{-1}(g(L'))$. By construction, there is a word $v' \in L'$ such that $g(v') = g(v)$. Since $v' \in L' = f(L)$, there is $w' \in L$ such that $v' = f(w')$. Moreover, since $g(v') = g(v)$, there is w such that $f(w) = v$ and $h(w') = h(w)$. Finally, since $L = h^{-1}(h(L))$, we conclude that $w \in L$, and hence $v = f(w) \in L'$.

Thanks to Lemma 3.4 and Corollary 3.12, the construction of $(\mathscr{P}(M),\tilde{\pi})$ can be performed at the level of $\circledast$-algebras. More precisely, any $\circledast$-algebra $(M,1,\cdot,\tau,\tau^*,\kappa)$ uniquely determines a $\circledast$-monoid (M,π), and from this, using the powerset construction, one defines the $\circledast$-monoid $(\mathscr{P}(M),\tilde{\pi})$, and finally the induced $\circledast$-algebra $(\mathscr{P}(M),\{1\},\tilde{\cdot},\tilde{\tau},\tilde{\tau}^*,\tilde{\kappa})$. The crux in this line of arguments is that the correspondence between the original $\circledast$-algebra $(M,1,\cdot,\tau,\tau^*,\kappa)$ and the final $\circledast$-algebra $(\mathscr{P}(M),\{1\},\tilde{\cdot},\tilde{\tau},\tilde{\tau}^*,\tilde{\kappa})$ may be, a priori, not effective. Below we explain why, in fact, this correspondence is effective, namely, we explain how each operator of the $\circledast$-algebra $(\mathscr{P}(M),\{1\},\tilde{\cdot},\tilde{\tau},\tilde{\tau}^*,\tilde{\kappa})$ can be computed using the initial $\circledast$-algebra $(M,1,\cdot,\tau,\tau^*,\kappa)$ and some saturation process.

We give the intuition for constructing the most difficult and interesting operator $\tilde{\kappa}$, that is, for computing $P^{\tilde{\kappa}} = \tilde{\pi}(P^\eta)$ for any given non-empty subset $P = \{A_1,\ldots,A_k\}$ of $\mathscr{P}(M)$, using the operators of the $\circledast$-algebra $(M,1,\cdot,\tau,\tau^*,\kappa)$. We recall that $P^{\tilde{\kappa}} = \{1\}$ if $A_1 = \ldots = A_k = \{1\}$, otherwise $P^{\tilde{\kappa}} = \big(P \smallsetminus \{1\}\big)^{\tilde{\kappa}}$. We also recall that $P^{\tilde{\kappa}}$ must represent the set $\{\pi(u) \mid u \in U,\ U \in P^\eta\}$ and hence the computation of $P^{\tilde{\kappa}}$ is very similar to that of $\{\pi(u) \mid u \in A^{\circledast}\}$, which was done in the proof of Theorem 3.13. The difference here is that one needs to relativise u to the words that belong to U, for some $U \in P^\eta$. This can be achieved by performing a product of the $\circledast$-algebra $(M,1,\cdot,\tau,\tau^*,\kappa)$ with a $\circledast$-algebra that recognizes the single-word language $\{P^\eta\}$, and then applying the saturation process of Theorem 3.13 on the resulting $\circledast$-algebra. ⊣

§5. From $\circledast$-algebras to monadic second-order logic. We have seen in the previous section that every MSO formula defines a recognizable $\circledast$-language. In this section, we prove the converse. Hereafter, we refer to the $\forall$-*fragment* (resp., $\exists$-*fragment*) of MSO logic as the set of formulas that start with a block

of universal (resp., existential) set quantifiers, followed by a first-order formula. Similarly, the $\exists\forall$-*fragment* consists of formulas starting with a block of existential set quantifiers followed by a formula of the $\forall$-fragment.

THEOREM 5.1. *The recognizable $\circledast$-languages are effectively MSO definable. Furthermore, such languages are definable in the $\exists\forall$-fragment of MSO logic.*

We fix for the remaining of the section a morphism h from $(A^{\circledast}, \prod)$ to a $\circledast$-monoid (M, π), with M finite, and a subset F of M. Let also $1, \cdot, \tau, \tau^*, \kappa$ be defined from π. Our goal is to show that $L = h^{-1}(F)$ is MSO definable. It is sufficient for this to show that for every $a \in M$, the language

$$\pi^{-1}(a) \;=\; \left\{ w \in M^{\circledast} \;:\; \pi(w) = a \right\},$$

can be defined by a suitable MSO sentence $\varphi_a^{\mathsf{value}}$. From this it will follow that that $L = \bigcup_{a \in F} h^{-1}(a)$ is defined by the disjunction $\bigvee_{a \in F} \hat{\varphi}_a^{\mathsf{value}}$, where $\hat{\varphi}_a^{\mathsf{value}}$ is obtained from $\varphi_a^{\mathsf{value}}$ by replacing every occurrence of an atom $b(x)$, with $b \in M$, by $\bigvee_{c \in h^{-1}(b) \cap A} c(x)$.

A reasonable approach for defining $\pi^{-1}(a)$ is to use a formula which, given $u \in M^{\circledast}$, guesses some object that 'witnesses' $\pi(u) = a$. The only objects that we have seen so far and that are able to "witness" $\pi(u) = a$ are evaluation trees. Unfortunately, there is no way an MSO formula can guess an evaluation tree, since their height cannot be bounded uniformly. That is why we use another kind of object for witnessing $\pi(u) = a$: the so-called Ramseian split, which is introduced just below.

5.1. Ramseian splits. Ramseian splits are not directly applied to words, but to additive labellings. Recall that an additive labelling σ from a linear ordering α to a semigroup $(M, \cdot)$ (which, in our case, will be induced by the $\circledast$-monoid (M, π)) is a function that maps any pair of elements $x < y$ from α to an element $\sigma(x, y) \in M$ in such a way that $\sigma(x, y) \cdot \sigma(y, z) = \sigma(x, z)$ for all $x < y < z$ in α.

Given two positions $x < y$ in a word u, denote by $[x, y)$ the interval $\{z \mid x \leq z < y\}$. Given a word u and two positions $x < y$ in it, we define $\sigma_u(x, y)$ to be the element $\pi\big(u|_{[x,y)}\big)$ of the $\circledast$-monoid (M, π). Quite naturally, σ_u is an additive labelling, since for all $x < y < z$, we have $\sigma_u(x, y) \cdot \sigma_u(y, z) = \pi\big(u|_{[x,y)}\big) \cdot \pi\big(u|_{[y,z)}\big) = \pi\big(u|_{[x,y)}\, w|_{[y,z)}\big) = \pi\big(u|_{[x,z)}\big) = \sigma_u(x, z)$.

DEFINITION 5.2. *A* split *of height n of a linear ordering α is a function $g : \alpha \to \{1, \ldots, n\}$. Two elements $x, y \in \alpha$ are called* (k-)neighbours *iff $g(x) = g(y) = k$ and $g(z) \leq k$ for all $z \in \alpha|_{[x,y] \cup [y,x]}$ (note that the neighbourhood relation is an equivalence). The split g is said to be* Ramseian *for an additive labelling $\sigma : \alpha \to M$ iff for all equivalence classes $X \subseteq \alpha$ of the neighbourhood relation, there is an idempotent $e \in M$ such that $\sigma(x, y) = e$ for all $x < y$ in X.*

THEOREM 5.3 (Colcombet [8]). *For every finite semigroup $(M, \cdot)$, every linear ordering α, and every additive labelling σ from α to $(M, \cdot)$, there is a split of α which is Ramseian for σ and which has height at most $2|M|$.*

5.2. Inductive construction of formulas. Below we construct a formula that, given a word u of domain α, guesses a split of α of height at most $2|M|$, and uses it for representing the function that associates with each convex subset I of α the value $\pi(u|_I)$ in M. For the sake of simplicity, we fix a word u of domain α and the corresponding additive labelling σ_u over α that is induced by u. We remark, however, that all constructions that follow are uniform and do not depend on the chosen word u.

In the following, we make extensive use of properties, functions, sets that are first-order definable from other parameters. For instance, when we say that a set X is *first-order definable* from some variables $\bar{Y}$, we mean that there exists a first-order formula $\xi(x, \bar{Y})$ that describes the membership of x in X on the basis of $\bar{Y}$, that is, $x \in X$ iff $\xi(x, \bar{Y})$ holds on the given interpretation of x and $\bar{Y}$. In practice, this means that it is never necessary to quantify over X for defining properties concerning X: it is sufficient to replace each predicate $x \in X$ by the corresponding formula $\xi(x, \bar{Y})$. This remark is crucial for understanding why the construction we provide yields a formula in the $\exists\forall$-fragment of MSO logic.

Recall that we aim at constructing, for each $a \in M$, a sentence $\varphi_a^{\mathsf{value}}$ that holds over the word u iff $\pi(u) = a$. The starting point is to guess:

1. a split g of α of height at most $2|M|$, and;
2. a function f mapping each position $x \in \alpha$ to an idempotent $f(x) \in M$.

The intention is that a choice of g and f is good when g is a Ramseian split for σ_u and the function f maps each position x to the idempotent $f(x)$ that arises when the neighbourhood class of x is considered (cf. Definition 5.2). In this a case, by a slight abuse of terminology, we say that (g, f) is a *Ramseian pair*.

Observe that neither g nor f can be represented by a single monadic variable. However, since both g and f are functions from α to sets of bounded size ($2|M|$ for g, and $|M|$ for f), one can guess them using a fixed number of monadic variables. This kind of encoding is quite standard, and from now on we shall use explicitly the mappings g and f in our formulas, rather than their encodings.

Knowing a Ramseian pair (g, f) is an advance towards computing the value of a word. Indeed, Ramseian splits can be used as "accelerating structures" in the sense that every computation of $\pi(u|_I)$ for some convex I becomes significantly easier when a Ramseian split is known, namely, it becomes first-order definable in terms of the Ramseian split. This is formalized by the following lemma.

LEMMA 5.4. *Given $a \in M$, one can construct a first-order formula $\mathsf{eval}_a(g, f, X)$ such that for every convex subset I of α:*

- *if (g, f) is Ramseian, then $\mathsf{eval}_a(g, f, I)$ holds iff $\pi(u|_I) = a$,*
- *if both $\mathsf{eval}_a(g, f, I)$ and $\mathsf{eval}_b(g, f, I)$ hold, then $a = b$.*

PROOF. As already mentioned, we encode both functions g and f by tuples of monadic predicates. This allows us to use shorthands such as $g(x) = k$, where x is a first-order variable and $1 \leq k \leq 2|M|$, for claiming that the point x of the underlying word u is mapped via g to the number k. Similarly, we encode the convex subset I of α by a monadic predicate and we write $x \in I$ as a shorthand for a formula that states that the point x belongs to I.

We assume from now that (g, f) is Ramseian. Under this assumption, it will be clear that the constructed formulas will satisfy the desired properties. We remark, however, that the following definitions make sense also in the case when (g, f) is not Ramseian, in which case only the second condition of the lemma will be guaranteed.

Given a convex I, we denote by $\mathsf{level}(g, I)$ the maximal value of $g(x)$ for x ranging over I. Of course, the properties $\mathsf{level}(g, I) = k$ and $\mathsf{level}(g, I) \leq k$ are first-order definable in terms of g and I.

We will construct by induction on $k \in \{0, 1, \ldots, 2|M|\}$ a *partial* function eval^k that maps some triples (g, f, I) to elements $\mathsf{eval}^k(g, f, I) \in M$ in such a way that the following properties hold:

- $\mathsf{eval}^k(g, f, I) = a$ is definable by a first-order formula, say $\mathsf{eval}^k_a(g, f, I)$, for each $a \in M$,
- $\mathsf{eval}^k(g, f, I)$ is defined iff $\mathsf{level}(g, I) \leq k$, and in this case it coincides with $\pi(u|_I)$ (provided (g, f) is Ramseian).

The base case is when $k = 0$. In this case, we define $\mathsf{eval}^k(g, f, I)$ to be the neutral element 1 when $I = \varnothing$, and we let $\mathsf{eval}^k(g, f, I)$ be undefined when $I \neq \varnothing$. Of course, this is first-order definable and satisfies the expected induction hypothesis.

Let us now construct the partial function $\mathsf{eval}^k(g, f, I)$ for any $k \geq 1$. First, if $\mathsf{level}(g, I) < k$, then one simply outputs $\mathsf{eval}^{k-1}(g, f, I)$. Otherwise, the convex subset I can be uniquely partitioned into $X < J < Y$ in such a way that $X \cup J \cup Y = I$ and J is the minimal convex subset containing $I \cap g^{-1}(k)$. Note that the sets X, J, and Y are first-order definable in the parameters I and g, that is, membership of any point x in X (resp., J, Y) is characterized by a first-order formula in the variables x, I, and g. Furthermore, fix e to be $f(x)$ for some $x \in I \cap g^{-1}(k)$. From the assumption that I has level k for g, we know that all elements in $I \cap g^{-1}(k)$ are neighbours. In particular, the fact that g is a Ramseian split for σ_u means that $\sigma_u(x, y) = e$ for all $x < y$ chosen in $I \cap g^{-1}(k)$. The mapping $\mathsf{eval}^k(g, f, I)$ is defined below by a case distinction (we remark that the following definitions are not symmetric with respect to the underlying order, and this reflects the asymmetry occurring in the definition of σ_u, that is, $\sigma_u(x, y) = \pi(u|_{[x,y)})$ for all $x < y \in \alpha$):

1. if J is a singleton $\{x\}$, then
$$\mathsf{eval}^k(g, f, I) \;=\; \mathsf{eval}^{k-1}(g, f, X) \cdot u(x) \cdot \mathsf{eval}^{k-1}(g, f, Y) \;,$$
2. if J has distinct minimal and maximal elements and $y = \mathsf{max}(J)$, then
$$\mathsf{eval}^k(g, f, I) \;=\; \mathsf{eval}^{k-1}(g, f, X) \cdot e \cdot u(y) \cdot \mathsf{eval}^{k-1}(g, f, Y) \;,$$
3. if J has no minimal element but has a maximal element y, then
$$\mathsf{eval}^k(g, f, I) \;=\; \mathsf{eval}^{k-1}(g, f, X) \cdot e^{\tau^*} \cdot u(y) \cdot \mathsf{eval}^{k-1}(g, f, Y) \;,$$
4. if J has a minimal element but no maximal element, then
$$\mathsf{eval}^k(g, f, I) \;=\; \mathsf{eval}^{k-1}(g, f, X) \cdot e^{\tau} \cdot \mathsf{eval}^{k-1}(g, f, Y) \;,$$

5. if J has no minimal element and no maximal element, then

$$\mathsf{eval}^k(g,f,I) \;=\; \mathsf{eval}^{k-1}(g,f,X) \;\cdot\; e^{\tau^*}\cdot e^{\tau} \;\cdot\; \mathsf{eval}^{k-1}(g,f,Y)\ .$$

One easily checks that the function eval^k can be defined by first-order formulas of the form $\mathsf{eval}^k_a(g,f,I)$, with $a \in M$. It is also easy to see that if (g,f) is Ramseian and $\mathsf{level}(g,I) \leq k$, then $\mathsf{eval}^k(g,f,I)$ coincides with $\pi(u|_I)$.

At this step, the first conclusion of the lemma is already satisfied by the first-order formulas $\mathsf{eval}^{2|M|}_a(g,f,I)$. The second point, however, is false in general. Indeed, we did not pay attention so far on what the formulas compute in the case where (g,f) is not Ramseian. In particular, it can happen that both formulas $\mathsf{eval}^{2|M|}_a(g,f,I)$ and $\mathsf{eval}^{2|M|}_b(g,f,I)$ hold for distinct elements $a,b \in M$. However, this can be easily fixed using the following formula:

$$\mathsf{eval}_a(g,f,I) \;=^{\text{def}}\; \mathsf{eval}^{2|M|}_a(g,f,I) \;\wedge\; \bigwedge_{b\neq a} \neg\mathsf{eval}^{2|M|}_b(g,f,I)\ .$$

This formula ensures the second property of the lemma by construction, and behaves like eval_s whenever (g,f) is Ramseian. ⊣

The formulas constructed in Lemma 5.4 can be seen as defining a partial function eval that maps g,f,I to some element $a \in M$ (the second item in the lemma enforces that there is no ambiguity about the value, namely, that this is a function and not a relation). Hereafter, we simply use the notation $\mathsf{eval}(g,f,I)$ as if it were a function.

One needs now to enforce that $\mathsf{eval}(g,f,I)$ coincides with $\pi(u|_I)$, even without assuming that (g,f) is Ramseian. For this, one uses condensations. A priori, a condensation is not representable by monadic variables, since it is a binary relation. However, any set $X \subseteq \alpha$ naturally defines the relation $\approx_X$ such that $x \approx_X y$ iff either $[x,y] \subseteq X$, or $[x,y] \cap X = \varnothing$. It is easy to check that this relation is a condensation. A form of converse result also holds:

LEMMA 5.5. *For every condensation $\sim$, there is X such that $\sim$ and $\approx_X$ coincide.*

PROOF. It is easy to see that, given a linear ordering β, there exists a subset Y of β such that for all $x < y$ in β, $[x,y]$ intersects both Y and its complement $\beta \setminus Y$: indeed, one can first prove this for scattered linear orderings and for dense linear orderings, and then combine the results for these subcases using the fact that every linear ordering is a dense sum of non-empty scattered linear orderings [23].

The lemma follows easily from the above argument: consider Y obtained from the claim above applied to the condensed ordering $\beta = \alpha/_\sim$. We construct the desired set X in such a way that it contains the elements of the equivalence classes of $\sim$ that belong to Y, i.e., $X = \{x \mid [x]_\sim \in Y\}$. It is easy to see that $x \sim y$ iff $x \approx_X y$. ⊣

Lemma 5.5 tells us that it is possible to work with condensations as if they were monadic variables. In particular, in the sequel we use variables for condensations and we tacitly assume that they are encoded by the sets obtained from Lemma 5.5.

Given a convex subset I of α and some condensation $\sim$ of $\alpha|_I$, we denote by $u[I,\sim]$ the word with domain $\beta = (\alpha|_I)/_\sim$ in which every $\sim$-equivalence class J is labelled by $\mathsf{eval}(g,f,J)$. One can easily define a formula $\mathsf{consistency}(g,f)$ that checks that, for all convex subsets I and all condensations $\sim$ of $\alpha|_I$, the following conditions hold:

(C1) if I is a singleton $\{x\}$, then $\mathsf{eval}(g,f,I) = u(x)$,
(C2) if $u[I,\sim] = a\,b$ for some $a,b \in M$, then $\mathsf{eval}(g,f,I) = a \cdot b$,
(C3) if $u[I,\sim] = e^\omega$ for some idempotent $e \in M$, then $\mathsf{eval}(g,f,I) = e^\tau$,
(C4) if $u[I,\sim] = e^{\omega^*}$ for some idempotent $e \in M$, then $\mathsf{eval}(g,f,I) = e^{\tau^*}$,
(C5) if $u[I,\sim] = P^\eta$ for some non-empty set $P \subseteq M$, then $\mathsf{eval}(g,f,I) = P^\kappa$.

For some fixed I and $\sim$, the above tests require access to the elements $u[I,\sim](J)$, where J is a $\sim$-equivalence class of $\alpha|_I$. Since the property of $\sim$-equivalence for two positions $x, y \in \alpha|_I$ is first-order definable, we know that for every position $x \in \alpha|_I$, the element $\mathsf{eval}(g,f,[x]_\sim)$ is first-order definable from x. This shows that the above properties can be expressed by first-order formulas and hence $\mathsf{consistency}(g,f)$ is a formula in the $\forall$-fragment of MSO logic.

The last key argument is to show how the 'local' consistency constraints C1–C5 imply a 'global' consistency property. This is done by the following lemma.

LEMMA 5.6. *If* $\mathsf{consistency}(g,f)$ *holds, then* $\mathsf{eval}(g,f,I) = \pi(u|_I)$ *for all convex subsets* I *of* α.

PROOF. Recall that, given a convex subset I of α and a condensation $\sim$ of $\alpha|_I$, $u[I,\sim]$ is the word with domain $\beta = (\alpha|_I)/_\sim$ in which every $\sim$-equivalence class J is labelled by $\mathsf{eval}(g,f,J)$. Suppose that $\mathsf{consistency}(g,f)$ holds, namely, that for all convex subsets I of α and all condensations $\sim$ of $\alpha|_I$, the conditions C1–C5 are satisfied.

To show that $\mathsf{eval}(g,f,I) = \pi(u|_I)$ for all convex subsets I, we use again evaluation trees. Precisely, we fix a convex subset I of α and an evaluation tree $\mathcal{T} = (T,\gamma)$ over the word $u|_I$ (the evaluation tree exists thanks to Proposition 3.8), and we prove, by an induction on $\mathcal{T}$, that

$$\mathsf{eval}(g,f,I) \;=\; \gamma(I)\ .$$

Since $\gamma(I) = \pi(u|_I)$ (by Proposition 3.9), it follows that $\mathsf{eval}(g,f,I) = \pi(u|_I)$.

If $\mathcal{T}$ consists of a single leaf, then I is a singleton of the form $\{x\}$. Condition C1 then immediately implies $\mathsf{eval}(g,f,I) = u(x) = \gamma(I)$.

If the root of $\mathcal{T}$ is not a leaf, then we let $\sim$ be the condensation of $\alpha|_I$ induced by the children of the root of $\mathcal{T}$ and we let $\beta = (\alpha|_I)/_\sim$ be the corresponding condensed ordering (formally, $\beta = \mathsf{children}(I)$). Note that for every class $J \in \beta$, $\mathcal{T}|_J$ is a subtree of $\mathcal{T}$. From the induction hypothesis on the evaluation tree $\mathcal{T}|_J$, we have $\mathsf{eval}(g,f,J) = \gamma(J)$ for all $J \in \beta$. Moreover, we know from the definition of $u[I,\sim]$ that $u[I,\sim](J) = \mathsf{eval}(g,f,J)$, for all $J \in \beta$, and hence $u[I,\sim]$ is isomorphic to the word $\prod_{J\in\beta} \gamma(J)$. We also know from the definition of $\mathcal{T}$ that the image under π_0 of the word $\prod_{J\in\beta} \gamma(J)$ is defined. From this we derive that $\prod_{J\in\beta} \gamma(J)$ is isomorphic to one of the following words:

1. a word $a\,b$, for some $a,b \in M$,
2. an ω-word e^ω, for some idempotent $e \in M$,

3. an ω^*-word e^{ω^*}, for some idempotent $e \in M$,
4. a shuffle P^η, for some non-empty subset P of M.

We only analyse the first two cases (the remaining cases are all similar).

If the word $\prod_{J \in \beta} \gamma(J)$ is of the form $a\,b$, with $a, b \in M$, then we let J_1 and J_2, with $J_1 < J_2$, be the two positions in it (recall that these are $\sim$-equivalence classes for $\alpha|_I$). Thanks to the inductive hypothesis, we have $\mathsf{eval}(g, f, J_1) = u[I, \sim](J_1) = \gamma(J_1) = a$ and $\mathsf{eval}(g, f, J_2) = u[I, \sim](J_2) = \gamma(J_2) = b$. From Condition C2, using the condensation $\sim$, we derive $\mathsf{eval}(g, f, I) = \mathsf{eval}(g, f, J_1 \cup J_2) = a \cdot b$, and from this we easily conclude that

$$\mathsf{eval}(g, f, I) \;=\; a \cdot b \;=\; \pi_0(a\,b) \;=\; \pi_0\big(\gamma(J_1)\,\gamma(J_2)\big) \;=\; \gamma(I).$$

Let us now consider the case where $\prod_{J \in \beta} \gamma(J)$ is an ω-word of the form e^ω, for some idempotent $e \in M$. We denote by $J_1 < J_2 < \dots$ the positions in $\prod_{J \in \beta} \gamma(J)$ (recall that these are $\sim$-equivalence classes for $\alpha|_I$). As in the previous case, we know from the inductive hypothesis that $\mathsf{eval}(g, f, J_i) = u[I, \sim](J_i) = \gamma(J_i) = e$ for all $i = 1, 2, \dots$. We know from Condition C3 that $\mathsf{eval}(g, f, I) = \mathsf{eval}(g, f, J_1 \cup J_2 \cup \dots) = e^\tau$. Finally, we derive

$$\mathsf{eval}(g, f, I) \;=\; e^\tau \;=\; \pi_0(e^\omega) \;=\; \pi_0\Big(\prod_{J \in \beta} \gamma(J)\Big) \;=\; \gamma(I)\ .$$

⊣

We conclude the section by showing how Lemma 5.6 implies Theorem 5.1. We claim that, given $a \in M$, the language $\pi^{-1}(a)$ is defined by the following sentence in the $\exists\forall$-fragment of MSO logic:

$$\varphi_a^{\mathsf{value}} \;=^{\mathrm{def}}\; \exists g.\ \exists f.\ \mathsf{consistency}(g, f) \;\wedge\; \mathsf{eval}(g, f, \alpha) = a\ .$$

Let $\pi(u) = a$. One can find a Ramseian pair (g, f) using Theorem 5.3. Lemma 5.4 then implies $\pi(u|_I) = \mathsf{eval}(g, f, I)$ for all convex subsets I. Since π is a product, the constraints C1–C5 are satisfied and $\mathsf{consistency}(g, f)$ holds. This proves that $\varphi_a^{\mathsf{value}}$ holds. Conversely, if $\varphi_a^{\mathsf{value}}$ holds, then $\mathsf{consistency}(g, f)$ holds for some (g, f). Lemma 5.6 then implies

$$\pi(u) \;=\; \pi(u|_\alpha) \;=\; \mathsf{eval}(g, f, \alpha) \;=\; a\ .$$

§6. Applications. In this section we present consequences of our results.

6.1. Collapse of the quantifier hierarchy. A first consequence of Theorems 4.1 and 5.1 is that the hierarchy of monadic quantifier alternation for MSO logic interpreted over countable words collapses to its $\exists\forall$-fragment. Clearly, since MSO logic is closed under complementation, it also collapses to its $\forall\exists$-fragment:

COROLLARY 6.1. *Every $\circledast$-language definable in MSO logic can be equally defined in the $\exists\forall$-fragment and in the $\forall\exists$-fragment.*

Moreover, the collapse result is optimal, in the sense that there exist MSO definable languages that are not definable in the $\exists$-fragment:

PROPOSITION 6.2. *The language $L_\forall$ of countable scattered words over the singleton alphabet $\{a\}$ cannot be defined in the $\exists$-fragment of MSO logic.*

PROOF. We first recall a folklore result that shows that the language $L_\forall$ cannot be defined in first-order logic. The argument is based on Ehrenfeucht-Fraïssé games (we refer the reader to [26, 16, 23] for basic knowledge on these games). One begins by fixing a number $n \in \mathbb{N}$ and suitable words $w \in L_\forall$ and $w' \notin L_\forall$, which may depend on n. One then considers n rounds of the Ehrenfeucht-Fraïssé game over w and w', where two players, called Spoiler and Duplicator, alternatively mark positions in w and w' inducing partial isomorphisms. More precisely, at each round $k = 1, \dots, n$, Spoiler marks a position in one of the two words, say either $x_k \in \mathsf{dom}(w)$ or $y_k \in \mathsf{dom}(w')$ – intuitively this corresponds to quantifying existentially or universally over w. Duplicator responds by choosing a corresponding position in the other structure, say either $y_k \in \mathsf{dom}(w')$ or $x_k \in \mathsf{dom}(w)$. The responses of Duplicator must enforce an isomorphism between the induced substructures $w|_{\{x_1,\dots,x_k\}}$ and $w'|_{\{y_1,\dots,y_k\}}$. If Duplicator cannot move while preserving the invariant, he loses the game. If he survives n rounds, he wins. We know from Fraïssé's Theorem that Duplicator can win the n-round game if, and only if, w and w' cannot be distinguished by any formula of first-order logic with n nested quantifiers – in particular, if this happens for arbitrarily large $n \in \mathbb{N}$, then $L_\forall$ cannot be defined in first-order logic.

Below, we show that, for all $n \in \mathbb{N}$, Duplicator has a strategy to survive n rounds of the Ehrenfeucht-Fraïssé game induced by the words

$$w =^{\text{def}} a^\omega \in L_\forall \qquad \text{and} \qquad w' =^{\text{def}} a^\omega \, (a^{\omega^*} a^\omega)^\eta \notin L_\forall \ .$$

Without loss of generality, we can assume that during the first round of the game the left endpoints of w and w' are marked. For the subsequent rounds, the strategy of Duplicator will enforce the following invariant: if the distance between two positions x_i, x_j that are marked in w at rounds $j < i$ is less than 2^{n-i}, then so is the distance between the corresponding positions y_i, y_j that are marked in w', and vice versa. On the other hand, if at round i Spoiler picks a position x_i in w that is at distance at least 2^{n-i} from all previously marked positions, then, Duplicator can responds by picking a position y_i inside a factor $a^{\omega^*} a^\omega$ of w' that has no marked positions, thus guaranteeing that y_i is at distance at least 2^{n-i} from all other marked positions. This strategy guarantees that Duplicator survives at least n rounds of the game. The fact that winning strategies for Duplicator exist for all $n \in \mathbb{N}$, proves that $L_\forall$ is not definable in first-order logic.

Now, it is straightforward to generalize the above argument to show that, for every first-order formula φ and every pair of finite words u, v over a finite alphabet, the following implication holds:

$$(\star) \qquad u\, v^\omega \vDash \varphi \qquad \text{implies} \qquad u\, v^\omega \, (v^{\omega^*} v^\omega)^\eta \vDash \varphi \ .$$

We can use this result to show that the language $L_\forall$ cannot be defined in the $\exists$-fragment of MSO logic. Suppose, by way of contradiction, that there is a sentence $\psi = \exists \bar{X}\, \varphi(\bar{X})$ that defines $L_\forall$, where φ is a first-order formula with free variables among $\bar{X} = X_1, \dots, X_m$. Since $a^\omega \in L_\forall$, we know that φ is satisfied by an interpretation of the free variables $\bar{X}$, and that this interpretation can be encoded by an ω-word w over the alphabet $\{a\} \times \{0,1\}^m$. By Büchi's result

(or, equally, by Theorem 3.13), we can assume, again without loss of generality, that w is ultimately periodic, namely, of the form $u\,v^\omega$, for some finite words u, v. By the indistinguishability result in $(\star)$, we know that φ is also satisfied by $u\,v^\omega\,(v^{\omega^*}\,v^\omega)^\eta$. It follows that $a^\omega\,(a^{\omega^*}\,a^\omega)^\eta$ is a model of ψ. However, the latter word does not belong to $L_\forall$, and this contradicts the fact that ψ defines $L_\forall$. ⊣

6.2. Definability with the cuts at the background. In [14] Gurevich and Rabinovich raised and left open the following question: given any MSO formula $\varphi(X_1, \ldots, X_m)$, does there exist another MSO formula $\tilde{\varphi}(X_1, \ldots, X_m)$ such that, for all sets of rational numbers $A_1, \ldots, A_m$,

$$(\mathbb{R}, <) \vDash \varphi(A_1, \ldots, A_m) \qquad \text{iff} \qquad (\mathbb{Q}, <) \vDash \tilde{\varphi}(A_1, \ldots, A_m) \qquad ?$$

In other words, they considered question of whether the ability to use all points of the real line does give more expressive power for stating properties of predicates over the rational line – Gurevich and Rabinovich use the suggestive terminology that the formula φ has access to the reals '*at the background*'. Note that here we implicitly use the fact that there is a fixed natural embedding of $(\mathbb{Q}, <)$ into $(\mathbb{R}, <)$.

Gurevich and Rabinovich answered positively the analogous question where the rational line is replaced by the order of the natural numbers:

THEOREM 6.3 ([14]). *For every MSO formula $\varphi(X_1, \ldots, X_m)$, there is an MSO formula $\tilde{\varphi}(X_1, \ldots, X_m)$ such that, for all sets $A_1, \ldots, A_m \subseteq \mathbb{N}$,*

$$(\mathbb{R}, <) \vDash \varphi(A_1, \ldots, A_m) \qquad \textit{iff} \qquad (\mathbb{N}, <) \vDash \tilde{\varphi}(A_1, \ldots, A_m)\ .$$

We will not enter the details of this result, which is superseded by what follows. However, already in this case an interesting phenomenon occurs: the existence of the formula $\tilde{\varphi}$ is inherently non-effective, and this holds even if $\tilde{\varphi}$ is allowed to use extra predicates with a decidable MSO theory:

THEOREM 6.4 ([14]). *Let $\bar{B} = B_1, \ldots, B_n \subseteq \mathbb{N}$ be a tuple of monadic predicates such that $(\mathbb{N}, <, \bar{B})$ has a decidable MSO theory. There is no algorithm that transforms an MSO formula $\varphi(X_1, \ldots, X_m)$ to an MSO formula $\tilde{\varphi}(X_1, \ldots, X_m)$ such that*

$$(\mathbb{R}, <) \vDash \varphi(A_1, \ldots, A_m) \qquad \textit{iff} \qquad (\mathbb{N}, <, \bar{B}) \vDash \tilde{\varphi}(A_1, \ldots, A_m)\ .$$

PROOF. Assume that such an algorithm exists, and consider a generic MSO sentence φ. We can apply the algorithm to φ to obtain a sentence $\tilde{\varphi}$ such that $(\mathbb{R}, <) \vDash \varphi$ iff $(\mathbb{N}, <, \bar{B}) \vDash \tilde{\varphi}$. Since the MSO theory of $(\mathbb{N}, <, \bar{B})$ is decidable, we could decide the MSO theory of $(\mathbb{R}, <)$. However, in [24, 15] it has been shown that MSO theory of the real line is undecidable. ⊣

Despite the inherent difficulty due to the non-effectiveness of the transformation, we are able to answer positively the question raised by Gurevich and Rabinovich.

We begin by describing more precisely the relationship between the rational line and the real line. In fact, for technical reasons, it is convenient to work, rather than on the real line, on a larger structure that is obtained by completing the rational line with all Dedekind cuts.

DEFINITION 6.5. *A* (Dedekind) cut *of a linear ordering α is a subset E of α such that $\alpha|_E$ is a prefix of α.*

The cuts of α are naturally order by the containment relation, that is, for all cuts E, F, we have $E < F$ iff $E \subsetneq F$. A cut is *extremal* if it is empty or contains all elements of the linear order α. Cuts can also be compared with the elements of α as follows: for all $x \in \alpha$ and all cuts E of α, we have $x < E$ (resp., $E < x$) iff $x \in E$ (resp., $x \notin E$). Note that every element x of α has two adjacent cuts: $x^- = \{y \in \alpha \mid y < x\}$ and $x^+ = \{y \in \alpha \mid y \leq x\}$. Cuts that are not of the form x^- or x^+ are called *natural*.

DEFINITION 6.6. *The* completion *of a linear order α, denoted $\hat{\alpha}$, is obtained from the disjoint union of the elements of α and the non-extremal cuts of α, and it is equipped with the extended ordering defined above.*

Note that the real line is obtained from the rational line using a similar notion of completion that only adds the non-extremal *natural* cuts. However, the difference between the real line and the completion, as defined above, of the rational line is negligible, especially as far as MSO definability of rational sets is concerned. In particular, since the natural cuts in $\hat{\mathbb{Q}}$ are definable by first-order formulas, one can easily transform any MSO formula $\varphi(X_1, \dots, X_m)$ to an MSO formula $\varphi'(X_1, \dots, X_m)$ such that, for all sets $A_1, \dots, A_m \subseteq \mathbb{Q}$, $(\mathbb{R}, <) \models \varphi(A_1, \dots, A_m)$ iff $(\hat{\mathbb{Q}}, <) \models \varphi'(A_1, \dots, A_m)$. As a consequence, to answer the question raised by Gurevich and Rabinovich, it is sufficient to prove the following result:

THEOREM 6.7. *For every MSO formula $\varphi(X_1, \dots, X_m)$, there is an MSO formula $\tilde{\varphi}(X_1, \dots, X_m)$ such that, for all countable linear orderings α and all sets $A_1, \dots, A_m \subseteq \alpha$,*

$$\hat{\alpha} \models \varphi(A_1, \dots, A_m) \qquad \textit{iff} \qquad \alpha \models \tilde{\varphi}(A_1, \dots, A_m) \ .$$

Next, we generalize the notion of completion to words. We fix a dummy letter c that is intended to label the cuts. The *completion* of a word $w : \alpha \to A$ is the word $\hat{w} : \hat{\alpha} \to A \uplus \{c\}$ defined by $\hat{w}(x) = w(x)$, for all elements $x \in \alpha$, and $\hat{w}(E) = c$ for all cuts $E \in \hat{\alpha} \setminus \alpha$.

A simple, yet important, property is the relationship between the operation of completion of a word and that of product of words, which is formalized in the following lemma (proof omitted). Intuitively, the completion of the product of a series of words is equivalent to a variant of the product on the completions of the words, where the variant of the product 'fills the missing cuts'.

LEMMA 6.8. *For all linear orderings α and all words $(u_i)_{i \in \alpha}$, we have*

$$\widehat{\left(\prod_{i \in \alpha} u_i\right)} \;=\; \widehat{\prod}_{i \in \alpha} \hat{u}_i$$

where the product variant $\widehat{\prod}$ is defined by $\widehat{\prod}_{i \in \alpha} \hat{v}_i = \prod_{i \in \hat{\alpha}} v'_i$, with $v'_i = v_i$ if $i \in \alpha$ and $v'_i = c$ if $i \in \hat{\alpha} \setminus \alpha$.

A language L of countable words is said to be *MSO definable with the cuts at the background* if there exists an MSO sentence φ such that $u \in L$ iff $\hat{u} \models \varphi$. The

following proposition is similar to the claim of Theorem 4.1 (note that here we omit the part about effectiveness).

PROPOSITION 6.9. *Languages of countable words that are MSO definable with the cuts at the background are recognizable by $\circledast$-monoids.*

PROOF. Recall that the proof of Theorem 4.1 was based on closure properties of recognizable $\circledast$-languages under boolean operations and projections, which could be easily implemented at the level of the $\circledast$-algebras. Because in this proof we do not have to deal with effectiveness, it is convenient to work directly at the level of $\circledast$-monoids. In particular, the monoids recognizing the considered languages will be defined using logical types and Shelah's composition method [24]. We shall consider MSO formulas up to syntactic equivalence, that is, up to associativity, commutativity, idempotency, and distributivity of conjunctions and disjunctions, commutativity of conjunctions with universal quantifications and disjunctions with existential quantifications, and renamings of quantified variables. Recall that, over a fixed finite signature with only relational symbols, there exist only finitely many sentences up to syntactic equivalence.

Let φ be an MSO sentence defining, with the cuts at the background, a language $L \subseteq A^{\circledast}$. Let k be the *quantifier rank of* φ, that is, the maximum number of nested quantifiers in φ. Given a word u of possibly uncountable domain, we define its *k-type* $\mathsf{type}_k(u)$ as the (finite) set of all sentences of quantifier rank at most k. We recall a simplified version of the composition theorem of Shelah, which shows that the type of a product of words is uniquely determined by the types of the words:

CLAIM (Shelah's composition theorem [24]). *Let α be a (possibly uncountable) linear ordering and, for every $i \in \alpha$, let u_i, v_i be words (of possibly uncountable domains). We have*

$$\forall i \in \alpha \quad \mathsf{type}_k(u_i) = \mathsf{type}_k(v_i) \qquad \textit{implies} \qquad \mathsf{type}_k\Big(\prod\nolimits_{i\in\alpha} u_i\Big) = \mathsf{type}_k\Big(\prod\nolimits_{i\in\alpha} v_i\Big)\,.$$

To show that L is recognizable, we need to construct a $\circledast$-monoid (M, π) and a morphism h from A to M such that $L = h^{-1}\big(h(L)\big)$. For this, we define the function $\mathsf{type}_k^{\wedge}$ that maps any countable word to the k-type of its completion, that is, $\mathsf{type}_k^{\wedge}(w) = \mathsf{type}_k(\hat{w})$. The domain M of the $\circledast$-monoid is precisely the range of the function $\mathsf{type}_k^{\wedge}$, that is,

$$M \;=^{\text{def}}\; \big\{\mathsf{type}_k^{\wedge}(w) \;\big|\; w \in A^{\circledast}\big\}\,.$$

We further let word be a function that maps any element $m \in M$ to a word $\mathsf{word}(m) \in A^{\circledast}$ such that $\mathsf{type}_k^{\wedge}\big(\mathsf{word}(m)\big) = m$. The product π of the $\circledast$-monoid is defined as follows:

$$\pi\Big(\prod\nolimits_{i\in\alpha} m_i\Big) \;=^{\text{def}}\; \mathsf{type}_k^{\wedge}\Big(\prod\nolimits_{i\in\alpha} \mathsf{word}(m_i)\Big)\,.$$

Even if we do not know yet that π is a product (e.g., that it satisfies generalized associativity), we can easily verify that the function $\mathsf{type}_k^{\wedge}$ behaves like a morphism. Formally, for all countable linear orderings α and all words $u_i \in A^{\circledast}$, we

have:

$$\begin{aligned}
&\text{(by definition of } \mathsf{type}^\wedge_k) && \mathsf{type}^\wedge_k\Big(\prod\nolimits_{i\in\alpha} u_i\Big) &&= \mathsf{type}_k\Big(\widehat{\textstyle\prod_{i\in\alpha} u_i}\Big)\\
&\text{(by Lemma 6.8)} && &&= \mathsf{type}_k\Big(\widehat{\prod}_{i\in\alpha} \hat{u}_i\Big)\\
&\text{(by Claim 6.2)} && &&= \mathsf{type}_k\Big(\widehat{\prod}_{i\in\alpha} \widehat{\mathsf{word}\big(\mathsf{type}^\wedge_k(u_i)\big)}\Big)\\
&\text{(by Lemma 6.8)} && &&= \mathsf{type}^\wedge_k\Big(\prod\nolimits_{i\in\alpha} \mathsf{word}\big(\mathsf{type}^\wedge_k(u_i)\big)\Big)\\
&\text{(by definition of } \pi) && &&= \pi\Big(\prod\nolimits_{i\in\alpha} u_i\Big)
\end{aligned}$$

Moreover, since $\mathsf{type}^\wedge_k$ is surjective from $A^\circledast$ to M, the property of being a $\circledast$-monoid is transferred from $(A^\circledast, \prod)$ to (M, π). Hence, (M, π) is a $\circledast$-monoid. Finally, if we let $h = \mathsf{type}^\wedge_k$ and we consider two words $u, v \in A^\circledast$ such that $h(u) = u(v)$, we get $u \in L$ iff $\hat{u} \vDash \varphi$ iff $\varphi \in \mathsf{type}^\wedge_k(u) = h(u)$ iff $\varphi \in \mathsf{type}^\wedge_k(v)$ iff $v \in L$. This shows that L is recognized by the $\circledast$-monoid (M, π) via the morphism $h = \mathsf{type}^\wedge_k$. ⊣

Proposition 6.9 combined with Theorem 5.1 shows that the languages definable in MSO logic with the cuts at the background are also definable in classical MSO logic:

COROLLARY 6.10. *Languages of countable words that are MSO definable with the cuts at the background are MSO definable.*

Finally, if we restate the above corollary in terms of relational structures, we get precisely the claim of Theorem 6.7.

6.3. Yields of tree languages. We conclude the section by considering another open problem related to countable words. More precisely, we will consider yields of trees, that is, words spelled out by frontiers of trees following the natural left-to-right order [10, 4].[3] We restrict ourselves to *labelled binary trees*, namely, trees in which every node has an associated label from a finite alphabet and every internal node has exactly two (ordered) successors. These trees may contain leaves as well as infinite paths.

DEFINITION 6.11. *The* yield *of a tree t is the word $\mathsf{yield}(t)$ whose domain is the set of leaves of t, ordered by the infix relation, such that $\mathsf{yield}(t)(x) = t(x)$ for all leaves x.*

Given two trees t, t' and a set X of leaves of t, we denote by $t[X/t']$ the tree resulting from the simultaneous substitution in t of all leaves $x \in X$ by t'. This substitution operation is compatible with the analogous operation of substitution on yields, that is, for all $X \subseteq \mathsf{dom}\big(\mathsf{yield}(t)\big)$, we have

$$\mathsf{yield}\big(t[X/t']\big) \;=\; \mathsf{yield}(t)\big[X/\mathsf{yield}(t')\big]\ .$$

[3] We remark that a different notion of yield was introduced in [11], based on a specific continuous function that maps trees to finite or ω-words.

By a slight abuse of notation, given a letter a occurring at some leaves of t, we denote by $t[a/t']$ the result of the simultaneous substitution in t of all a-labelled leaves by t', and similarly for $\mathsf{yield}(t)\big[a/\mathsf{yield}(t')\big]$.

Every word of countable domain can be seen as the yield of some tree. Indeed, this holds trivially for every word indexed over the rationals. Moreover, every word w of countable domain can be obtained from a word w' over the rationals by removing some positions. This latter operation of removing positions can be implemented at the level of trees by a substitution: if $w = \mathsf{yield}(t)$ and $X \subseteq \mathsf{dom}(w)$, then $w[X/\varepsilon] = \mathsf{yield}\big(t[X/t_\varepsilon]\big)$, where t_ε is the infinite complete binary tree, whose yield is the empty word.

We can also extend the yield function to any language T of trees by letting $\mathsf{yield}(T) = \big\{\mathsf{yield}(t) \,\big|\, t \in T\big\}$. Similarly, given a language L of words, we define the corresponding tree language as $\mathsf{yield}^{-1}(L) = \big\{t \,\big|\, \mathsf{yield}(t) \in L\big\}$. We say that a tree language T is *yield-invariant* if, for all trees t, t' such that $\mathsf{yield}(t) = \mathsf{yield}(t')$, we have $t \in T$ iff $t' \in T$ (or, equally, if $T = \mathsf{yield}^{-1}(\mathsf{yield}(T))$).

It is known (see, for instance, [25]) that the yield of a regular language T of finite trees is a context-free language, and in general it is not regular. However, when the regular tree language T is also yield-invariant, the yield language $\mathsf{yield}(T)$ is shown to be regular [13]. A converse result also holds: if L is a regular language of finite words, then $T = \mathsf{yield}^{-1}(L)$ is yield-invariant and regular. The work [4] raises the natural question of whether analogous properties hold between languages of possibly infinite trees and languages of words of countable domains. Below, we answer positively to this question by exploiting again the correspondence between MSO logic and $\circledast$-algebras.

THEOREM 6.12. *Let L be a language of countable words and let $T = \mathsf{yield}^{-1}(L)$ be the corresponding yield-invariant language of trees. Then, L is MSO definable iff T is MSO definable.*

The proof of the left-to-right direction is straightforward: if L is defined by an MSO sentence φ, then we can construct another MSO sentence φ' that, when interpreted on a tree, checks that the frontier satisfies φ; the sentence φ' defines precisely the language $T = \mathsf{yield}^{-1}(L)$.

The proof of the converse direction is not immediate, since, a priori, checking whether a given word w belongs to L requires guessing some tree $t \in T$ such that $\mathsf{yield}(t) = w$. To show that L is recognizable by $\circledast$-monoids, and hence definable in MSO logic, we will construct a $\circledast$-algebra on the basis of a suitable congruence defined from T.

DEFINITION 6.13. *Let T be a tree language over the alphabet A and let $c \notin A$ be a fresh letter that will be used as a placeholder for substitution. We denote by $\cong_T$ the equivalence on trees defined by $t_1 \cong_T t_2$ iff, for all trees t labelled over the alphabet $A \uplus \{c\}$, we have $t[c/t_1] \in L \;\leftrightarrow\; t[c/t_2] \in L$. We say that a tree t_1* inhabits *a $\cong_T$-equivalence class $[t_2]_{\cong_T}$ if $t_1 \cong_T t_2$.*

We now show some simple but fundamental properties of the relation $\cong_T$. The first property is that $\cong_T$ correctly abstracts trees with the same yield, provided that the language T is yield-invariant. Formally, if T is yield-invariant and t_1 and t_2 are two trees such that $\mathsf{yield}(t_1) = \mathsf{yield}(t_2)$, then we have $t_1 \cong_T t_2$. It

is also easy to verify that $\cong_T$ is a *congruence* with respect to the substitution operation, that is, $t_1 \cong_T t_2$ implies $t[c/t_1] \cong_T t[c/t_2]$.

Another crucial property that is used to prove Theorem 6.12 is based on Rabin's tree theorem [20], which shows that MSO definable tree languages can be equivalently described by means of automata. Below, we recall some basic knowledge about tree automata, their problems, and the translation from MSO logic. We begin by introducing a variant of parity tree automaton that can parse trees containing leaves and/or infinite paths:

DEFINITION 6.14. *A* parity tree automaton *is a tuple* $\mathcal{A} = (A, Q, I, \Delta, \Omega)$, *where* A *is a finite set of node labels,* Q *is a finite set of states,* $I \subseteq Q$ *is a set of initial states,* $\Delta \subseteq (Q \times A) \uplus (Q \times A \times Q \times Q)$ *is a set of transition rules, and* $\Omega : Q \to \mathbb{N}$ *is a priority function. A* successful run of $\mathcal{A}$ on a tree t *is a tree* ρ *that has the same domain as* t *and satisfies:*

- $\rho(x_0) \in I$, *where* x_0 *is the root of* ρ;
- *for all leaves* x *of* ρ, $\big(\rho(x), t(x)\big) \in \Delta$;
- *for all internal nodes* x *of* ρ, $\big(\rho(x), t(x), \rho(x_1), \rho(x_2)\big) \in \Delta$, *where* x_1 *and* x_2 *are the left and right successors of* x, *respectively;*
- *for all infinite paths* π *in* ρ, $\limsup\big(\Omega(\rho|_\pi)\big)$ *is even, where* $\Omega(\rho|_\pi)$ *denotes the sequence of priorities associated with the states along the path* π *and* $\limsup\big(\Omega(\rho|_\pi)\big)$ *returns the maximal priority that occurs infinitely often in the sequence* $\Omega(\rho|_\pi)$.

The language recognized *by* $\mathcal{A}$ *is the set* $\mathscr{L}(\mathcal{A})$ *of all trees* t *that admit a successful run of* $\mathcal{A}$.

We recall that the *emptiness problem* for parity tree automata, that is, the problem of testing whether $\mathscr{L}(\mathcal{A}) = \varnothing$ for any given parity tree automaton $\mathcal{A}$, is decidable. The *containment* and *equivalence problems* can be reduced to the emptiness problem by exploiting effective closures of automata under intersection and complementation: indeed, we have $\mathscr{L}(\mathcal{A}) \subseteq \mathscr{L}(\mathcal{A}')$ iff $\mathscr{L}(\mathcal{A}) \cap \mathscr{L}(\overline{\mathcal{A}'}) = \varnothing$, where $\overline{\mathcal{A}'}$ denotes the automaton recognizing the complement of the language $\mathscr{L}(\mathcal{A}')$. There is another fundamental problem that is known to be decidable, called *membership problem*. This amounts at testing whether a given tree t belongs to the language recognized by a given parity tree automaton $\mathcal{A}$. For this problem to make sense, however, we need to specify how the tree t is provided in input. A simple solution is to restrict to *regular trees*, that is, trees that contain only finitely many non-isomorphic subtrees. It is easy to see that any regular tree can be finitely represented by a parity tree automaton $\mathcal{B}$ that recognizes the singleton language $\{t\}$. The closure of parity tree automata under intersection implies that the membership problem is decidable: for every regular tree t represented by the singleton language $\mathscr{L}(\mathcal{B}) = \{t\}$, we have $t \in \mathscr{L}(\mathcal{A})$ iff $\mathscr{L}(\mathcal{B}) \cap \mathscr{L}(\mathcal{A}) \neq \varnothing$.

We recall below the correspondence between MSO sentences interpreted on trees and parity tree automata. A proof of this correspondence can be found in [27] and is based on closure properties of parity tree automata under boolean operations and projections (originally, this was established by Rabin in [20] using a different model of automaton).

THEOREM 6.15 (Translation of MSO to tree automata [27]). *One can effectively translate any MSO sentence φ that defines a tree language T into a parity tree automaton $\mathcal{A}$ that recognizes T.*

We are now ready to prove the following key lemma:

LEMMA 6.16. *For every MSO definable tree language T, $\cong_T$ has finite index, namely, there exist only finitely many $\cong_T$-equivalence classes. Moreover, given an MSO sentence defining T, one can decide whether $t_1 \cong_T t_2$, for any pair of regular trees t_1 and t_2, and one can compute a finite set of regular trees that inhabit all $\cong_T$-equivalence classes.*

PROOF. Let φ be an MSO sentence defining the tree language T and let $\mathcal{A} = (A, Q, I, \Delta, \Omega)$ be the corresponding parity tree automaton recognizing T, obtained from Theorem 6.15. Given a generic tree t, we abstract the behaviour of $\mathcal{A}$ on t by introducing the $\mathcal{A}$-*type* of t, defined as

$$\mathsf{type}_{\mathcal{A}}(t) \ =^{\text{def}} \ \big\{q \in Q \bigm| t \in \mathscr{L}(\mathcal{A}^q)\big\}$$

where $\mathcal{A}^q$ is the automaton obtained from $\mathcal{A}$ by replacing the set I of initial states with the singleton $\{q\}$. Note that there are at most $2^{|Q|}$ different $\mathcal{A}$-types of trees. Based on this, we can establish the first claim of the lemma by simply showing that the type-equivalence induced by $\mathcal{A}$ refines the $\cong_T$-equivalence, namely, that for all trees t_1 and t_2, $\mathsf{type}_{\mathcal{A}}(t_1) = \mathsf{type}_{\mathcal{A}}(t_2)$ implies $t_1 \cong_T t_2$. Consider two trees t_1, t_2 such that $\mathsf{type}_{\mathcal{A}}(t_1) = \mathsf{type}_{\mathcal{A}}(t_2)$ and another tree t labelled over the extended alphabet $A \uplus \{c\}$.

We first prove that $\mathcal{A}$-types are compatible with tree substitutions, that is, knowing that $\mathsf{type}_{\mathcal{A}}(t_1) = \mathsf{type}_{\mathcal{A}}(t_2)$, we get

$$\mathsf{type}_{\mathcal{A}}\big(t[c/t_1]\big) \ = \ \mathsf{type}_{\mathcal{A}}\big(t[c/t_2]\big) \ .$$

Consider a state $q \in \mathsf{type}_{\mathcal{A}}\big(t[c/t_1]\big)$, namely, such that $t[c/t_1] \in \mathscr{L}(\mathcal{A}^q)$. Let ρ be a successful run of $\mathcal{A}^q$ on $t[c/t_1]$ and let X be the set of c-labelled leaves of t. The set X can be equally seen as a set of nodes of ρ. We partition X into some subsets $X_{q'}$, where $q' \in Q$ and $X_{q'} = \{x \in X \mid \rho(x) = q'\}$, and for every $x \in X_{q'}$, we let ρ_x be the subtree of ρ starting at node x. Note that each subrun ρ_x, with $x \in X_{q'}$, is a successful run of the automaton $\mathcal{A}^{q'}$ on the tree t_1. This means that $q' \in \mathsf{type}_{\mathcal{A}}(t_1)$ for all non-empty sets $X_{q'}$. Since $\mathsf{type}_{\mathcal{A}}(t_1) = \mathsf{type}_{\mathcal{A}}(t_2)$, we derive that $q' \in \mathsf{type}_{\mathcal{A}}(t_2)$ for all non-empty sets $X_{q'}$. Thus, there exist successful runs ρ'_x of $\mathcal{A}^{q'}$ on t_2, for all $x \in X_{q'}$. Next, we define the tree ρ' by substituting in ρ every subtree ρ_x starting at node $x \in X$ with the tree ρ'_x (note that the substitution is performed simultaneously on nodes that may not be leaves, but these nodes are still pairwise incomparable with respect to the descendant relation). Since $\rho(x) = \rho'(x)$ for all $x \in X$, we deduce that ρ' is a successful run of $\mathcal{A}^q$ on $t[c/t_2]$. This proves that $q \in \mathsf{type}_{\mathcal{A}}\big(t[c/t_2]\big)$. Symmetric arguments show that $q \in \mathsf{type}_{\mathcal{A}}\big(t[c/t_2]\big)$ implies $q \in \mathsf{type}_{\mathcal{A}}\big(t[c/t_1]\big)$.

Now that we know that $\mathsf{type}_{\mathcal{A}}\big(t[c/t_1]\big) = \mathsf{type}_{\mathcal{A}}\big(t[c/t_2]\big)$, we can conclude the proof of the first claim by observing that

$$t[c/t_1] \in L \quad \text{iff} \quad \mathsf{type}_{\mathcal{A}}\big(t[c/t_1]\big) \cap I \neq \varnothing \quad \text{iff} \quad \mathsf{type}_{\mathcal{A}}\big(t[c/t_2]\big) \cap I \neq \varnothing \quad \text{iff} \quad t[c/t_2] \in L$$

and hence $t_1 \cong_T t_2$. This shows that $\cong_T$ has finite index.

We turn to the proof of the second claim. Consider two regular trees t_1 and t_2 represented by singleton languages $\mathscr{L}(\mathcal{B}_1) = \{t_1\}$ and $\mathscr{L}(\mathcal{B}_2) = \{t_2\}$, respectively. Recall that $t_1 \cong_T t_2$ iff for all trees t labelled over $A \uplus \{c\}$, either both trees $t[c/t_1]$ and $t[c/t_2]$ are inside $\mathscr{L}(\mathcal{A})$, or neither of them are. Further note that $t[c/t_i] \in \mathscr{L}(\mathcal{A})$ iff there is a state $q \in Q$ such that $t_i \in \mathscr{L}(\mathcal{A}^q)$ and $t \in \mathscr{L}(\mathcal{A}_q)$, where $\mathcal{A}_q$ is the automaton obtained from $\mathcal{A}$ by replacing the transition relation Δ with

$$\Delta_q \;=^{\text{def}}\; \big(\Delta \cap (Q \times A \times Q \times Q)\big) \;\uplus\; \big(\Delta \cap (Q \times (A \setminus \{c\}))\big) \;\uplus\; \big(\{q, c\}\big)$$

(intuitively, $\mathcal{A}_q$ behaves exactly as $\mathcal{A}$ on all nodes of the tree t, with the only exception of the c-labelled leaves, which must be associated with state q). Using the above properties, we can restate the equivalence $t_1 \cong_T t_2$ as a (decidable) boolean combination of emptiness problems:

$$\begin{aligned}
t_1 \cong_T t_2 \quad \text{iff} \quad & \bigwedge_{q\in Q}\bigvee_{q'\in Q} \left(\underbrace{\mathcal{B}_1 \cap \mathcal{A}^q \neq \varnothing}_{t_1 \in \mathcal{A}^q} \wedge \underbrace{\mathcal{A}_q \cap \mathcal{A}_{q'} \neq \varnothing}_{\exists\, t \in \mathcal{A}_q \cap \mathcal{A}_{q'}} \wedge \underbrace{\mathcal{B}_2 \cap \mathcal{A}^{q'} \neq \varnothing}_{t_2 \in \mathcal{A}^{q'}} \right) \\
\wedge \quad & \bigwedge_{q'\in Q}\bigvee_{q\in Q} \left(\underbrace{\mathcal{B}_1 \cap \mathcal{A}^q \neq \varnothing}_{t_1 \in \mathcal{A}^q} \wedge \underbrace{\mathcal{A}_q \cap \mathcal{A}_{q'} \neq \varnothing}_{\exists\, t \in \mathcal{A}_q \cap \mathcal{A}_{q'}} \wedge \underbrace{\mathcal{B}_2 \cap \mathcal{A}^{q'} \neq \varnothing}_{t_2 \in \mathcal{A}^{q'}} \right) \\
\wedge \quad & \bigvee_{q\in Q}\bigwedge_{q'\in Q} \left(\underbrace{\mathcal{B}_1 \cap \mathcal{A}^q = \varnothing}_{t_1 \notin \mathcal{A}^q} \wedge \underbrace{\mathcal{B}_2 \cap \mathcal{A}^{q'} = \varnothing}_{t_2 \notin \mathcal{A}^{q'}} \right) \vee \left(\underbrace{\mathcal{A}_q \cup \mathcal{A}_{q'} = \varnothing}_{\nexists\, t \in \mathcal{A}_q \cap \mathcal{A}_{q'}} \wedge \right) \\
\wedge \quad & \bigvee_{q'\in Q}\bigwedge_{q\in Q} \left(\underbrace{\mathcal{B}_1 \cap \mathcal{A}^q = \varnothing}_{t_1 \notin \mathcal{A}^q} \wedge \underbrace{\mathcal{B}_2 \cap \mathcal{A}^{q'} = \varnothing}_{t_2 \notin \mathcal{A}^{q'}} \right) \vee \left(\underbrace{\mathcal{A}_q \cup \mathcal{A}_{q'} = \varnothing}_{\nexists\, t \in \mathcal{A}_q \cap \mathcal{A}_{q'}} \wedge \right).
\end{aligned}$$

(for simplicity, we identified automata and the recognized languages).

Finally, to compute a set of regular trees that inhabit all $\cong_T$-equivalence classes, we consider again $\mathcal{A}$-types. We first show how to associate with each $\mathcal{A}$-type σ a corresponding regular tree t_σ such that $\mathsf{type}_{\mathcal{A}}(t_\sigma) = \sigma$. We do so by solving a series of emptiness problems. Indeed, we recall that an $\mathcal{A}$-type is any set σ of states of $\mathcal{A}$ such that the language $\bigcap_{q\in\sigma} \mathscr{L}(\mathcal{A}^q) \cap \bigcap_{q\notin\sigma} \mathscr{L}(\overline{\mathcal{A}^q})$ is non-empty. Moreover, if the latter language is non-empty, then it contains a regular tree t_σ that can be effectively constructed from σ. Clearly, we have $\mathsf{type}_{\mathcal{A}}(t_\sigma) = \sigma$ and hence t_σ can be used as a representant of the $\mathcal{A}$-type σ. Towards a conclusion, we can construct a list of regular trees $t_1, \dots, t_n$, one for each $\mathcal{A}$-type. Since the equivalence $\cong_T$ is refined by the type-equivalence induced by $\mathcal{A}$, we know that every $\cong_T$-equivalence class is inhabited by at least one tree among $t_1, \dots, t_n$. If needed, we can also exploit the decidability of $\cong_T$ to select a minimal subsequence $t_{i_1}, \dots, t_{i_m}$ of regular inhabitants of all $\cong_T$-equivalence classes. ⊣

We can now prove the right-to-left direction of Theorem 6.12. Let T be a yield-invariant language defined by an MSO sentence φ. We will exploit Lemma 6.16 and the fact that $\cong_T$ is a yield-invariant equivalence compatible with tree substitutions to construct a $\circledast$-algebra $(M, 1, \cdot, \tau, \tau^*, \kappa)$ recognizing the language $L = \mathsf{yield}(T)$. Formally, we define M to be the set of all $\cong_T$-equivalence classes. We recall that this set is finite and that $\cong_T$-equivalence classes can be effectively

manipulated through their *regular inhabitants*, that is, by means of representants that have the form of regular trees. We define the operators of the algebra as follows:

- 1 is the $\cong_T$-equivalence class of the infinite complete tree t_ε. Note that this tree t_ε has no leaves, and hence its yield is the empty word. Moreover, t_ε is regular, and hence it can be used as a regular inhabitant of its $\cong_T$-equivalence class.
- $\cdot$ is the function that maps any pair of $\cong_T$-equivalence classes $[t_1]_{\cong_T}$ and $[t_2]_{\cong_T}$ to the $\cong_T$-equivalence class

$$[t_1]_{\cong_T} \cdot [t_2]_{\cong_T} \;=^{\text{def}}\; \big[t_{a_1a_2}[a_1/t_1][a_1/t_2]\big]_{\cong_T}$$

  where $t_{a_1a_2}$ is a fixed tree such that $\mathsf{yield}(t) = a_1\, a_2$, and a_1, a_2 are distinct fresh letters not occurring in the alphabet of t_1 and t_2. For example, $t_{a_1a_2}$ can be chosen to be the tree

$$t_{a_1a_2} \;=\; \begin{array}{ccc} & \bullet & \\ a_1 & & a_2 \end{array}$$

  where the label $\bullet$ of the root is immaterial. Note that the $\cong_T$-equivalence class $[t_1]_{\cong_T} \cdot [t_2]_{\cong_T}$ is well defined thanks to the fact that $\cong_T$ is a congruence. Moreover, because the tree $t_{a_1a_2}$ is regular, a regular inhabitant of the class $[t_1]_{\cong_T} \cdot [t_2]_{\cong_T}$ can be effectively constructed from some regular inhabitants of $[t_1]_{\cong_T}$ and $[t_2]_{\cong_T}$.
- τ is the function that maps any $\cong_T$-equivalence class $[t_1]_{\cong_T}$ to the $\cong_T$-equivalence class

$$[t_1]^\tau_{\cong_T} \;=^{\text{def}}\; \big[t_\omega[a/t_1]\big]_{\cong_T} \qquad \text{where} \quad t_\omega \;=\; \begin{array}{ccccc} \bullet & & & & \\ & a & \bullet & & \\ & & & a & \ddots \end{array}$$

  Again, since t_ω is a regular tree, a regular inhabitant of the class $[t_1]^\tau_{\cong_T}$ can be computed from a regular inhabitant of the class $[t_1]_{\cong_T}$.
- τ^* is defined similarly to τ, where t_ω is replaced by the tree

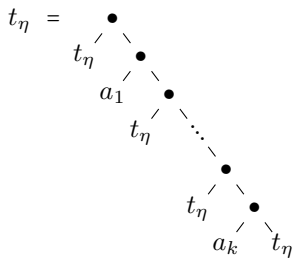
$$t_{\omega^*} \;=\; \begin{array}{ccccc} & & \bullet & & \\ & \bullet & & & a \\ \iddots & & & a & \end{array}$$

- κ is the function that maps any set $\big\{[t_1]_{\cong_T}, \dots, [t_k]_{\cong_T}\big\}$ of $\cong_T$-equivalence classes to the $\cong_T$-equivalence class

$$\big\{[t_1]_{\cong_T}, \dots, [t_k]_{\cong_T}\big\}^\kappa \;=^{\text{def}}\; \big[t_\eta[a_1/t_1]\dots[a_k/t_k]\big]_{\cong_T}$$

  where t_η is a fixed regular tree with yield $\{a_1, \dots, a_k\}^\eta$ and $a_1, \dots, a_k$ are fresh letters. For example, t_η can be defined by a parity tree automaton so

as to satisfy the following equation:



Below, we verify that the structure $(M, 1, \cdot, \tau, \tau^*, \kappa)$ obtained from the automaton $\mathcal{A}$ is indeed a $\circledast$-algebra, that is, it satisfies Axioms A1-A5 of Definition 3.2.

LEMMA 6.17. *The structure* $(M, 1, \cdot, \tau, \tau^*, \kappa)$ *obtained from* $\cong_T$ *is a* $\circledast$*-algebra.*

PROOF. The fact that the structure $(M, 1, \cdot, \tau, \tau^*, \kappa)$ satisfies Axioms A1-A5 follows almost directly from its definition and from the fact that the equivalence $\cong_T$ is yield-invariant, that is, $t_1 \cong_T t_2$ whenever $\mathsf{yield}(t_1) = \mathsf{yield}(t_2)$. For example, recall the definition of the binary operator $\cdot$: for all pairs of trees t_1, t_2, we have

$$[t_1]_{\cong_T} \cdot [t_2]_{\cong_T} = \left[\begin{array}{c} \bullet \\ t_1 \quad t_2 \end{array} \right]_{\cong_T} .$$

From this, we easily deduce that $\cdot$ satisfies Axiom A1:

(by definition) $\big([t_1]_{\cong_T} \cdot [t_2]_{\cong_T}\big) \cdot [t_3]_{\cong_T} = \left[\begin{array}{c} \bullet \\ \bullet \quad t_3 \\ t_1 \quad t_2 \end{array} \right]_{\cong_T}$

(by yield-invariance) $= \left[\begin{array}{c} \bullet \\ t_1 \quad \bullet \\ t_2 \quad t_3 \end{array} \right]_{\cong_T}$

(by definition) $= [t_1]_{\cong_T} \cdot \big([t_2]_{\cong_T} \cdot [t_3]_{\cong_T}\big)$.

We omit the analogous arguments showing that 1, τ, τ^*, and κ satisfy the remaining Axioms A2-A5. ⊣

Combining the above lemma, Corollary 3.12, and Theorem 5.1 gives the right-to-left direction of Theorem 6.12.

We also remark that, if the MSO definable tree language T is not known to be yield-invariant, we can still construct the structure $(M, 1, \cdot, \tau, \tau^*, \kappa)$ from $\cong_T$. Below, we explain how to use this structure to decide whether T is yield-invariant. We follow the same approach described in Section 5 and we construct, using the operators of $(M, 1, \cdot, \tau, \tau^*, \kappa)$, a family of MSO sentences of the form $\varphi_\sigma^{\mathsf{value}}$, where σ ranges over the set of possible $\cong_T$-equivalence classes. Given a word w, these sentences can be used to derive the $\cong_T$-equivalence class of some tree t_w such that $\mathsf{yield}(t_w) = w$. In particular, we can define in MSO logic a word language of the form $L = \big\{w \,\big|\, t_w \in \mathscr{L}(\mathcal{A})\big\}$. We can then use the left-to-right implication of Theorem 6.12 to derive an MSO sentence defining the tree

language $T' = \mathsf{yield}^{-1}(L)$. Now, if T is yield-invariant, then $T' = \mathsf{yield}^{-1}(L) = \mathsf{yield}^{-1}(\mathsf{yield}(T)) = T$, as shown by Theorem 6.12. Conversely, if $T' = T$, then T is clearly yield-invariant. We thus reduced the problem of deciding whether an MSO definable tree language T is yield-invariant to the equivalence problem for MSO sentences interpreted on trees, which is known to be decidable.

THEOREM 6.18. *The problem of deciding whether a tree language T defined by an MSO sentence is yield-invariant is decidable.*

§7. Conclusion. We have introduced an algebraic notion of recognizability for languages of countable words and we have shown the correspondence with the family of languages definable in MSO logic. As a side-product of this result, we obtained that the hierarchy of monadic quantifier alternation for MSO logic interpreted over countable words collapses to its $\exists\forall$-fragment (or, equally, to its $\forall\exists$-fragment). The collapse result is optimal in the sense that there are recognizable languages that are not definable in the $\exists$-fragment. Our techniques are then used to solve an open problem posed by Gurevich and Rabinovich, concerning the definability of properties of sets of rationals using MSO formulas interpreted over the real line (definability with the cuts at the background). Finally, we exploited the correspondence between logic and algebras to solve another open problem posed by Bruyère, Carton, and Sénizergues, concerning the characterization of properties of trees that can be defined in MSO logic and that are yield-invariant.

We conclude by mentioning the possibility of defining models of automata that extend those from [3] and that capture precisely the expressiveness of MSO logic over words of countable domains. However, such automata need to have complicated acceptance conditions in order to distinguish between scattered and non-scattered words and, more generally, to enjoy closure properties under boolean operations and projections. The definition of an automaton model for languages of countable words is thus not as natural as that of $\circledast$-monoid.

REFERENCES

[1] N. BEDON, A. BÈS, O. CARTON, and C. RISPAL, *Logic and Rational Languages of Words Indexed by Linear Orderings*, ***Theoretical Computer Science***, vol. 46 (2010), no. 4, pp. 737–760.

[2] V. BRUYÈRE and O. CARTON, *Automata on linear orderings*, ***Journal of Computer and System Sciences***, vol. 73 (2007), no. 1, pp. 1–24.

[3] V. BRUYÈRE and O. CARTON, *Automata on Linear Orderings*, ***Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS)***, LNCS, vol. 2136, Springer, 2001, pp. 236–247.

[4] V. BRUYÈRE, O. CARTON, and G. SÉNIZERGUES, *Tree Automata and Automata on Linear Orderings*, ***RAIRO Theoretical Informatics and Applications***, vol. 43 (2009), pp. 321–338.

[5] J.R. BÜCHI, *Transfinite automata recursions and weak second order theory of ordinals*, ***Proceedings of the 3rd International Congress for Logic, Methodology, and Philosophy of Science***, North Holland, 1967, pp. 2–23.

[6] J.R. BÜCHI, *On a Decision Method in Restricted Second Order Arithmetic*, ***Proceedings of the International Congress for Logic, Methodology and Philosophy of Science***, Stanford University Press, 1962, pp. 1–11.

[7] O. Carton, T. Colcombet, and G. Puppis, *Regular Languages of Words Over Countable Linear Orderings*, ***Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP)***, LNCS, vol. 6756, Springer, 2011, pp. 125–136.

[8] T. Colcombet, *Factorisation Forests for Infinite Words and Applications to Countable Scattered Linear Orderings*, ***Theoretical Computer Science***, vol. 411 (2010), pp. 751–764.

[9] ———, *Composition with Algebra at the Background – On a Question by Gurevich and Rabinovich on the Monadic Theory of Linear Orderings*, ***Proceedings of the 8th International Computer Science Symposium in Russia (CSR)***, LNCS, vol. 7913, Springer, 2013, pp. 391–404.

[10] B. Courcelle, *Frontiers of Infinite Trees*, ***RAIRO Theoretical Informatics and Applications***, vol. 12 (1978), no. 4, pp. 319–337.

[11] M. Dauchet and E. Timmerman, *Continuous Monoids and Yields of Infinite Trees*, ***RAIRO Theoretical Informatics and Applications***, vol. 20 (1986), no. 3, pp. 251–274.

[12] S. Feferman and R. Vaught, *The First-order Properties of Products of Algebraic Systems*, ***Fundamenta Mathematicae***, vol. 47 (1959), pp. 57–103.

[13] F. Gécseg and M. Steinby, *Tree languages*, ***Handbook of Formal Languages, Vol. 3: Beyond Words*** (G. Rozenberg and A. Salomaa, editors), Springer, 1997, pp. 1–68.

[14] Y. Gurevich and A.M. Rabinovich, *Definability and Undefinability with Real Order at The Background*, ***The Journal of Symbolic Logic***, vol. 65 (2000), no. 2, pp. 946–958.

[15] Y. Gurevich and S. Shelah, *Monadic Theory of Order and Topology in ZFC*, ***Annals of Mathematical Logic***, vol. 23 (1982), no. 2-3, pp. 179–198.

[16] Wilfrid Hodges, ***Model Theory***, vol. 42, Cambridge University Press, 1993.

[17] H. Läuchli and J.Leonard, *On the elementary theory of linear order*, ***Fundamenta Mathematicae***, vol. 59 (1966), no. 1, pp. 109–116.

[18] ———, *A decision procedure for the weak second order theory of linear order*, ***Contributions to Mathematical Logic, Proceedings of Logic Colloquium***, Studies in Logic and the Foundations of Mathematics, vol. 50, Elsevier, 1968, pp. 189–197.

[19] J.E. Pin and D. Perrin, ***Infinite Words: automata, semigroups, logic and games***, Elsevier Science Publishers, 2004.

[20] M.O. Rabin, *Decidability of Second-order Theories and Automata on Infinite Trees*, ***Transactions of the American Mathematical Society***, vol. 141 (1969), pp. 1–35.

[21] F.P. Ramsey, *On a Problem of Formal Logic*, ***Proceedings of the London Mathematical Society***, vol. 30, 1929, pp. 264–286.

[22] C. Rispal and O. Carton, *Complementation of Rational Sets on Countable Scattered Linear Orderings*, ***International Journal of Foundations of Computer Science***, vol. 16 (2005), no. 4, pp. 767–786.

[23] J.G. Rosenstein, ***Linear Orderings***, Academic Press, 1982.

[24] S. Shelah, *The Monadic Theory of Order*, ***Annals of Mathematics***, vol. 102 (1975), pp. 379–419.

[25] J.W. Thatcher, *Characterizing Derivation Trees of Context-free Grammars Through a Generalization of Finite Automata Theory*, ***Journal of Computer and System Sciences***, vol. 1 (1967), no. 4, pp. 317–322.

[26] W. Thomas, *On the Ehrenfeucht-Fraïssé Game in Theoretical Computer Science*, ***Theory and Practice of Software Development (TAPSOFT)***, LNCS, vol. 668, Springer, 1993, pp. 559–568.

[27] ———, *Languages, Automata, and Logic*, ***Handbook of Formal Languages***, vol. 3, Springer, 1997, pp. 389–455.

[28] T. Wilke, *An Algebraic Theory For Regular Languages of Finite and Infinite Words*, ***International Journal of Algebra and Computation***, vol. 3 (1993), no. 4, pp. 447–489.

[1] IRIF, PARIS, FRANCE
E-mail: olivier.carton@irif.fr

[2] CNRS / IRIF, PARIS, FRANCE
E-mail: thomas.colcombet@irif.fr

[3] CNRS / LABRI, BORDEAUX, FRANCE
E-mail: gabriele.puppis@labri.fr