

Recognisable Languages over Monads

Mikołaj Bojańczyk

February 18, 2015

Contents

I	Introduction	3
1	Introduction	4
2	Monads and their algebras	6
3	Syntactic morphisms	8
3.1	Proof of the Syntactic Morphism Theorem	10
4	Pseudovarieties	17
4.1	The Syntactic Pseudovariety Theorem.	20
4.2	The Polynomial Pseudovariety Theorem	22
5	Representing an algebra	27
6	Monadic second-order logic	30
6.1	Language theoretic definition of MSO	30
6.2	Deciding satisfiability of MSO	32
II	Example Monads	34
7	Monads for chains	35
8	Pointed words	36
8.1	Unary queries definable in two-variable first-order logic	37
9	Monads for trees	46
9.1	Ranked trees over a fixed alphabet	46
9.2	Clones	49
9.3	Forests of unranked trees	53
9.4	A monad for infinite unranked forests.	57

10 Future work	58
 III Profinite Monads	 59
11 Stone duals and topology on an algebra	60
11.1 Stone duals of Boolean algebras	60
11.2 Stone duals of \mathbf{T} -algebras	62
11.3 Uniform continuity	67
 12 Profinite monads	 72
12.1 Definition of the profinite monad	72
12.2 From a \mathbf{T} -algebra to a $\overline{\mathbf{T}}$ -algebra.	79
12.3 From a $\overline{\mathbf{T}}$ -algebra to a \mathbf{T} -algebra.	82
12.4 Clopen languages and Stone algebras	84
 13 Profinite words	 86
13.1 The unboundedness language	86

Part I

Introduction

In this part, we introduce monads and their algebras. Section 2 contains the basic definitions: first illustrated on examples of finite words and ∞ -words, and then formally defined. Sections 3-6 show how some results about languages can be stated and proved on the level of monads, including: the Myhill-Nerode theorem (Section 3), Eilenberg's pseudovariety theorem (Section 4), and some parts of the connection between regular languages and MSO (Sections 5 and 6).

1 Introduction

The principle behind algebraic language theory for various kinds of structures, such as words or trees, is to use a compositional function from the structures into a finite set. To talk about compositionality, one needs some way of composing structures into bigger structures. It so happens that category theory has an abstract concept for this, namely a monad. The goal of this paper is to propose monads as a unifying framework for discussing existing algebras and designing new algebras. To introduce monads and their algebras, we begin with two examples, which use a monad style to present algebras for finite and infinite words.

Example 1. Consider the following non-standard definition of a semigroup. Define a $+$ -algebra \mathbf{A} to be a set A called its *universe*, together with a *multiplication operation* $\text{mul}_{\mathbf{A}} : A^+ \rightarrow A$, which is the identity on single letters, and which is associative in the sense that the following diagram commutes.

$$\begin{array}{ccc} (A^+)^+ & \xrightarrow{\mu_A} & A^+ \\ (\text{mul}_{\mathbf{A}})^+ \downarrow & & \downarrow \text{mul}_{\mathbf{A}} \\ A^+ & \xrightarrow{\text{mul}_{\mathbf{A}}} & A \end{array},$$

In the diagram, $(\text{mul}_{\mathbf{A}})^+$ is the function that applies $\text{mul}_{\mathbf{A}}$ to each label of a word where the alphabet is A^+ , and μ_A is the function which flattens a word of words into a word, e.g.

$$(abc)(aa)(acaa) \mapsto abcaaacaa.$$

Restricting the multiplication operation in a $+$ -algebra to words of length two (the semigroup binary operation) is easily seen to be a one-to-one correspondence between $+$ -algebras and semigroups. \square

The second example will be running example in the paper.

Running Example 1. Let us define an algebra for infinite words in the spirit of the previous example. Define A^∞ to be the ∞ -words over A , i.e. $A^+ \cup A^\omega$. Define an ∞ -algebra \mathbf{A} to be a set A , called its *universe*, together with a *multiplication operation* $\text{mul}_{\mathbf{A}} : A^\infty \rightarrow A$, which is the identity on single letters, and which is associative in the sense that the following diagram commutes.

$$\begin{array}{ccc} (A^\infty)^\infty & \xrightarrow{\mu_A} & A^\infty \\ (\text{mul}_{\mathbf{A}})^\infty \downarrow & & \downarrow \text{mul}_{\mathbf{A}} \\ A^\infty & \xrightarrow{\text{mul}_{\mathbf{A}}} & A \end{array}$$

In the diagram, $(\text{mul}_{\mathbf{A}})^\infty$ is the function that applies $\text{mul}_{\mathbf{A}}$ to the label of every position in a ∞ -words where the alphabet is A^∞ , and μ_A flattens an ∞ -word of ∞ -words into an ∞ -word. If the argument of μ_A contains an infinite word on some position, then all subsequent positions are ignored.

An ∞ -algebra is essentially the same thing as an ω -semigroup, see [PP04], with the difference that ω -semigroups have separate sorts for finite and infinite words. There is also a close connection with Wilke semigroups [Wil91], which will be described as the running example develops. \square

The similarities in the examples suggest that there should be an abstract notion of algebra, which would cover the examples and possibly other settings, e.g. trees. A closer look at the examples reveals that concepts of algebraic language theory such as “algebra”, “morphism”, “language”, “recognisable language” can be defined only in terms of the following four basic concepts (written below in the notation appropriate to $+$ -algebras):

1. how a set A is transformed into a set A^+ ;
2. how a function $f : A \rightarrow B$ is lifted to a function $f^+ : A^+ \rightarrow B^+$;
3. a flattening operation from $(A^+)^+ \rightarrow A^+$;
4. how to represent an element of A as an element of A^+ .

These four concepts, subject to certain axioms, are what constitutes a monad, a fundamental concept in category theory (and recently, programming languages).

The point of this paper is that, based on a monad one can also define things like: “syntactic algebra”, “pseudovariety”, “MSO logic”, “profinite object”, and even prove some theorems about them. Furthermore, monads as an abstraction cover practically every setting where algebraic language theory has been applied so far, including labelled scattered orderings [BR12], labelled countable total orders [CCP11], ranked trees [Ste92], unranked trees [BW08], preclones [ÉW03].

The paper has three parts.

Part I of this paper shows that several results of formal language theory can be stated and proved on the abstract level of monads, including: the Myhill-Nerode theorem on syntactic algebras (Section 3), the Eilenberg pseudovariety theorem (Section 4), or the Reiterman theorem (Section 11) on profinite identities defining pseudovarieties. Another example is decidability of MSO (Section 6), although here monads only take care of the symbol-pushing part, leaving out the combinatorial part that is specific to individual monads, like applying the Ramsey theorem in the case of infinite words. When proving such generalisations of classical theorems, one is naturally forced to have a closer look at notions such as “derivative of a language”, or “finite algebra”, which are used in the assumptions of the theorems.

Part II includes shows how existing algebraic settings can be seen as a special case of monads. Part II also contains some new settings, illustrating how new kinds of algebras can be easily produced using monads. Specifically, Section 8 describes a monad for words with a distinguished position, where standard theorems and definitions come for free by virtue of being a monad.

Part III, is devoted to profinite constructions. It is shown that every monad has a corresponding profinite monad, which, like any monad, has its own notion of recognisability, which does not reduce to recognisability in the original

monad. For example, the monad for finite words has a corresponding monad of profinite words, and recognisable languages of profinite words turn out to be a generalisation of languages of infinite words definable in the logic $\text{MSO}+\text{U}$.

Thanks. I would like to thank Bartek Klin (who told me what a monad is), Szymon Toruńczyk, Joost Winter and Marek Zawadowski for discussions on the subject.

2 Monads and their algebras

This paper uses only the most rudimentary notions of category theory: the definitions of a category (objects and composable morphisms between them), and of a functor (something that maps objects to objects and morphisms to morphisms in a way that is consistent with composition). Almost all examples in this paper use the category of sets, where objects are sets and morphisms are functions; or possibly the category of sorted sets, where objects are sorted sets for some fixed set of sort names, and morphisms are sort-preserving functions.

A *monad* over a category is defined to be a functor T from the category to itself, and for every object X in the category, two morphisms

$$\eta_X : X \rightarrow \mathsf{T}X \quad \text{and} \quad \mu_X : \mathsf{T}\mathsf{T}X \rightarrow \mathsf{T}X,$$

which are called the unit and multiplication operations. The monad must satisfy the axioms given in Figure 1. In the language of sets, an intuition appropriate for this paper is that a monad inputs a set X , and produces the set of all “structures” whose “nodes” are labelled by elements of X . Depending on the monad, the structures could be words, or trees, or graphs, etc. The function η_X inputs a label and produces a one-node structure that uses this label; while the function μ_X , which is the essence of the monad, flattens a structure of structures into a single structure. Basing on this intuition, we will use the name *T -structures over X* for elements of $\mathsf{T}X$.

We already saw two monads in Example 1 and in the running example.

For this paper, the most important thing about monads is that they have a natural corresponding notion of algebra. An *Eilenberg-Moore algebra in a monad T* , or simply T -algebra, is a pair \mathbf{A} consisting of a *universe* A , which is an object in the category underlining the monad, together with a multiplication morphism

$$\text{mul}_{\mathbf{A}} : \mathsf{T}A \rightarrow A,$$

such that the $\text{mul}_{\mathbf{A}} \circ \eta_A$ is the identity, and which is associative in the sense that the following diagram commutes.

$$\begin{array}{ccc} \mathsf{T}\mathsf{T}A & \xrightarrow{\mu_A} & \mathsf{T}A \\ \mathsf{T}\text{mul}_{\mathbf{A}} \downarrow & & \downarrow \text{mul}_{\mathbf{A}} \\ \mathsf{T}A & \xrightarrow{\text{mul}_{\mathbf{A}}} & A \end{array}$$

$$\begin{array}{ccc}
X & \xrightarrow{f} & Y \\
\eta_X \downarrow & & \downarrow \eta_Y \\
\mathsf{T}X & \xrightarrow{\mathsf{T}f} & \mathsf{T}Y
\end{array}
\qquad
\begin{array}{ccc}
\mathsf{T}\mathsf{T}X & \xrightarrow{\mathsf{T}\mathsf{T}f} & \mathsf{T}\mathsf{T}Y \\
\mu_X \downarrow & & \downarrow \mu_Y \\
\mathsf{T}X & \xrightarrow{\mathsf{T}f} & \mathsf{T}Y
\end{array}
.$$

$$\begin{array}{ccc}
\mathsf{T}\mathsf{T}\mathsf{T}X & \xrightarrow{\mu_{\mathsf{T}X}} & \mathsf{T}\mathsf{T}X \\
\mathsf{T}\mu_X \downarrow & & \downarrow \mu_X \\
\mathsf{T}\mathsf{T}X & \xrightarrow{\mu_X} & \mathsf{T}X
\end{array}
\qquad
\begin{array}{ccc}
\mathsf{T}X & \xrightarrow{\eta_{\mathsf{T}X}} & \mathsf{T}\mathsf{T}X \\
\mathsf{T}\eta_X \downarrow & \searrow \text{id}_X & \downarrow \mu_X \\
\mathsf{T}\mathsf{T}X & \xrightarrow{\mu_X} & \mathsf{T}X
\end{array}$$

Figure 1: The axioms of a monad are that these four diagrams commute for every object X in the category and every morphism $f : X \rightarrow Y$. The upper diagrams say that the unit and multiplication are natural. The lower left diagram says that multiplication is associative, and the lower right says that the unit is consistent with multiplication.

Observe that this associativity is similar to the lower left axiom in Figure 1. In fact, the lower left axiom in Figure 1 and the upper half of the lower right axiom say that μ_X induces a T -algebra with universe $\mathsf{T}X$.

We use the convention that an algebra is denoted by a boldface letter, while its universe is written without boldface. A T -*morphism* between two T -algebras \mathbf{A} and \mathbf{B} is defined to be a function h between their universes which respects their multiplication operations in the sense that the following diagram commutes.

$$\begin{array}{ccc}
\mathsf{T}A & \xrightarrow{\mathsf{T}h} & B \\
\text{mul}_{\mathbf{A}} \downarrow & & \downarrow \text{mul}_{\mathbf{B}} \\
A & \xrightarrow{h} & B
\end{array}$$

This completes the definition of monads and their algebras.

Languages and colourings. To develop the basic definitions of recognisable languages over a monad, we require the following parameters, which we call the *setting*: the underlying category, the monad, a notion of finite alphabet, and a notion of finite T -algebra. So far, we do not place any restrictions on the notions of finiteness, e.g. when considering sets with infinitely many sorts, reasonable settings will often have finite algebras whose universe is not be finite in the same sense as a finite algebra. Actually, for some monads, it is not clear what a finite algebra should be, e.g. this is the case for infinite trees, and this paper sheds little new light on the question. Fix a setting, with the monad being T , for the following definitions.

A *colouring* of a \mathbf{T} -algebra is defined to be a morphism from its universe to some object in the underlying category. For example, when the category is sets, then a coloring is like a multivalued language, i.e. instead of saying only “yes” or “no” to each input, a colouring can have multiple values. A coloring is said to be recognised by a \mathbf{T} -morphism if it factors through it. A coloring is called *\mathbf{T} -recognisable* if it is recognised by some \mathbf{T} -morphism with a finite target, according to the notion of finite \mathbf{T} -algebra given in the setting.

Almost all examples in this paper are in sets, or in sorted sets. When the category is sets or sorted sets, we will focus mainly on the special case of colourings, namely languages, where colourings have two possible values on every sort. Consider a finite alphabet, according to the notion of finite alphabet given in the setting. In all of the examples of this paper where the category is sorted sets, a finite alphabet will be a possibly sorted set with finitely many elements. In particular, if there are infinitely many sorts, then a finite alphabet will use only finitely many. A \mathbf{T} -language over a finite alphabet Σ is defined to be any subset $L \subseteq \mathbf{T}\Sigma$. Notions of recognisability are inherited from colourings, using the characteristic function of a language. Colourings are a mild generalisation of languages, for example, when the category is sets, then a colouring with finitely many colours is \mathbf{T} -recognisable if and only if for every color, its inverse image is a recognisable language.

When the monad \mathbf{T} is clear from the context, we will sometimes skip the prefix \mathbf{T} -, and simply write language, algebra, morphism, structure.

Beyond recognisable languages. The recognisable languages will play the role of regular languages in the monad. One could go beyond regular languages. For instance, there is a natural monad version of context-free grammars, where the production rules have right hand sides in the monad applied to the terminals and nonterminals, and one can prove some theorems, like closure of context-free languages under intersection with recognisable languages. Context-free languages are beyond the scope of this paper.

3 Syntactic morphisms

This section presents a monad generalisation of the Myhill-Nerode theorem, which gives a sufficient condition for colourings, and therefore also languages, to have a syntactic (i.e. minimal) morphism. The generalisation is proved only in the setting of sorted sets, and therefore also in the setting of normal sets¹. Fix a category of sorted sets, for some choice of, possibly infinitely many, sort names. A finite sorted set is one which has finitely many elements, in particular it can use only finitely many sorts.

Finitary algebras. If \mathbf{T} is a monad, then a \mathbf{T} -algebra \mathbf{A} is called *finitary* if for every $w \in \mathbf{T}A$, there is some finite $A_0 \subseteq A$ such that $w \in \mathbf{T}A_0$. Sometimes,

¹Bartek Klin has an alternative proof, which works in arbitrary categories, but requires some additional assumptions.

a monad is such that every T -algebra is finitary, e.g. this is the case for the monad of finite words A^+ .

Theorem 3.1 [*Syntactic Morphism Theorem*] *Consider a monad T in a category of sorted sets. Let f be a colouring of an algebra \mathbf{A} , which is recognised by a T -morphism h into some finitary T -algebra. There exists a surjective T -morphism into a T -algebra*

$$\text{synt}f : \mathbf{A} \rightarrow \mathbf{A}_f,$$

called the syntactic morphism of f , which recognises f and which factors through every surjective T -morphism recognising f . Furthermore, $\text{synt}f$ is unique up to isomorphisms on \mathbf{A}_f .

Note that if \mathbf{A} itself is finitary, then f is recognised by the identity T -morphism on \mathbf{A} . Therefore, if a monad T is such that every T -algebra is finitary, then every colouring of a T -algebra has a syntactic morphism. This implies that every colouring has a syntactic morphism in monads such as the monad of finite words that corresponds to monoids, the monad of nonempty finite words that corresponds to semigroups, and several monads for describing finite trees that will be described later in the paper. Before proving the theorem, we give an example which shows how that a syntactic morphism might not exist in general.

Running Example 2. Consider the monad of ∞ -words and the language

$$L = \{a^{n_1}ba^{n_2}b \cdots : \text{the sequence } n_i \text{ is unbounded, i.e. } \limsup n_i = \infty.\}$$

We will prove that L does not have a syntactic morphism. Consider an equivalence relation \sim on natural numbers such that every equivalence class is finite. For example, \sim could identify all numbers that are between two consecutive powers of two. Define a function

$$h_{\sim} : \{a, b\}^{\infty} \rightarrow \underbrace{\mathbb{N} \cup (\mathbb{N}^2 \times \mathbb{N}/\sim)}_A \cup \{\perp, \top\}$$

as follows. If the input is infinite, then h_{\sim} returns \perp or \top depending on whether the input belongs to L . If the input has no b 's, then h_{\sim} returns the length. Finally, if the input contains at least one b , then h returns the triple consisting of: the number of a 's before the first b ; the number of a 's after the last b ; the equivalence class of the largest n such that the input has an infix $ba^n b$ (or the equivalence class of 0 if there is no such n). One can show that the kernel of h_{\sim} is a congruence in the natural sense, and therefore A can be equipped with the structure of an ∞ -algebra which makes h an ∞ -morphism recognising L .

Consider two equivalence relations \sim_1 and \sim_2 on natural numbers, such that their transitive closure has infinite equivalence classes, e.g. \sim_1 identifies even numbers with their successors, while \sim_2 identifies even numbers with their predecessors. If there were a syntactic morphism h , then it would need to factor through both h_{\sim_1} and h_{\sim_2} , and therefore it would need to assign the same

value to all words in ba^*b . By associativity, h would assign the same value to all ∞ -words with infinitely many b 's, and therefore it would not recognise L . \square

The rest of Section 3 is devoted to proving the Syntactic Morphism Theorem.

3.1 Proof of the Syntactic Morphism Theorem

We are working in a category of sorted sets; fix therefore a set of sort names, and a monad T . We first show that the syntactic morphism, if it exists, is unique up to isomorphisms on the target algebra. This is a consequence of the following lemma. In the lemma, the crucial distinction is between a function between universes of two T -algebras, and such a function which is a T -morphism, i.e. one that is consistent with the multiplication in the two algebras.

Lemma 3.2 *Let T be a monad, let $\mathbf{A}, \mathbf{B}, \mathbf{C}$ be T -algebras, let*

$$f : \mathbf{A} \rightarrow \mathbf{B} \quad \text{and} \quad g : \mathbf{A} \rightarrow \mathbf{C}$$

be T -morphisms, with f being surjective, and let $h : B \rightarrow C$ be a function such that, as functions on universes, the following diagram commutes.

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ & \searrow g & \downarrow h \\ & & C \end{array}$$

Then h is a T -morphism.

Proof.

This might be a standard lemma on Eilenberg-Moore algebras, although this proof uses right inverses, and it will therefore not work in every category. Consider the following diagram.

$$\begin{array}{ccccc} & & \mathsf{T}B & & \\ & \nearrow \mathsf{T}f & & \searrow \text{mul}_B & \\ \mathsf{T}A & \xrightarrow{\text{mul}_A} & A & \xrightarrow{f} & B \\ & \searrow \mathsf{T}g & & \downarrow g & \downarrow h \\ & & \mathsf{T}C & \xrightarrow{\text{mul}_C} & C \end{array}$$

$\mathsf{T}h$ (curved arrow from $\mathsf{T}A$ to $\mathsf{T}C$)

The right triangular face (involving A, B, C) commutes by assumption of the lemma, and the left triangular face (involving $\mathsf{T}A, \mathsf{T}B, \mathsf{T}C$) commutes by T applied to the assumption of the lemma. The two quadrangular faces commute because f and g are T -morphisms. Therefore, the entire diagram commutes. Because f is surjective, and we are in the category of sorted sets, f has a right inverse, i.e. a function $f^{-1} : B \rightarrow A$ such that $f \circ f^{-1}$ is the identity on B . By

the previous commuting diagram, the two paths from $\mathbb{T}A$ to C in the following diagram describe the same function.

$$\begin{array}{ccccc} \mathbb{T}B & \xrightarrow{\mathbb{T}f^{-1}} & \mathbb{T}A & \xrightarrow{\mathbb{T}f} & \mathbb{T}B & \xrightarrow{\text{mul}_{\mathbb{T}B}} & B \\ & & & \downarrow \mathbb{T}h & & \downarrow h & \\ & & & \mathbb{T}C & \xrightarrow{\text{mul}_{\mathbb{T}C}} & C \end{array}$$

Because \mathbb{T} is a functor, it follows that the path connecting the two copies of $\mathbb{T}B$ in the above diagram is actually the identity on $\mathbb{T}B$, and therefore the square face above commutes, which proves that h is a \mathbb{T} -morphism. \square

Congruences. Define a *congruence* in an \mathbb{T} -algebra \mathbf{A} to be a surjective function $g : A \rightarrow B$ from the universe of \mathbf{A} to some set such that $g \circ \text{mul}_{\mathbf{A}}$ factors through $\mathbb{T}g$.

Lemma 3.3 *If \mathbf{A} is a \mathbb{T} -algebra and $g : A \rightarrow B$ is a congruence, then there is a multiplication operation on B which makes g into a \mathbb{T} -morphism.*

Proof.

The assumption that g is a congruence says that there is a function, call it $\text{mul}_{\mathbf{B}}$, which makes the following diagram commute.

$$\begin{array}{ccc} \mathbb{T}A & \xrightarrow{\mathbb{T}g} & \mathbb{T}B \\ \text{mul}_{\mathbf{A}} \downarrow & & \downarrow \text{mul}_{\mathbf{B}} \\ A & \xrightarrow{g} & B \end{array}$$

To prove the lemma, we need to show that $\text{mul}_{\mathbf{B}}$ is associative, which is explained in the following diagram.

$$\begin{array}{ccccc} \mathbb{T}\mathbb{T}B & \xrightarrow{\mu_B} & \mathbb{T}B & & \\ \downarrow \mathbb{T}\text{mul}_B & \swarrow \mathbb{T}\mathbb{T}g & \searrow \mathbb{T}g & & \\ & \mathbb{T}\mathbb{T}A & \xrightarrow{\mu_A} & \mathbb{T}A & \\ & \downarrow \mathbb{T}\text{mul}_A & & \downarrow \text{mul}_A & \\ & \mathbb{T}A & \xrightarrow{\text{mul}_A} & A & \\ & \swarrow \mathbb{T}g & & \searrow g & \\ \mathbb{T}B & \xrightarrow{\text{mul}_B} & B & & \end{array}$$

The upper face commutes because $\mathbb{T}g$ is a \mathbb{T} -morphism between the free algebras $\mathbb{T}A$ and $\mathbb{T}B$. The right and lower faces commute by the assumption on $\text{mul}_{\mathbf{B}}$,

while the left face commutes by T applied to this assumption. It follows that all paths that begin in TTA and end in B denote the same function. Since g is surjective, we can use the same argument as in the end of Lemma 3.2 to show that the perimeter of the diagram commutes. \square

Therefore, a congruence is simply a T -morphism with the algebraic structure on the target being omitted.

Polynomials. In universal algebra, a polynomial is a term with some constants from the algebra. We generalise this notion to monads. For a set X , define the set of *polynomials over \mathbf{A} with variables X* to be

$$\mathsf{pol}_X \mathbf{A} \stackrel{\text{def}}{=} \mathsf{T}(A \sqcup X).$$

For a valuation $v : X \rightarrow A$, we consider the evaluation function

$$\llbracket _ \rrbracket(v) : \mathsf{pol}_X \mathbf{A} \rightarrow A$$

which first replaces the variables in the argument polynomial by the valuation v , and then applies the multiplication in \mathbf{A} . The notion of polynomials makes sense in arbitrary categories, not just those in sorted sets, but the following definition is specific to the category of sorted sets. Suppose that p is a polynomial over \mathbf{A} with variables X . Define

$$\llbracket p \rrbracket : \mathbf{A}^X \rightarrow \mathbf{A}$$

to be the function $v \mapsto \llbracket p \rrbracket(v)$. A problem is that although $\llbracket p \rrbracket$ is a well-defined function, it is not a morphism in the category, because it is not necessarily sort preserving. For example, if X has just one variable x , then the function $\llbracket p \rrbracket$ is sort preserving only when the (output) sort of p is the same as the sort of the variable x . In the language of category theory, this problem is that monads in sorted sets need not be strong. The problem goes away when there is only one sort.

If $h : \mathbf{A} \rightarrow \mathbf{B}$ is a T -morphism, and $p \in \mathsf{pol}_X \mathbf{A}$, then $h(p) \in \mathsf{pol}_X \mathbf{B}$ is defined by applying h to the constants in p and leaving the variables alone. Formally speaking $h(p)$ is obtained by applying $\mathsf{T}h'$ to p , where h' is the disjoint union of h and the identity on the variables. From the definition of T -morphism, it follows that the T -morphisms commute with polynomials in the sense that the following diagram commutes:

$$\begin{array}{ccc} \mathbf{A}^X & \xrightarrow{\llbracket p \rrbracket} & \mathbf{A} \\ h^X \downarrow & & \downarrow h \\ \mathbf{B}^X & \xrightarrow{\llbracket h(p) \rrbracket} & \mathbf{B} \end{array} \tag{1}$$

Note that the diagram is not in the category of sorted sets, because the horizontal arrows are not necessarily sort preserving.

Unary polynomials. In our proof of the Syntactic Morphism Theorem, special attention is devoted to unary polynomials. In the setting of (unsorted) sets, which covers the well-known versions of the Syntactic Morphism Theorem for monoids or finite automata, the classical construction is to identify elements that cannot be distinguished by unary polynomials. To define unary polynomials in the setting of sorted sets, one needs a little care with the sorts. In the following, we assume that the name of each sort is also an element of its own sort. For sort names τ and σ , a *unary polynomial with input sort τ and output sort σ over \mathbf{A}* is defined to be a polynomial over \mathbf{A} , which has sort σ , and which uses just one variable, namely the sort name τ . By abuse of notation, if τ is a sort name then we write $\text{pol}_\tau \mathbf{A}$ and $\llbracket p \rrbracket(a)$, respectively, instead of the formally correct $\text{pol}_{\{\tau\}} \mathbf{A}$ and $\llbracket p \rrbracket(\tau \mapsto a)$. In the setting of (unsorted) sets, there is only one sort and unary polynomials can be composed forming a monoid. In the setting of sorted sets, to compose unary polynomials one needs to take care that the output sort of one unary polynomial matches the input sort of the other.

Definition of the syntactic morphism. Consider a colouring

$$f : \mathbf{A} \rightarrow C,$$

as in the assumptions of the Syntactic Morphism Theorem. Define an equivalence relation \sim on the universe \mathbf{A} which identifies $a, b \in A$ if they have the same sort τ and

$$f(\llbracket p \rrbracket(a)) = f(\llbracket p \rrbracket(b)) \quad \text{for every } p \in \text{pol}_\tau \mathbf{A}.$$

Define A_f to be the equivalence classes of \sim , and define the syntactic morphism

$$\text{synt}f : A \rightarrow A_f$$

to be the function which maps a to its equivalence class under \sim . We will show that $\text{synt}f$ is a congruence, and therefore by Lemma 3.3 there is a multiplication operation on A_f image which makes $\text{synt}f$ into a surjective T-morphism. Let us begin by showing that \sim is a congruence with respect to polynomials with finitely many variables, as expressed in the following lemma.

Lemma 3.4 *Let X be a finite set of variables, and let $p \in \text{pol}_X \mathbf{A}$. If $v_1, v_2 : X \rightarrow A$ are valuations then*

$$\bigwedge_{x \in X} v_1(x) \sim v_2(x) \quad \text{implies} \quad \llbracket p \rrbracket(v_1) \sim \llbracket p \rrbracket(v_2).$$

Proof.

The idea is that the definition of \sim guarantees the lemma for unary polynomials, and then induction extends the result to polynomials of higher finite arities.

Consider first the case when X has exactly one variable, i.e. p is a unary polynomial. Let a_1, a_2 be the values of the valuations v_1, v_2 on the unique variable. We need to show that $a_1 \sim a_2$ implies

$$\llbracket p \rrbracket(a_1) \sim \llbracket p \rrbracket(a_2).$$

Unraveling the definition of \sim , we need to show that

$$(f \circ \llbracket q \rrbracket \circ \llbracket p \rrbracket)(a_1) = (f \circ \llbracket q \rrbracket \circ \llbracket p \rrbracket)(a_2)$$

holds for every unary polynomial q whose input sort is the output sort of p . Composing the polynomials q and p yields a unary polynomial r such that that $\llbracket q \rrbracket \circ \llbracket p \rrbracket$ and $\llbracket r \rrbracket$ describe the same function. By assumption that a_1, a_2 are \sim -equivalent, they have the same values under $f \circ \llbracket r \rrbracket$, which proves the above equality, and completes the proof of the special case of the lemma when X has one variable.

The case when X has more than one variable is proved by a straightforward induction on the size of X as follows. Let then $v_1, v_2 : X \rightarrow A$ be as in the assumption of the lemma. Choose some partition $X = X_1 \cup X_2$ with both X_i being nonempty, and define p_i for $i \in \{1, 2\}$ to be the polynomial obtained from p by substituting the variables from X_i with their values under v_i . Then

$$\llbracket p \rrbracket(v_1) = \llbracket p_1 \rrbracket(v_1|_{X_2}) \sim \llbracket p_1 \rrbracket(v_2|_{X_2}) = \llbracket p_2 \rrbracket(v_2|_{X_1}) \sim \llbracket p_2 \rrbracket(v_1|_{X_1}) = \llbracket p \rrbracket(v_2).$$

□

Let us restate a special case of the above lemma in terms of commuting diagrams.

Corollary 3.5 *If X is a finite set then*

$$\begin{array}{ccc} X & \xrightarrow{v_1} & A \\ v_2 \downarrow & & \downarrow \text{synt} f \\ A & \xrightarrow{\text{synt} f} & A_f \end{array} \quad \text{implies} \quad \begin{array}{ccccc} TX & \xrightarrow{\tau v_1} & TA & \xrightarrow{\text{mul}_A} & A \\ \tau v_2 \downarrow & & \downarrow & & \downarrow \text{synt} f \\ TA & & & & \\ \text{mul}_A \downarrow & & & & \\ A & \xrightarrow{\text{synt} f} & A_f \end{array}$$

Proof.

This is a special case of Lemma 3.4 where the polynomials have no constants in them, i.e. they are built entirely out of variables. □

Lemma 3.6 *If $h : \mathbf{A} \rightarrow \mathbf{B}$ is a T -morphism that recognises f , then $\text{synt} f$ factors through h .*

Proof.

The value of $\text{synt} f$ for an element of \mathbf{A} is determined by the values of $f \circ \llbracket p \rrbracket$ on the element, ranging over all unary polynomials p of appropriate input sort. Therefore, to prove the lemma it suffices to show that $f \circ \llbracket p \rrbracket$ factors through h for every unary polynomial p . This is the content of diagram (1) and the assumption that h recognises f . □

We now resume the proof of the Syntactic Morphism theorem. Recall the assumption that the coloring f is recognised by a morphism

$$h : \mathbf{A} \rightarrow \mathbf{B}$$

into a finitary \mathbf{T} -algebra. By Lemma 3.6, the syntactic morphism factors through h , and therefore there is a function $\text{synt}_{\mathbf{B}}f$ which makes the following diagram commute.

$$\begin{array}{ccc} \mathbf{A} & \xrightarrow{h} & \mathbf{B} \\ & \searrow \text{synt}f & \downarrow \text{synt}_{\mathbf{B}}f \\ & & A_f \end{array}$$

We will show in the following lemma that $\text{synt}_{\mathbf{B}}f$ is a congruence on \mathbf{B} . The lemma will complete the proof of the Syntactic Morphism Theorem, because by Lemma 3.3, there is a multiplication operation on A_f which makes it into an algebra \mathbf{A}_f such that $\text{synt}_{\mathbf{B}}f$ is a \mathbf{T} -morphism. Therefore, $\text{synt}f$ is a \mathbf{T} -morphism from \mathbf{A} to \mathbf{A}_f , as the composition of \mathbf{T} -morphisms $\text{synt}_{\mathbf{B}}f$ and h .

Lemma 3.7 *$\text{synt}_{\mathbf{B}}f$ is a congruence in \mathbf{B} .*

Proof.

By the assumption that we are in a category of sorted sets, and the assumption that $\text{synt}_{\mathbf{B}}f$ is surjective, there is a right inverse

$$\text{synt}_{\mathbf{B}}f^{-1} : A_f \rightarrow B,$$

i.e. a function such that $\text{synt}_{\mathbf{B}}f \circ \text{synt}_{\mathbf{B}}f^{-1}$ is the identity on A_f . Define

$$i \stackrel{\text{def}}{=} \text{synt}_{\mathbf{B}}f^{-1} \circ \text{synt}_{\mathbf{B}}f.$$

Similarly, let $h^{-1} : B \rightarrow A$ be a right inverse of h , i.e. a function such that $h \circ h^{-1}$ is the identity on B . From the definitions of h^{-1} and i we see that the following diagram commutes.

$$\begin{array}{ccccc} & & B & & \\ & \nearrow i & \downarrow \text{synt}_{\mathbf{B}}f & \searrow h^{-1} & \\ A & \xrightarrow{h} B & & & A \\ & \searrow \text{synt}_{\mathbf{B}}f & \downarrow \text{synt}_{\mathbf{B}}f & \nearrow \text{synt}f & \\ & & A_f & & \end{array} \quad (2)$$

Later in the proof, we will use the above commuting diagram to show that that the assumptions of Corollary 3.5 are satisfied, when the mappings v_1, v_2 from the Corollary are the identity and $h^{-1} \circ i \circ h$, restricted to a finite subset of A .

We will prove that $\mathsf{T}i$ does not affect the value under $\mathsf{synt}_{\mathbf{B}}f \circ \mathsf{mul}_{\mathbf{B}}$, i.e. that the following diagram commutes

$$\begin{array}{ccc}
 \mathsf{T}B & \xrightarrow{\mathsf{T}i} & \mathsf{T}B \\
 \mathsf{mul}_{\mathbf{B}} \downarrow & & \downarrow \mathsf{mul}_{\mathbf{B}} \\
 B & & B \\
 \searrow \mathsf{synt}_{\mathbf{B}}f & & \downarrow \mathsf{synt}_{\mathbf{B}}f \\
 & & A_f
 \end{array} \tag{3}$$

Before proving that the diagram above commutes, we show how it implies the statement of the lemma. The statement is that $\mathsf{synt}_{\mathbf{B}}f$ is a congruence, which means that if $w, w' \in \mathsf{T}B$ have the same image under $\mathsf{Tsynt}_{\mathbf{B}}f$, then they have the same image under $\mathsf{synt}_{\mathbf{B}}f \circ \mathsf{mul}_{\mathbf{B}}$. By definition i factors through $\mathsf{synt}_{\mathbf{B}}f$, and therefore if w, w' have the same image under $\mathsf{Tsynt}_{\mathbf{B}}f$, then they have the same image under $\mathsf{T}i$, and therefore they have the same image under $\mathsf{synt}_{\mathbf{B}}f \circ \mathsf{mul}_{\mathbf{B}}$ thanks to (3).

To prove (3), we use the assumption that \mathbf{B} is finitary, i.e. every structure in $\mathsf{T}B$ already belongs to $\mathsf{T}B_0$ for some finite subset $B_0 \subseteq B$. Therefore, to prove that the above diagram commutes, it suffices to prove that it commutes when the upper left $\mathsf{T}B$ is replaced by $\mathsf{T}B_0$ for some finite B_0 . Let then B_0 be a finite subset of B . Define A_0 to be the image of B_0 under h^{-1} , and define $j : A \rightarrow A$ to be the restriction of $h^{-1} \circ i \circ h$ to A_0 .

$$\begin{array}{ccccc}
 & \mathsf{T}B_0 & & & \\
 & \downarrow \mathsf{T}h^{-1} & & & \\
 & \mathsf{T}A_0 & \xrightarrow{\mathsf{T}j} & \mathsf{T}A & \\
 & \downarrow \mathsf{T}h & \nearrow \mathsf{T}h^{-1} & \downarrow \mathsf{T}h & \\
 \mathsf{mul}_{\mathbf{A}} \swarrow & \mathsf{T}B_0 & \xrightarrow{\mathsf{T}i} & \mathsf{T}B & \searrow \mathsf{mul}_{\mathbf{A}} \\
 & \downarrow \mathsf{mul}_{\mathbf{B}} & & \downarrow \mathsf{mul}_{\mathbf{B}} & \\
 A & \xrightarrow{h} & B & & B \xleftarrow{h} A \\
 & \searrow \mathsf{synt}f & \searrow \mathsf{synt}_{\mathbf{B}}f & \swarrow \mathsf{synt}_{\mathbf{B}}f & \swarrow \mathsf{synt}f \\
 & & A_f & &
 \end{array}$$

We claim that all paths that begin in the upper $\mathsf{T}B_0$ and end in A_f denote the same function. Thanks to (2), the functions j and the identity on A_0 satisfy the assumptions in Corollary 3.5. By the Corollary, the two paths on the perimeter that begin in $\mathsf{T}B_0$ and end in A_f describe the same function. The two upper quadrangular face commutes by definition of j . The face which uses $\mathsf{T}B$ twice commutes because $h \circ h^{-1}$ is the identity on B . The two faces which use $\mathsf{mul}_{\mathbf{A}}$

commute because h is a T -morphism. The two triangular faces commute by definition of $\text{synt}_{\mathbf{B}}f$.

Since $Th \circ Th^{-1}$ is the identity on TB_0 , we have proved that the diagram (3) commutes assuming that the top left corner is replaced by TB_0 ; and therefore by the assumption on \mathbf{B} being finitary we have proved that the diagram (3) commutes in general. \square

4 Pseudovarieties

This section is dedicated to a monad version of Eilenberg's pseudovariety theorem. Eilenberg's theorem says that, in the case of semigroups, language pseudovarieties and algebra pseudovarieties, which will be defined below, are in bijective correspondence. The theorem implies that if \mathbb{L} is a language pseudovariety, then the membership problem $L \in \mathbb{L}$ can be decided only by looking at the syntactic semigroup of L , and one need not look at the accepting set, nor at the information about which letters are mapped to which elements of the semigroup. A typical application of the pseudovariety theorem is that definability in first-order logic, or various fragments thereof, can be determined based only on the syntactic monoid. The theorem does not give an algorithm to determine this, the algorithm needs to be found in a case-by-case way.

In this section we prove that the pseudovariety theorem works in general for monads when the category is (possibly sorted) sets, with the same proof as in the case of monoids. Surely Eilenberg must have known this, since he invented both the pseudovariety theorem and algebras in abstract monads, but I have not found this result in his book [Eil74]. Our generalised pseudovariety theorem subsumes pseudovariety theorems for: finite words in both monoid and semigroup variants [Eil74], ∞ -words [Wil91], scattered linear orderings [BR12], finite trees [Ste92]; it also gives pseudovariety theorems for other known settings which have not had their pseudovariety theorems yet, such as forest algebra.

Algebra pseudovarieties. The definition of an algebra pseudovariety is a straightforward generalisation of the definition given by Eilenberg for semigroups or monoids. It is a class of finite algebras, according to the notion of finiteness given in the setting, which is closed under products, morphic images and subalgebras, as defined below in more detail.

- **Products.** A class of T -algebras is called *closed under products* if whenever \mathbf{A}, \mathbf{B} are in the class, then so is $\mathbf{A} \times \mathbf{B}$.
- **Morphic images.** A class of T -algebras is called *closed under morphic images* if whenever $h : \mathbf{A} \rightarrow \mathbf{B}$ is a surjective T -morphism and \mathbf{A} is in the class, then so is \mathbf{B} .
- **Subalgebras.** A class of T -algebras is called *closed under subalgebras* if whenever \mathbf{A} is in the class, then every subalgebra of \mathbf{A} is in the class. \mathbf{A}

subalgebra of \mathbf{A} is obtained by restricting the universe to a subset B such that $\text{mul}_{\mathbf{A}}$ maps elements of $\mathsf{T}B$ to B .

- **Algebra pseudovariety.** A class of finite T -algebras is called an *algebra pseudovariety* if it has all three closure properties defined above.

Running Example 3. Call an ∞ -algebra \mathbf{A} *definite* if the multiplication operation

$$\text{mul}_{\mathbf{A}} : A^{\infty} \rightarrow A$$

is such that the value of the multiplication depends only on the first n letters of the argument, for some n depending only on the algebra. Definite ∞ -algebras are easily seen to form an algebra pseudovariety. \square

Language pseudovarieties. Unlike for algebras, the notion of language pseudovariety requires some discussion. For intuition, let us recall the original notion of language pseudovariety for semigroups that was introduced by Eilenberg. Eilenberg defines a language pseudovariety for semigroups to be a class of recognisable languages of finite words which is closed under Boolean combinations, inverse images of semigroup morphisms $h : \Sigma^+ \rightarrow \Gamma^+$, and derivatives. Here a derivative of a language $L \subseteq \Sigma^+$ is defined to be any language of the form

$$w^{-1}Lv^{-1} \stackrel{\text{def}}{=} \{u \in \Sigma^+ : wuv \in L\}.$$

for some $w, v \in \Sigma^*$.

It is not immediately obvious how to generalise the notion of derivative to abstract monads. We propose two solutions: one using unary polynomials, which we call a polynomial derivative, and one using syntactic morphisms, which we call a syntactic derivative. The advantage of polynomial derivatives is that they are closer to the derivatives used by Eilenberg, while the advantage of syntactic derivatives is that they make sense in settings without a clear notion of unary polynomials (recall that unary polynomials have only been defined for sorted sets). The two notions of derivative lead to different notions of language pseudovariety, which happen to coincide in settings that use sorted sets. The precise definitions are given below.

- **Boolean combinations.** A class of T -languages is called *closed under Boolean combinations*, if whenever it contains languages $L \subseteq \mathsf{T}\Sigma$ and $K \subseteq \mathsf{T}\Gamma$, then it also contains

$$L \cap K \quad L \cup K \quad \mathsf{T}\Sigma - L$$

Of course, in the presence of complementation, only one of \cup, \cap is needed.

- **Morphic preimages.** A class of T -languages is called *closed under morphic preimages* if whenever the class contains a language $L \subseteq \mathsf{T}\Sigma$ and $h : \mathsf{T}\Gamma \rightarrow \mathsf{T}\Sigma$ is a T -morphism with Γ being a finite alphabet, then the class also contains $h^{-1}(L)$.

- **Polynomial derivatives.** (This definition assumes that the setting has a notion of unary polynomial, which have only been defined for sorted sets in this paper.) A class of \mathbf{T} -languages is called *closed under polynomial derivatives* if whenever the class contains a language $L \subseteq \mathbf{T}\Sigma$ and p is a unary polynomial in the \mathbf{T} -algebra $\mathbf{T}\Sigma$, then the class also contains the language

$$p^{-1}L \stackrel{\text{def}}{=} \{w \in \mathbf{T}\Sigma : \llbracket p \rrbracket(w) \in L\}.$$

- **Syntactic derivatives.** A class of \mathbf{T} -languages is called *closed under polynomial derivatives* if whenever the class contains a language, then it also contains all other languages recognised by its syntactic algebra.
- **Polynomial language pseudovariety.** A *polynomial language pseudovariety* is a class of recognisable \mathbf{T} -languages that is closed under Boolean combinations, morphic preimages, and polynomial derivatives.
- **Syntactic language pseudovariety.** A *polynomial language pseudovariety* is a class of recognisable \mathbf{T} -languages that is closed under Boolean combinations, morphic preimages, and syntactic derivatives. (Since complementation is covered by syntactic derivatives, it suffices to have only closure under union and not all Boolean combinations.)

As usual for pseudovarieties, a \mathbf{T} -language in the above definitions is formally treated as its characteristic function, which means that a language comes with a description of its domain. The reason for this is that it is sometimes important to know the input alphabet of a language. Before continuing, let us observe the following simple fact.

Fact 4.1 *Polynomial derivatives are a special case of syntactic derivatives.*

Proof.

Thanks to (1), any \mathbf{T} -morphism, not necessarily the syntactic morphism, which recognises L will also recognise every polynomial derivative $p^{-1}L$. \square

Running Example 4. Call an ∞ -language *definite* if there is some $n \in \mathbb{N}$ such that membership in the language depends only on the first n letters. Examples of definite ∞ -languages include: “words that begin with a ”, or “words of length at least two”. Clearly definite ∞ -languages are closed under Boolean combinations. They are also closed under inverse images of ∞ -morphisms, because if

$$h : \Sigma^\infty \rightarrow \Gamma^\infty$$

is an ∞ -morphism, then the first n letters of $h(w)$ are uniquely determined by the first n letters (or less) of w . Here it is important that the monad of ∞ -words does not allow the empty word; there is a natural variant of the monad which does have the empty word, and in this variant the definite ∞ -languages do not form a pseudovariety.

The same argument as for ∞ -morphisms applies to functions $\Sigma^\infty \rightarrow \Sigma^\infty$ defined by unary polynomials. Therefore, definite ∞ -languages are closed under polynomial derivatives as well. Summing up, definite ∞ -languages form a polynomial language pseudovariety. Definite ∞ -languages also form a syntactic language pseudovariety, but this takes a little more effort to check, and will follow from Corollary 4.5. \square

4.1 The Syntactic Pseudovariety Theorem.

We have defined two versions of language pseudovarieties, syntactic and polynomial, and therefore there will be two version of Pseudovariety Theorem. In this section we present the version which talks about syntactic varieties. The proof is essentially a monad version of half of Eilenberg's proof, because the definition of syntactic derivative eliminates the other half. A closer similarity with Eilenberg's full theorem is the polynomial version, which is presented in the next section, but which comes at the cost of restricting to settings that use sorted sets.

For a class \mathbb{L} of recognisable \mathbf{T} -languages, define $\mathbf{Alg} \mathbb{L}$ to be the class of finite \mathbf{T} -algebras which only recognise \mathbf{T} -languages from \mathbb{L} . For a class \mathbb{A} of finite \mathbf{T} -algebras, define $\mathbf{Lan} \mathbb{A}$ to be the \mathbf{T} -languages recognised by \mathbf{T} -algebras from \mathbb{A} . The Pseudovariety Theorem says that these mappings are mutual bijections when restricted to pseudovarieties.

Theorem 4.2 [*Syntactic Pseudovariety Theorem*] *Consider a setting with the following properties.*

- *Every recognisable language has a syntactic morphism;*
- *Every finite algebra is finitely generated, i.e. its universe has a finite subset G such that multiplication is surjective when restricted to $\mathbf{T}G$;*
- *Every finite algebra \mathbf{A} has a finite subset of its universe A_0 with the following property. If $h : \mathbf{A} \rightarrow \mathbf{B}$ is a surjective \mathbf{T} -morphism which is injective when restricted to A_0 , then h is an isomorphism.*

Then the mapping \mathbf{Lan} is a bijection between algebra pseudovarieties and syntactic language pseudovarieties, and its inverse is \mathbf{Alg} .

The rest of Section 4.1 is devoted to proving the Syntactic Pseudovariety Theorem. We begin by showing that \mathbf{Alg} and \mathbf{Lan} produce pseudovarieties when given pseudovarieties (of appropriate types, respectively); actually not all closure properties are needed for this part. If \mathbb{L} is a syntactic language pseudovariety, then $\mathbf{Alg} \mathbb{L}$ is easily seen to be an algebra pseudovariety. Actually, to prove this, we only need to assume that \mathbb{L} is closed under Boolean combinations. This is because every \mathbf{T} -language recognised by $\mathbf{A} \times \mathbf{B}$ is a Boolean combination of \mathbf{T} -languages recognised by \mathbf{A} and \mathbf{B} . If \mathbb{A} is any class of finite \mathbf{T} -algebras, in particular an algebra pseudovariety, then $\mathbf{Lan} \mathbb{A}$ is easily to be a syntactic language pseudovariety.

To finish the proof of the Syntactic Pseudovariety Theorem, it remains to show that if \mathbb{L} and \mathbb{A} are pseudovarieties of \mathbf{T} -languages and \mathbf{T} -algebras respectively, then

$$\text{Alg Lan } \mathbb{A} = \mathbb{A} \quad \text{and} \quad \text{Lan Alg } \mathbb{L} = \mathbb{L}.$$

By definition, the class $\text{Lan Alg } \mathbb{L}$ consists of \mathbf{T} -languages that are recognised by some finite \mathbf{T} -algebra which only recognises \mathbf{T} -languages from \mathbb{L} . Therefore

$$\text{Lan Alg } \mathbb{L} \subseteq \mathbb{L}.$$

For the converse inclusion, consider a language $L \in \mathbb{L}$. By assumption on the setting, L has a syntactic algebra, and by definition of language pseudovarieties, every language recognised by this syntactic algebra belongs to \mathbb{L} . Therefore, L is recognised by some algebra which only recognises languages from \mathbb{L} . Here we have profited from the definition of syntactic derivatives; with polynomial derivatives this part of the proof will need to be more involved.

More effort is required for the equality

$$\text{Alg Lan } \mathbb{A} = \mathbb{A}.$$

By definition, the class $\text{Alg Lan } \mathbb{A}$ consists of finite \mathbf{T} -algebras \mathbf{A} such that every finitely sorted \mathbf{T} -language recognised by \mathbf{A} is recognised by some \mathbf{T} -algebra from \mathbb{A} . This gives the right-to-left inclusion. The converse inclusion is proved in the following lemma.

Lemma 4.3 *Let \mathbf{A} be a finite \mathbf{T} -algebra such that every \mathbf{T} -language recognised by \mathbf{A} is recognised by some \mathbf{T} -algebra from \mathbb{A} . Then $\mathbf{A} \in \mathbb{A}$.*

Proof.

Let G be a finite generating subset of the universe of \mathbf{A} , i.e. a subset such that $\text{mul}_{\mathbf{A}}$ is surjective when restricted to $\mathbf{T}G$. By the assumptions on the setting, there is a finite subset A_0 of the universe of \mathbf{A} such that if a surjective \mathbf{T} -morphism $f : \mathbf{A} \rightarrow \mathbf{B}$ is injective on A_0 then it is an isomorphism. For $a \in A_0$ define

$$L_a = \{w \in \mathbf{T}G : \text{mul}_{\mathbf{A}}(w) = a\}.$$

Let the syntactic morphism of L_a be

$$h_a : \mathbf{T}G \rightarrow \mathbf{B}_a.$$

The syntactic morphism exists because L_a is recognised by a finite algebra, namely \mathbf{A} , and therefore the Syntactic Morphism Theorem can be applied. Furthermore, by the assumption of the lemma, L_a , like any language recognised by \mathbf{A} , is also recognised by some algebra from \mathbb{A} . Therefore, the syntactic algebra \mathbf{B}_a is an image of some algebra in \mathbb{A} , and therefore itself belongs to \mathbb{A} by closure of algebra pseudovarieties under morphic images. Using the definition

of syntactic morphism again, the syntactic morphism h_a must factor through $\text{mul}_{\mathbf{A}}$. Summing up, $\mathbf{B}_a \in \mathbb{A}$ and there is a surjective morphism f_a which makes the following diagram commute.

$$\begin{array}{ccc} \mathsf{T}G & \xrightarrow{\text{mul}_{\mathbf{A}}} & \mathbf{A} \\ & \searrow h_a & \downarrow f_a \\ & & \mathbf{B}_a \end{array}$$

Define h to be the product of the morphisms h_a ranging over $a \in A_0$, which is surjective onto its image

$$h : \mathsf{T}G \rightarrow \mathbf{B} \subseteq \prod_{a \in A_0} \mathbf{B}_a.$$

The algebra \mathbf{B} belongs to \mathbb{A} , by closure of \mathbb{A} under finite products and subalgebras. Defining f to be the product of all f_a , we see that the following diagram commutes.

$$\begin{array}{ccc} \mathsf{T}G & \xrightarrow{\text{mul}_{\mathbf{A}}} & \mathbf{A} \\ & \searrow h & \downarrow f \\ & & \mathbf{B} \end{array}$$

To prove that f is actually an isomorphism, it suffices to show that f is injective when restricted to A_0 . Because G are generators, every element $a \in A_0$ is the image under $\text{mul}_{\mathbf{A}}$ of some $w_a \in \mathsf{T}G$. Furthermore, if $a \neq b$, then $h(w_a) \neq h(w_b)$, because only one of w_a, w_b belongs to the language L_a that is recognised by h . \square

4.2 The Polynomial Pseudovariety Theorem

In this section, we prove that if the setting uses sorted sets with finitely many sorts, then both the syntactic and polynomial versions of language pseudovariety coincide. The key result is Lemma 4.4 below, which says that syntactic derivatives can be represented as inverse morphic images of Boolean combinations of polynomial derivatives. This lemma is essentially the other half of Eilenberg's proof, which was not used in the syntactic version of the pseudovariety theorem. We state Lemma 4.4 in the more general setting with possibly infinitely many sorts. In such a setting, call a language finitely sorted if on all but finitely many sorts it is full or empty. When there are finitely many sorts, then all languages are finitely sorted.

Lemma 4.4 *Consider a setting where the category is sorted sets, with possibly infinitely many sorts, and that the notion of finite algebra is such that finiteness of an algebra implies that the universe is finite on every sort. Then for every*

recognisable T -language L , every finitely sorted syntactic derivative of L is an inverse image, under some T -morphism, of a Boolean combination of polynomial derivatives of L .

Before proving the above lemma, let us note two corollaries, and an example of a setting where the conclusion of the lemma is violated.

Corollary 4.5 *Consider a setting where the category is sorted sets with finitely many sorts, finite alphabets are finite sorted sets, and the notion of finite algebra is such that finiteness of an algebra implies that the universe is finite. Then syntactic language pseudovarieties are the same thing as polynomial language pseudovarieties.*

Proof.

Lemma 4.4 implies that every polynomial language pseudovariety is closed under syntactic derivatives, and is therefore a syntactic language pseudovariety. The converse is Fact 4.1, which says that closure under syntactic derivatives implies closure under polynomial derivatives. \square

Actually, the above corollary would also be true with infinitely many sorts, with a modified definition of language pseudovariety where only finitely sorted languages are allowed. By combining the above corollary with the Syntactic Pseudovariety Theorem, we get the Polynomial Pseudovariety Theorem stated below.

Corollary 4.6 [*Polynomial Pseudovariety Theorem*] *Under assumptions on the setting as in Corollary 4.5, Lan is a bijection between T -algebra pseudovarieties and polynomial T -language pseudovarieties, and its inverse is Alg .*

An advantage of the polynomial version of the pseudovariety theorem is that it is sometimes easier to check if a class is closed under polynomial derivatives, as compared to syntactic derivatives. This was the case for definite ∞ -languages discussed previously in the running example.

Running Example 5. As an illustration of the Polynomial Pseudovariety Theorem, it is easy to see that Alg takes the class of definite ∞ -languages to the class of definite ∞ -algebras, and the mapping Lan goes the other way. An ∞ -language is definite if and only if its syntactic ∞ -algebra is definite. \square

Here is an example of a setting which violates the conclusions of the Polynomial Pseudovariety Theorem, and therefore also the conclusions of Lemma 4.4.

Example 2. In the proof of the Lemma 4.4, we will use the following property of the category of sorted sets: if $g : X \rightarrow Y$ is surjective, then there is an inverse $g^{-1} : Y \rightarrow X$ such that $g \circ g^{-1}$ is the identity on Y . An example of a category where this assumption fails is nominal sets. In nominal sets, the Polynomial Pseudovariety Theorem also fails, as we show in this example. The example assumes familiarity with nominal sets, and orbit-finite sets.

Consider the category where objects are finitely supported nominal sets and morphisms are finitely supported functions. Consider the monad of finite words in this category, where algebras are finitely supported semigroups. To complete the definition of the setting, define finite alphabets to be finitely supported sets which are orbit-finite, and define finite algebras to be finitely supported orbit-finite semigroups. This setting was studied in [Boj13], although not using the monad terminology. The Syntactic Morphism Theorem holds in this setting, as was shown in Lemmas 3.3 and 3.4 of [Boj13]. We will show that the Pseudovariety Theorem fails in this setting.

Here is the property of finitely supported functions that will make the Polynomial Pseudovariety Theorem fail. Let \mathbb{A} denote the atoms underlying the nominal sets, let $P_2\mathbb{A}$ be size two sets of atoms, i.e. unordered pairs of atoms. One can show that if

$$f : P_2\mathbb{A} \rightarrow \mathbb{A}^+$$

is a function supported by a finite set S of atoms,

$$f(\{a, b\}) = f(\{c, d\}) \quad \text{for } a, b, c, d \notin S. \quad (4)$$

Define \mathbb{L} to be the polynomial language pseudovariety generated by all recognisable languages over the alphabet \mathbb{A} . It is not difficult to see that a language $L \subseteq \Sigma^+$ belongs to \mathbb{L} if and only if there is a finitely supported monoid morphism

$$h : \Sigma^+ \rightarrow \mathbb{A}^+$$

such that L is an inverse image under h of some recognisable subset of \mathbb{A}^+ . We will show that $\text{Alg } \mathbb{L}$ is not an algebra pseudovariety, because it is not closed under morphic images.

Consider the following two languages.

1. The alphabet is ordered pairs of atoms, i.e. \mathbb{A}^2 . The language consists of two letter words over this alphabet such that the two atoms which appear in the first letter are pairwise distinct from the two atoms that appear in the second letter. In other words, this language is

$$L_1 = \{(a, b)(c, d) : \{a, b\} \cap \{c, d\} = \emptyset\} \subseteq (\mathbb{A}^2)^+$$

This language is recognised by a semigroup, call it \mathbf{S}_1 , whose universe is

$$\mathbb{A}^2 \cup \{\top, \perp\},$$

with elements of \mathbb{A}^2 describing one letter words, with \top describing words in the language, and with \perp describing words of length at least two that are outside the language. Although \mathbf{S}_1 recognises the language L_1 , it is not its syntactic semigroup. To get the syntactic semigroup, one needs to identify ordered pairs that correspond to the same set, i.e. the syntactic semigroup, call it \mathbf{S}_2 , has universe

$$P_2\mathbb{A} \cup \{\top, \perp\}.$$

Clearly \mathbf{S}_2 is an image of \mathbf{S}_1 under a finitely supported semigroup morphism, namely the function which forgets the order in pairs. Therefore, any algebra pseudovariety with \mathbf{S}_1 will also contain \mathbf{S}_2 .

2. Here is a language that is recognised by \mathbf{S}_2 . The alphabet is unordered pairs of atoms, i.e. $\mathbf{P}_2\mathbb{A}$. The language consists of two letter words over this alphabet such that the set in the first letter is disjoint with the set in the second letter. In other words, this language is

$$L_2 = \{\{a, b\}\{c, d\} : \{a, b\} \cap \{c, d\} = \emptyset\} \subseteq (\mathbf{P}_2\mathbb{A})^+$$

We claim that $\text{Alg } \mathbb{L}$ contains \mathbf{S}_1 but not \mathbf{S}_2 , and is therefore not an algebra pseudovariety. It is not difficult to show that \mathbf{S}_1 recognises only languages from \mathbb{L} , and therefore it belongs to $\text{Alg } \mathbb{L}$. We only show that L_2 is not in \mathbb{L} , and therefore \mathbf{S}_2 is not in $\text{Alg } \mathbb{L}$. To this end, we need to show that there is no finitely supported semigroup morphism

$$h : (\mathbf{P}_2\mathbb{A})^+ \rightarrow \mathbb{A}^+$$

such that L_2 is an inverse image of some recognisable subset of \mathbb{A}^+ . Indeed, by (4), the function h would need to assign the same value to two different letters in $\mathbf{P}_2\mathbb{A}$, and therefore it could not recognise L_2 . \square

The rest of this section is devoted to proving Lemma 4.4.

Lemma 4.7 *Assume the assumptions of Lemma 4.4. Let L be a recognisable T-language. Every finitely sorted language recognised by the syntactic morphism of L is a Boolean combination of polynomial derivatives of L .*

Proof.

Let $L \subseteq \mathbf{T}\Sigma$ be a recognisable T-language. Let K be a finitely sorted language recognised by the syntactic morphism of L , in particular the K is also a subset of $\mathbf{T}\Sigma$. We want to show that K is a Boolean combination of derivatives of L . A finitely sorted language is a finite union of single-sorted languages, and therefore without loss of generality, we can assume that K entirely included in a single sort, call it τ .

Claim 4.7.1 *There is a finite set $P \subseteq \text{pol}_\tau \mathbf{T}\Sigma$ such that structures $w, w' \in \mathbf{T}\Sigma$ of sort τ have the same image under the syntactic morphism of L if and only if*

$$p(w) \in L \quad \text{iff} \quad p(w') \in L \quad \text{for every } p \in P. \quad (5)$$

Proof.

By construction of the syntactic morphism in the proof of the Syntactic Morphism Theorem, structures w, w' have the same image under the syntactic morphism if and only if (5) holds for every polynomial $p \in \text{pol}_\tau \mathbf{T}\Sigma$, not necessarily from some finite set P . In other words, one can choose for every w, w' a polynomial $p_{w, w'}$ such that w and w' have the same image under the syntactic morphism if and only if

$$\llbracket p_{w, w'} \rrbracket(w) \in L \quad \text{iff} \quad \llbracket p_{w, w'} \rrbracket(w') \in L.$$

Furthermore, because the syntactic morphism recognises L , the choice of $p_{w,w'}$ need need only depend on the images of w and w' under the syntactic morphism, for which there are finitely many possibilities. \square

Stated differently, the claim says that structures in sort τ have the same image under the syntactic morphism if and only if they belong to the same polynomial derivatives $p^{-1}L$ for p belonging to the finite set P in the statement of the claim. This means that K , being a subset of sort τ that is recognised by the syntactic morphism, is a Boolean combination of finitely many derivatives. \square

Lemma 4.8 *Let Γ be a set and let $f : \mathbf{A} \rightarrow \mathbf{B}$ and $h : \mathsf{T}\Sigma \rightarrow \mathbf{B}$ be T -morphisms. If f is surjective, then there is some T -morphism g which makes the following diagram commute*

$$\begin{array}{ccc} \mathsf{T}\Sigma & \xrightarrow{g} & \mathbf{A} \\ & \searrow h & \downarrow f \\ & & \mathbf{B} \end{array}$$

Proof.

Because f is surjective, there is a function g' which makes the following diagram commute.

$$\begin{array}{ccccc} \mathbf{A} & \xleftarrow{g'} & \Sigma & \xrightarrow{\eta_\Sigma} & \mathsf{T}\Sigma \\ & \searrow f & & \searrow h & \\ & & \mathbf{B} & & \end{array}$$

Consider the following diagram.

$$\begin{array}{ccccc} & & \mathsf{T}A & & \\ & \swarrow \text{mul}_{\mathbf{A}} & \uparrow \eta_A & \nwarrow \mathsf{T}g' & \\ & \mathbf{A} & A & & \mathsf{T}\Sigma \\ & \swarrow g' & \uparrow g' & \nwarrow \eta_\Sigma & \\ \mathbf{A} & \xleftarrow{g'} & \Sigma & \xrightarrow{\eta_\Sigma} & \mathsf{T}\Sigma \\ & \searrow f & & \searrow h & \\ & & \mathbf{B} & & \end{array}$$

By definition of g' , the lower face commutes. The upper left face commutes because multiplication in an algebra must maps units to themselves, while the upper right face comes from the assumption that the unit in a monad is a natural transformation. All arrows on the perimeter of the diamond-shaped diagram

describe \mathbf{T} -morphisms. Therefore, both paths which begin with the edge η_Σ and end in \mathbf{B} describe the same function. Because $\mathbf{T}\Sigma$ is generated by the units of Σ , and both paths from $\mathbf{T}\Sigma$ to \mathbf{B} are (compositions of) \mathbf{T} -morphisms, it follows that both paths from $\mathbf{T}\Sigma$ to \mathbf{B} describe the same \mathbf{T} -morphism. Therefore, g in the statement of the lemma can be taken to be $\text{mul}_{\mathbf{A}} \circ \mathbf{T}g'$. \square

Proof. (of Lemma 4.4)

The lemma says that if $L \subseteq \mathbf{T}\Gamma$ is a recognisable \mathbf{T} -language, then every finitely sorted syntactic derivative of L is an inverse image, under some \mathbf{T} -morphism, of a Boolean combination of polynomial derivatives of L . Let the syntactic morphism of L be

$$f : \mathbf{T}\Gamma \rightarrow \mathbf{B}.$$

Suppose that $K \subseteq \mathbf{T}\Sigma$ is a finitely sorted syntactic derivative of L , i.e. it is recognised by some \mathbf{T} -morphism

$$h : \mathbf{T}\Sigma \rightarrow \mathbf{B}.$$

By Lemma 4.8, there is a \mathbf{T} -morphism g which makes the following diagram commute.

$$\begin{array}{ccc} \mathbf{T}\Sigma & \xrightarrow{g} & \mathbf{T}\Gamma \\ & \searrow h & \downarrow f \\ & & \mathbf{B} \end{array}$$

In other words, K is an inverse image, under g , of some language M recognised by the syntactic morphism f . We can assume without loss of generality that M is empty (respectively, full) on sorts where K is empty (respectively, full), and therefore M is also finitely sorted. By Lemma 4.7, M is a Boolean combination of polynomial derivatives of L . \square

This completes the proof of the Polynomial Pseudovariety Theorem.

5 Representing an algebra

In all interesting cases, the monad \mathbf{T} produces infinite sets, even on finite arguments. Therefore, the finiteness of the universe of a \mathbf{T} -algebra \mathbf{A} does not, on its own, imply that the algebra itself has a finite representation, because one needs some way of representing the algebra's multiplication operation

$$\text{mul}_{\mathbf{A}} : \mathbf{T}A \rightarrow A.$$

In this section, we present one such way. We assume that the monad is in the category of sets, or sorted sets. The idea is to find a function \mathbf{T}_0 , which chooses for every finite set A a finite subset $\mathbf{T}_0 A \subseteq \mathbf{T}A$ such that:

1. for every finite T -algebra \mathbf{A} with universe A , the multiplication operation is uniquely determined by its values on $\mathsf{T}_0 A$;
2. the function $A \mapsto \mathsf{T}_0 A$ can be computed, modulo some representation of elements in $\mathsf{T}_0 A \subseteq \mathsf{T} A$.

For instance, in the monad of finite words, the function T_0 maps a set A to word over A of length two, because a semigroup is uniquely determined by its neutral element and its binary multiplication. In the example of ∞ -algebras, the function T_0 maps A to words over A of length two and to infinite words of the form a^ω for some $a \in A$. We now describe these notions in more detail.

Subfunctors. Because the monad is in the category of sets, or sorted sets, the notion of subset can be used. Define a *subfunctor* of a monad T to be a mapping which takes every set X to a subset $\mathsf{T}_0 X \subseteq \mathsf{T} X$. A subfunctor on its own is not a monad (as defined here it is not even a functor), however it can be used to generate a monad as follows. For an ordinal number α , define $\mathsf{T}_0^\alpha X \subseteq \mathsf{T} X$ as follows by transfinite induction: $\mathsf{T}_0^0 X$ is the units of X , while for $\alpha > 0$ we have

$$\mathsf{T}_0^\alpha X \stackrel{\text{def}}{=} \bigcup_{\beta < \alpha} \text{mul}_{\mathsf{T} X} \mathsf{T}_0 \mathsf{T}_0^\beta X.$$

By monotonicity, this sequence must stabilise at some value, which is denoted by $\mathsf{T}_0^* X$. If the monad is finitary, i.e. every element $w \in \mathsf{T} X$ belongs to $w \in \mathsf{T} Y$ for some finite $Y \subseteq X$, then the sequence stabilises at ω , i.e. induction only on natural numbers is needed. It is not difficult to show that T_0^* is a submonad of T , i.e. a subfunctor with the monad structure inherited from T . A subfunctor T_0 is said to *span* an algebra \mathbf{A} if

$$\text{mul}_{\mathbf{A}} \mathsf{T}_0^* X = \text{mul}_{\mathbf{A}} \mathsf{T} X$$

holds for every subset X of the universe. A subfunctor is called *complete* if it spans every T -algebra, and *finitely complete* if it spans every finite T -algebra; note how this depends on the notion of finite T -algebra.

Running Example 6. Consider the monad ∞ for infinite words. Define

$$\mathsf{T}_0 X \stackrel{\text{def}}{=} \{xy, x^\omega : x, y \in X\}.$$

It is not difficult to check that the submonad T_0^* maps X to the finite and ultimately periodic words over alphabet X . Using the Ramsey Theorem, in the same way as it is used explicitly by Wilke in [Wil91], and implicitly by Büchi in [Büc62], we show that T_0 is finitely complete. Indeed, let X be a subset of the universe in some finite ∞ -algebra \mathbf{A} . To show that T_0 spans \mathbf{A} , we need to show that if $w \in X^\infty$, then there is some ultimately periodic word v over X such that

$$\text{mul}_{\mathbf{A}}(w) = \text{mul}_{\mathbf{A}}(v).$$

If w is finite, then it already is ultimately periodic. Otherwise, using the Ramsey Theorem, one can decompose w as

$$w = w_0 w_1 w_2 \cdots \quad \text{with } w_0, w_1, \dots \in X^+$$

such that $\text{mul}_{\mathbf{A}}$ gives the same result for all the finite words w_1, w_2, \dots . Let a_i be the image of w_i under $\text{mul}_{\mathbf{A}}$. By assumption that all a_i are the same for $i \geq 1$ and by associativity, we have

$$\text{mul}_{\mathbf{A}}(w) = \text{mul}_{\mathbf{A}}(a_0 a_1 a_1 \cdots) = \text{mul}_{\mathbf{A}}(w_0 (w_1)^\omega),$$

and the latter uses an ultimately periodic word. As we shall see, the argument made in this example is the only part of the proof of decidability of MSO on ∞ -words that needs to be proved by hand; the remainder of the proof will follow from abstract principles stated in Theorem 6.3. \square

Reducts. Consider a subfunctor T_0 that is finitely complete for a monad T . For a finite T -algebra \mathbf{A} , define its T_0 -*reduct* to be the pair consisting of the universe A of \mathbf{A} , and the restriction of the multiplication operation from \mathbf{A} to the subfunctor:

$$\text{mul}_{\mathbf{A}}|_{T_0 A} : T_0 A \rightarrow A$$

The T_0 -reduct is a special case of what category theorists call an *algebra over signature* T_0 . Straight from the definition it follows that if T_0 spans \mathbf{A} , then \mathbf{A} is uniquely determined by its T_0 -reduct. In particular, if T_0 is complete, then every algebra over signature T_0 extends to at most one T -algebra. Note the “at most one” in the previous sentence; some algebras over signature T_0 might not extend to T -algebras, e.g. not every binary operation extends to a semigroup operation, because for this associativity is needed. The same holds for finite completeness and finite algebras. The point of using T_0 -reducts is that sometimes T_0 can be chosen so that it preserves finiteness, and therefore T_0 -reducts can be manipulated by algorithms, at least as long as finite objects and functions between them can be manipulated by algorithms.

Running Example 7. The T_0 -reduct of a finite ∞ -algebra consists of a finite universe A together with two operations, of arities two and one:

$$\cdot : A \times A \rightarrow A \quad \omega : A \rightarrow A.$$

This is essentially the same thing as a Wilke semigroup. Not every choice of finite universe and three operations above will yield an T_0 -representation of some finite ∞ -algebra; this requires the operations to satisfy certain axioms, e.g. Wilke gives such axioms in Definition 3 of [Wil93]. \square

Computing the syntactic T -morphism. The point of T_0 -reducts is to have a finite representation of T -algebras so that they can be manipulated by algorithms. We give one example of such an algorithm, namely the Moore² algorithm. This algorithm computes the syntactic morphisms in polynomial time. To state this result, we need to explain how morphisms are represented. Consider a subfunctor T_0 . We assume that it is *effective*, in the sense that $T_0\Sigma$ can be computed up to isomorphism based on Σ for finite Σ , in particular T_0 preserves finiteness. The T_0 -representation of a finite T -algebra is simply the finite multiplication table that gives the values of $\text{mul}_{\mathbf{A}}$ for arguments from T_0A . The T_0 -representation of a T -morphism $h : T\Sigma \rightarrow \mathbf{A}$ consists of the T_0 -representation of the algebra, as well as the values of h for units. If T_0 has polynomial size increase, as is the case in the examples of monoids or ∞ -algebras discussed in Examples 6 and 7, then the T_0 -representation of an algebra will be of size polynomial with respect to the size of the universe. However, there will be examples where T_0 has exponential size increase, e.g. in Section 7 in the case of countable chains.

Lemma 5.1 *Let T be a monad in a category of sorted sets, with finitely many sorts, and let T_0 be a subfunctor that is complete for finite algebras. Then syntactic T -morphisms can be computed for T -recognisable languages, in polynomial time with respect to T_0 -representation.*

Proof.

Using the Moore algorithm. \square

6 Monadic second-order logic

An important part of the theory of regular languages is the connection between recognisability and definability in monadic second-order logic MSO. This connection says that languages recognised by finite recognisers are the same thing as MSO definable languages. Examples where this connection holds include: finite words (as proved independently by Büchi, Elgot and Trakhtenbrot), infinite words (as proved by Büchi), finite trees (as proved by Thatcher and Wright), infinite trees (as proved by Rabin), etc. There are common ingredients in all of the proofs, and there are parts that are specific to each domain. In this section, we show that the common ingredients can be stated and proved on the abstract level of monads. This takes care of much of the symbol pushing in the proofs, and leaves only the combinatorial parts to be proved in each specific case, e.g. nothing is left to be proved for finite words or trees, or only the Ramsey theorem needs to be applied in the case of ∞ -words.

6.1 Language theoretic definition of MSO

To establish the connection between MSO and recognisability, consider the following lemma, see [Tho96], which characterises MSO in a way that does not talk

²This is not the same Moore as in Eilenberg-Moore algebras.

about “positions” or “sets of positions” of a structure, but is defined in purely language theoretic terms.

Lemma 6.1 *A language $L \subseteq \Sigma^*$ is definable in MSO if and only if it belongs to the least class of languages that is closed under Boolean combinations, images and inverse images of morphisms $h : \Sigma^* \rightarrow \Gamma^*$, and which contains the languages*

$$0^* \subseteq \{0, 1\}^* \quad \text{and} \quad 0^*1^* \subseteq \{0, 1\}^*.$$

A similar lemma holds for infinite words (instead of 0^*1^* one uses 0^*1^∞), and also for finite and infinite trees, etc. Motivated by the above, we define an abstract notion of MSO in a monad T . In the abstract version, predicates are modelled by languages. For a set \mathcal{L} of T -languages, define $\text{MSO}_{\mathsf{T}}(\mathcal{L})$ to be the smallest class of T -languages which contains \mathcal{L} , is closed under Boolean operations, images and inverse images of T -morphisms.

The following lemma is in the category of sets, or more generally, in categories which have a powerset functor that preserves finiteness. A non-example is the category of nominal sets with orbit-finite sets, where powerset does not preserve orbit-finiteness, and also MSO contains non-recognisable languages, see [Boj13].

Lemma 6.2 *If \mathcal{L} contains only T -recognisable T -languages, then so does $\text{MSO}_{\mathsf{T}}(\mathcal{L})$.*

Proof.

To prove the lemma, one needs to show that T -recognisable languages are closed under Boolean operations, images of T -morphisms, inverse images of T -morphisms. For Boolean operations we use products, for inverse images the property is immediate. The only nontrivial part is the images, where we use the powerset construction, defined as follows. We write $\mathsf{P}X$ for the powerset of X . If X is a set, then we say that $w \in \mathsf{T}X$ belongs pointwise to $v \in \mathsf{T}\mathsf{P}X$ if there is some element of

$$\mathsf{T}\{(a \in X, b \in \mathsf{P}X) : a \in b\}$$

which projects to w and v respectively on the first and second coordinates. For a T -algebra \mathbf{A} , define its powerset to be the T -algebra

$$\mathsf{P}\mathbf{A} : \mathsf{T}\mathsf{P}\mathbf{A} \rightarrow \mathsf{P}\mathbf{A}$$

whose multiplication operation maps $w \in \mathsf{T}\mathsf{P}\mathbf{A}$ to the set

$$\{\text{mul}_{\mathbf{A}}(v) : v \in \mathsf{T}\mathbf{A} \text{ belongs pointwise to } w\}.$$

It is not difficult to check that this is indeed a T -algebra, for the distrustful see Johnstone [J]. \square

6.2 Deciding satisfiability of MSO

For a monad T , we define *MSO satisfiability over T* to be the following decision problem. An instance is what one can see as an MSO formula, which is formalised as an expression that uses the constructors of MSO formulas, with the predicates being represented by T -morphisms recognising them. The question is whether the language corresponding to the instance is nonempty.

In this section we give a sufficient criterion for the decidability of MSO satisfiability. We assume that the monad is in the setting of finitely sorted sets.

Strongly effective subfunctor. Recall the notion of an effective subfunctor T_0 from Section 5, which said that if Σ is finite then $\mathsf{T}_0\Sigma$ is also finite and can be computed based on Σ . As discussed in Section 5, if a monad T has an effective subfunctor T_0 that is finitely complete, then a finite T -algebra can be represented by its multiplication table restricted to T_0 , while a T -morphism

$$h : \mathsf{T}\Sigma \rightarrow \mathbf{A}$$

where Σ is a finite alphabet and \mathbf{A} is finite can be represented by its values on generators, i.e. units of Σ . For the results on MSO of this section, we will need a stronger assumption, which says that algebras recognising singleton sets can be computed. A subfunctor T_0 is called *strongly effective* if for every finite set Σ and every $w \in \mathsf{T}_0\Sigma$, one can compute a representation of a T -morphism

$$h : \mathsf{T}\Sigma \rightarrow \mathbf{A}$$

into a finite T -algebra that recognises $\{w\}$.

Example 3. Consider the monad ∞ of infinite words, and the subfunctor

$$\mathsf{T}_0X \stackrel{\text{def}}{=} \{\epsilon, xy, x^\omega : x, y \in X\}.$$

which was considered in Examples 6 and 7, and proved to be finitely complete. We claim that T_0 is strongly effective. Clearly T_0 preserves finiteness and can be computed, as T_0X is isomorphic to $1 \sqcup X^2 \sqcup X$. For a finite alphabet Σ and $a, b \in \Sigma$ it is not difficult to compute T_0 -reducts of ∞ -algebras that recognise the languages $\{ab\}$ and $\{a^\omega\}$. Let us do the case of $\{a^\omega\}$. The ∞ -algebra has four elements in its universe, representing the empty word, finite words in a^+ , the unique infinite word a^ω , and finally words that use some letter other than a . \square

The following theorem shows that a sufficient criterion for decidable MSO satisfiability is having a subfunctor that is finitely complete and strongly effective.

Theorem 6.3 *Let T be a monad in the setting of finitely sorted sets. If there is a subfunctor T_0 that is strongly effective and finitely complete, then MSO satisfiability is decidable.*

As mentioned at the beginning of this section, Theorem 6.3 is abstract non-sense in the sense that it does not resolve the actual combinatorics necessary to prove satisfiability of MSO. This can be seen in the series of Examples 6, 7 and 3, which show that the monad of infinite words has a subfunctor that is finitely complete and effective, and therefore Theorem 6.3 can be invoked to show that satisfiability of MSO is decidable over infinite words. The decidability proof that comes from these examples has the same structure as the original proof of Büchi [Büc62], or its algebraic version in [Wil93]. What the examples show is that a large part of the proof is sufficiently generic to be stated on the abstract level of monads; and the only challenge is finding a subfunctor that is finitely complete and strongly effective, with finite completeness being essential part.

Theorem 6.3 follows immediately from the following lemma.

Lemma 6.4 *From multiplication tables of T_0 -reducts of a finite T -algebras \mathbf{A}, \mathbf{B} , one can compute multiplication tables of the T_0 -reducts of \mathbf{PA} and $\mathbf{A} \times \mathbf{B}$.*

Proof.

The Cartesian product is immediate, the interesting case is the powerset \mathbf{PA} . For $w \in T_0(\mathbf{PA})$, we need to compute $\text{mul}_{\mathbf{PA}}(w)$. By strong effectivity of T_0 , we can compute a T -morphism

$$h : T(\mathbf{PA}) \rightarrow \mathbf{B}$$

that recognises the singleton $\{w\}$. Define Σ to be the finite set of pairs (a, A_0) such that $a \in A_0 \subseteq A$ and consider the T -morphism

$$g : T\Sigma \rightarrow \mathbf{A} \times \mathbf{B}$$

which works like $\text{mul}_{\mathbf{A}}$ on the first coordinate, and like h on the second coordinate. By definition of the powerset algebra,

$$\text{mul}_{\mathbf{PA}}(w) = \{a : \text{some } v \in T\Sigma \text{ satisfies } g(v) = (a, h(w))\}.$$

Therefore, to compute the above, it suffices to be able to compute the image

$$g(T\Sigma) \subseteq \mathbf{A} \times \mathbf{B}.$$

Because T_0 spans every finite T -algebra, the above image is the same thing as the smallest subset of $\mathbf{A} \times \mathbf{B}$ that contains images of single letters from Σ , and which is closed under g restricted to T_0 . This subset can be computed. \square

Part II

Example Monads

In this part, we give examples of how monads can be used to describe algebraic approaches to the languages for labelled chains (Section 7), unary queries over finite words (Section 8) and various kinds of trees (Section 9). These examples illustrate the general theorems from the first part, i.e. the Syntactic Morphism Theorem, the Eilenberg Pseudovariety Theorem, and the results on MSO.

7 Monads for chains

In this section, we show monads for representing chains, which are a generalisation of infinite words, where the set of positions can be any total order, e.g. the rational or even real numbers. A *chain* over an alphabet Σ is defined to be a nonempty totally ordered set of *positions*, together with a labelling of these positions by Σ . Chains form a monad, modulo the issue that all chains over a given alphabet do not form a set. The unit of this monad interprets an element $a \in \Sigma$ as a chain with a single position labelled by a . The multiplication of a chain of chains w is defined by taking positions to be pairs (i, j) such that i is a position in w , and j is a position in the label of position i , ordered lexicographically.

Shelah showed in [She75] that it is undecidable if a sentence of MSO is true in ordered real numbers (\mathbb{R}, \leq) , which can be seen as an unlabelled chain, or equivalently, a chain over a one-letter alphabet. This implies that satisfiability of MSO is undecidable on arbitrary chains, or even on chains of cardinality continuum, i.e. one cannot decide, given an MSO formula with a binary predicate for the order, whether or not the formula is true in some chain. The binary predicate for the order can be seen as the language of chains over the alphabet $\{0, 1\}$ where all zeros are before all ones. It follows that the assumptions of Theorem 6.3 cannot be met, even for chains of cardinality at most continuum. These problems go away if one considers countable chains.

Countable chains. A *countable* chain is one where the set of positions is countable. A countable chain is called *scattered* if its indexing set is scattered, i.e. its positions do not embed an isomorphic copy of the rational numbers. A special case of a scattered chain is a countable well-chain, i.e. one where the positions are well-ordered. These three kinds of chains are submonads of the monad of chains, i.e. they form monads when equipped with the unit and multiplication inherited from the monad of all chains.

The following theorem shows that in all three cases, the algebras admit finitely complete subfunctors, as defined in Section 5, which are also strongly effective as defined in Section 6. The cases of countable well-founded and countable scattered chains are simple enough to warrant a self-contained proof, modulo the Hausdorff theorem on scattered chains. The case of arbitrary countable chains is more involved and follows from [She75], see also [CCP11].

Theorem 7.1

1. *Every finite algebra in the monad of countable well-chains is spanned by*

$$X \mapsto \{x \cdot y, x^\omega : x, y \in X\}$$

2. *Every finite algebra in the monad of countable scattered chains is spanned by*

$$X \mapsto \{x \cdot y, x^\omega, x^{-\omega} : x, y \in X\}$$

3. Every finite algebra in the monad of countable chains is spanned by

$$X \mapsto \{x \cdot y, x^\omega, x^{-\omega}, \text{shuffle}Y : x, y \in X, Y \subseteq X\}$$

where $\text{shuffle}Y$ is the chain where the positions are rational numbers and where every $y \in Y$ labels a dense subset (such a chain is unique up to isomorphism).

Proof. (of the first two cases)

The Hausdorff theorem on scattered chains says that scattered chains are the smallest class of chains that contains the finite chains, chains indexed by ω and $-\omega$, and is closed under substitution. For well-founded countable chains, the same holds, but $-\omega$ is not allowed. The result then follows, using the Ramsey theorem in the same way as in the case of ∞ -algebras. \square

Corollary 7.2 *Satisfiability for MSO is decidable on: all countable chains, scattered chains, and well-ordered countable chains.*

Proof.

It is easy to see that the subfunctors given in Theorem 7.1 are strongly effective. Therefore, the result follows from Theorem 6.3. \square

In particular, for the well-chains and the scattered chains, we get a simple self-contained proof of decidability for MSO. This proof is no different from the known ones, but the advantage of using monads is that they clearly identify which part of the argument is specific to the monad being used.

8 Pointed words

This section presents a monad which generates a new kind of algebra, which, although simple, has not appeared in the literature up to the author's best knowledge. The monad, call it **Point**, is defined by

$$\text{Point}A \stackrel{\text{def}}{=} A^* \underline{A} A^*,$$

where \underline{A} is a disjoint copy of the set A . Elements of $\text{Point}A$ are called *pointed words*³. The idea is that a pointed word represents a nonempty word over A where the underlined position is selected, and therefore a pointed word can be used as an input to a unary query that tests properties of positions in a word. Therefore we will use the term unary query for a set of pointed words. The unit operation is $a \mapsto \underline{a}$, while the monad multiplication operation is the same as in the monad of finite words, except that the underlined position is the underlined position in the underlined word.

A pointed word can be viewed in two ways: as a nonempty word over alphabet Σ with a distinguished position, or as a special case of a non-pointed

³Similar ideas would work for pointed chains, pointed trees, etc.

word over an extended alphabet $\Sigma \cup \underline{\Sigma}$. In logical terms, the first view proposes that sets of pointed words are defined by unary queries (i.e. formulas with one free individual variable) over the alphabet Σ , and the second view proposes that sets of pointed words are defined by Boolean queries (i.e. with no free variables) over the extended alphabet $\Sigma \cup \underline{\Sigma}$. For some logics, the two views are essentially the same. For instance a set of pointed words is MSO definable in the first view if and only if it is MSO-definable in the second view. The same is true for first-order logic with the order predicate. Therefore, for some logics such as MSO or first-order logic, characterising unary queries reduces to characterising Boolean queries. However, for some logics this is not the case.

In Section 8.1, we will show that for two-variable first-order logic, characterising unary queries is not easily reducible to characterising Boolean queries over extended alphabets. We also show how finite **Point**-algebras are useful in characterising unary queries. Along the way, we use much of the machinery developed in Part I of this paper, in particular the Syntactic Morphism theorem, the Pseudovariety Theorem, and the results on representation. All of these would be relatively straightforward to prove by hand in the special case of the monad **Point**, but deducing them from abstract nonsense allows us to focus on the more specific and combinatorial parts of the proof.

Much of the material in Section 8.1 is specific to unary queries definable in two-variable first-order logic, and the reader who is more interested in the general principles of monads is advised to skip it.

8.1 Unary queries definable in two-variable first-order logic

To illustrate the monad of pointed words **Point**, consider the fragment of first-order logic that uses only two variables, but which is allowed to reuse them by requantifying. The logic has access to predicates for the labels and the order, but not for the successor, although similar results are true for other choices of predicates. We say that a set of pointed words, i.e. a unary query, is two-variable definable if it can be defined by a formula of two-variable first-order logic that has one free variable, say x , and which uses the predicates described above. In the semantics of the formula, the free variable binds the selected position, but once the free variable of the query is requantified, the selected position is forgotten. For example the unary query “the distinguished position is followed by at least two positions with label a ” can be defined in two-variable first-order logic, although only thanks to using requantification:

$$\varphi(x) = \exists y (x < y \wedge a(y) \wedge \exists x (y < x \wedge a(x))).$$

It is not immediately clear how to define the unary query “the successor of the distinguished position has label a ”, because the natural formula would use three variables to define successor in terms of order:

$$\psi(x) = \exists y (x < y \wedge a(y) \wedge \forall z (z \leq x \vee y \leq z)).$$

In fact, the unary query $\psi(x)$ cannot be defined using two variables, as long as the vocabulary has predicates just for the order and labels, which is our chosen

setting in this section. This example illustrates that with only two variables, the choice of vocabulary is more important than in first-order logic with arbitrarily many variables.

The two-variable fragment of first-order logic is a well-studied logic for non-pointed words, i.e. for Boolean queries on words, see e.g. [TW98], but it also makes sense for unary queries, as it corresponds to unary queries definable in XPath with only the transitive axis $//$ and its inverse⁴.

We will show that two-variable definable languages form a pseudovariety, and therefore by the Pseudovariety Theorem, definability of a language in two-variable logic depends only the syntactic **Point**-algebra of the language. The Pseudovariety Theorem alone does not give an algorithm to decide this definability, but such an algorithm is given in Theorem 8.2.

The transformation monoids. In every **Point**-algebra there is a hidden monoid, actually two monoids. Consider a **Point**-algebra \mathbf{A} . For $a \in A$, define its *left transformation* to be the function $A \rightarrow A$ defined by

$$b \mapsto \text{mul}_{\mathbf{A}}(a\underline{b}).$$

Likewise we define the right transformation. Left transformations form a monoid, equipped with function composition, call it the *left monoid*. If A is finite then so is the left monoid. Likewise one can define right transformations and the right monoid. It is not difficult to see that a **Point**-algebra is uniquely specified by its universe A and the the left and right transformations for each $a \in A$. In other words, using the terminology of Section 5, the subfunctor

$$A \mapsto \{a\underline{b}, \underline{a}b : a, b \in A\}$$

is complete for all **Point**-algebras. It is also strongly effective as defined in Section 6. It follows that a finite **Point**-algebra can be represented in space polynomial in the size of its universe; and that syntactic algebras can be computed in polynomial time (by Lemma 5.1).

The following example shows that just looking at the left and right monoids of a unary query is not sufficient to decide if it is two-variable definable. Stated in the language of temporal logic, the example shows that two-variable logic does not have the separation property.

Example 4. Let us revisit the successor query discussed at the beginning of this section. Let the alphabet be $\{a, b\}$, and consider the unary query “the successor of the selected position has label a ”, i.e.

$$\{w\underline{\sigma}av : w, v \in \{a, b\}^*, \sigma \in \{a, b\}\} \subseteq \text{Point}\{a, b\}.$$

When seen as a language over an extended alphabet, the above is definable by a formula of two-variable logic without free variables. The formula says that

⁴To be fair, the XPath motivation would be best justified by studying the tree variant of the logic. Preliminary research indicates that the results from this section can be generalised to trees.

there exists a position with label a , such that one can go one step to the left and find the underlined position, but one cannot go two steps to the left and find the underlined position. When seen as a unary query over the alphabet Σ , the above is not two-variable definable, which will follow from Theorem 8.2.

Also, one can observe that just looking at the left and right monoids is not sufficient to understand the query. In this case, the left monoid is trivial, i.e. contains only the identity transformation, while the right monoid is the syntactic monoid of the language “words beginning with a ”. Both monoids have the property that they recognise only languages definable in two-variable first-order logic. \square

The above example shows that characterising unary queries definable in two-variable logic does not simply reduce to characterising languages (i.e. Boolean queries) definable in two-variable logic over an extended alphabet.

An Ehrenfeucht-Fraïssé game. We now show that two-variable definable unary queries form a pseudovariety of Point-languages. Therefore, by the Pseudovariety Theorem, the syntactic Point-algebra of a unary query has sufficient information (unlike the left and right monoids) to decide if the query is two-variable definable.

We do this using Ehrenfeucht-Fraïssé games in a standard way. Consider two pointed words w_0, w_1 . For $n \in \mathbb{N}$, define the following game, which is played by players Spoiler and Duplicator. At the beginning of the game, the labels of the selected positions in the two pointed words are checked; if they are different then Spoiler wins immediately and the game is terminated. If the selected positions have the same labels, then n rounds of the game are played as follows. At the beginning of each round Spoiler chooses $i \in \{0, 1\}$ and a direction, which is one of “left”, “stay” or “right”. Then Spoiler changes the selected position in the pointed word w_i according to the direction, i.e. if the direction is “left” then the selected position is moved somewhere to the left, if it is “stay” then it is not changed, and if it is “right” then it is moved to the right. Duplicator responds by choosing a new selected position in the other pointed word w_{1-i} , according to the direction chosen by Spoiler, and such that the new selected positions have the same labels. If Duplicator cannot do this, then Spoiler wins immediately and the game is terminated. Otherwise, another round is played with the new selected positions; and if all n rounds are played without Spoiler winning, then Duplicator wins.

We write $w_0 \sim_n w_1$ if Duplicator has a winning strategy in the n -round game. It is not difficult to show that $w_0 \sim_n w_1$ holds if and only if w_0, w_1 satisfy the same unary queries of two-variable logic of quantifier depth n . The following lemma, which is proved by composing winning strategies for Duplicator in an obvious way, says that equivalence under \sim_n is preserved under unary polynomials and Point-morphisms.

Lemma 8.1 *If pointed words satisfy $w_0 \sim_n w_1$ then*

$$\begin{array}{ll} \llbracket p \rrbracket(w_0) \sim_n \llbracket p \rrbracket(w_1) & \text{for every unary polynomial in } \mathbf{Point}\Sigma \\ h(w_0) \sim_n h(w_1) & \text{for every Point-morphism } f : \mathbf{Point}\Sigma \rightarrow \mathbf{Point}\Gamma \end{array}$$

A corollary of the above lemma is that unary queries that are two-variable definable form a pseudovariety of **Point**-languages. Closure under Boolean combinations is immediate, while for closures under derivatives and inverse images under **Point**-morphisms, one uses Lemma 8.1 and the fact that a **Point**-language is two-variable definable if and only if it is a finite union of equivalence classes of \sim_n for some n .

An effective characterization As stated above, two-variable unary queries form a pseudovariety of **Point**-languages, and therefore the Pseudovariety Theorem can be invoked to show that whether or not a unary query is two-variable depends only on its syntactic **Point**-algebra. Is this dependency effective? In the case of Boolean queries, i.e. languages of non-pointed words, this problem was solved in [TW98], where it was shown that a language $L \subseteq \Sigma^*$ is two-variable definable if and only if its syntactic monoid belongs to a class of monoids called **DA**. The class **DA** is a pseudovariety of monoids that can be defined by two identities, and therefore membership in it is decidable. In the following theorem, we extend the result of [TW98] from Boolean queries to unary queries, i.e. from non-pointed words to pointed words.

Theorem 8.2 *Let $q \subseteq \mathbf{Point}\Sigma$ be a unary query recognisable by a finite **Point**-algebra. Then q is two-variable definable if and only if its syntactic **Point**-algebra **A** satisfies:*

- the left and right monoids of **A** belong to **DA**;
- for every $w \in A^+$ which all letters in a set $B \subseteq A$, every $b \in B$ and every $v \in \mathbf{Point}B$, the following equality holds for all but finitely many $n \in \mathbb{N}$:

$$\text{mul}_{\mathbf{A}}(w^n b v w^n) = \text{mul}_{\mathbf{A}}(w^n v w^n) = \text{mul}_{\mathbf{A}}(w^n v b w^n).$$

The rest of Section 8.1 is devoted to proving the above theorem. We begin with a corollary of the theorem, which says that definability of unary queries in two-variable logic can be decided in polynomial time. When talking about polynomial time, we refer to representation of **Point**-algebras with respect to

$$A \mapsto \{a\bar{b}, \bar{a}b : a, b \in A\}.$$

When a, b are in the universe of a **Point**-algebra **A**, we will treat $a\bar{b}$ as an element of **A**, although the more formally correct notation would be $\text{mul}_{\mathbf{A}}(a\bar{b})$.

Corollary 8.3 *Whether or not a recognisable $q \subseteq \mathbf{Point}\Sigma$ is two-variable definable can be decided in polynomial time with respect to the recognising morphism.*

Proof.

By Lemma 5.1, the syntactic morphism can be computed in polynomial time based on any recognising morphism into a finite algebra. Therefore, it suffices to show that the conditions in Theorem 8.2 can be checked in polynomial time, when given on input an **Point**-algebra \mathbf{A} .

For the first condition, one computes the left and right monoids. These monoids are quotients of \mathbf{A} under an equivalence relation that can be checked in polynomial time, and therefore they can be computed in polynomial time. Then one checks in polynomial time if the left and right monoids satisfy the identities for DA.

Let us show how to check the second condition. A naive algorithm would check all possible subsets $B \subseteq A$, which would take exponential time. To overcome this, define an ordering \preceq on the universe of \mathbf{A} , such that $a \preceq b$ holds if there exist pointed words $w, v \in \text{Point}A$ such that

$$a = \text{mul}_{\mathbf{A}}(w) \quad b = \text{mul}_{\mathbf{A}}(v)$$

and every letter that appears in w also appears in v , ignoring the underlining. The relation \preceq is not necessarily transitive, due to taking the image under $\text{mul}_{\mathbf{A}}$. In terms of \preceq , the second condition in the statement of the theorem says that for all but finitely many n ,

$$\text{mul}_{\mathbf{A}}(a^n b \underline{c} a^n) = \text{mul}_{\mathbf{A}}(a^n \underline{c} a^n) = \text{mul}_{\mathbf{A}}(a^n \underline{c} b a^n) \quad \text{for every } a \succeq b, c$$

It is not difficult to show that the above need only be checked for n which are linear in the size of left and right monoids of \mathbf{A} , and therefore the only remaining thing to do is compute \preceq .

It is not difficult to show that \preceq is the smallest relation which contains every pair $a \preceq a$ and which satisfies the following implications for every a, b, c, d in \mathbf{A} .

$$\begin{aligned} a \preceq b & \text{ implies } a \preceq b \underline{c} \\ a \preceq b & \text{ implies } a \preceq \underline{b} c \\ a \preceq b \text{ and } c \preceq d & \text{ implies } a \underline{c} \preceq b \underline{d} \\ a \preceq b \text{ and } c \preceq d & \text{ implies } \underline{a} c \preceq \underline{b} d. \end{aligned}$$

In particular, \preceq can be computed in polynomial time using a fixpoint algorithm.

□

The rest of Section 8.1 is devoted to proving Theorem 8.2. We begin with the easier implication.

Lemma 8.4 *If a unary query is two-variable definable, then its syntactic algebra satisfies the conditions in Theorem 8.2.*

Proof.

Let $q \subseteq \text{Point}\Sigma$ be a unary query definable in two-variable first-order logic, and let n be the quantifier depth of the defining formula. By Lemma 8.1, the equivalence relation \sim_n is preserved under unary polynomials, and therefore by

the Syntactic Morphism theorem it is a congruence, i.e. the set of equivalence classes can be equipped a multiplication operation which makes it into a finite **Point**-algebra, call it \mathbf{A}_n . Since q is a finite union of equivalence classes under \sim_n , it is recognised by \mathbf{A}_n , and therefore the syntactic algebra of q is an image of \mathbf{A}_n under a **Point**-morphism. The conditions in Theorem 8.2 are easily seen to be closed under images of **Point**-morphisms, and therefore it suffices to show that these conditions are satisfied by \mathbf{A}_n . We only sketch the proof for the second condition: this boils down to showing that if $w \in \Sigma^*$ is a word which uses all letters in a set $B \subseteq \Sigma$, then

$$w^m v_1 \underline{b} v_2 w^m \sim_n w^m \underline{b} w^m \quad \text{for every } m \geq n, v_1, v_2 \in B^* \text{ and } b \in B.$$

This is proved by induction on n . Here is one of the cases that needs to be considered: if in the first round, Spoiler moves the selected position of the first pointed word to some position in v_1 , the Duplicator responds by moving the selected position in the second pointed word to a position in the last copy of w before \underline{b} which has the same label, such a position exists by assumption on w using all letters from B . \square

The rest of Section 8.1 is devoted to showing the converse implication in Theorem 8.2. A possibly partial function $f : \mathbf{Point}\Sigma \rightarrow X$ with X finite is called two-variable definable if the inverse image of every $x \in X$ is two-variable definable. We will prove that if \mathbf{A} is a finite **Point**-algebra that satisfies the conditions in the theorem, then every **Point**-morphism $h : \mathbf{Point}\Sigma \rightarrow \mathbf{A}$ is definable in two-variable logic. The proof is by induction on the size of the alphabet Σ . We begin by introducing some auxiliary results.

Filtering. For a unary query $q \subseteq \mathbf{Point}\Sigma$, define

$$\text{filter}q : (\Sigma^* \cup \mathbf{Point}\Sigma) \rightarrow (\Sigma^* \cup \mathbf{Point}\Sigma)$$

to be the function which inputs a pointed or non-pointed word, and only keeps positions that are selected by q . The output is a pointed word if the input was a pointed word and q selected the selected position, otherwise the the output is a non-pointed word. For example, if $q \subseteq \mathbf{Point}\{a, b\}$ is the set of pointed words where the distinguished position has label a , then

$$(\text{filter}q)(a\underline{b}a) = aa \in \Sigma^* \quad (\text{filter}q)(a\underline{a}a) = a\underline{a}a \in \mathbf{Point}\Sigma.$$

The following simple fact is proved by relativising formulas in the obvious way.

Fact 8.5 *If $f : \mathbf{Point}\Sigma \rightarrow X$ and $q \subseteq \mathbf{Point}\Sigma$ are definable in two-variable logic, then so is the partial function $f \circ \text{filter}q$. (The function is partial because it is undefined when $\text{filter}q$ removes the selected position.)*

A monoid. Let $\text{mon}\mathbf{A}$ be the product of the left and right transformation monoids of \mathbf{A} . It is not difficult to see that the left and right transformations of $h(w)$ for $w \in \text{Point}\Sigma$ do not depend on the selected position. In other words, there is a function $\text{mon}h$ which makes the following diagram commute

$$\begin{array}{ccc} \text{Point}\Sigma & \xrightarrow{\text{deselect}} & \Sigma^* \\ & \searrow \text{mon} & \downarrow \text{mon}h \\ & & \text{mon}\mathbf{A} \end{array} \quad (6)$$

where deselect is the function that ignores the selected position and mon is the function that computes the left and right transformation. It is not difficult to show that $\text{mon}h$ is a monoid morphism. By the assumption of the theorem, the monoid $\text{mon}\mathbf{A}$ is in DA , as a product of two monoids in DA .

Known results about DA . We now recall some results about sets of words (not pointed words) definable in two-variable logic. Let \mathbf{M} be a monoid. Define the right ideal generated by $m \in \mathbf{M}$ to be the set

$$\{mn : n \in \mathbf{M}\} \subseteq \mathbf{M}.$$

We say $m, n \in \mathbf{M}$ are \mathcal{R} -equivalent, denoted by $m \sim_{\mathcal{R}} n$, if they generate the same right ideals. There is a symmetric notion of \mathcal{L} -equivalence that uses left ideals. The theorem below summarises some results from [TW98]. It only mentions \mathcal{R} -classes, i.e. equivalence classes under \mathcal{R} -equivalence, but the symmetric results also hold for \mathcal{L} -classes.

Theorem 8.6 *For a monoid morphism $g : \Sigma^* \rightarrow \mathbf{M}$ into a monoid in DA :*

1. *every language recognised by g is a two-variable definable Boolean query, i.e. it is definable by a formula of two-variable first-order logic without free variables;*
2. *for every \mathcal{R} -class R of \mathbf{M} there is a two-variable definable unary query which selects a position if and only if the prefix up to and including that position is mapped by g to R .*
3. *for every $m \in \mathbf{M}$ the set $\{n : mn \sim_{\mathcal{R}} m\}$ is a submonoid of \mathbf{M} .*

Apply the above theorem to the morphism $\text{mon}h$ defined in diagram (6). By the second item of the theorem and its symmetric variant for \mathcal{L} -classes, for every \mathcal{R} -class R and every \mathcal{L} -class L of the monoid $\text{mon}\mathbf{A}$, there are two-variable definable unary queries, call them q_R and q_L , such that

$$\begin{array}{lll} uav \in q_R & \text{iff} & \text{mon}h(ua) \in R \\ uav \in q_L & \text{iff} & \text{mon}h(av) \in L \end{array}$$

holds for every $u, v \in \Sigma^*$ and $a \in \Sigma$.

The more difficult implication from Theorem 8.2 will follow from the following lemma.

Lemma 8.7 *Let R be an \mathcal{R} -class in the monoid \mathbf{monA} , and let L be an \mathcal{L} -class in the monoid \mathbf{monA} . Then the partial function obtained from h by restricting its domain to $q_R \cap q_L$ is two-variable definable.*

Before proving the lemma, observe that it implies that h is two-variable definable. This is because every pointed word belongs to $q_R \cap q_L$ for some choice of R and L .

Proof.

Call an \mathcal{R} -class *minimal* if the corresponding right ideal is minimal with respect to inclusion. Likewise we define a minimal \mathcal{L} -class. Consider three cases: when R is not minimal, when L is not minimal, and when both R and L are minimal. The first two cases are not disjoint.

The \mathcal{R} -class R is not minimal. We prove a stronger result, namely the restriction of h to q_R is two-variable definable. Let us decompose q_R into a disjoint union of two unary queries, both of which are two-variable definable: q_R^0 selects the leftmost position that satisfies q_R , and q_R^+ selects the remaining positions.

- Let us first show that h is two-variable definable when its domain is restricted to q_R^0 . Define q_{left} be the unary query which selects positions that are strictly to the left of some position that satisfies q_R^0 . Likewise define q_{right} to be the positions that are strictly to the right of some position that satisfies q_R^0 . Because q_R^0 is two-variable definable, then the queries q_{left} and q_{right} are also two-variable definable. Consider some $w \in q_R^0$. The word $\text{deselect}w$ underlying w splits into three consecutive intervals:

1. first come the positions that satisfy q_{left} ;
2. then comes the single position that satisfies q_R^0 ;
3. finally come the positions that satisfy q_{right} .

Because the intervals are consecutive, when restricted to arguments from q_R^0 , the function h factors through the following three functions.

$$\begin{array}{lll} \text{mon}h \circ \text{filter}q_{\text{left}} & : & \text{Point}\Sigma \rightarrow \mathbf{monA} \\ h \circ \text{filter}q_R^0 & : & \text{Point}\Sigma \rightarrow \mathbf{A} \\ \text{mon}h \circ \text{filter}q_{\text{right}} & : & \text{Point}\Sigma \rightarrow \mathbf{monA} \end{array}$$

The first and third functions are two-variable definable by Theorem 8.6 and Fact 8.5. The middle function depends only on the label of the selected position, and is therefore also two-variable definable. Therefore, h is two-variable definable when restricted to arguments from q_R^0 .

- Let us now show that h is two-variable definable when restricted to q_R^+ . The proof is the same as above, the only difference is in the proof that

$$h \circ \text{filter}q_R^+ : \text{Point}\Sigma \rightarrow \mathbf{A} \tag{7}$$

is two-variable definable. To prove this, let m be an element of the \mathcal{R} -class R , and consider the set

$$M_m \stackrel{\text{def}}{=} \{n : mn \sim_{\mathcal{R}} m\}$$

which is a submonoid of \mathbf{monA} by item 3 of Theorem 8.6. Because M_m is a submonoid, it is not difficult to show that it does not depend on the choice of $m \in R$, i.e. it only depends on the \mathcal{R} -class R . Furthermore, this submonoid cannot be all of \mathbf{monA} , since otherwise R would be a minimal \mathcal{R} -class. Therefore, there is some $a \in \Sigma$ such that $\mathbf{mon}h(a)$ does not belong to the submonoid, which means that

$$m\mathbf{mon}h(a) \not\sim_{\mathcal{R}} m \quad \text{for every } m \in R$$

It follows that positions selected by q_R^+ cannot have label a . Therefore the function in (7) is two-variable definable by induction assumption on a smaller alphabet.

The \mathcal{L} -class R is not minimal. This case is symmetric to the previous one.

Both L and R are minimal. We are left with the case where L is a minimal \mathcal{L} -class of \mathbf{monA} , and R is a minimal \mathcal{R} -class of \mathbf{monA} . We will prove that, when restricted to pointed words in $q_R \cap q_L$, the value of h depends only on the label of the selected position. In other words, we claim that if

$$w = w_{\text{left}} \underline{a} w_{\text{right}} \quad v = v_{\text{left}} \underline{a} v_{\text{right}}$$

are pointed words in $q_R \cap q_L$ with the same label $a \in \Sigma$ of the distinguished position, then both have the same value under h . Let $u \in \Sigma^*$ be a non-pointed word which uses all letters in Σ , whose \mathcal{R} -class is R and whose \mathcal{L} -class is L . Such a word exists by minimality, for example, u can be obtained from either of the pointed words w or v by replacing the selected position by some non-pointed word which uses all letters from Σ . Let n be a sufficiently large number. We will show that

$$h(w) = h(u^n \underline{a} u^n) = h(v).$$

By symmetry, we only prove the left equality above. Because $\mathbf{mon}h(w_{\text{left}})$ is in the same \mathcal{R} -class as $\mathbf{mon}h(w_{\text{left}} u^n)$, we have

$$\mathbf{mon}h(w_{\text{left}} u^n) m = \mathbf{mon}h(w_{\text{left}}) \quad \text{for some } m \in \mathbf{monA}.$$

Therefore, there must exist some $x_{\text{left}} \in \Sigma^*$ such that $w_{\text{left}} u^n x_{\text{left}}$ induces the same left transformation in \mathbf{A} as w_{left} . Using a symmetric reasoning for \mathcal{L} -classes, we obtain some $x_{\text{right}} \in \Sigma^*$ such that $h(w)$ is equal to

$$h(w_{\text{left}} u^n x_{\text{left}} \underline{a} x_{\text{right}} u^n w_{\text{right}})$$

The elements $\text{mon}h(w_{\text{left}}u^n)$ and $\text{mon}h(u^n)$ are in the \mathcal{R} -class R , because both have prefixes in this \mathcal{R} -class and the class is minimal; and they are both in the \mathcal{L} -class L because both have suffixes in this \mathcal{L} -class. (Here we use minimality.) In a monoid from DA, or more generally in an aperiodic monoid, an element is uniquely determined by its \mathcal{L} -class and \mathcal{R} -class. Therefore, $w_{\text{left}}u^n$ and u^n induce the same left transformation in \mathbf{A} . By this observation, and a symmetric one for \mathcal{L} -classes, it follows that $h(w)$ is equal to

$$h(u^n x_{\text{left}} \underline{a} x_{\text{right}} u^n).$$

From the assumption on \mathbf{A} in the theorem, the above is equal to

$$h(u^n \underline{a} u^n).$$

By the same reasoning, $h(v)$ is also equal to the above, which completes the proof of the lemma. \square

9 Monads for trees

In this section, we present a series of monads for modelling trees.

9.1 Ranked trees over a fixed alphabet

We begin with a monad that represents finite trees over a fixed ranked alphabet. Algebras in this monad will be deterministic bottom up tree automata over the ranked alphabet.

Consider a ranked alphabet Σ , i.e. a finite set where each element has an associated rank, which is a natural number. A ranked tree over such an alphabet is a finite tree labelled by Σ , where a node has as many children as the rank of its alphabet, and these children are ordered. In other words, this is a ground term over Σ seen as a signature. We define a monad \mathbf{T}_Σ , which is parametrised by Σ , and which will model ranked trees over Σ . Although the alphabet Σ is ranked, the monad \mathbf{T}_Σ itself is in the category of sets, i.e. sets without any arity structure imposed.

Define \mathbf{T}_Σ to be the monad which maps a set Γ to the set of terms over the signature Σ extended by variables from Γ (i.e. trees where labels from Γ can occur in the leaves). The multiplication operation

$$\text{mul}_{\mathbf{T}_\Sigma \Gamma} : \mathbf{T}_\Sigma \mathbf{T}_\Sigma \Gamma \rightarrow \mathbf{T}_\Sigma \Gamma$$

is term substitution, while the unit maps a variable $a \in \Gamma$ to the term that consists only of this variable. In the language of category theory, this is the monad generated by Σ interpreted as a polynomial functor.

If Γ is a finite alphabet, then a \mathbf{T}_Σ -language over Γ is a set of trees over the ranked alphabet Σ , extended by rank zero symbols for letters from Γ . In the special case of $\Gamma = \emptyset$, a \mathbf{T}_Σ -language over the empty alphabet is a set of ranked trees over the alphabet Σ .

Example 5. In this example, we show that when Σ contains only letters of rank one, then the monad T_Σ can be seen as modelling deterministic word automata with input alphabet Σ . Consider a ranked set Σ , which has only letters of rank one. If Q is a set, then elements of $T_\Sigma Q$ are trees with unary branching where inner nodes are from Σ and the unique leaf is from Q . For example an element of $T_{\{a,b\}}\{q\}$ can look like this:



When written bottom-up, such a tree can be seen as a word in $Q \cdot \Sigma^*$. A finite algebra Eilenberg-Moore for this monad, i.e. a finite T_Σ -algebra, consists of a universe, call it Q , and a multiplication operation, which can be seen as a function

$$\delta : Q \cdot \Sigma^* \rightarrow Q.$$

The associativity of multiplication says that

$$\delta(\delta(q \cdot w) \cdot v) = \delta(q \cdot w \cdot v),$$

and therefore δ is uniquely defined by its values on $Q \cdot \Sigma$. Stated differently, a T_Σ algebra is the same thing as a deterministic finite word automaton with input alphabet Σ , without designated initial and accepting states. \square

Connections with Σ -algebras. Recall that in universal algebra, a Σ -algebra consists of a universe A together, together with an operation $f : A^n \rightarrow A$ for each $f \in \Sigma$ of rank n . To go from a T_Σ -algebra \mathbf{A} in the sense of Eilenberg-Moore to a Σ -algebra in the sense of universal algebra, one defines the universe to be A , and the operation corresponding to a n -ary letter $f \in \Sigma$ to be

$$(a_1, \dots, a_n) \in A^n \mapsto \text{mul}_{\mathbf{A}}(f(a_1, \dots, a_n)).$$

In the terminology of Section 5, this is the Σ -reduct of \mathbf{A} , where we view Σ as the (finitely complete and effective) subfunctor

$$\Sigma A = \{f(a_1, \dots, a_n) : f \text{ is an } n\text{-ary symbol in } \Sigma\} \subseteq T_\Sigma A$$

Every Σ -algebra is obtained this way, and therefore the two notions are essentially the same. This sameness extends to morphisms⁵.

⁵Actually, this sameness works for a more general notion of ranked set used in category theory, i.e. when Σ is an arbitrary functor. This more general setting can be used to describe

Connection with tree automata. As shown in Example 5, if the alphabet Σ contains letters of rank at most one, then T_Σ -algebras are essentially the same thing as deterministic word automata. For other alphabets, the correspondence is with deterministic bottom-up tree automata. For a T_Σ algebra \mathbf{A} , there is a unique T_Σ -morphism

$$h : \mathsf{T}_\Sigma \emptyset \rightarrow \mathbf{A}.$$

When interpreting an element of $\mathsf{T}_\Sigma \emptyset$ as a tree over the ranked alphabet, the algebra \mathbf{A} maps every tree to an element of its universe. When the algebra is finite, this is the same thing as a deterministic bottom-up tree automaton, with the only difference being that an automaton also has an accepting subset of states, which indicates when a tree belongs to the language. Therefore, T_Σ -recognisable languages are the same thing as the classical notion of regular languages of finite trees over the ranked alphabet Σ .

Example 6. Consider the following variant of first-order logic on trees over a ranked alphabet Σ . To a tree $t \in \mathsf{T}_\Sigma \Gamma$, one assigns a logical structure, where the universe is the nodes of the tree, and there are the following predicates: a unary predicate that is true in nodes with label a , a binary predicate for the descendant relation, and binary predicates for the i -th child relation for every i . A subset of $\mathsf{T}_\Sigma \Gamma$ is called definable in first-order logic if there is a formula of first-order logic that is true in the logical structures corresponding to trees in the subset, and false in logical structures corresponding to tree outside the subset. A well-known open problem stated in [Tho84] is: can one decide if a recognisable language of trees is definable in first-order logic?

Here we show the, already known, result that tree languages definable in first-order logic form a language pseudovariety. Since the assumptions of the Pseudovariety Theorem apply to the monad T_Σ , this will imply that first-order definability of a tree language depends only on its syntactic algebra (whether or not this dependence is computable is the open problem).

Recall that a language pseudovariety is a class of languages that is closed under Boolean combinations, polynomial derivatives, and inverse images under morphisms. (The other conditions in the definition are vacuous when there is only one sort, as is the case here.) Boolean combinations are for free in first-order logic. Closure under polynomial derivatives is shown the same way as closure under inverse morphisms, so we only show the latter closure. We need to show that for every T_Σ -morphism

$$h : \mathsf{T}_\Sigma \Gamma \rightarrow \mathsf{T}_\Sigma \Delta, \tag{8}$$

unranked trees, when Σ is the functor

$$X \mapsto X^*$$

or unranked trees without sibling order, when Σ is the functor which takes X to finite multisets over X . A problem with these more general settings is that their Eilenberg-Moore algebras model automata that are too strong, in the sense that the transition function need not be describable in a finite way.

inverse images under h of first-order languages are also first-order definable. One way of proving this statement is to show that h is a special case of a more general notion of *copying first-order interpretation*, and first-order definable languages are closed under inverse images of such interpretations. Another way, which we use here, is to use Ehrenfeucht-Fraïssé games. Let us write $s \sim_n t$ if player Duplicator has a winning strategy in the n -round Ehrenfeucht-Fraïssé game for the logical structures corresponding to s and t . It is easy to see that closure under inverse morphisms of first-order definable languages is implied by the following observation:

$$s \sim_n t \text{ implies } h(s) \sim_n h(t) \quad \text{for every } n \in \mathbb{N}. \quad (9)$$

The above observation is proved using a straightforward strategy copying argument, which we describe in more detail below. The key to the strategy copying argument is that every node in an image tree $h(t)$ is uniquely identified by two pieces of information, which we call the *origin* and *offset*, whose definition is explained by example in Figure 2.

To prove (9), in the game corresponding to $h(s)$ and $h(t)$, Duplicator preserves the following invariant.

- (*) Suppose that i rounds have been played so far, and that the nodes selected in those rounds were x_1, \dots, x_i in the tree $h(s)$ and y_1, \dots, y_i in the tree $h(t)$. Then the offsets, if defined, are the same for each x_j and y_j , and Duplicator has a winning strategy for the remaining rounds in the game for s and t , assuming that the selected nodes x_1, \dots, x_i and y_1, \dots, y_i are replaced by their origins.

Assuming that $s \sim_n t$ holds, it is not difficult to show that Duplicator can preserve the invariant for n rounds in the game between $h(s)$ and $h(t)$. Although simple, the strategy copying argument is a bit delicate – as we will see in Example 7, closure under inverse morphisms will fail in a different monad for modelling trees, where morphisms can duplicate subtrees. \square

Dependence on Σ . In the monad T_Σ , there is a different monad for every Σ . In the following two sections, we present two approaches where the monad is independent of the alphabet. The price we will pay is using categories of ranked sets.

9.2 Clones

In this section, we consider a monad which is used to describe clones. We begin by recalling the definition of a clone from universal algebra: a clone over a universe A is a set of functions of the form $A^n \rightarrow A$, of possibly different arities $n \in \mathbb{N}$, which includes all projections, i.e. functions of the form $(a_1, \dots, a_n) \mapsto a_i$, and which is closed under composition in the sense that if the clone contains an n -ary operation f and k -ary operations f_1, \dots, f_n , then it also contains the

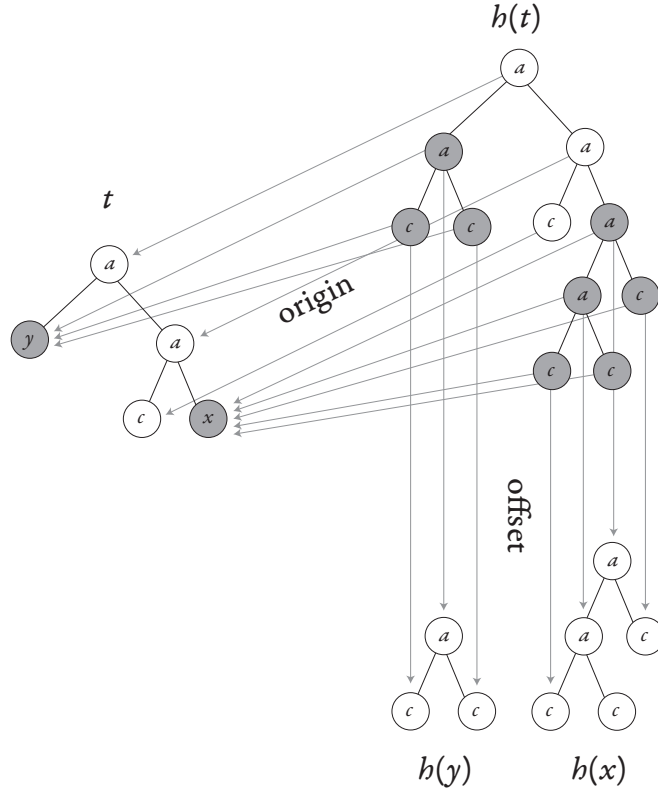


Figure 2: The origin and offset functions. In this example, Σ has one symbol a of rank two, and one symbol c of rank zero. The origin function is from nodes of $h(t)$ to nodes of t . The offset function is defined on the “new” nodes in $h(t)$, i.e. those nodes in $h(t)$ that have labels in Γ , and which have darker colour in the picture. The offset function maps such a node to the corresponding node $h(\sigma)$, where $\sigma \in \Gamma$ is the label of the origin.

k -ary operation

$$\bar{a} \in A^k \quad \mapsto \quad f(f_1(\bar{a}), \dots, f_n(\bar{a})) \in A.$$

The category of ranked sets. To model clones by a monad, we use a different category than sets. The category is ranked sets, i.e. sorted sets where the sort names are natural numbers. Recall that the notions of language theory are parametrised by notions of finite object and finite algebra. We make the following design decisions for the clone monad: a finite ranked set is one with finitely many elements, in particular only finitely many ranks can be achieved in a finite ranked set. We come back to the notion of finite algebra later on.

The clone monad. The *clone monad* maps a ranked set Σ to the ranked set $\text{clo}\Sigma$, where elements of rank n are terms over Σ that use n variables x_1, \dots, x_n (the sequence of variables x_1, x_2, \dots is chosen so that they are fresh with respect to Σ). The terms need not use all variables, and variables may appear with repetitions. The monad multiplication operation

$$\mu_\Sigma : \text{clo clo}\Sigma \rightarrow \text{clo}\Sigma$$

is substitution, as illustrated in Figure 3.

Comparison with clones. We use the name clo-algebra for an Eilenberg-Moore algebra in the monad of clones. A clo-algebra is almost the same thing as a clone in the sense of universal algebra, with the following differences.

- Clones are more general than clo-algebras in the sense that clones admit a distinction between the universe and the operations of rank zero (constants). In other words, it is not necessarily the case that every element of a clone's universe is a constant. (If this is the case, then a clone is called a *polynomial clone*.)
- Clones are less general than clo-algebras in the sense that in a clone, unlike in a clo-algebra, there is an extensionality property with respect to the universe: elements of the clone are uniquely determined by the transformations that they induce on the universe. This is similar to the finite observability condition used in the Pseudovariety Theorem from Section 4.

Therefore, a polynomial clone is the same thing as a clo-algebra that is zero-extensional in the sense every element is determined by its transformation on rank zero elements.

Finitary clones. There is no sense in considering clo-algebras that have a finite universe, because the requirement on projections means that the universe is nonempty on every rank. In clo-algebras, we call a clo-algebra *finite* if it has finitely many elements for every rank, and is finitely generated. The finite generation axiom is natural in the context of recognising languages (every

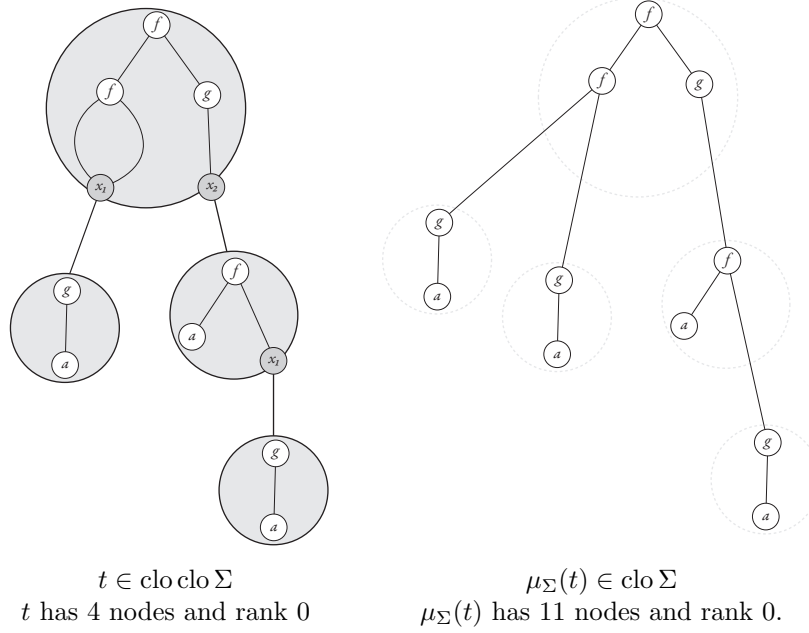


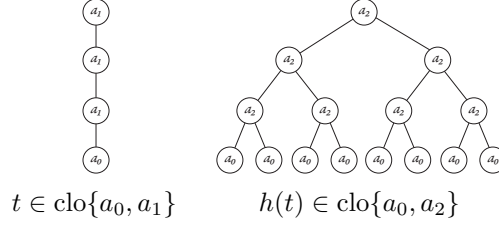
Figure 3: Example of multiplication in a clo-algebra. The ranked alphabet Σ has elements a, g, f of arities 0, 1, 2 respectively. The left picture represents a tree $t \in \text{clo clo } \Sigma$, where variable x_1 is used twice in the label of the root, which is drawn using parallel edges. This double use results in duplication after multiplication is applied. The light grey dotted circles on the right are not part of $\text{mul}_{\text{clo } \Sigma}(t)$, they just highlight how $\mu_\Sigma(t)$ is obtained from t .

recognisable clo-language over a finite alphabet is recognised by a finitely generated clo-algebra), but it is not superfluous – there exist clones over a three element universe that are not finitely generated, as shown by Yanov and Muchnik in [YM59], and this is even the case for polynomial clones [ÁDH83].

Example 7. This is a non-example of a pseudovariety. Let us revisit first-order logic on trees as defined in Example 6. A language of ranked trees can be seen as a special case of a clo-language, which happens to contain only elements of rank zero. Such languages are not closed under inverse images of clo-morphisms, which is witnessed by the following example, essentially due to Potthoff [Pot95]. (Recall that first-order definable language were closed under inverse morphisms for the monad of ranked trees, as shown in Example 6. What worked in Example 6 and no longer works in this example is that for clo-morphisms, a node is not uniquely determined by its offset and origin.) Consider letters a_0, a_1, a_2 with ranks 0, 1, 2 respectively, and consider the clo-morphism

$$h : \text{clo}\{a_0, a_1\} \rightarrow \text{clo}\{a_0, a_2\}$$

which maps a_0 to a_0 , and which maps a_1 to the term $a_2(x_1, x_1)$. This morphism sends trees that look like words to complete binary trees, as shown below:



There is a first-order formula φ that is true in complete binary trees of even depth, and false in complete binary trees of odd depth. The formula says that if one follows the unique path that begins in the root, and then turns left, right, left, right, etc., then one ends up in a leaf that is a left child. The inverse image, under the clo-morphism h , of the language defined by φ is the set of trees over alphabet $\{a_0, a_1\}$ which have even depth. This inverse image is not definable in first-order logic, and therefore first-order definable tree languages are not closed under inverse images of clo-morphisms.

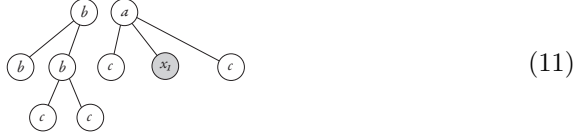
In particular, first-order logic does not form a pseudovariety of clo-languages. Therefore clones, or at least syntactic clones, are not the right tool to study first-order logic on trees. As shown in [ÉW03], his problem can be solved by using preclones, which are a variant of nonduplicating clones where every variable is used only once. The inadequacy of clones in this context is a bit of a shame, because clones have a better developed theory than preclones, e.g. Rosenberg classifies clones with a minimal set of operations that contains something other than projections [Ros86] or clones with a maximal set of operations that does not contain all operations [Ros70], while Hobby and McKenzie classify congruences in a finite clone [HM88]. \square

9.3 Forests of unranked trees

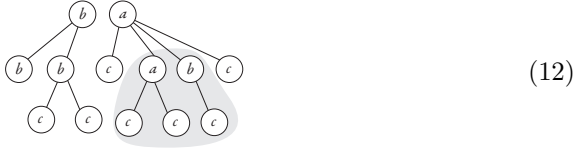
We present a monad for modelling trees, which corresponds to forest algebra [BW08]. As in the previous two monads, the trees are finite (finitely many nodes) and labelled (each node comes with a label). Unlike for the two previous monads trees are unranked, i.e. the number of children of a node is not determined by its label, and can be arbitrarily large. We also assume that trees are sibling-ordered, i.e. the children of a node come with a total order. Finally, instead of trees it will be more convenient to talk about *forests*, which we define to be ordered sequences of trees, i.e. ordered sequences of trees that are unranked, labelled and sibling-ordered. Here is a picture of a forest:



Forests and contexts. The monad in this section will correspond to forest algebra. The principal idea behind forest algebra is to use two kinds of objects, namely forests and contexts. Forests have already been described above. A *context* is defined to be a forest with exactly one distinguished leaf, which is called the *port* of the context⁶. Here is a picture of a context, with the port being labelled by x_1 :



The idea behind the port is that it can be replaced with a forest (or another context). One needs to be careful with the notion of replacement, because a port is a single node, while the forest that will replace it might have multiple roots, e.g. the forest in (10). The result of the replacement is that the all the roots of the inserted forest become children of the parent of the port, e.g. the result of replacing the port of (11) by the forest (10) is illustrated below, with the grey background indicating what used to be the port:



The forest monad. We now define the monad for forests and contexts, which is called the *forest monad*. Line in all previously considered monads, the main idea is that one can replace any node with another element of the monad. In the forest monad, we will use the following discipline: leaves in a forest or context are be replaced by forests, while non-leaves are be replaced by contexts. This leads to a two-sorted alphabet: there are *forest labels*, which are found in leaves, and there are *context labels*, which are found on non-leaves.

More formally, the forest monad, denoted by F , is in the category of two-sorted sets, where the sort names are “forest” and “context”. When applied to a sorted set Σ , the forest monad F yields the following sorted set $F\Sigma$

- on the forest sort, $F\Sigma$ contains nonempty forests labelled by Σ such that leaves are labelled by letters of “forest” sort, while non-leaves are labelled by letters of “context” sort;
- on the context sort, $F\Sigma$ contains contexts labelled by Σ in the same way as in the previous item.

⁶One could consider a variant of this monad without the requirement that the port appears in exactly one leaf, we keep this requirement so that the monad ends up describing forest algebra introduced in [BW08]. Furthermore, allowing ports in many leaves would break the argument in Example 8, actually first-order logic would no longer be a pseudovariety.

The unit operation in the monad F maps a forest element a to *unit forest* that looks like this



and maps a context element a to a *unit context* that looks like this



The multiplication operation in the monad is based on the intuitions of replacement depicted in pictures (10), (11) and (12). The operation is illustrated in Figure 4.

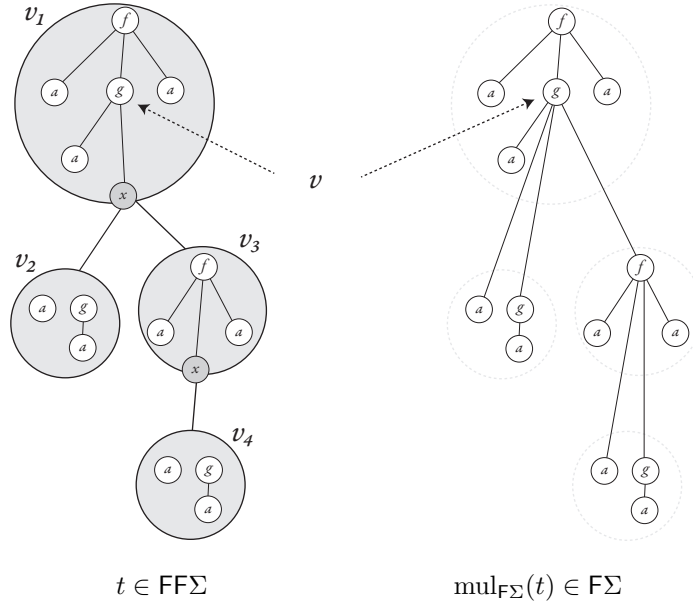
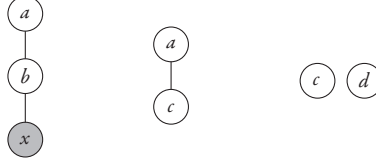


Figure 4: Example of multiplication in the forest monad. Before multiplication, t has two context nodes v_1 and v_3 and two forest nodes v_2 and v_4 . After multiplication, t has fourteen nodes, which correspond to the non-variable nodes in the labels of v_1, \dots, v_4 . Note how the x in the label of node v_1 is replaced by three nodes, namely the two roots of v_2 and the one root of v_3 , resulting in a change of the number of children for node v .

A *finite alphabet* is a two-sorted set finite that is finite on both sorts, and a *finite F-algebra* is one whose universe is finite.

Lemma 9.1 *Every F-algebra is spanned by the subfunctor F_0 which maps Σ to*



where a, b are context elements of Σ , and c, d are forest element of Σ .

Proof.

The lemma boils down to the following easy fact. Every forest or context can be built out of units forests and unit contexts by the following operations: replacing the port of a context by another context or a forest, and concatenating two forests. \square

By the above lemma, every F-algebra is uniquely determined by its F_0 -reduct. This reduct is exactly the same thing as a forest algebra from [BW08], in the variant of forest algebra where there is no empty forest or context. One advantage of seeing forest algebras as a special case of monads is the we can apply the general theorems from the first part to see that forest algebra has a syntactic morphism theorem (already known) or a pseudovariety theorem (not present in the literature).

Example 8. Let us revisit first-order logic on trees, as considered in Examples 6 and 7. To a forest one can assign a logical structure, where the universe is the nodes of the forest, there are unary predicates for the labels, and two binary predicates for the descendant and document orders (document order is the order in which nodes are visited in depth first search, which takes into account the order on siblings). For a context, the structure is defined the same way, except there is constant which denotes the port. A language $L \subseteq F\Sigma$ is called first-order definable if it can be defined by a formula of first-order logic in terms of the logical structure defined above. One can show that first-order definable language form a pseudovariety. The interesting case is to show that if

$$h : F\Sigma \rightarrow F\Gamma$$

is a F-morphism, then

$$t \sim_n s \quad \text{implies} \quad h(t) \sim_n h(s) \quad \text{for every } n \in \mathbb{N},$$

where \sim_n says that the corresponding logical structures have the same first-order theory of rank n . This is proved using the same origin and offset argument as in Example 6. \square

9.4 A monad for infinite unranked forests.

The ω -forest monad, denoted by $X \mapsto \omega F X$, is defined like the monad F , with the difference that infinite forests and infinite contexts are also allowed (assume finite branching, though). The problem with this monad is that it is unclear what finite algebra should be in this case. Clearly, the algebra needs to be finite on both sorts, but this is not sufficient, as the following example shows.

Example 9. Consider an alphabet Σ in the sense of the monad ωF , i.e. an alphabet with elements of sorts “forest” and “context”. Let L be an arbitrary set of trees over the alphabet L , not necessarily MSO definable. Define $\text{dense}L$ to be those forests where every node has a some descendant with a subtree in L . We claim that $\text{dense}L$ is recognised by an ωF -algebra \mathbf{A} with a four element universe. There forest sort has elements “forests in L ” and “forests not in L ”. The context sort has elements: “every node outside the port path has some descendant with a subtree in L ” and “some node outside the port path has no descendants with a subtree in L ”; where the port path is defined to be the ancestors of the port. The dependence on L in the algebra \mathbf{A} is seen in the multiplication operation

$$\text{mul}_{\mathbf{A}} : \omega F A \rightarrow A$$

which maps infinite objects to elements of the universe A . In particular, there are uncountably many ωF -algebras with finite universes, and there is no hope of representing them in a finite way. \square

As witnessed by the above example, the notion of finite algebra should have some additional requirements. Let us make the design decision that languages recognised by finite algebras should be exactly those that can be defined in MSO. The question of finding an adequate notion of finite ωF -algebra is a monad formulation of an open problem in the community of algebraic language theory, namely the problem of a finding an algebraic model for MSO on infinite trees. The fact that we use monads, or that the trees are unranked, does not seem to be important.

A simpleminded solution is to define an ωF -algebra \mathbf{A} to be finite if its universe is finite on both sorts, and the multiplication operation is MSO definable, in the sense that every language

$$\text{mul}_{\mathbf{A}}^{-1}(a) \subseteq \omega F A \quad \text{with } a \in A$$

is MSO definable. Adjusting for a different terminology, this is the solution proposed in [BI09], where it is shown that syntactic algebras can be computed, one can check if an algebra satisfies given equalities, and the algebras can be used to decide questions such as “is a given language of infinite trees definable in the temporal logic EF ?”. This definition of finite algebra is compatible with the results from Part I, in particular with the Syntactic Morphism Theorem and the Pseudovariety Theorem. Examples of language classes that are pseudovarieties include: languages defined in weak MSO, i.e. only using existential quantification over finite sets; languages recognised by nondeterministic (respectively, alternating) tree automata that use parity ranks from a given subset $\Omega \subseteq \mathbb{N}$.

10 Future work

This section sketches some potential monads to study in the future, with reasons for studying them.

- Unranked trees with possibly infinite branching (or graphs, which should not make a difference) modulo bisimulation. The hope would be that recognisable languages, under a suitably chosen notion of finite algebra, would be the same thing as definable in μ -calculus.
- Edge labelled hypergraphs. This looks like a monad, because a hypergraph with n distinguished port vertices can be substituted for a hyperedge of rank n , in the same spirit as Figure 3. The hope would be to describe tree width or clique width as submonads generated by finite subfunctors (as defined in Section 5).
- Typed terms of λ -calculus with fixpoints, modulo equivalence. The hope would be to describe the work of Salvati and Walukiewicz.
- Relations on words with origin information, as a generalisation of transducers with origin information from [Boj14]. The hope would be to give an algebraic framework for asynchronous relations on words with origin. The origin information would cure problems like no syntactic object, or undecidability of universality, which plague asynchronous relations without origin.

Part III

Profinite Monads

In this part, we show that for every monad \mathbb{T} , at least in the category of sets, there is a profinite version $\overline{\mathbb{T}}$. This gives immediately definitions, and basic theorems about, things like profinite words, profinite countable chains, profinite trees, etc. We also study the special case of profinite words, and show how the generic notion of recognisable language instantiates to an interesting class of languages of profinite words.

11 Stone duals and topology on an algebra

Profinite constructions are common in mathematics. For recognisable languages, the best known profinite construction is the semigroup of profinite words. In this section, we show how profinite are defined on the abstract level of monads. The main results of this section are:

- Lattices of languages are exactly those families of languages that can be defined by profinite implications. This generalises to monads a result that was proved for semigroups in [GGP08]. As a corollary, we get a monad generalisation of the Reiterman theorem [Rei82], which says that pseudovarieties are exactly those families of languages that can be defined by profinite identities, which are a stronger form of profinite implications.
- Every class of languages, e.g. context-free or decidable, can be used to yield get some kind of profinite object, but only recognisable languages can be used if we want algebraic operations to be uniformly continuous. These results are monad generalisations of results that were proved for semigroups in [GGP10].

11.1 Stone duals of Boolean algebras

In this section, we recall the definition of the Stone dual of a Boolean algebra, and how Stone duals can be used to characterise lattices. Section 11.1 does not talk about monads.

Consider a Boolean algebra

$$(A, \cap, \cup, \neg).$$

Define an *ultrafilter* in A to be a proper subset $U \subset A$ which is closed under intersections, and which contains every element of A or its complement but not both. The *Stone dual* of A , denoted by $\mathbf{Stone}A$, is defined to be the following topological space. The points in the space are ultrafilters in the Boolean algebra. The topology is generated by base open sets which are of the form

$$\bar{a} \stackrel{\text{def}}{=} \{U : U \text{ is an ultrafilter containing } a\} \quad \text{for } a \in A.$$

The topology on the Stone dual is known to compact and Hausdorff. One of the advantages of the Stone dual is that it can be used to describe lattices of languages, including the special case of pseudovarieties of languages.

Profinite implications. We begin by repeating a result from [GGP08], which says that for an arbitrary Boolean algebra, lattices are exactly those sets which are defined by profinite implications. (This terminology is different than [GGP08], which uses the name “equation” for what we call an implication.)

Consider a Boolean algebra A . (In the context of this paper, it is convenient to think of A as being all recognisable subsets of $\mathsf{T}\Sigma$. In this case, elements of

the Boolean algebra are themselves sets.) A *profinite implication* over a Boolean algebra A is an expression of the form $w \rightarrow v$, where $w, v \in \text{Stone}A$. The arrow is just part of the syntax, so formally a profinite implication is simply a pair of elements from $\text{Stone}A$. An element $a \in A$ (e.g. a recognisable language, when the Boolean algebra consists of recognisable languages) is said to satisfy the implication if $a \in w$ implies $a \in v$. A subset $B \subseteq A$ (e.g. a family of recognisable languages when the Boolean algebra consists of recognisable languages) is said to be defined by a set of profinite implications if it contains exactly the elements that satisfy all profinite implications from the set. In the following lemma, B is called a *lattice* if it contains the 0 and 1 in a Boolean algebra, and is closed under finite unions and finite intersections. The following theorem can be found implicitly in [GGP08], and maybe earlier as well.

Theorem 11.1 *Let A be a Boolean algebra and let $B \subseteq A$. Then B is a lattice if and only if it is definable by profinite implications.*

Proof.

It is easy to see that if a subset of A is definable by profinite implications, then it is a lattice. To prove the other implication, consider a lattice $B \subseteq A$. We claim that B is defined by the set of profinite implications:

$$\{x \rightarrow y : \text{for every } a \in B, \text{ if } a \in x \text{ then } a \in y\}. \quad (13)$$

By definition, every element of B satisfies the profinite implications above. Let then $a \in A$ be such that a satisfies all the profinite implications above. To prove the theorem, we will show $a \in B$.

Recall the definition of \bar{a} in the definition of the Stone dual, which is that \bar{a} is the set defined by $a \in x$ iff $x \in \bar{a}$. We first claim that every $x \in \bar{a}$ satisfies

$$x \in \bar{b} \subseteq \bar{a} \quad \text{for some } b \in B \quad (14)$$

For $x \in \bar{a}$, define $[x] \subseteq \text{Stone}A$ to be the intersection

$$\bigcap_{b \in B \cap x} \bar{b}.$$

It is easy to see that $[x]$ is the set of all $y \in \bar{A}$ such that the profinite implication $x \rightarrow y$ belongs to the set (13). By assumption that a satisfies all these profinite implications, it follows that $[x] \subseteq \bar{a}$. Note that $[x]$ is an intersection of sets that are closed. By compactness of the Stone dual, $[x]$ is equal to an intersection of finitely many \bar{b} with $x \in b \in B$. Furthermore, the intersection is nonempty, because B contains the greatest element of the Boolean algebra, being a lattice. Because B is closed under finite intersections, it follows that $[x] = \bar{b}$ for some $b \in B$ with $x \in \bar{b}$. Together with $[x] \subseteq \bar{a}$, this proves (14).

From (14) it follows that \bar{a} is the union of all \bar{b} ranging over $b \in B$ such that $\bar{b} \subseteq \bar{a}$. By compactness, the union can be made finite, and by closure of B under finite union, it follows that there is some $b \in B$ such that $\bar{a} = \bar{b}$. Finally, since a, b are in the Boolean algebra, it follows that $a = b$. \square

11.2 Stone duals of T-algebras

Section 11.1 did not use monads and recognisability. In this section, we consider the special case of Stone duals of recognisable languages in a T-algebra. Using this Stone dual, we prove a monad version of the Reiterman theorem, which says that a class of recognisable languages is a language pseudovariety if and only if it can be defined by profinite identities.

The Stone dual of a T-algebra. Fix a monad T in the category of sets. We assume that finite alphabets are finite sets, and finite algebras are algebras with finite universes. The results can be easily generalised to sorted sets. Let \mathbf{A} be a T-algebra, not necessarily finite. Define $\text{rec}\mathbf{A}$ to be the subsets of the universe of \mathbf{A} that are recognised by T-morphisms from \mathbf{A} into finite T-algebras. Since $\text{rec}\mathbf{A}$ is a Boolean algebra, it has a Stone dual, which we denote by $\text{Stone}\mathbf{A}$.

Example 10. Consider the monad of finite nonempty words, where +-algebras are semigroups, and +-morphisms are semigroup morphisms. Consider the semigroup Σ^+ where Σ is a finite alphabet. An element of $\text{Stone}\Sigma^+$ is an ultrafilter in the Boolean algebra of recognisable languages over Σ . Recalling the definition of an ultrafilter, an element of $\text{Stone}\Sigma^+$ is a family of recognisable languages over Σ , which is closed under intersection, and which contains every recognisable language or its complement.

A simple example of such an ultrafilter is one that is induced by a word $w \in \Sigma^+$, namely the ultrafilter of recognisable languages which contain w . Stated differently, $\text{Stone}\Sigma^+$ can be seen as a generalisation of Σ^+ .

Here is a more exciting ultrafilter, which corresponds to taking the idempotent power of a finite word. Recall the well known fact that in every finite semigroup \mathbf{A} of size n , the function $a \mapsto a^{n!}$ maps every element of \mathbf{A} to an idempotent, i.e.

$$a^{n!} \cdot a^{n!} = a^{n!},$$

and this element is the unique idempotent power of a , i.e. if

$$a^k \cdot a^k = a^k \quad \text{implies} \quad a^k = a^{n!}.$$

We write $a^\#$ for this idempotent power. A common notation would be w^ω , but choose $\#$ to avoid conflict with the ω power in infinite words. For every semigroup morphism $h : \Sigma^+ \rightarrow \mathbf{A}$, and every $w \in \Sigma^+$, we have

$$h(w^{n!}) = h(w)^\# \quad \text{for all but finitely many } n.$$

This implies that for every recognisable language $L \subseteq \Sigma^+$ and every word $w \in \Sigma^+$, either L contains $w^{n!}$ for all but finitely many n ; or L does not contain $w^{n!}$ for all but finitely many n . This in turn implies that the set of languages

$$\{L \subseteq \Sigma^+ : L \text{ is recognisable and } w^{n!} \in L \text{ for all but finitely many } n\}$$

is an ultrafilter, which we denote by $w^\#$. \square

Running Example 8. Consider the monad of ∞ -words used in the running example. As for semigroups, for every ∞ -algebra \mathbf{A} and every $a \in \mathbf{A}$ there is a unique idempotent power $a^\#$. Let $w \in \Sigma^+$ is a finite nonempty word. As in Example 10, one can also define profinite ∞ -word $w^\#$, namely the ultrafilter

$$\{L \subseteq \Sigma^\infty : L \text{ is recognisable and } w^{n!} \in L \text{ for all but finitely many } n\}. \quad (15)$$

Note that in the monad for ∞ -words, the notation w^ω stands for an actual infinite word, which can then be treated as a profinite word, i.e.

$$\{L \subseteq \Sigma^\infty : w^\omega \in L\} \quad (16)$$

\square

Theorem 11.1 can be applied to \mathbf{StoneA} ; for instance if the monad is the monad of finite words, and \mathbf{A} is Σ^+ , then Theorem 11.1 says that a family of recognisable languages over Σ is a lattice if and only if it is definable by a set of profinite identities.

Running Example 9. Consider a language $L \subseteq \Sigma^\infty$. Define the *first difference distance* between two ∞ -words to be zero if they are equal, and otherwise to be $1/n$ where n is the first position where the words have a different label. Define a *safety* language to be a set of ∞ -words which is closed under limits with respect to first difference distance. In other words, safety says that if w is an infinite word such that every finite prefix of w can be extended to some word from the language, then w itself belongs to the language.

It is easy to see that recognisable safety ∞ -languages form a lattice, and therefore by Theorem 11.1 they must be characterised by a set of profinite implications. One can show that the set of profinite implications is

$$vw^\#u \rightarrow vw^\omega, \quad (17)$$

where $v, w, u \in \Sigma^+$ and the powers $\#$ and $^\omega$ are understood as profinite words in the same sense as in (15) and (16), i.e. the two sides of the above profinite implication are the following ultrafilters, respectively.

$$\begin{aligned} &\{L \subseteq \Sigma^\infty : L \text{ is recognisable and } vw^{n!}u \in L \text{ for all but finitely many } n\} \\ &\{L \subseteq \Sigma^\infty : vw^\omega \in L\} \end{aligned}$$

Indeed, suppose that L is a safety language, and it satisfies the left side of the profinite implication for some v, w, u , which means that it contains $vw^{n!}u$ for almost all n . By safety, the language L must also contain the limit of the sequence $vw^{n!}u$, which is vw^ω , and therefore L satisfies the right side of the profinite implication. The more interesting case is the converse, i.e. showing that if L satisfies all profinite implications of the form (17), then it is a safety language. To prove that L is a safety language, assume that it contains all words

$$w_1, w_2, \dots$$

which tend, under first difference distance, to some word w . We need to show that L also contains w . If w is finite, then all but finitely many of the words w_i are equal to w , and therefore $w \in L$. Assume therefore that w is infinite. By the Ramsey Theorem, w can be factorised as

$$w = v_0 v_1 v_2 \cdots$$

such that all word v_1, v_2, \dots have the same image under some ∞ -morphism

$$h : \Sigma^\infty \rightarrow \mathbf{A}$$

which recognises L . This means that for every i , all but finitely many of the words w_j have a prefix of the form $v_0 v_1 \cdots v_i$. Without loss of generality, we may assume that

$$w_i = v_0 v_1 \cdots v_i u_i,$$

and also without loss of generality we may assume that all words u_i have the same image under the ∞ -morphism h . Since L is recognised by h , it follows that L contains all words of the form $v_0(v_1)^n u_1$. By (17), L also contains $v_0(v_1)^\omega$, which has the same image under h as w , and therefore L also contains w . \square

Defining pseudovarieties by identities As mentioned in its proof, Theorem 11.1 does not use any properties of recognisability over a monad. We now present a corollary of the theorem, which is more specific to monads, and which says that pseudovarieties can be defined by identities.

To define profinite identities, we observe that both \mathbf{T} -morphisms and unary polynomials can be naturally lifted to profinite objects, as described below. Suppose that

$$f : \mathbf{A} \rightarrow \mathbf{B}$$

is a function, not necessarily a \mathbf{T} -morphism, which has the property that recognisable languages are preserved under inverse images of f , i.e.

$$L \in \text{rec}\mathbf{B} \quad \text{implies} \quad f^{-1}(L) \in \text{rec}\mathbf{A}. \quad (18)$$

Then for every ultrafilter U of recognisable languages subsets of \mathbf{A} , the family

$$(\text{Stone}f)(U) \stackrel{\text{def}}{=} \{L \in \text{rec}\mathbf{B} : f^{-1}(L) \in U\}$$

is an ultrafilter of recognisable subsets of \mathbf{B} . In other words, f lifts to a function

$$\text{Stone}f : \text{Stone}\mathbf{A} \rightarrow \text{Stone}\mathbf{B}.$$

One can show that the mapping Stone defined this way is a functor, whose domain is the category of \mathbf{T} -algebras with functions that satisfy (18). We will be interested in two special cases of functions f with property (18), i.e. when f is a \mathbf{T} -morphism and when f is a unary polynomial. The following fact is an immediate consequence of the definitions.

Fact 11.2 *If $f : \mathbf{A} \rightarrow \mathbf{B}$ satisfies (18), and $L \subseteq \mathbf{B}$, then*

$$f^{-1}(L) \text{ satisfies } w \rightarrow v \quad \text{iff} \quad L \text{ satisfies } (\mathbf{Stone}f)(w) \rightarrow (\mathbf{Stone}f)(v)$$

Define a *profinite identity* to be an expression of the form $w = v$ where $w, v \in \mathbf{StoneTX}$ for some finite set X of variables. As in profinite implications, the equality sign is just part of the syntax, and formally a profinite identity is simply the pair (w, v) . If \mathbf{A} is a \mathbf{T} -algebra, then we say that $L \in \mathbf{recA}$ satisfies a profinite identity $w = v$ if it satisfies

$$(\mathbf{Stone}(p \circ h))(w) \leftrightarrow (\mathbf{Stone}(p \circ h))(v)$$

for every unary polynomial $p \in \mathbf{pol}_1 \mathbf{A}$ and every \mathbf{T} -morphism $h : \mathbf{TX} \rightarrow \mathbf{A}$, where \leftrightarrow means that the profinite implication is satisfied both ways. As mentioned above, the mapping \mathbf{Stone} is a functor, and therefore $\mathbf{Stone}(p \circ h)$ is the same as $(\mathbf{Stone}p) \circ (\mathbf{Stone}h)$. Intuitively speaking, for every substitution of the variables, i.e. every morphism h , and in every environment, i.e. for every unary polynomial p , the two sides of the profinite identity are equivalent.

Example 11. Consider the monad of finite words, and the profinite identity

$$xy = yx.$$

Formally speaking, the profinite identity uses profinite words, call them “ xy ” and “ yx ”, which correspond to the finite words xy and yx , as described in the second paragraph of Example 10. A recognisable language $L \subseteq \Sigma^+$ satisfies this profinite identity if

$$p^{-1}L \in \text{“}wv\text{”} \quad \text{iff} \quad p^{-1}L \in \text{“}vw\text{”}$$

holds for every unary polynomial p over Σ^+ and every $w, v \in \Sigma^+$. Unraveling the definitions of the profinite words “ xy ” and “ yx ”, this means that

$$wv \in p^{-1}L \quad \text{iff} \quad vw \in p^{-1}L.$$

This means that the language must be commutative. \square

Example 12. Consider again the monad of finite words. Recall the profinite word $w^\#$ that was described in Example 10. In a similar way, we can define a profinite word $w^{\#+1}$ to be the ultrafilter

$$\{L \subseteq \Sigma^+ : L \text{ is recognisable and } w^{n!+1} \in L \text{ for all but finitely many } n\}.$$

Consider the following profinite identity over a single variable x :

$$x^\# = x^{\#+1},$$

which the reader might recognise as the identity defining aperiodic semigroups. We now check that this is the case under the definitions of this section. A recognisable language $L \subseteq \Sigma^+$ satisfies this profinite identity if for every unary polynomial p over the semigroup Σ^+ and every $w \in \Sigma^+$, the following conditions are equivalent

- $w^{n!}$ belongs to $p^{-1}L$ for all but finitely many n ;
- $w^{n!+1}$ belongs to $p^{-1}L$ for all but finitely many n .

This implies that for every word w and unary polynomial p , the language $p^{-1}L$ contains either finitely many, or all but finitely many, of the powers w^n . This means that the syntactic semigroup of the language is aperiodic, which means that the language is definable in first-order logic, by Schützenberger’s theorem. \square

The above two examples showed that, in the monad of finite words, some classes of recognisable languages can be characterised via profinite identities. The following theorem, which is a monad version of the Reiterman Theorem [Rei82], says that this is the case for all pseudovarieties, although infinite sets of identities might need to be used. Note that although profinite identities can be evaluated in a recognisable subset of an arbitrary \mathbf{T} -algebra, in the following theorem we talk only about \mathbf{T} -languages, i.e. recognisable subsets of algebras of the form $\mathbf{T}\Sigma$ where Σ is a finite alphabet.

The following theorem uses the polynomial variant of language pseudovarieties that is mentioned in Section 4.2, i.e. this is a class of recognisable languages that is closed under polynomial derivatives, inverse morphisms, and Boolean combinations.

Theorem 11.3 *Let \mathbb{L} be class of recognisable \mathbf{T} -languages. Then \mathbb{L} is a pseudovariety if and only if it is defined by a set of profinite identities.*

Proof.

The right-to-left implication is essentially checking the definitions, while the left-to-right implication is a corollary of Theorem 11.1.

Right-to-left implication. Suppose that I is a set of profinite identities, and let \mathbb{L} be the set of recognisable \mathbf{T} -languages which satisfy all of these identities. We need to show that \mathbb{L} is a pseudovariety. As mentioned in the proof of Theorem 11.1, satisfying profinite implications is closed under unions and intersections. It is easy to see that satisfying a profinite identity is invariant under complementation. Therefore, \mathbb{L} is closed under Boolean combinations. It remains to show that \mathbb{L} is closed under inverse images of morphisms and under polynomial derivatives. Let then $L \subseteq \mathbf{T}\Gamma$ be a language that satisfies all identities from I , and suppose that

$$h : \mathbf{T}\Sigma \rightarrow \mathbf{T}\Gamma$$

is a \mathbf{T} -morphism. We will show that $h^{-1}(L) \subseteq \mathbf{T}\Sigma$ also satisfies all identities in I . (The proof for polynomial derivatives is the same and is omitted.) By definition, we need to show that for every profinite identity $w = v$ in I which is over variables X , and every

$$f : \mathbf{T}X \rightarrow \mathbf{T}\Sigma \quad q \in \text{pol}_1 \mathbf{T}\Sigma$$

which are a \mathbf{T} -morphism and unary polynomial, respectively, we have

$$h^{-1}(L) \text{ satisfies } (\mathbf{Stone}(q \circ f))(w) \leftrightarrow (\mathbf{Stone}(q \circ f))(v)$$

By Fact 11.2 and functoriality of \mathbf{Stone} , the above is equivalent to saying that

$$L \text{ satisfies } (\mathbf{Stone}(h \circ q \circ f))(w) \leftrightarrow (\mathbf{Stone}(h \circ q \circ f))(v),$$

which is the same as saying that

$$L \text{ satisfies } (\mathbf{Stone}(r \circ h \circ f))(w) \leftrightarrow (\mathbf{Stone}(r \circ h \circ f))(v),$$

where $r \in \mathbf{pol}_1 \mathbf{T}\Gamma$ is the image of q under h , see (1). Since $h \circ f$ is itself a \mathbf{T} -morphism, the above holds by assumption that L satisfies all profinite identities from L .

Left-to-right implication. Let \mathbb{L} be a class of recognisable \mathbf{T} -languages which is a language pseudovariety. We need to show that \mathbb{L} is definable by profinite identities. For a finite alphabet Σ , define \mathbb{L}_Σ to be all languages from \mathbb{L} over alphabet Σ , and let I_Σ be the set of profinite implications that are satisfied by all languages in \mathbb{L}_Σ . Define I to be the set of profinite identities $w = v$ such that some I_Σ contains the profinite implication $w \rightarrow v$. We show below that \mathbb{L} is defined by I , i.e. a language belongs to \mathbb{L} if and only if it satisfies all identities in I .

- Suppose that L satisfies all profinite identities from I . In particular, this means that L satisfies all profinite implications from I_Σ . Since \mathbb{L} is a pseudovariety, it follows that \mathbb{L}_Σ is a lattice, and therefore by Theorem 11.1, \mathbb{L}_Σ is defined by the profinite implications from I_Σ , which means that L belongs to \mathbb{L}_Σ .
- Suppose that L belongs to \mathbb{L} . We need to show that L satisfies all profinite identities from I . In other words, we need to show that if $w \rightarrow v$ is a profinite implication from I_Γ , then

$$L \text{ satisfies } (\mathbf{Stone}(p \circ h))(w) \leftrightarrow (\mathbf{Stone}(p \circ h))(v)$$

for every unary polynomial $p \in \mathbf{pol}_1 \mathbf{T}\Sigma$ and every \mathbf{T} -morphism $h : \mathbf{T}\Gamma \rightarrow \mathbf{T}\Sigma$. By Fact 11.2, and closure properties of a pseudovariety, this boils down to the profinite implications $w \leftrightarrow v$ being satisfied by a language from \mathbb{L}_Γ , which holds by definition of I_Γ .

□

11.3 Uniform continuity

The results in Section 11.1 and 12.4 did not really use assumptions on recognisability. Actually, Theorem 11.1 would also be true for non-recognisable languages, as shown in the following example.

Example 13. Consider the monad of finite words. For the purpose of this example, define $\text{Stone}\Sigma^+$ to be the Stone dual of the Boolean algebra of decidable languages over the alphabet Σ , as opposed to the recognisable languages considered in the previous section. Also for the purpose of this example, define a pseudovariety to be a class of decidable languages that is closed under Boolean combinations, inverse morphisms and polynomial derivatives, e.g. the polynomial time complexity class P is such a pseudovariety. Inspection of the proofs in Section 11.1 shows that Theorem 11.3 would also work in this setup, in particular P is definable by profinite identities. \square

In this section, we show that recognisable languages are special in some sense. The result in this section is a generalisation of Theorem 4.1 in [GGP10] from semigroups to a certain class of monads over sets.

Uniformly continuous operations. We begin by defining the notion of a uniformly continuous operation in a \mathbf{T} -algebra, with respect to a chosen class of languages. Let

$$\mathcal{L} = \{L_1, L_2, \dots\}$$

be a countable family of subsets of a set A , along with some enumeration. We say that $L \in \mathcal{L}$ separates two elements of A if it contains exactly one of them. Define the \mathcal{L} -distance on A to be

$$\text{dist}_{\mathcal{L}}(a, b) = \frac{1}{2^n} \quad \text{where } n \text{ is minimal such that } L_n \text{ separates } a, b.$$

It is easy to see that this is a distance, assuming that every two elements of A are separated by some element of \mathcal{L} . Note how countability is used in the definition. Unravelling the classical definition of uniform continuity, a function

$$f : A^n \rightarrow A$$

is uniformly continuous with respect to \mathcal{L} -distance if for every finite set $\mathcal{K} \subseteq \mathcal{L}$ there is some finite set $\mathcal{M} \subseteq \mathcal{L}$ such that

$$\bigwedge_i v_i \equiv_{\mathcal{M}} w_i \quad \text{implies} \quad f(v_1, \dots, v_n) \equiv_{\mathcal{K}} f(w_1, \dots, w_n)$$

where $\equiv_{\mathcal{K}}$ says that elements cannot be separated by languages from \mathcal{K} , and $\equiv_{\mathcal{M}}$ is the analogous equivalence but lifted pointwise to functions. It follows that although the definition of \mathcal{L} -distance depends on the enumeration of \mathcal{L} , the notion of uniformly continuous function does not.

The goal of this section is to investigate conditions on \mathcal{L} which guarantee that all polynomials of finite arity define uniformly continuous functions. The answer will be that \mathcal{L} needs to contain only recognisable languages. We begin by two examples, which show the result for the special case of semigroups.

Example 14. Let Σ be a finite alphabet, let X be a set of finite semigroups (e.g. all finite semigroups, or all aperiodic semigroups), and consider the \mathcal{L} -distance on Σ^+ where \mathcal{L} is all subsets of Σ^+ recognised by semigroups in X . We

claim that concatenation, which can be seen as a binary polynomial

$$(w, v) \mapsto wv$$

is uniformly continuous with respect to \mathcal{L} -distance. We need to show that for every finite $\mathcal{K} \subseteq \mathcal{L}$ there is some finite $\mathcal{M} \subseteq \mathcal{L}$ such that

$$w_1 \equiv_{\mathcal{M}} w_2 \text{ and } v_1 \equiv_{\mathcal{M}} v_2 \text{ implies } w_1 v_1 \equiv_{\mathcal{L}} w_2 v_2.$$

holds for every words $w_1, w_2, v_1, v_2 \in \Sigma^+$. The languages \mathcal{M} can be taken to be all languages recognised by those semigroups that are used to recognise the languages from \mathcal{K} . The family \mathcal{M} is finite because there are finitely many possible semigroup morphisms from Σ^+ to a finite set of finite semigroups. The same solution works for other operations in Σ^+ that can be built using concatenation. \square

Example 15. As in the previous example, consider the \mathcal{L} -distance on Σ^+ where \mathcal{L} contains some language that is not recognisable. We show that with respect to \mathcal{L} -distance, concatenation might be continuous, but not uniformly continuous.

To show that concatenation might be continuous, suppose that \mathcal{L} contains all singleton languages, e.g. \mathcal{L} is the decidable languages. This implies that the topology generated by \mathcal{L} -distance is discrete, because every singleton set is open. Therefore, the topology on finite powers of Σ^+ is also discrete, and thus concatenation is continuous with respect to \mathcal{L} -distance, like any other operation on this semigroup.

Let us now show that concatenation is not uniformly continuous. Consider some non-recognisable language $L \in \mathcal{L}$. We will show that there is no finite set \mathcal{M} of decidable languages such that

$$w_1 \equiv_{\mathcal{M}} w_2 \text{ and } v_1 \equiv_{\mathcal{M}} v_2 \text{ implies } w_1 v_1 \equiv_{\{L\}} w_2 v_2.$$

for every words $w_1, w_2, v_1, v_2 \in \Sigma^+$. Let then \mathcal{M} be a finite set of languages from \mathcal{L} , or any languages for that matter. Because L is not recognisable, there are infinitely many left derivatives, i.e. languages of the form $x^{-1}L$. Since there are finitely many equivalence classes of $\equiv_{\mathcal{M}}$, there must exist some two words w_1, w_2 such that

$$w_1 \equiv_{\mathcal{M}} w_2 \quad \text{and} \quad w_1^{-1}L \neq w_2^{-1}L.$$

The inequality of derivatives means that there is some v such that

$$w_1 v \not\equiv_{\{L\}} w_2 v,$$

which proves that concatenation is not uniformly continuous. \square

The two examples above are essentially Theorem 4.1 of [GGP10]. The goal of this section is to generalise that result to algebras over abstract monads. The role of concatenation will be played by polynomials of finite arity. In our generalisation we assume that the monad is finitary, and that it is over the category of (unsorted) sets. The proof can be easily generalised to finitely sorted sets. There is one additional assumption in our generalisation, which will require some more definitions.

Observationally complete polynomials. The idea behind observational completeness is that sometimes, instead of using all unary polynomials in the definition of the syntactic congruence, one can use a smaller subset, e.g. unary polynomials that use the variable only once (whatever that may mean in an abstract monad).

Let \mathbf{A} be a \mathbf{T} -algebra. We write $\text{pol}_n \mathbf{A}$ for polynomials with n arguments in the algebra \mathbf{A} ; we do not need to indicate the sorts of these arguments because we use unsorted sets. We use the convention that the variables in a polynomial from $\text{pol}_n \mathbf{A}$ are called x_1, \dots, x_n . Therefore, formally

$$\text{pol}_n \mathbf{A} = \mathbf{T}(A \sqcup \{x_1, \dots, x_n\}).$$

A set $P \subseteq \text{pol}_1 \mathbf{A}$ is called *observationally complete for \mathbf{A}* if the following conditions are equivalent for every a, b in the universe of \mathbf{A} and every subset L of the universe of \mathbf{A} :

$$w \in p^{-1}L \quad \text{iff} \quad w' \in p^{-1}L \quad \text{for every } p \in \text{pol}_1 \mathbf{A} \quad (19)$$

$$w \in p^{-1}L \quad \text{iff} \quad w' \in p^{-1}L \quad \text{for every } p \in \text{pol}_1 \mathbf{A} \cap P. \quad (20)$$

Recall that the condition in (19) is the equivalence relation defined in the proof of the Syntactic Morphism Theorem.

Example 16. Consider the monad of finite words where algebras are semigroups. Call a unary polynomial nonduplicating if it uses its variable exactly once. Such a polynomial is of the form wx_1v where w, v are possibly empty words over the universe of the semigroup. Without loss of generality one could also assume that w, v have length zero or one. It is not difficult to show that the unary nonduplicating polynomials are observationally complete in every semigroups. \square

Running Example 10. Consider the monad of ultimately periodic ∞ -words. It is not difficult to show that in every algebra \mathbf{A} for this monad, an observationally complete set of unary polynomials is

$$\{wx_1v, w(x_1v)^\omega : \text{ where } w, v \in A^*\}.$$

These unary polynomials correspond to the *Arnold congruence* from [Arn85]. \square

Finite covers. Define an *n -ary term* to be an element of $\mathbf{T}\{x_1, \dots, x_n\}$, where x_1, \dots, x_n are the variables used for polynomials. In other words, a term is the special case of a polynomial that does not use any constants, and therefore an *n -ary term* is an *n -ary polynomial* in every algebra. We say that a unary polynomial $p \in \text{pol}_1 \mathbf{A}$ is *covered* by an *n -ary term* q if there exist a_2, \dots, a_n in the universe of \mathbf{A} such that

$$p = q(x_1, a_2, \dots, a_n).$$

A set $P \subseteq \text{pol}_1 \mathbf{A}$ is said to have a finite cover, if there is a finite set Q of terms, of possibly different arities, such that every polynomial in P is covered by some term in Q .

Example 17. The nonduplicating polynomials mentioned in Example 16 are covered by the 3-ary term $x_2x_1x_3$. The nonduplicating polynomials mentioned in Running Example 10 are covered by the two 3-ary terms $x_2x_1x_3$ and $x_2(x_1x_3)^\omega$. Summing up, in both these monads, every algebra has a finite cover for the set of nonduplicating unary polynomials. \square

Characterisation of uniformly continuous term operations. We are now ready to state the theorem that characterises recognisability as a necessary and sufficient condition for uniform continuity of term operations. In the theorem, we write $\text{der}\mathcal{L}$ for the set of all polynomial derivatives of languages from \mathcal{L} .

Theorem 11.4 *Consider a finitary monad T in the setting of sets. Let \mathbf{A} be a finitely generated T -algebra which has an observationally complete set of unary polynomials that has a finite cover. Let \mathcal{L} be a countable family of subsets of the universe of \mathbf{A} . Then*

1. *if \mathcal{L} contains only T -recognisable languages, then every term operation is uniformly continuous for $\text{der}\mathcal{L}$ -distance;*
2. *if \mathcal{L} contains at least one language that is not T -recognisable, then some term operation is not uniformly continuous for \mathcal{L} -distance.*

Before proving the theorem, note that by the discussion in Example 17, the assumptions of the theorem are satisfied by every algebra in the monad of finite words, and by every algebra in the monad of ultimately periodic words.

Proof.

We skip the proof of the first item, which is proved as in Example 14, and does not use the assumption on the observationally complete set of unary polynomials, but uses the assumption on finite generation.

Let us consider the second item. Let P be a set of observationally complete polynomials with a finite cover Q , as in the assumptions of the theorem. Let $L \in \mathcal{L}$ be some language that is not recognisable. Since \mathbf{A} is finitary, we can use the Syntactic Morphism Theorem. It follows that the equivalence in (19) has infinite index, and therefore the equivalence relation defined in (20) as applied to P has infinite index. For a unary polynomial q in the finite cover Q , define \sim_q to be the relation as in (20) but with polynomials restricted to those that are covered by q , i.e. \sim_q identifies $a, b \in A$ if

$$a \in p^{-1}L \quad \text{iff} \quad b \in p^{-1}L \quad \text{for every } p \in \text{pol}_1 \mathbf{A} \text{ covered by } q.$$

Since the relation (20) has infinite index and is the intersection of the finitely many relations \sim_q , there must be some n -ary term $q \in Q$ such that \sim_q has

infinite index. We claim that q , when seen as a polynomial in $\mathbf{pol}_n \mathbf{A}$, is not uniformly continuous. To prove this, consider some finite set $\mathcal{K} \subseteq \mathcal{L}$. Because the index of \sim_q is infinite and the index of $\equiv_{\mathcal{K}}$ is finite, there must be some $a, b \in A$ such that

$$a \not\sim_q b \quad \text{and} \quad a \equiv_{\mathcal{K}} b.$$

Unraveling the definition of \sim_q , this means that there are some a_2, \dots, a_n in the universe of \mathbf{A} such that

$$q(a, a_2, \dots, a_n) \not\equiv_{\{L\}} q(b, a_2, \dots, a_n),$$

which proves that q is not uniformly continuous. \square

12 Profinite monads

In this section, we prove that the Stone dual considered in the previous section has sufficient structure to make it into a monad, i.e. for every monad \mathbf{T} the mappings

$$\begin{array}{ccc} \Sigma & \mapsto & \mathbf{Stone}(\mathbf{T}\Sigma) \\ f : \Sigma \rightarrow \Gamma & \mapsto & \mathbf{Stone}(\mathbf{T}f) \end{array}$$

can be equipped with unit and multiplication to make it a monad, which we will denote by $\overline{\mathbf{T}}$. Because $\overline{\mathbf{T}}$ is a monad, it has its own notion of recognisability, which is related to but richer than the notion of recognisability of the original \mathbf{T} . This richer notion of recognisability is studied in Section 13, on the example of profinite words.

12.1 Definition of the profinite monad

Fix for the rest of this section a monad \mathbf{T} , in the category of sets, the generalisation to sorted sets being straightforward. We explain how to convert \mathbf{T} into a monad, which we denote by $\overline{\mathbf{T}}$, that describes profinite objects over \mathbf{T} .

Types. Instead of Stone duals as studied in the previous section, we will use in this section an alternative definition, which has a more algebraic flavour. For a \mathbf{T} -algebra \mathbf{A} , not necessarily finite, define a *\mathbf{T} -morphism type over \mathbf{A}* to be a function τ which maps every surjective \mathbf{T} -morphism

$$h : \mathbf{A} \rightarrow \mathbf{B} \quad \text{with } \mathbf{B} \text{ finite} \tag{21}$$

to an element $h^\tau \in \mathbf{B}$, subject to the condition that

$$(g \circ h)^\tau = g(h^\tau) \quad \text{for every } h : \mathbf{A} \rightarrow \mathbf{B} \text{ and } g : \mathbf{B} \rightarrow \mathbf{C} \tag{22}$$

where g is a surjective \mathbf{T} -morphism between finite \mathbf{T} -algebras. The set of \mathbf{T} -morphism types over a \mathbf{T} -algebra \mathbf{A} is called its *compactification*, and is denoted by $\bar{\mathbf{A}}$. As a topological space, the set of \mathbf{T} -morphism types is an equivalent definition of the Stone dual defined in the previous section, as stated in the following fact.

Fact 12.1 *If \mathbf{A} is a \mathbf{T} -algebra, then $\text{Stone}\mathbf{A}$ is homeomorphic to $\bar{\mathbf{A}}$, assuming that the base open sets are of the form*

$$\{\tau \in \bar{\mathbf{A}} : h^\tau = b\}$$

for $h : \mathbf{A} \rightarrow \mathbf{B}$ a surjective \mathbf{T} -morphism into a finite \mathbf{T} -algebra and $b \in \mathbf{B}$.

Nevertheless, we use \mathbf{T} -morphism types instead of the Stone dual because they will be more convenient to study the algebraic structure. Define the *profinite extension* of a surjective \mathbf{T} -morphism

$$h : \mathbf{A} \rightarrow \mathbf{B}$$

into a finite \mathbf{T} -algebra \mathbf{B} to be the function

$$\bar{h} : \bar{\mathbf{A}} \rightarrow \mathbf{B}$$

defined by $\bar{h}(\tau) = h^\tau$. In terms of profinite extensions (22), says that the following diagram commutes.

$$\begin{array}{ccc} \bar{\mathbf{A}} & \xrightarrow{\bar{h}} & \mathbf{B} \\ & \searrow \scriptstyle \overline{g \circ h} & \downarrow \scriptstyle \bar{g} \\ & & \mathbf{C} \end{array} \quad (23)$$

Example 18. Consider the monad $+$ of finite words, where $+$ -algebras are semigroups, and $+$ -morphisms are semigroup morphisms. Consider the semigroup Σ^+ where Σ is a finite alphabet. As stated in Fact 12.1, $+$ -morphism types, or semigroup morphism types, over Σ^+ are the same thing as elements of the Stone dual $\text{Stone}\Sigma^+$, which are profinite words as described in Example 10. We now revisit the element $w^\#$ of the Stone dual that was described in Example 10, and describe its corresponding semigroup morphism types.

By definition, a semigroup morphism type over Σ^+ is a function which maps every semigroup morphism

$$h : \Sigma^+ \rightarrow \mathbf{A} \quad \text{with } \mathbf{A} \text{ a finite semigroup}$$

to an element of \mathbf{A} , in a way that is consistent with composition. The idempotent power $w^\#$ described in Example 10 is the morphism type which maps a morphism h to $h(w)^\#$. Let us check that $w^\#$ defined this way is indeed a semigroup morphism type, we need to show that for every semigroup morphisms

$$h : \Sigma^* \rightarrow \mathbf{A} \quad g : \mathbf{A} \rightarrow \mathbf{B}$$

with \mathbf{A} and \mathbf{B} being finite semigroups, and g being surjective, we have

$$g(h(w)^\#) = (g \circ h(w))^\#.$$

This is checked below, assuming that n and m are the sizes of \mathbf{A} and \mathbf{B} .

$$\begin{aligned} g(h(w)^\#) &= \text{(by definition)} \\ g(h(w)^{n!}) &= \text{(because } g \text{ is a semigroup morphism)} \\ g(h(w))^{n!} &= \text{(because } m \leq n \text{ and } m! \text{ is an idempotent power)} \\ (g(h(w)))^{m!} &= \text{(by definition)} \\ (g \circ h(w))^\# &. \end{aligned}$$

□

The functor of $\bar{\mathsf{T}}$. We now define the profinite monad $\bar{\mathsf{T}}$. We assume that the unit and multiplication in the original monad T are denoted by η_Σ and μ_Σ . An object Σ is mapped by $\bar{\mathsf{T}}$ the compactification of the T -algebra $\mathsf{T}\Sigma$, i.e.

$$\bar{\mathsf{T}}\Sigma \stackrel{\text{def}}{=} \overline{\mathsf{T}\Sigma}.$$

The remaining components of the monad, i.e. how $\bar{\mathsf{T}}$ acts on functions, as well as the unit and multiplication operations, are defined and proved correct in the following theorem.

Theorem 12.2 *There are unique operations*

$$\begin{aligned} \bar{\mathsf{T}}f &: \bar{\mathsf{T}}\Gamma \rightarrow \bar{\mathsf{T}}\Sigma \quad \text{for } f: \Gamma \rightarrow \Sigma \\ \bar{\eta}_\Sigma &: \Sigma \rightarrow \bar{\mathsf{T}}\Sigma \\ \bar{\mu}_\Sigma &: \bar{\mathsf{T}}\bar{\mathsf{T}}\Sigma \rightarrow \bar{\mathsf{T}}\Sigma \end{aligned}$$

such that for every finite T -algebra \mathbf{A} and every surjective T -morphism

$$h: \mathsf{T}\Sigma \rightarrow \mathbf{A},$$

into a finite T -algebra, the following diagrams commute

$$\begin{array}{ccccc} \bar{\mathsf{T}}\Gamma & \xrightarrow{\bar{\mathsf{T}}f} & \bar{\mathsf{T}}\Sigma & & \Sigma & \xrightarrow{\bar{\eta}_\Sigma} & \bar{\mathsf{T}}\Sigma & & \bar{\mathsf{T}}\bar{\mathsf{T}}\Sigma & \xrightarrow{\bar{\mu}_\Sigma} & \bar{\mathsf{T}}\Sigma \\ & \searrow \bar{h \circ \mathsf{T}f} & \downarrow \bar{h} & & \downarrow \eta_\Sigma & & \downarrow \bar{h} & & \downarrow \bar{\mathsf{T}}h & & \downarrow \bar{h} \\ & & \mathbf{A} & & \mathsf{T}\Sigma & \xrightarrow{h} & \mathbf{A} & & \bar{\mathsf{T}}\mathbf{A} & \xrightarrow{\overline{\text{mul}}_{\mathbf{A}}} & \mathbf{A} \end{array}$$

Furthermore, equipped with the above operations, $\bar{\mathsf{T}}$ is a monad.

The rest of Section 12.1 is devoted to proving the above theorem. First observe that the operations from the statement of the theorem, if they exist, are uniquely specified by the diagrams in the statement of the theorem, because

an element of $\overline{\mathbf{T}}\Sigma$ is uniquely specified by its values under all possible profinite extensions \bar{h} . We need to check that the operations actually produce morphism types, i.e. the values that they produce satisfy (22). Let us first check that $\overline{\mathbf{T}}f$ produces types, i.e. that

$$\overline{g \circ h \circ \mathbf{T}} = g \circ \bar{h} \circ \overline{\mathbf{T}}f$$

holds for every finite \mathbf{T} -morphisms

$$h : \mathbf{T}\Sigma \rightarrow \mathbf{A} \quad g : \mathbf{A} \rightarrow \mathbf{B}$$

where \mathbf{A}, \mathbf{B} are finite. This is checked below:

$$\begin{aligned} \overline{g \circ h \circ \mathbf{T}}f &= \text{(by definition of } \overline{\mathbf{T}}f) \\ \overline{g \circ h \circ \mathbf{T}}f &= \text{(by (23))} \\ g \circ \bar{h} \circ \overline{\mathbf{T}}f &= \text{(by definition of } \overline{\mathbf{T}}f) \\ g \circ \bar{h} \circ \overline{\mathbf{T}}f & \end{aligned}$$

For the unit operation, the check is even simpler:

$$\begin{aligned} \overline{g \circ h \circ \bar{\eta}_\Sigma} &= \text{(by definition of } \bar{\eta}_\Sigma) \\ g \circ h \circ \eta_\Sigma &= \text{(by definition of } \bar{\eta}_\Sigma) \\ g \circ \bar{h} \circ \bar{\eta}_\Sigma & \end{aligned}$$

Before checking that the multiplication operation defined in the theorem produces types, we check that $\overline{\mathbf{T}}$ is a functor, i.e.

$$\overline{\mathbf{T}}(f \circ g) = \overline{\mathbf{T}}f \circ \overline{\mathbf{T}}g \quad \text{for every } f : \Delta \rightarrow \Sigma \text{ and } g : \Gamma \rightarrow \Delta.$$

To prove the above equality, we show that the two sides of the equality are equal after being composed with functions of the form \bar{h} with

$$h : \mathbf{T}\Sigma \rightarrow \mathbf{A}$$

a \mathbf{T} -morphism into a finite \mathbf{T} -algebra. This is checked below and illustrated in Figure 5.

$$\begin{aligned} \bar{h} \circ \overline{\mathbf{T}}(f \circ g) &= \text{(by definition of } \overline{\mathbf{T}}(f \circ g)) \\ \bar{h} \circ \overline{\mathbf{T}}(f \circ g) &= \text{(because } \mathbf{T} \text{ is a functor)} \\ \bar{h} \circ \overline{\mathbf{T}}f \circ \overline{\mathbf{T}}g &= \text{(by definition of } \overline{\mathbf{T}}g) \\ \bar{h} \circ \overline{\mathbf{T}}f \circ \overline{\mathbf{T}}g &= \text{(by definition of } \overline{\mathbf{T}}f) \\ \bar{h} \circ \overline{\mathbf{T}}f \circ \overline{\mathbf{T}}g & \end{aligned}$$

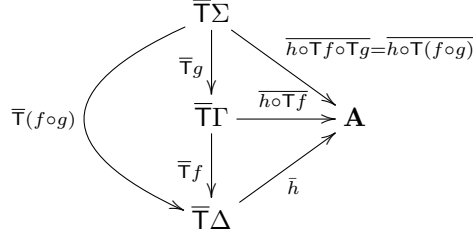


Figure 5: \bar{T} is a functor.

Multiplication in \bar{T} To prove that the multiplication operation of the monad \bar{T} produces types, one uses the following lemma in the special case of $\mathbf{A} = T\Sigma$.

Lemma 12.3 *If \mathbf{A} is a T -algebra, then there is a unique operation*

$$\text{mul}_{\bar{\mathbf{A}}} : \bar{T}\bar{\mathbf{A}} \rightarrow \bar{\mathbf{A}},$$

which makes the following diagram commute

$$\begin{array}{ccc} \bar{T}\bar{\mathbf{A}} & \xrightarrow{\text{mul}_{\bar{\mathbf{A}}}} & \bar{\mathbf{A}} \\ \bar{T}\bar{h} \downarrow & & \downarrow \bar{h} \\ \bar{T}\bar{\mathbf{B}} & \xrightarrow{\text{mul}_{\bar{\mathbf{A}}}} & \bar{\mathbf{B}} \end{array} \quad (24)$$

for every T -morphism $h : \mathbf{A} \rightarrow \mathbf{B}$ into a finite T -algebra \mathbf{B} .

Proof.

The diagram (24) leaves no choice in the definition of $\text{mul}_{\bar{\mathbf{A}}}$, since an element of $\bar{\mathbf{A}}$ is uniquely defined by its images under all possible \bar{h} . We check below that the multiplication operation is well-defined, i.e. it produces T -morphism types. Let then

$$h : \mathbf{A} \rightarrow \mathbf{B} \quad g : \mathbf{B} \rightarrow \mathbf{C}$$

be T -morphisms with \mathbf{B} and \mathbf{C} being finite T -algebras. We need to show that

$$\overline{g \circ h} \circ \bar{\mu}_{\bar{\mathbf{A}}} = g \circ \bar{h} \circ \bar{\mu}_{\bar{\mathbf{A}}}.$$

This is done below and illustrated in Figure 6.

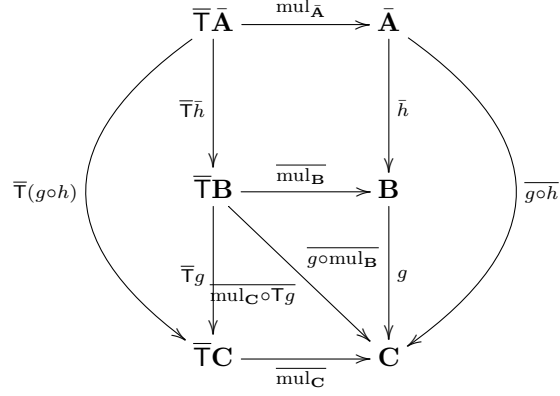


Figure 6: Multiplication in \bar{T} is well-defined.

$$\begin{aligned}
\overline{g \circ h} \circ \bar{\mu}_{\bar{\mathbf{A}}} &= \text{(by (24))} \\
\overline{\text{mul}_{\mathbf{C}}} \circ \bar{T} \overline{g \circ h} &= \text{(by (23))} \\
\overline{\text{mul}_{\mathbf{C}}} \circ \bar{T}(g \circ \bar{h}) &= \text{(because } \bar{T} \text{ is a functor)} \\
\overline{\text{mul}_{\mathbf{C}}} \circ \bar{T}g \circ \bar{T}\bar{h} &= \text{(by definition of } \bar{T}g\text{)} \\
\overline{\text{mul}_{\mathbf{C}}} \circ \bar{T}g \circ \bar{T}\bar{h} &= \text{(because } g \text{ is a } T\text{-morphism)} \\
\overline{g \circ \text{mul}_{\mathbf{B}}} \circ \bar{T}\bar{h} &= \text{(by (23))} \\
g \circ \overline{\text{mul}_{\mathbf{B}}} \circ \bar{T}\bar{h} &= \text{(by (24))} \\
g \circ \bar{h} \circ \text{mul}_{\bar{\mathbf{A}}} &
\end{aligned}$$

□

So far we have proved that the operations in the statement of Theorem 12.2 are well defined, i.e. they produce T -morphism types, and that \bar{T} is a functor. We now check the remaining axioms of a monad. We skip proving that multiplication and unit are natural, i.e. the upper two diagrams in Figure 1. We only show that multiplication is associative and consistent with the unit, i.e. the lower two diagrams in Figure 1.

To prove that the multiplication operation in the monad is associative, we apply the following lemma to the special case of $\mathbf{A} = T\Sigma$.

Lemma 12.4 *Let \mathbf{A} be a T -algebra, and let $\text{mul}_{\bar{\mathbf{A}}}$ be as in Lemma 12.3. Then the following diagram commutes:*

$$\begin{array}{ccc}
\bar{T}\bar{T}\bar{\mathbf{A}} & \xrightarrow{\bar{\mu}_{\bar{\mathbf{A}}}} & \bar{T}\bar{\mathbf{A}} \\
\bar{T}\text{mul}_{\bar{\mathbf{A}}} \downarrow & & \downarrow \text{mul}_{\bar{\mathbf{A}}} \\
\bar{T}\bar{\mathbf{A}} & \xrightarrow{\text{mul}_{\bar{\mathbf{A}}}} & \bar{\mathbf{A}}
\end{array}$$

Proof.

Because an element of $\bar{\mathbf{A}}$ is uniquely determined by its values under \bar{h} , with h ranging over \mathbf{T} -morphisms from \mathbf{A} into finite \mathbf{T} -algebras, it suffices to show that the diagram commutes when extended with such a \bar{h} , i.e.

$$\bar{h} \circ \text{mul}_{\bar{\mathbf{A}}} \circ \bar{\mathbf{T}}\text{mul}_{\bar{\mathbf{A}}} = \bar{h} \circ \text{mul}_{\bar{\mathbf{A}}} \circ \bar{\mu}_{\bar{\mathbf{A}}}.$$

Let us then fix $h : \mathbf{A} \rightarrow \mathbf{B}$ and prove the above equality. The calculation is performed below and also illustrated in Figure 7.

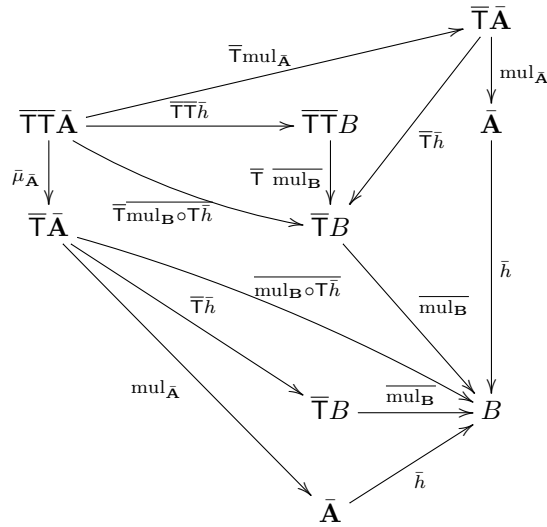


Figure 7: The four-sided faces in the diagram commute by (24), or by $\bar{\mathbf{T}}$ applied to (24). The three-sided faces in the diagram commute by the definition of $\bar{\mathbf{T}}$ on functions, or by $\bar{\mathbf{T}}$ applied to the definition of $\bar{\mathbf{T}}$ on functions.

$$\begin{aligned} \bar{h} \circ \text{mul}_{\bar{\mathbf{A}}} \circ \bar{\mathbf{T}}\text{mul}_{\bar{\mathbf{A}}} &= \text{(by (24))} \\ \overline{\text{mul}_{\mathbf{B}}} \circ \bar{\mathbf{T}}\bar{h} \circ \bar{\mathbf{T}}\text{mul}_{\bar{\mathbf{A}}} &= \text{(by } \bar{\mathbf{T}} \text{ applied to (24))} \\ \overline{\text{mul}_{\mathbf{B}}} \circ \bar{\mathbf{T}} \overline{\text{mul}_{\mathbf{B}}} \circ \bar{\mathbf{T}}\bar{h} &= \text{(because } \bar{\mathbf{T}} \text{ is a functor)} \\ \overline{\text{mul}_{\mathbf{B}}} \circ \bar{\mathbf{T}}(\overline{\text{mul}_{\mathbf{B}}} \circ \bar{\mathbf{T}}\bar{h}) &= \text{(by definition of } \bar{\mathbf{T}}\bar{h}) \\ \overline{\text{mul}_{\mathbf{B}}} \circ \overline{\bar{\mathbf{T}}\text{mul}_{\mathbf{B}} \circ \bar{\mathbf{T}}\bar{h}} &= \text{(by (24) with the } \mathbf{T}\text{-morphism being } \text{mul}_{\mathbf{B}} \circ \bar{\mathbf{T}}\bar{h}) \\ \overline{\text{mul}_{\mathbf{B}}} \circ \bar{\mathbf{T}}\bar{h} \circ \bar{\mu}_{\bar{\mathbf{A}}} &= \text{(by definition of } \bar{\mathbf{T}}\bar{h}) \\ \overline{\text{mul}_{\mathbf{B}}} \circ \bar{\mathbf{T}}\bar{h} \circ \bar{\mu}_{\bar{\mathbf{A}}} &= \text{(by (24))} \\ \bar{h} \circ \text{mul}_{\bar{\mathbf{A}}} \circ \bar{\mu}_{\bar{\mathbf{A}}} & \end{aligned}$$

□

We now check the last axiom of a monad, namely that the following diagram commutes:

$$\begin{array}{ccc}
\bar{T}\Sigma & \xrightarrow{\bar{\eta}_{\bar{T}\Sigma}} & \bar{T}\bar{T}\Sigma \\
\bar{T}\bar{\eta}_{\Sigma} \downarrow & \searrow \text{id}_{\Sigma} & \downarrow \bar{\mu}_{\Sigma} \\
\bar{T}\bar{T}\Sigma & \xrightarrow{\bar{\mu}_{\Sigma}} & \bar{T}\Sigma
\end{array}$$

Let us first check the upper triangular face of the diagram:

$$\begin{aligned}
\bar{h} \circ \bar{\mu}_{\Sigma} \circ \bar{\eta}_{\bar{T}\Sigma} &= \text{(by (24))} \\
\overline{\text{mul}_{\mathbf{A}} \circ \bar{T}\bar{h} \circ \bar{\eta}_{\bar{T}\Sigma}} &= \text{(by definition of } \bar{T}\bar{h}) \\
\overline{\text{mul}_{\mathbf{A}} \circ \bar{T}\bar{h} \circ \bar{\eta}_{\bar{T}\Sigma}} &= \text{(by definition of } \bar{\eta}_{\Sigma}) \\
\overline{\text{mul}_{\mathbf{A}} \circ \bar{T}\bar{h} \circ \eta_{\bar{T}\Sigma}} &= \text{(because } T \text{ is a monad)} \\
\overline{\text{mul}_{\mathbf{A}} \circ \eta_A \circ \bar{h}} &= \text{(because } \text{mul}_{\mathbf{A}} \text{ is the identity on units)} \\
&\bar{h}
\end{aligned}$$

Let us now check the lower triangular face of the diagram:

$$\begin{aligned}
\bar{h} \circ \bar{\mu}_{\Sigma} \circ \bar{T}\text{unit}_{\bar{T}\Sigma} &= \text{(by (24))} \\
\overline{\text{mul}_{\mathbf{A}} \circ \bar{T}\bar{h} \circ \bar{T}\text{unit}_{\bar{T}\Sigma}} &= \text{(because } \bar{T} \text{ is a functor)} \\
\overline{\text{mul}_{\mathbf{A}} \circ \bar{T}(\bar{h} \circ \text{unit}_{\bar{T}\Sigma})} &= \text{(by definition of } \bar{\eta}_{\Sigma}) \\
\overline{\text{mul}_{\mathbf{A}} \circ \bar{T}(h \circ \text{unit}_{T\Sigma})} &= \text{(because } \bar{T} \text{ is a functor)} \\
\overline{\text{mul}_{\mathbf{A}} \circ \bar{T}\bar{h} \circ \bar{T}\text{unit}_{T\Sigma}} &= \text{(by definition of } \bar{T}\bar{h}) \\
\overline{\text{mul}_{\mathbf{A}} \circ \bar{T}\bar{h} \circ \bar{T}\text{unit}_{T\Sigma}} &= \text{(by definition of } \bar{T}\eta_{\Sigma}) \\
\overline{\text{mul}_{\mathbf{A}} \circ \bar{T}h \circ \bar{T}\text{unit}_{T\Sigma}} &= \text{(because } h \text{ is a } T\text{-morphism)} \\
\bar{h} \circ \bar{\mu}_{\Sigma} \circ \bar{T}\text{unit}_{T\Sigma} &= \text{(because } T \text{ is a monad)} \\
&\bar{h}
\end{aligned}$$

This completes the proof that \bar{T} is a monad.

12.2 From a T -algebra to a \bar{T} -algebra.

Having defined the monad \bar{T} , it is natural to ask what are finite \bar{T} -algebras, and what are the languages recognised by them. In this section, we discuss how every T -algebra can be transformed into a \bar{T} -algebra. Since this transformation preserves finiteness, it gives a source of examples of finite \bar{T} -algebras. However, the algebras produced by this transformation are not very interesting, because they are essentially decorations of T -algebras. More interesting examples will be given in Section 13.

From \mathbf{A} to $\bar{\mathbf{A}}$. An element of $a \in \mathbf{A}$ can be interpreted as an element of $\bar{\mathbf{A}}$, namely as the T -morphism type which maps a T -morphism h to $h(a)$. We

denote this interpretation by $\iota_{\mathbf{A}}$, by definition it makes the following diagram commute:

$$\begin{array}{ccc} \mathbf{A} & \xrightarrow{\iota_{\mathbf{A}}} & \bar{\mathbf{A}} \\ & \searrow h & \downarrow \bar{h} \\ & & \mathbf{B} \end{array} . \quad (25)$$

for every \mathbf{T} -morphism h into a finite \mathbf{T} -algebra \mathbf{B} . It is tempting to think of $\iota_{\mathbf{A}}$ as an embedding. However, for $\iota_{\mathbf{A}}$ to be an embedding, one would require that every distinct elements of \mathbf{A} can be distinguished by some \mathbf{T} -morphism into a finite \mathbf{T} -algebra. This additional assumption is true in all monads studied in this paper, at least for finitely generated \mathbf{T} -algebras, but it can be false, e.g. with a very restrictive notion of finite \mathbf{T} -algebra.

An algebraic structure on $\bar{\mathbf{A}}$. From Lemmas 12.3 and 12.4 it follows that if \mathbf{A} is a \mathbf{T} -algebra, then there is a multiplication operation

$$\text{mul}_{\bar{\mathbf{A}}} : \bar{\mathbf{T}}\bar{\mathbf{A}} \rightarrow \bar{\mathbf{A}}$$

which turns the compactification $\bar{\mathbf{A}}$ into a $\bar{\mathbf{T}}$ -algebra. The following lemma implies that compactification preserves finiteness of algebras.

Lemma 12.5 *If \mathbf{A} is a finite \mathbf{T} -algebra, then $\bar{\mathbf{A}}$ is isomorphic to the $\bar{\mathbf{T}}$ -algebra where the universe is the universe of \mathbf{A} , and multiplication is defined to be the profinite extension of $\text{mul}_{\mathbf{A}}$, i.e. by*

$$\overline{\text{mul}_{\mathbf{A}}} : \bar{\mathbf{T}}\mathbf{A} \rightarrow \mathbf{A}.$$

Proof.

We claim that the isomorphism is the profinite extension

$$\overline{\text{id}_{\mathbf{A}}} : \bar{\mathbf{A}} \rightarrow \mathbf{A}$$

of the identity on \mathbf{A} . We claim that the above is a bijection, because its inverse is $\iota_{\mathbf{A}}$. To prove bijectivity, we need to show that

$$\iota_{\mathbf{A}} \circ \overline{\text{id}_{\mathbf{A}}} = \overline{\text{id}_{\mathbf{A}}} \circ \iota_{\mathbf{A}}$$

are the identity functions on $\bar{\mathbf{A}}$ and \mathbf{A} respectively. For the latter, we invoke (25). The former is explained in the following diagram

$$\begin{array}{ccc} \bar{\mathbf{A}} & \xrightarrow{\overline{\text{id}_{\mathbf{A}}}} & \mathbf{A} \\ \bar{h} \downarrow & \swarrow h & \downarrow \iota_{\mathbf{A}} \\ \mathbf{B} & \xleftarrow{\bar{h}} & \bar{\mathbf{A}} \end{array}$$

□

Lemma 12.6 *If $h : \mathbf{A} \rightarrow \mathbf{B}$ is a T -morphism, then there is a unique function*

$$\bar{h} : \bar{\mathbf{A}} \rightarrow \bar{\mathbf{B}}$$

which makes the following diagram commute

$$\begin{array}{ccc} \bar{\mathbf{A}} & \xrightarrow{\bar{h}} & \bar{\mathbf{B}} \\ & \searrow \scriptstyle \overline{g \circ h} & \downarrow \scriptstyle g \\ & & \mathbf{C} \end{array}$$

for every T -morphism $g : \mathbf{B} \rightarrow \mathbf{C}$ with \mathbf{C} finite.

Proof.

Note that the definition of $\bar{\mathsf{T}}f$ is actually a special case of this lemma, because $\bar{\mathsf{T}}f$ makes the diagram in the lemma commute for $\mathsf{T}f$, i.e. $\bar{\mathsf{T}}f = \bar{\mathsf{T}}\bar{f}$. The lemma is proved the same way as we proved that $\bar{\mathsf{T}}f$ is well defined. \square

The above lemma introduces a little clash of notation. If

$$h : \mathbf{A} \rightarrow \mathbf{B}$$

is a T -morphism such that \mathbf{B} is finite, then \bar{h} has two definitions: namely the profinite extension of h , which is of type $\bar{\mathbf{A}} \rightarrow \bar{\mathbf{B}}$, and the definition from the above lemma, which is of the type $\bar{\mathbf{A}} \rightarrow \bar{\mathbf{B}}$. However, the two definitions are essentially the same mapping, because they are equal up to the isomorphism from Lemma 12.5.

Lemma 12.7 *If $h : \mathbf{A} \rightarrow \mathbf{B}$ is a T -morphism, then \bar{h} defined in Lemma 12.6 is a $\bar{\mathsf{T}}$ -morphism and makes the following diagram commute:*

$$\begin{array}{ccc} \mathbf{A} & \xrightarrow{h} & \mathbf{B} \\ \downarrow \scriptstyle \iota_{\mathbf{A}} & & \downarrow \scriptstyle \iota_{\mathbf{B}} \\ \bar{\mathbf{A}} & \xrightarrow{\bar{h}} & \bar{\mathbf{B}} \end{array}$$

Proof.

We first check the diagram in the statement of the lemma. It suffices to show that the diagram commutes after the lower right corner is extended with the profinite extension of a T -morphism $f : \mathbf{B} \rightarrow \mathbf{C}$ into a finite T -algebra. This is shown in the following diagram:

$$\begin{array}{ccccc} \mathbf{A} & & \xrightarrow{h} & & \mathbf{B} \\ & & & \searrow \scriptstyle f & \downarrow \scriptstyle \iota_{\mathbf{B}} \\ & & & \mathbf{C} & \\ & \swarrow \scriptstyle \bar{f} \circ \bar{h} & & \swarrow \scriptstyle \bar{f} & \\ \bar{\mathbf{A}} & & \xrightarrow{\bar{h}} & & \bar{\mathbf{B}} \end{array}$$

The proof that \bar{h} is a \bar{T} -morphism is in the following diagram.

$$\begin{array}{ccccc}
\bar{T}\bar{\mathbf{A}} & \xrightarrow{\text{mul}_{\bar{\mathbf{A}}}} & \bar{\mathbf{A}} & & \\
\downarrow \bar{T}h & \searrow \bar{T}\bar{f \circ h} & \searrow f \circ h & & \downarrow h \\
& & \bar{T}\mathbf{C} & \xrightarrow{\text{mul}_{\bar{\mathbf{C}}}} & \mathbf{C} \\
& \nearrow \bar{T}\bar{f} & \nearrow \bar{T}f & & \nearrow f \\
\bar{T}\bar{\mathbf{B}} & \xrightarrow{\text{mul}_{\bar{\mathbf{B}}}} & \bar{\mathbf{B}} & &
\end{array}$$

The upper and lower faces commute by Lemma 12.3, the right face commutes by Lemma 12.6, and the left face commutes by applying the functor \bar{T} to Lemma 12.6. \square

12.3 From a \bar{T} -algebra to a T -algebra.

In the previous section, we showed how to convert a T -algebra into a \bar{T} -algebra. We now discuss the opposite direction. To go from a \bar{T} -algebra \mathbf{A} to a T -algebra, call it \mathbf{A}_T , one keeps the same universe and defines the multiplication operation by

$$\begin{array}{ccc}
TA & \xrightarrow{\iota_A} & \bar{T}A \\
& \searrow \text{mul}_{\mathbf{A}_T} & \downarrow \text{mul}_{\bar{\mathbf{A}}} \\
& & A
\end{array}$$

The following lemma shows that this construction is correct. In the specific case of the monad of finite words, the lemma says that a profinite semigroup is actually a semigroup (it has other structure as well).

Lemma 12.8 *If \mathbf{A} is a \bar{T} -algebra, then \mathbf{A}_T is a T -algebra. If $h : \mathbf{A} \rightarrow \mathbf{B}$ is a \bar{T} -morphism, then the function underlying h is a T -morphism from \mathbf{A}_T to \mathbf{B}_T .*

Proof.

By Lemma 12.7 applied to \bar{h} being

$$T\iota_{T\Sigma} : TT\Sigma \rightarrow T\bar{T}\Sigma$$

we see that the following diagram commutes

$$\begin{array}{ccc}
TT\Sigma & \xrightarrow{\iota_{T\Sigma}} & T\bar{T}\Sigma \\
\downarrow T\iota_\Sigma & & \downarrow \bar{T}\iota_\Sigma \\
T\bar{T}\Sigma & \xrightarrow{\iota_{\bar{T}\Sigma}} & \bar{T}\bar{T}\Sigma
\end{array}$$

Let us write μ_Σ for the diagonal of the above diagram.

To prove that \mathbf{A}_T is a T -algebra, we will show that the following diagram commutes (the outer perimeter of the diagram says that $\text{mul}_{\mathbf{A}_T}$ is associative as required in a T -algebra):

$$\begin{array}{ccccc}
 TTA & \xrightarrow{\mu_A} & TA & & \\
 \downarrow T\text{mul}_{\mathbf{A}_T} & \searrow \iota_A & \downarrow \iota_A & & \downarrow \text{mul}_{\mathbf{A}_T} \\
 & TTA & \xrightarrow{\bar{\mu}_A} & TA & \\
 & \downarrow T\text{mul}_A & & \downarrow \text{mul}_A & \\
 & TA & & A & \\
 & \swarrow \iota_A & \nearrow \text{mul}_A & & \\
 TA & \xrightarrow{\text{mul}_{\mathbf{A}_T}} & A & &
 \end{array}$$

The middle face of the diagram is the assumption that \mathbf{A} is a \bar{T} -algebra. The right and bottom faces are the definition of $\text{mul}_{\mathbf{A}_T}$. The top face can be shown using the definition of multiplication in $\bar{T}A$, and does not use the algebraic structure on A . Finally, for the left face, we use the following diagram:

$$\begin{array}{ccccc}
 TTA & & & & \\
 \downarrow T\text{mul}_{\mathbf{A}_T} & \searrow \iota_A & \searrow \mu_\Sigma & & \\
 & TTA & \xrightarrow{\iota_{TA}} & TTA & \\
 & \downarrow T\text{mul}_A & & \downarrow \bar{T}\text{mul}_A & \\
 & TA & \xrightarrow{\iota_A} & TA &
 \end{array}$$

The rectangular face commutes by Lemma 12.7. The lower triangular face commutes by applying the functor T to the definition of $\text{mul}_{\mathbf{A}_T}$. The upper triangular face commutes because it is the definition of ι_A .

This completes the proof that \mathbf{A}_T is a T -algebra. To prove that h as in the statement of the lemma is a T -morphism, we consider the following diagram:

$$\begin{array}{ccccc}
 TA & \xrightarrow{Th} & TB & & \\
 \downarrow \text{mul}_{\mathbf{A}_T} & \searrow \iota_A & \searrow \iota_B & & \downarrow \text{mul}_{\mathbf{B}_T} \\
 & TA & \xrightarrow{\bar{T}h} & TB & \\
 & \swarrow \text{mul}_A & & \swarrow \text{mul}_B & \\
 & A & \xrightarrow{h} & B &
 \end{array}$$

The left and right faces commute by definitions of multiplication in \mathbf{A}_T and \mathbf{B}_T . The bottom face commutes by assumption that h is a \bar{T} -morphism. The top face commutes by Lemma 12.7. \square

12.4 Clopen languages and Stone algebras

A well known result for profinite words is that there is a one-to-one correspondence between clopen subsets of the profinite monoid over Σ , and recognisable subsets of Σ^+ . In this section we prove Theorem 12.9, which generalises this observation to monads.

Recall that if \mathbf{A} is a T -algebra, then by Fact 12.1 there is a topological structure on $\bar{\mathbf{A}}$, which is homeomorphic to the Stone dual \mathbf{StoneA} . Recall also the mapping $\iota_{\mathbf{A}} : \mathbf{A} \rightarrow \bar{\mathbf{A}}$. Using these two notions, for $L \subseteq \mathbf{A}$, we define $\bar{L} \subseteq \bar{\mathbf{A}}$ to be the closure, in the topology of $\bar{\mathbf{A}}$, of the image of L under the $\iota_{\mathbf{A}}$. In Theorem 12.9, we will show that the clopen subsets of $\bar{\mathbf{A}}$ are exactly the closures, in the sense just defined, of T -recognisable subsets of \mathbf{A} .

Theorem 12.9 *Let \mathbf{A} be a T -algebra. A subset of $\bar{\mathbf{A}}$ is clopen if and only if it is equal to \bar{L} for some recognisable $L \subseteq \mathbf{A}$.*

Before proving the theorem, we present a lemma.

Lemma 12.10 *If $h : \mathbf{A} \rightarrow \mathbf{B}$ is a T -morphism into a finite T -algebra, then*

$$\overline{h^{-1}(F)} = (\bar{h})^{-1}(F) \quad \text{for every } F \subseteq \mathbf{B}.$$

Proof.

Recall that a base open set in $\bar{\mathbf{A}}$ is a set of the form

$$\bar{h}^{-1}(b) \quad \text{for some } \mathsf{T}\text{-morphism } h : \mathbf{A} \rightarrow \mathbf{B} \text{ and } b \in B,$$

and such sets are also closed. Therefore, the set on the right side of the equality in the statement of the lemma is closed, as a finite union of base sets. To complete the proof, we show that the image of $h^{-1}(F)$ under $\iota_{\mathbf{A}}$ is dense in the set on the right side, i.e. every open subset of the right side contains $\iota_{\mathbf{A}}(a)$ for some $a \in \mathbf{A}$ with $h(a) \in F$. Every open set contains a base open set, and therefore it suffices to show that if

$$g : \mathbf{A} \rightarrow \mathbf{C}$$

is a T -morphism into a finite T -algebra, and the base open set $\bar{g}^{-1}(c)$ is included in the right side of the equality, then $\bar{g}^{-1}(c)$ contains $\iota_{\mathbf{A}}(a)$ for some $a \in \mathbf{A}$ with $h(a) \in F$. For a it suffices to choose any element of $g^{-1}(c)$, which is easily shown to belong to $h^{-1}(F)$. \square

Proof. (of Theorem 12.9)

Let us begin with the left-to-right implication. Consider a clopen subset of $\bar{\mathbf{A}}$. Like any clopen set in a compact space, this is a finite union of base open sets. By definition of closed base open sets in $\bar{\mathbf{A}}$ and Lemma 12.10, we see that every clopen subset of $\bar{\mathbf{A}}$ can be represented as a finite union

$$\bigcup_i \overline{h_i^{-1}(F_i)} = \overline{\bigcup_i h_i^{-1}(F_i)}.$$

The right side is as required in the statement of the theorem. The right-to-left implication is done by reversing the above reasoning. \square

Stones In Section 12.2, we showed how to convert T -algebras into $\overline{\mathsf{T}}$ -algebras, and how to convert T -morphisms into $\overline{\mathsf{T}}$ -morphisms. We now explain that the T -algebras and T -morphisms produced this way have special topological properties.

A $\overline{\mathsf{T}}$ -algebra \mathbf{A} is called *Stone* if its universe is finite, and the multiplication operation is continuous assuming the discrete topology on the universe. The reason for this name is that the discrete topology is the only one which makes the finite universe a Stone space, i.e. a compact totally disconnected Hausdorff topological space⁷. As shown in the following lemma, Stone $\overline{\mathsf{T}}$ -algebras are essentially the same thing as finite T -algebras. In Section 13 we will show examples of finite $\overline{\mathsf{T}}$ -algebras that are not Stone.

Theorem 12.11 *Up to isomorphism, the mappings*

$$\begin{array}{ccc} \mathbf{A} & \mapsto & \overline{\mathbf{A}} \\ h : \mathbf{A} \rightarrow \mathbf{B} & \mapsto & \bar{h} : \overline{\mathbf{A}} \rightarrow \overline{\mathbf{B}} \end{array}$$

are one-to-one correspondences between, respectively:

- *finite T -algebras and finite $\overline{\mathsf{T}}$ -algebras that are Stone; and*
- *T -morphisms into finite T -algebras and continuous $\overline{\mathsf{T}}$ -morphisms into finite $\overline{\mathsf{T}}$ -algebras that are Stone.*

Proof.

We only consider the first mapping, the second is proved in a similar way. We will show that if \mathbf{A} is a finite T -algebra then $\overline{\mathbf{A}}$ is Stone, and if \mathbf{A} is a Stone, then \mathbf{A}_{T} is a finite T -algebra. In Lemma 12.5, we have shown that if \mathbf{A} is a finite T -algebra, then $\overline{\mathbf{A}}$ is isomorphic to the algebra whose universe is the universe of \mathbf{A} , and whose multiplication operation is

$$\overline{\text{mul}_{\mathbf{A}}} : \overline{\mathsf{T}}A \rightarrow A,$$

i.e. the profinite extension of the original multiplication. Every profinite extension is continuous (assuming the discrete topology on the image) by definition of the topology in $\overline{\mathbf{A}}$, and therefore $\overline{\mathbf{A}}$ is Stone. To prove that the correspondence is one-to-one up to isomorphism, we need to show that if \mathbf{A} is a Stone $\overline{\mathsf{T}}$ -algebra then it is isomorphic to $\overline{\mathbf{A}_{\mathsf{T}}}$, and if \mathbf{A} is a finite T -algebra then it is isomorphic to $(\overline{\mathbf{A}})_{\mathsf{T}}$. We only prove the former isomorphism. Let then \mathbf{A} be a Stone algebra. By Lemma 12.5, it suffices to show that the two multiplication operations

$$\begin{array}{ccc} \text{mul}_{\mathbf{A}} & : & \overline{\mathsf{T}}A \rightarrow A \\ \overline{\text{mul}_{\mathbf{A}_{\mathsf{T}}}} & : & \overline{\mathsf{T}}A \rightarrow A \end{array}$$

⁷Actually, already the Hausdorff requirement implies discreteness, but Stone spaces are closely connected to profiniteness.

are equal. By definition, the two operations agree on elements in the image of $\mathsf{T}A$ under

$$\iota_{\mathsf{T}A} : \mathsf{T}A \rightarrow \overline{\mathsf{T}A}.$$

This image is a dense subset of $\overline{\mathsf{T}A}$. Because A is finite, $\overline{\mathsf{T}A}$ is a metric space (we implicitly assume that there are countably many finite T -algebras up to isomorphism), and therefore both multiplication operations are uniformly continuous functions that agree on a dense subset. Such functions must be equal. \square

13 Profinite words

In this section, we illustrate the profinite monad construction from Section 12 in the special case of words. Consider the monad $\Sigma \mapsto \Sigma^+$ of finite words. Let us denote by $\Sigma \mapsto \Sigma^{\bar{+}}$ the profinite version of this monad, as defined in Section 12. In particular, $\Sigma^{\bar{+}}$ is a semigroup thanks to Lemma 12.8, and $\Sigma^{\bar{+}}$ is has a topology which makes it a Stone space by Fact 12.1. An element of $\Sigma^{\bar{+}}$ is called a *profinite word* over the alphabet Σ . One of the results of this section is Theorem 13.3, which implies that MSO is undecidable over profinite words, already with two predicates.

As shown in Section 12.2, every finite semigroup \mathbf{S} , can be extended to a $\bar{+}$ -algebra $\bar{\mathbf{S}}$ with the same universe, where the multiplication operation

$$\text{mul}_{\mathbf{S}} : S^{\bar{+}} \rightarrow S$$

is continuous assuming the profinite topology on the domain and the discrete topology on the image. In this section we give an example of a finite $\bar{+}$ -algebra that is not obtained this way, because the multiplication operation is not going to be continuous assuming the discrete topology on the image.

13.1 The unboundedness language

We say that a profinite word $w \in \Sigma^{\bar{+}}$ has *at least n letters* in a subset $\Gamma \subseteq \Sigma$ if it has value n under \bar{h} where

$$h : \Sigma^+ \rightarrow \{0, 1, \dots, n\}$$

is the semigroup morphism which counts the number of letters in Γ up to threshold n . A profinite word is said to have exactly n letters from a set if it has at least n letters from the set but not at least $n + 1$. If a profinite word has at least n letters in Γ for every n , then we say that it has *an unbounded number* of letters in Γ .

Lemma 13.1 *The set of profinite words in $\{0, 1\}^{\bar{+}}$ which have unboundedly many ones is $\bar{+}$ -recognisable.*

Proof.

We show that the set in the statement of the lemma is recognised by a $\bar{+}$ -morphism

$$h : \{0, 1\}^{\bar{+}} \rightarrow \mathbf{A}$$

where \mathbf{A} is the finite $\bar{+}$ -algebra defined as follows. The universe of \mathbf{A} has three elements, call them 0, 1 and ∞ , which represent profinite words that have zero ones, a bounded number of ones, and an unboundedly number of ones respectively. The multiplication operation

$$\text{mul}_{\mathbf{A}} : A^{\bar{+}} \rightarrow A$$

is defined as follows. If the argument has only zeros, the value is zero. If the argument has at least one letter ∞ , or unboundedly many ones, then the value is ∞ . Otherwise the value is one. Note that the multiplication operation is *not* continuous, at least assuming a discrete topology on the universe, because the inverse image of 1 is not closed. We now prove that this multiplication is associative, i.e. that the following diagram commutes:

$$\begin{array}{ccc} (A^{\bar{+}})^{\bar{+}} & \xrightarrow{\bar{\mu}_A} & A^{\bar{+}} \\ (\text{mul}_{\mathbf{A}})^{\bar{+}} \downarrow & & \downarrow \text{mul}_{\mathbf{A}} \\ A^{\bar{+}} & \xrightarrow{\text{mul}_{\mathbf{A}}} & A \end{array}$$

where $\bar{\mu}_A$ denotes the multiplication operation of the profinite monad.

To prove that the above diagram commutes, we need to show that

$$\text{mul}_{\mathbf{A}}((\text{mul}_{\mathbf{A}})^{\bar{+}}(w)) = \text{mul}_{\mathbf{A}}(\bar{\mu}_A(w)). \quad (26)$$

holds for every profinite word of profinite words $w \in (A^{\bar{+}})^{\bar{+}}$. We consider two cases, depending on whether w has an unbounded number of letters in the set $A^{\bar{+}} - 0^{\bar{+}}$.

- The word w has an unbounded number of letters outside $0^{\bar{+}}$. We will show that (26) holds, because both sides are equal to ∞ . Consider first the left side. Let

$$h_n : A^+ \rightarrow \{0, \dots, n\}$$

be the semigroup morphism that counts the number of nonzero letters. Our assumption on w says that $(\bar{h}_1)^{\bar{+}}(w)$ has unboundedly many ones. Since the image of h_1 is a subset of A , it makes sense to compare values of \bar{h}_1 with values of $\text{mul}_{\mathbf{A}}$, in particular the following observation is easy to get:

$$\bar{h}_1(v) \leq \text{mul}_{\mathbf{A}}(v) \quad \text{for every } v \in A^{\bar{+}}.$$

As in the proof of Lemma 6.2, a binary relation $R \subseteq X \times Y$ lifts to a relation $R^\top \subseteq \mathbb{T}X \times \mathbb{T}Y$. Apply this construction to the natural ordering on A , and call \leq the resulting relation on A^\top . As we have observed,

$$(\bar{h}_1)^\top(w) \leq^\top (\text{mul}_\mathbf{A})^\top(w).$$

The profinite word on the left of the above inequality has an unbounded number of ones by our assumption, and therefore it is mapped by $\text{mul}_\mathbf{A}$ to ∞ . It is not difficult to see that the mapping $\text{mul}_\mathbf{A}$ is monotone with respect to \leq , and therefore the left side of the equality in (26) is ∞ .

To prove that the right side of the equality in (26) is also ∞ , by definition of $\text{mul}_\mathbf{A}$ we need to show that every n satisfies

$$\overline{h_n}(\bar{\mu}_A(w)) = n.$$

Let mul_n be the multiplication operation in the semigroup $\{0, \dots, n\}$. Theorem 12.2 says that

$$\begin{array}{ccc} A^{\top\top} & \xrightarrow{\bar{\mu}_A} & \overline{\mathbb{T}A} \\ \overline{h_n}^\top \downarrow & & \downarrow \overline{h_n} \\ \overline{\mathbb{T}\{0, \dots, n\}} & \xrightarrow{\overline{\text{mul}_n}} & \{0, \dots, n\} \end{array}$$

To prove that the right side of the equality in (26) is also ∞ , from the definition of $\text{mul}_\mathbf{A}$ we need to show that for every n , if start with w and consider the right-down path in the above diagram, then we get n . Because the diagram commutes, we can also consider the down-right path. Our assumption on w says that $(\bar{h}_n)^\top(w)$ is a profinite word which has an unbounded number of nonzero letters. On such words, $\overline{h_n}$ gives result n .

- The other case is when w has a bounded number of letters outside 0^\top . We begin with a straightforward lemma, which uses the semigroup structure of profinite words that was described in Lemma 12.8. Let us denote the unit of the profinite monad by $\bar{\eta}_\Sigma$, i.e. if $a \in \Sigma$ then $\bar{\eta}_\Sigma(a) \in \Sigma^\top$ is the corresponding profinite word.

Lemma 13.2 *If $w \in \Sigma^\top$ has a bounded number of letters in $\Gamma \subseteq \Sigma$ then it admits a finite decomposition*

$$w = w_0 \cdot \bar{\eta}_\Sigma(a_1) \cdot w_1 \cdots w_{n-1} \cdot \bar{\eta}_\Sigma(a_n) \cdot w_n$$

where w_0, \dots, w_n are profinite words over the alphabet $\Sigma - \Gamma$, a_1, \dots, a_n are letters in Γ , and the dot stands for concatenation in the profinite semigroup.

By applying Lemma 13.2, there is a decomposition

$$w = w_0 \cdot \bar{\eta}_{A^\top}(a_1) \cdot w_1 \cdots w_{n-1} \cdot \bar{\eta}_{A^\top}(a_n) \cdot w_n$$

where $w_i \in (0^\mp)^\mp$, $a_i \in A^\mp - 0^\mp$, and the dot is concatenation in the profinite semigroup over alphabet A^\mp . Lemma 12.8 implies that

$$\bar{\mu}_A(w) = \bar{\mu}_A(w_0) \cdot a_1 \cdot \bar{\mu}_A(w_1) \cdots \bar{\mu}_A(w_{n-1}) \cdot a_n \cdot \bar{\mu}_A(w_n)$$

where the dot is concatenation in the profinite semigroup over alphabet A . Since $\text{mul}_\mathbf{A}$ is a semigroup morphism, and it maps words in 0^\mp to the identity in A , it follows that

$$\text{mul}_\mathbf{A}(\bar{\mu}_A(w)) = \text{mul}_\mathbf{A}(a_1) \cdots \text{mul}_\mathbf{A}(a_n).$$

Let us now consider $\text{mul}_\mathbf{A}((\text{mul}_\mathbf{A})^\mp(w))$. Lemma 12.8 says that $(\text{mul}_\mathbf{A})^\mp$ is a semigroup morphism, and therefore $\text{mul}_\mathbf{A}^\mp(w)$ is equal to

$$\text{mul}_\mathbf{A}^\mp(w_0) \cdot \text{mul}_\mathbf{A}^\mp(\bar{\eta}_{A^\mp}(a_1)) \cdot \text{mul}_\mathbf{A}^\mp(w_1) \cdots \text{mul}_\mathbf{A}^\mp(w_{n-1}) \cdot \text{mul}_\mathbf{A}^\mp(\bar{\eta}_{A^\mp}(a_n)) \cdot \text{mul}_\mathbf{A}^\mp(w_n).$$

By the axioms of a monad, we have

$$\text{mul}_\mathbf{A}^\mp(\bar{\eta}_{A^\mp}(a_i)) = \bar{\eta}_A(\text{mul}_\mathbf{A}(a_i)).$$

Each word $\text{mul}_\mathbf{A}^\mp(w_i)$ in the decomposition of $\text{mul}_\mathbf{A}^\mp(w)$ belongs to 0^\mp . Therefore, because $\text{mul}_\mathbf{A}$ is a semigroup morphism that maps 0^\mp to the identity, we get

$$\text{mul}_\mathbf{A}(\text{mul}_\mathbf{A}^\mp(w)) = \text{mul}_\mathbf{A}(\bar{\eta}_A(\text{mul}_\mathbf{A}(a_1))) \cdots \text{mul}_\mathbf{A}(\bar{\eta}_A(\text{mul}_\mathbf{A}(a_n))).$$

The result follows because $\text{mul}_\mathbf{A}(\bar{\eta}_A(a)) = a$ holds for every $a \in A$.

This completes the proof that $\text{mul}_\mathbf{A} : A^\mp \rightarrow A$ is a \mp -morphism. \square

Let us define $\text{MSO}+\text{inf}$ to be by applying the abstract notion of MSO defined in Section 6.1, with the base predicates being the language of unboundedly many ones from the previous lemma, and the profinite closure of the language “some a comes before some b ”. This class of languages of profinite words was considered in [Tor11] and [Tor12], adjusting for the monad terminology. From Lemma 6.2 it follows that $\text{MSO}+\text{inf}$ contains only \mp -recognisable languages. It is not clear if it contains all \mp -recognisable languages.

Theorem 13.3 *The satisfiability problem for $\text{MSO}+\text{inf}$ is undecidable.*

Proof.

Consider $\text{MSO}+\text{U}$ on infinite words, which is an extension of MSO . This logic is shown undecidable in [MB15]. Corollary 2 of [Tor12] shows that decidability of $\text{MSO}+\text{U}$ on infinite words reduces to decidability of $\text{MSO}+\text{inf}$ on profinite words.

\square

References

- [ÁDH83] I Ágoston, J Demetrovics, and L Hannák. The number of clones containing all constants (a problem of R. McKenzie). In *Colloquia mathematica ocietatis Janos Bolyai*, volume 43, pages 21–25, 1983.
- [Arn85] André Arnold. A syntactic congruence for rational omega-language. *Theor. Comput. Sci.*, 39:333–335, 1985.
- [BI09] Mikolaj Bojanczyk and Tomasz Idziaszek. Algebra for infinite forests with an application to the temporal logic EF. In *CONCUR*, pages 131–145, 2009.
- [Boj13] Mikolaj Bojanczyk. Nominal monoids. *Theory Comput. Syst.*, 53(2):194–222, 2013.
- [Boj14] Mikolaj Bojanczyk. Transducers with origin information. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2014.
- [BR12] Nicolas Bedon and Chloé Rispal. Schützenberger and Eilenberg theorems for words on linear orderings. *J. Comput. Syst. Sci.*, 78(2):517–536, 2012.
- [Büc62] Julius Richard Büchi. On a decision method in restricted second-order arithmetic. In *Proc. 1960 Int. Congr. for Logic, Methodology and Philosophy of Science*, pages 1–11, 1962.
- [BW08] Mikolaj Bojanczyk and Igor Walukiewicz. Forest algebras. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, volume 2 of *Texts in Logic and Games*, pages 107–132. Amsterdam University Press, 2008.
- [CCP11] Olivier Carton, Thomas Colcombet, and Gabriele Puppis. Regular languages of words over countable linear orderings. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, volume 6756 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 2011.
- [Eil74] S. Eilenberg. *Automata, languages, and machines. Vol. A*. 1974.
- [ÉW03] Zoltán Ésik and Pascal Weil. On logically defined recognizable tree languages. In *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science*, pages 195–207. Springer, 2003.

- [GGP08] Mai Gehrke, Serge Grigorieff, and Jean-Éric Pin. Duality and equational theory of regular languages. In *Automata, languages and programming*, pages 246–257. Springer, 2008.
- [GGP10] Mai Gehrke, Serge Grigorieff, and Jean-Éric Pin. A topological approach to recognition. In *Automata, Languages and Programming*, pages 151–162. Springer, 2010.
- [HM88] David Charles Hobby and Ralph McKenzie. *The structure of finite algebras*, volume 76. American Mathematical Society Providence, 1988.
- [MB15] Szymon Toruńczyk Mikołaj Bojańczyk, Paweł Parys. The $\text{MSO}+\text{U}$ theory of $(\mathbb{N}, <)$ is undecidable. *CoRR*, arXiv:1502.04578, 2015.
- [Pot95] A. Potthoff. First-order logic on finite trees. *Lecture Notes in Computer Science*, 915:125–139, 1995.
- [PP04] Dominique Perrin and Jean-Éric Pin. *Infinite Words: Automata, Semigroups, Logic and Games*. Elsevier, 2004.
- [Rei82] Jan Reiterman. The Birkhoff theorem for finite algebras. *Algebra Universalis*, 14(1):1–10, 1982.
- [Ros70] Ivo Rosenberg. *Über die funktionale Vollständigkeit in den mehrwertigen Logiken: Struktur der Funktionen von mehreren Veränderlichen auf endlichen Mengen*. Academia, 1970.
- [Ros86] Ivo Rosenberg. Minimal clones i: the five types. In *Lectures in Universal Algebra (Proc. Conf. Szeged 1983)*, volume 43, pages 405–427. North-Holland Amsterdam, 1986.
- [She75] Saharon Shelah. The monadic theory of order. *The Annals of Mathematics*, 102(3):379–419, 1975.
- [Ste92] Magnus Steinby. A theory of tree language varieties. In *Tree Automata and Languages*, pages 57–82. 1992.
- [Tho84] Wolfgang Thomas. Logical aspects in the study of tree languages. In *CAAP*, pages 31–50, 1984.
- [Tho96] Wolfgang Thomas. Languages, automata and logics. Technical Report 9607, Institut für Informatik und Praktische Mathematik, Christian-Albsechts-Universität, Kiel, Germany, 1996.
- [Tor11] Szymon Toruńczyk. *Languages of profinite words and the limitedness problem*. PhD thesis, University of Warsaw, 2011.
- [Tor12] Szymon Toruńczyk. Languages of profinite words and the limitedness problem. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, pages 377–389, 2012.

- [TW98] Denis Thérien and Thomas Wilke. Over words, two variables are as powerful as one quantifier alternation. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 234–240. ACM, 1998.
- [Wil91] Thomas Wilke. An Eilenberg theorem for infinity-languages. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez-Artalejo, editors, *Automata, Languages and Programming, 18th International Colloquium, ICALP91, Madrid, Spain, July 8-12, 1991, Proceedings*, volume 510 of *Lecture Notes in Computer Science*, pages 588–599. Springer, 1991.
- [Wil93] Thomas Wilke. An algebraic theory for regular languages of finite and infinite words. *Int. J. Alg. Comput.*, 3:447–489, 1993.
- [YM59] Y. I. Yanov and A. A. Muchnik. Existence of k-valued closed classes without a finite basis. *Dokl. Akad. Nauk.*, 127:44–46, 1959.