Recognisable Languages over Monads

Mikołaj Bojańczyk*

December 12, 2014

Contents

*Supported by ERC Starting Grant "Sosna"

Part I
Introduction

In this part, we introduce monads and their algebras. As a gentle introduction, we use finite words and ω-words as examples of monads in Section 1. In Section 2, we present the formal definition of monads and their algebras. In the following sections, we show how some results about languages can be stated and proved on the level of monads. This includes a monad version of the Myhill-Nerode theorem (Section 3), a monad version of Eilenberg's pseudovariety theorem (Section 4), and some parts of the connection between regular languages and MSO (Sections 5 and 6).

1 Examples of monads for words

Before introducing monads, we describe how monoids can be viewed as a special case of algebras in a monad. Define a $*$-algebra $\mathbf{A}$ to be a set A called its *universe*[1], together with a multiplication operation $\mathrm{mul}_{\mathbf{A}} : A^* \to A$, which maps the single letter word to its unique letter, and which is associative in the sense that the following diagram commutes

$$\begin{array}{ccc} (A^*)^* & \xrightarrow{\mathrm{mul}_{A^*}} & A^* \\ {\scriptstyle (\mathrm{mul}_{\mathbf{A}})^*}\downarrow & & \downarrow{\scriptstyle \mathrm{mul}_{\mathbf{A}}} \\ A^* & \xrightarrow[\mathrm{mul}_{\mathbf{A}}]{} & A \end{array},$$

wher $(\mathrm{mul}_{\mathbf{A}})^*$ is the function that applies $\mathrm{mul}_{\mathbf{A}}$ to each label of a word, and mul_{A^*} is the function which flattens a word of words into a word, e.g.

$$\mathrm{mul}_{A^*}((abc)(\epsilon)(acaa)) = abcacaa.$$

A monoid can be interpreted as a $*$-algebra, by taking $\mathrm{mul}_{\mathbf{A}}$ to be the associative product operation in the monoid, and every $*$-algebra is obtained this way. Therefore $*$-algebras are the same thing as monoids.

A morphism between two $*$-algebras $\mathbf{A}$ and $\mathbf{B}$, which is the same thing as a monoid morphism, is a function $h : A \to B$ between their universes such that the following diagram commutes

$$\begin{array}{ccc} A^* & \xrightarrow{h^*} & B^* \\ {\scriptstyle \mathrm{mul}_{\mathbf{A}}}\downarrow & & \downarrow{\scriptstyle \mathrm{mul}_{\mathbf{B}}} \\ A & \xrightarrow[h]{} & B \end{array},$$

The set of all words A^* forms a $*$-algebra, with mul_{A^*} as the operation, and this $*$-algebra is the free one generated by A. A $*$-language is a subset of Σ^*, for some finite alphabet Σ. A $*$-language $L \subseteq \Sigma^*$ is called *recognisable* if there is a finite $*$-algebra $\mathbf{A}$ and a $*$-morphism $h : \Sigma^* \to \mathbf{A}$ such that membership $w \in L$ is uniquely determined by $\alpha(w)$. These are the standard notions of languages recognised by finite monoids, which coincide with languages recognised by automata or defined by regular expressions.

The reader will observe that the above definitions of "algebra", "morphism", "language", "recognisable language" were defined only in terms of the following four notions:

1. how a set A is transformed into a set A^*;
2. how a function $f : A \to B$ is lifted to a function $f^* : A^* \to B^*$;
3. the flattening operation from $(A^*)^* \to A^*$;

[1]From now on, we adopt the convention that algebras are written in boldface as in $\mathbf{A}, \mathbf{B}, \mathbf{C}$, while their universes are written without boldface as in A, B, C.

4. how to represent an element of A as an element of A^*.

These four notions, subject to certain axioms, are what constitutes a monad. As we shall see, based on these notions one can also define notions such as recognisable language, syntactic algebra, pseudovariety, MSO logic, profinite object, and even prove some theorems about them.

Before presenting a formal definition of monads, we introduce more examples. One example is the monad of nonempty words A^+, which corresponds to semigroups in the way that A^* corresponds to monoids. The following section contains a more interesting example, namely a monad for infinite words.

1.1 Possibly infinite words

For a set A, define A^∞ to be the set of finite or ω-words over A, i.e.

$$A^\infty \quad \stackrel{\text{def}}{=} \quad A^* \cup A^\omega.$$

There is a natural multiplication operation

$$\text{mul}_{A^\infty} : (A^\infty)^\infty \to A^\infty$$

which substitutes each position for the word that it contains. In particular, if the argument of the composition contains an infinite word on some position, then all subsequent positions are ignored.

Define an ∞-algebra $\mathbf{A}$ to be a set A, called its universe, together with a multiplication operation $\text{mul}_{\mathbf{A}} : A^\infty \to A$ such that a single letter word a is mapped to the letter a, and the following diagram commutes

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

where $(\text{mul}_{\mathbf{A}})^\infty$ is the function that applies $\text{mul}_{\mathbf{A}}$ to the label of every position in a word from $(A^\infty)^\infty$. An ∞-algebra is essentially the same thing as an ω-semigroup, see [11], with the difference that ω-semigroups have separate sorts for finite and infinite words.

A finite ∞-algebra is one with a finite universe. In a truly finite algebra, one would also want the multiplication operation to be somehow finitely represented. It turns out that such a finite representation necessarily exists, as stated in the following lemma, which was implicit in the original Büchi paper [6] on infinite words, and explicit in Wilke's work [19] on ω-semigroups.

Theorem 1.1 *Consider an ∞-algebra $\mathbf{A}$. Then $\text{mul}_{\mathbf{A}}$ is uniquely determined by its values on elements of the form ab and a^ω.*

Proof.
Using the Ramsey theorem, one shows that for every $w \in A^\infty$ one can find a

decomposition $v \in (A^*)^\infty$ such that all letters in the decomposition v, which are elements of A^*, have the same value under the multiplication operation, with the possible exception of the first letter.$\square$

2 Monads and their algebras

For people who are afraid of categories (like the author), we underline that this paper uses only the most rudimentary notions of category theory: namely the definition of a category (objects and composable morphisms between them), and of a functor (something that maps objects to object 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 maybe the category of multi-sorted sets, where objects are multi-sorted sets for some fixed set of sort names, and morphisms are sort-preserving functions.

2.1 Definition of monads

A monad over a category consists of a functor T from the category to itself, and for every object X in the category, two morphisms

$$\mathrm{unit}_X : X \to \mathsf{T}X \qquad \text{and} \qquad \mathrm{mul}_{\mathsf{T}X} : \mathsf{T}\mathsf{T}X \to \mathsf{T}X,$$

which are called the unit and multiplication operations. The monad must satisfy the axioms given in Figure 1.

One example of a monad that we have already seen is the functor which maps a set X to the set X^* of finite words over X, and which maps a function $f : X \to Y$ to the letter-to-letter lifting $f^* : X^* \to Y^*$. Other examples that we have seen are the monad for finite nonempty words X^+, and the monad for possibly infinite words X^∞. In the second part of this paper, we give a large number of other monads, which model things like labelled total orders, trees, or words with distinguished positions.

2.2 Eilenberg-Moore Algebras

For this paper, the most important thing about monads is that they have a natural corresponding algebras. 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

$$\mathrm{mul}_{\mathbf{A}} : \mathsf{T}A \to A,$$

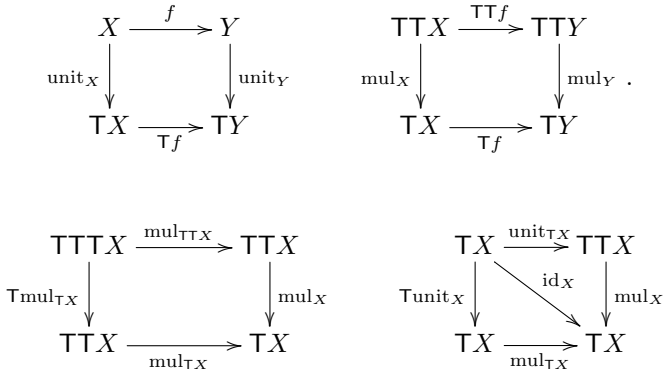


Figure 1: The axioms of a monad are that the above four diagrams commute for every object X in the category and every morphism $f : X \to Y$. The upper diagrams say that the unit and multiplication are natural. The lower left diagram says that multiplication is associative, and the lower right says the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

such that the $\mathrm{mul}_{\mathbf{A}} \circ \mathrm{unit}_A$ is the identity, and which is associative in the sense that the following diagram commutes:

$$\begin{array}{ccc} \mathsf{T}\mathsf{T}A & \xrightarrow{\mathrm{mul}_{\mathsf{T}A}} & \mathsf{T}A \\ {\scriptstyle \mathsf{T}\mathrm{mul}_{\mathbf{A}}}\downarrow & & \downarrow{\scriptstyle \mathrm{mul}_{\mathbf{A}}} \\ \mathsf{T}A & \xrightarrow[\mathrm{mul}_{\mathbf{A}}]{} & A \end{array} .$$

The axioms of a monad say that $\mathsf{T}A$, equipped with the operation $\mathrm{mul}_{\mathsf{T}A}$, forms a T-algebra. A T-*morphism* between two T-algebras $\mathbf{A}$ and $\mathbf{B}$ is a function h between their universes which respect their multiplication operations in the sense that the following diagram commutes:

$$\begin{array}{ccc} \mathsf{T}A & \xrightarrow{\mathsf{T}h} & B \\ {\scriptstyle \mathrm{mul}_{\mathbf{A}}}\downarrow & & \downarrow{\scriptstyle \mathrm{mul}_{\mathbf{B}}} \\ A & \xrightarrow[h]{} & B \end{array} .$$

2.3 Languages and colourings

In this section we develop the basic definitions of recognisable languages over a monad. These notions require the following parameters, apart from the monad itself, which we call the *setting*: the underlying category, 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, the universe of a finite algebra will not be finite in the same sense as a finite algebra.

Many examples are in the setting of sets, where the category is the category of sets (objects are sets and morphisms are functions), finite alphabets are finite sets, and finite algebras are algebras with finite universes. Fix a setting and a monad T for the following definitions.

Recognisable colourings and languages. A *coloring* of a T-algebra $\mathbf{A}$ is a morphism from its universe to some object in the underlying category. A coloring is said to be recognised by a T-morphism $h : \mathbf{A} \to \mathbf{B}$ if it factors through h. A coloring is called T-*recognisable* if it is recognised by some T-morphism with a finite target, according to the notion of finite T-algebra given in the setting.

Consider an alphabet Σ, according to the notion of alphabet given in the setting. In all of the examples where sets or sorted sets are used, an alphabet will be a possibly sorted set with finitely many elements. A T-language over an alphabet Σ is a subset $L \subseteq \mathsf{T}\Sigma$, which assumes that the underlying category has a notion of subset. Notions of recognisability are inherited from colourings, by using the characteristic function of a language.

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 languages, 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 only works in the setting of sorted sets, and therefore also in the setting of normal sets, and I have trouble generalising it beyond that. Fix the setting 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 T is a monad, then a T-algebra $\mathbf{A}$ is called *finitary* if for every $w \in \mathsf{T}A$, there is some finite $A_0 \subseteq A$ such that $w \in \mathsf{T}A_0$. Sometimes, 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 the setting 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 finite* T*-algebra*

$$\mathsf{synt} f : \mathbf{A} \to \mathbf{A}_f,$$

called the syntactic morphism of f*, which recognises* f *and which factors through every surjective* T*-morphism recognising* f*. Furthermore,* $\mathsf{synt} f$ *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.

Example 1. Consider the monad of infinite words and the language

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

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. Define a function

$$h_\sim : \{a, b\}^\infty \to \underbrace{\mathbb{N} \cup (\mathbb{N}^2 \times \mathbb{N}/_\sim) \cup \{\bot, \top\}}_{A}$$

as follows. If the input is infinite, then $h_\sim$ returns $\bot$ 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 identify the same value to all words in ba^*b. By associativity, h would identify 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.

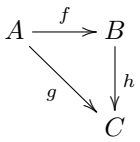
3.1 Proof of the Syntactic Morphism Theorem

We are working in the setting of sorted sets; fix therefore a set of sort names, and a monad T in this setting. We first show that the syntactic morphism is unique up to ismorphisms on the target algebra. This is a consequence of the following lemma, which is not specific to the setting of sorted sets. In the lemma, the crucial distinction is between a morphism (in the underlying category) between universes of two T-algebras, and such a morphism 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, and let*

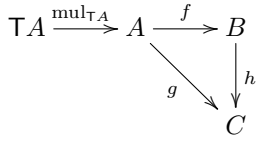
$$f : \mathbf{A} \to \mathbf{B} \qquad \text{and} \qquad g : \mathbf{A} \to \mathbf{C}$$

be T-morphisms such that, as morphisms on the universes,

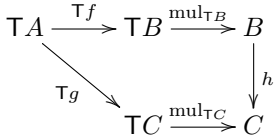


commutes for some h, and f is surejctive. Then h is a T-morphism.

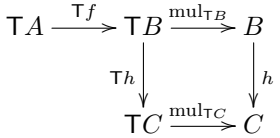
Proof.
This should be a standard lemma on Eilenberg-Moore alebras, citation needed. By the assumption of the lemma, the following diagram commutes



Because both f and g are T-morphisms, it follows that the diagram



commutes. By the commuting diagram in assumption of the lemma, with T applied to it, we see that the following diagram commutes.



Because f is surjective, and therefore also $\mathsf{T}f$, it follows that the diagram above commutes after removing $\mathsf{T}A$ and $\mathsf{T}f$, which is the definition of h being a T-morphism. □

Congruences. Define a *congruence* in an T-algebra $\mathbf{A}$ to be a morphism $g : A \to B$ from the universe of $\mathbf{A}$ to some object in the underlying category, such that $g \circ \mathrm{mul}_{\mathbf{A}}$ factors through $\mathsf{T}g$. For a congruence, one can define a multiplication operation on B, which makes it into a T-algebra such that g is a T-morphism. Therefore, a congruence is simply a T-morphism with the algebraic structure on the target being ommitted.

Polynomials. For an object X in the category, define the set of *polynomials over* $\mathbf{A}$ *with variables* X to be

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

For a valuation $v : X \to A$, we consider the evaluation morphism

$$[\![\_]\!](v) : \mathsf{pol}_X \mathbf{A} \to A$$

which first replaces the variables in the argument polynomial by the valuation v, and then applies the multiplication in $\mathbf{A}$. It is not clear how to define the other kind of evaluation $[\![p]\!](\_)$ in a meaningful way so that it is a morphism in the category, i.e. it is a sort preserving function. The problem is that the valuation might have a different sort than the polynomial (in other words, the input and output might have different sorts). The same problem applies to defining $[\![\_]\!](\_)$.

Unary polynomials. In the 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. For a sort τ, a *unary polynomial with input sort* τ *over* $\mathbf{A}$ is defined to be a polynomial over $\mathbf{A}$ with variables $\{\tau\}$. By abuse of notation, if τ is a sort then we write $\mathsf{pol}_\tau \mathbf{A}$ and $[\![p]\!](a)$, respectively, instead of the formally correct $\mathsf{pol}_{\{\tau\}} \mathbf{A}$ and $[\![p]\!](\tau \mapsto a)$.

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, unary polynomials form a category in the following sense. The objects of the category are sorts. Morphisms from τ to σ are unary polynomials, where the sort of the variable is τ and the sort of the polynomial itself is σ. Composition is defined as follows: for polynomials

$$\tau \xrightarrow{\ p\ } \sigma \xrightarrow{\ q\ } \pi \ ,$$

their composition is the polynomial $q \circ p$ obtained by substituting p for variable σ in q, this polynomial satisfies

$$[\![q \circ p]\!](a) = [\![q]\!]([\![p]\!](a)) \qquad \text{for every } a \in A \text{ of sort } \tau$$

Definition of the syntactic morphism. Consider a colouring

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

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

$$f([\![p]\!](a)) = f([\![p]\!](b)) \qquad \text{for every } p \in \mathsf{pol}_\tau \mathbf{A}.$$

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

$$\mathsf{synt} f : A \to A_f$$

to be the function which maps a to its equivalence class under $\sim$. A tempting alternative would be to use the function $a \mapsto f_a$, unfortunately it is not clear how to make this function into a morphism of the underlying category.

We will show that $\mathsf{synt} f$ is a congruence, and therefore there is a multiplication operation on A_f image which makes $\mathsf{synt} f$ into a surjective T-morphism.

Lemma 3.3 *If $h : \mathbf{A} \to \mathbf{B}$ recognises f, then $\mathsf{synt} f$ factors through h.*

Proof.
We need to show that every $a_1, a_2 \in A$ satisfy

$$h(a_1) = h(a_2) \qquad \text{implies} \qquad a_1 \sim a_2.$$

Because a_1, a_2 have the same images under h, they must have the same sort, call it τ. To prove $a_1 \sim a_2$ we need to show that

$$f([\![p]\!](a_1)) = f([\![p]\!](a_2)).$$

holds for every $p \in \mathsf{pol}_\tau \mathbf{A}$. Because h recognises f, it suffices to show

$$h([\![p]\!](a_1)) = h([\![p]\!](a_2)). \tag{1}$$

Define $h(p) \in \mathsf{pol}_\tau \mathbf{B}$ by applying h to all values from A that appear in p and leaving the variable τ alone. Because h is a morphism of algebras, we have

$$h([\![p]\!](a_i)) = [\![h(p)]\!](h(a_i))) \qquad \text{for } i = 1, 2,$$

which implies (1) because the above right side only depends on $h(a_i)$. □

We write $g_1 \approx g_2$ for morphisms $g_1, g_2 : X \to A$ if their values are pointwise equivalent under $\sim$, i.e. when $\mathsf{synt} f \circ g_1 = \mathsf{synt} f \circ g_2$.

Lemma 3.4 *Let X be finite, and let $g_1, g_2 : X \to A$. Then*

$$g_1 \approx g_2 \qquad \textit{implies} \qquad \mathrm{mul}_{\mathbf{A}} \circ \mathsf{T} g_1 \approx \mathrm{mul}_{\mathbf{A}} \circ \mathsf{T} g_2$$

Proof.
We begin by proving the lemma for a special case.

A special case. Consider the special case when $F = A \sqcup \{\tau\}$ for some sort τ, and g_1, g_2 are the identity on A. (Strictly speaking, this is not really a special case, because the universe of a finite algebra might not be finite, which is the case for instance in categories with infinitely many sorts.) Let us write a_i for $g_i(\tau)$. The assumption of the lemma says that $a_1 \sim a_2$. To prove the lemma for the special case, we need to show that

$$[\![w]\!](a_1) \sim [\![w]\!](a_2))$$

holds for every $w \in \mathsf{pol}_\tau \mathbf{A}$. Let w be as above, and let σ be its sort. By unraveling the definition of $\sim$, the above equality says that every $v \in \mathsf{pol}_\sigma \mathbf{A}$ satisfies

$$f([\![v]\!]([\![w]\!](a_1))) = f([\![v]\!]([\![w]\!](a_2))). \tag{2}$$

By composing the polynomials v and w, we get a unary polynomial $v \circ w \in \mathsf{pol}_\tau \mathbf{A}$ such that

$$[\![v]\!]([\![w]\!](a_i)) = [\![v \circ w]\!](a_i) \qquad \text{for } i = 1, 2$$

By assumption that $a_1 \sim a_2$, the right side above does not depend on i, which proves (2). This completes the proof of the special case of the lemma.

General case. The proof is by induction on the number of arguments where g_1 and g_2 give different results. When there are no such arguments, the result is immediate. When there is one argument, the result follows from the special case. The induction step is straightforward. Choose some x with $g_1(x) \neq g_2(x)$, and define $g_{1\frac{1}{2}} : X \to A$ to be the same as g_1, on all arguments, except for x which is mapped to $g_2(x)$. The result follows by applying the induction assumption to the pair g_1 and $g_{1\frac{1}{2}}$, and then to the pair $g_{1\frac{1}{2}}$ and g_2. □

Recall the assumption that the coloring f is recognised by a morphism

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

into a finitary T-algebra. By Lemma 3.3, there is some

$$\mathsf{synt}_{\mathbf{B}} f : \mathbf{B} \to A_f \qquad \text{with } \mathsf{synt} f = \mathsf{synt}_{\mathbf{B}} f \circ h.$$

We will prove that $\mathsf{synt}_{\mathbf{B}} f$ is a congruence in $\mathbf{B}$.

Lemma 3.5 $\mathsf{synt}_{\mathbf{B}} f$ *is a congruence in* $\mathbf{B}$.

Proof.
Recall that the definition of a congruence says that $\mathsf{synt}_{\mathbf{B}} f \circ \mathrm{mul}_{\mathbf{B}}$ factors through $\mathsf{T}(\mathsf{synt}_{\mathbf{B}} f)$. Because $\mathbf{B}$ is finitary, it suffices to show that

$$\mathsf{synt}_{\mathbf{B}} f \circ \mathrm{mul}_{\mathbf{B}} \circ \mathsf{T}\eta \qquad \text{factors through} \qquad \mathsf{T}(\mathsf{synt}_{\mathbf{B}} f) \circ \mathsf{T}\eta$$

holds for $\eta : F \to B$ with finite domain. By surjectivity of h, it suffices to show

$$\mathsf{synt}_{\mathbf{B}} f \circ \mathrm{mul}_{\mathbf{B}} \circ \mathsf{T}(h \circ \eta) \qquad \text{factors through} \qquad \mathsf{T}(\mathsf{synt}_{\mathbf{B}} f) \circ \mathsf{T}(h \circ \eta) \tag{3}$$

holds for every finite $\eta : F \to A$ with finite domain. Let us simplify the left side of the above:

$$\mathsf{synt}_{\mathbf{B}} f \circ \mathrm{mul}_{\mathbf{B}} \circ \mathsf{T}(h \circ \eta) = \mathsf{synt}_{\mathbf{B}} f \circ h \circ \mathrm{mul}_{\mathbf{A}} \circ \mathsf{T}\eta = \mathsf{synt} f \circ \mathrm{mul}_{\mathbf{A}} \circ \mathsf{T}\eta \tag{4}$$

with the first equality coming T being a functor and h being a T-morphism, and the second equality coming from the definition of $\mathsf{synt}_{\mathbf{B}} f$. Let us simplify the right side of (3):

$$\mathsf{T}(\mathsf{synt}_{\mathbf{B}} f) \circ \mathsf{T}(h \circ \eta) = \mathsf{T}(\mathsf{synt}_{\mathbf{B}} f \circ h \circ \eta) = \mathsf{T}(\mathsf{synt} f \circ \eta) \tag{5}$$

with the first equality coming from T being a functor, and the second equality coming from the definition of $\mathsf{synt}_{\mathbf{B}} f$. Therefore, we are left to prove that the morphism in (4) factors through the one in (5). In other words, we need to show that for every $w_1, w_2 \in \mathsf{T}F$ satisfy

$$\mathsf{T}\eta(w_1) \sim \mathsf{T}\eta(w_2) \qquad \text{implies} \qquad \mathrm{mul}_{\mathbf{A}} \circ \mathsf{T}\eta(w_1) \sim \mathrm{mul}_{\mathbf{A}} \circ \mathsf{T}\eta(w_2)$$

where $\sim$ on the left side is, formally speaking, the pointwise lifting of $\sim$ to $\mathsf{T}A$. Let then w_1, w_2 be as in the assumption of the above implication.

Assumption 1 *If $w_1, w_2 \in \mathsf{T}F$ have the same value under some $\mathsf{T}g$, then there is some $w \in \mathsf{T}(F \times F)$ which projects to w_1 and w_2, respectively.*

This means that there is some $w \in \mathsf{T}(F \times F)$ such that

$$w_i = \mathsf{T}\pi_i(w) \qquad \text{and} \qquad \eta \circ \pi_1 \approx \eta \circ \pi_2.$$

The result then follows by applying Lemma 3.4 to $\eta \circ \pi_1$ and $\eta \circ \pi_2$. □

By the above lemma, there is a multiplication operation on A_f which makes it into an algebra $\mathbf{A}_f$ such that $\mathsf{synt}_{\mathbf{B}} f$ is a T-morphism. Therefore, $\mathsf{synt} f = \mathsf{synt}_{\mathbf{B}} f \circ h$ is a T-morphism from $\mathbf{A}$ to $\mathbf{A}_f$, as the composition of T-morphisms. This completes the proof of the Syntactic Morphism Theorem.

4 Pseudovarieties

Eilenberg's pseudovariety theorem says that, in the case of monoids, 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 monoid 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 monoid. A typical application 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, though, the algorithm is usually found in a case-by-case way.

Here we prove that the theorem works in general for monads in the setting of 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.

Actually, we prove a slightly more general version, for the setting of sorted sets, because many of our examples monad will be for sorted sets. This generalisation subsumes pseudovariety theorems for: finite words in both monoid and semigroup variants [8], ∞-words [18], scattered linear orderings [1], finite trees [14]. When there are infinitely many sorts, which will be the case for some of our examples, additional assumptions are needed, namely languages are required to be finitely sorted, and algebras are required to be finitely observable and finitely generated, as defined below.

Language pseudovarieties. Call a language *finitely sorted* if there are finitely many sorts where the language is nontrivial (a language is trivial on some sort if it contains everything or nothing on that sort). When there are finitely many sorts, every language is finitely sorted. If $L \subseteq \mathsf{T}\Sigma$ is a T-language then a *derivative* of L is a language of the form $[\![p]\!]^{-1}(L)$, where p is a unary polynomial. Note that polynomials are typed in the sense that they have input and output sorts; and the derivative will contain only elements of the input sort and will only depend on elements of L in the output sort.

Definition 4.1 *A* T-language pseudovariety *is a class finitely sorted* T*-recognisable* T*-languages which is closed under Boolean combinations, derivatives, and preimages under* T*-morphisms.*

In the above definition, a T-language 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 for instance when T is the monad of finite words, then there are language pseudovarieties (e.g. languages that only depend on the length of a word) that contain the language a^* over the alphabet $\{a\}$, but do not contain the same set a^* when seen as a language over the alphabet $\{a, b\}$.

Algebra pseudovarieties. Call a T-algebra $\mathbf{A}$ *finitely observable* if there is a finite set of sorts S_0 such that for every $a, b \in \mathbf{A}$ with sort τ, $a = b$ holds if and only if

$$[\![p]\!](a_1) = [\![p]\!](a_2)$$

holds for every unary polynomial with input sort τ and output sort in S_0. It is not difficult to show that a T-algebra is finitely observable if and only if it is the syntactic algebra of some colouring that is constant on all but finitely many sorts. A T-algebra $\mathbf{A}$ is called *finitely generated* if there is some finite subset A_0 of the universe such that multiplication is surjective when restricted to $\mathsf{T}A_0$.

When there are finitely many sorts, every algebra where the universe is finite on every sort is finitely observable and finitely generated.

Definition 4.2 *A* T-algebra pseudovariety *is a class of* T*-algebras that are finite, finitely observable and finitely generated, and such that the class is closed under finite products, images of surjective* T*-morphisms, and subalgebras.*

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

Theorem 4.3 *[Pseudovariety Theorem] Let* T *be a monad in the setting of sorted sets. The mapping* Lan *is a bijection between* T*-algebra pseudovarieties and* T*-language pseudovarieties, and its inverse is* Alg*.*

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

To prove the Pseudovariety Theorem, it remains to show that if $\mathbb{C}$ and $\mathbb{A}$ are pseudovarieties of T-colorings and T-algebras respectively, then

$$\mathsf{Alg}\,\mathsf{Lan}\,\mathbb{A} = \mathbb{A} \qquad \text{and} \qquad \mathsf{Lan}\,\mathsf{Alg}\,\mathbb{L} = \mathbb{L}.$$

By definition, the class $\mathsf{Alg}\,\mathsf{Lan}\,\mathbb{A}$ consists of finite T-algebras $\mathbf{A}$ that are finitely observable, finitely generated, and such that every finitely sorted T-language recognised by $\mathbf{A}$ is recognised by some T-algebra from $\mathbb{A}$. Thererefore

$$\mathsf{Alg}\,\mathsf{Lan}\,\mathbb{A} \supseteq \mathbb{A}.$$

Again by definition, the class $\mathsf{Lan}\,\mathsf{Alg}\,\mathbb{L}$ consists of finitely sorted T-languages that are recognised by some finite T-algebra that is finitely observable, finitely generated, and which only recognises T-languages from $\mathbb{L}$. Therefore

$$\mathsf{Lan}\,\mathsf{Alg}\,\mathbb{L} \subseteq \mathbb{L}.$$

The remaining inclusions are shown in the following two lemmas.

Lemma 4.4 *Let* $\mathbf{A}$ *be a finite* T*-algebra, which is finitely generated and finitely observable, and such that every* T*-language recognised by* $\mathbf{A}$ *is recognised by some* T*-algebra from* $\mathbb{A}$*. Then* $\mathbf{A} \in \mathbb{A}$*.*

Proof.
Let A_0 be a subset of the universe of $\mathbf{A}$ that contains all elements of observable sorts, and all generators. For $a \in A_0$ define

$$L_a = \{w \in \mathsf{T}A_0 : \mathrm{mul}_{\mathbf{A}} w = a\}.$$

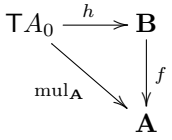
By the assumption of the lemma, the language L_a is recognised by some

$$h_a : \mathsf{T}A_0 \to \mathbf{B}_a \in \mathbb{A},$$

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

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

This is a T-morphism h that recognises every language L_a with $a \in A_0$. The T-algebra $\mathbf{B}$ belongs to $\mathbb{A}$, by closure of $\mathbb{A}$ under finite products and subalgebras. We claim that there is some function f from the universe of $\mathbf{B}$ to the universe of $\mathbf{A}$ which makes the following diagram commute:



If the claim is proved, then Lemma 3.2 will imply that f is a T-morphism. Since f is surjective by the assumption that A_0 contains a set of generators, it will follow that $\mathbf{A} \in \mathbb{A}$ by closure of $\mathbb{A}$ under images of surjective T-morphisms.

We are left with proving the claim, which says that

$$h(w) = h(w') \qquad \text{implies} \qquad \mathrm{mul}_{\mathbf{A}}(w) = \mathrm{mul}_{\mathbf{A}}(w') \qquad \text{for every } w, w' \in \mathsf{T}A_0.$$

Let then w, w' be as in the assumption of the above implication. They must have the same sort, call it τ, because they have equal image under h. By assumption on $\mathbf{A}$ being finitely observable, it suffices to prove

$$p(\mathrm{mul}_{\mathbf{A}}(w)) = p(\mathrm{mul}_{\mathbf{A}}(w'))$$

for every $p \in \mathsf{pol}_\tau \mathbf{A}$ with observable output sort. Let then p be such a polynomial. Because A_0 are generators of $\mathbf{A}$, there must be some $q \in \mathsf{pol}_\tau A_0$ which is mapped to p by $\mathrm{mul}_{\mathbf{A}}$. Because h is a T-morhpism,

$$h(q(w)) = (h(q))(h(w)) = (h(q))(h(w')) = h(q(w')).$$

Because h recognises both languages $L_{\mathrm{mul}_{\mathbf{A}}(q(w))}$ and $L_{\mathrm{mul}_{\mathbf{A}}(q(w'))}$,

$$\mathrm{mul}_{\mathbf{A}}(q(w)) = \mathrm{mul}_{\mathbf{A}}(q(w')),$$

which proves the result of the claim by invoking the definition of q and the fact that $\mathrm{mul}_{\mathbf{A}}$ is a T-morphism. □

Lemma 4.5 *Every $L \in \mathbb{L}$ is recognised by some finite T-algebra $\mathbf{A}$ which is finitely generated, finitely observable, and which only recognised languages in $\mathbb{L}$.*

Proof.
Let $L \subseteq \mathsf{T}\Sigma$ be as in the statement of the lemma. Apply the Syntactic Morphism Theorem, yielding a syntactic morphism

$$\mathsf{synt}f : \mathsf{T}\Sigma \to \mathbf{A}_f$$

that recognises L. The algebra $\mathbf{A}_f$ is finitely generated, namely generated by the images of letters in Σ.

To prove that $\mathbf{A}_f$ is finitely observable, we need to recall the way $\mathbf{A}_f$ is constructed in the Syntactic Morphism Theorem. We say that $p \in \mathsf{pol}_\tau \mathsf{T}\Sigma$ distinguishes $w, w' \in \mathsf{T}\Sigma$ of type τ if

$$[\![p]\!](w) \in L \qquad \text{iff} \qquad [\![p]\!](w') \notin L$$

By construction of the syntactic morphism, elements w, w' with different images under $\mathsf{synt}f$ can be distinguished by some polynomial in $\mathsf{pol}_\tau \mathsf{T}\Sigma$; this polynomial must have output sort such that L nontrivial on the sort. Therefore, the algebra $\mathbf{A}_f$ is finitely observable.

Claim 4.5.1 *Let τ be a sort. There is a finite set*

$$P_\tau \subseteq \mathsf{pol}_\tau \mathsf{T}\Sigma$$

such that if $w, w' \in \mathsf{T}\Sigma$ have type τ but different images under $\mathsf{synt}f$, then they are distinguished by some polynomial from P_τ.

Proof.
As we have already observed, elements w, w' with different images under $\mathsf{synt}f$ can be distinguished by some polynomial in $\mathsf{pol}_\tau \mathsf{T}\Sigma$. Furthermore, because $\mathsf{synt}f$ recognises L, the choice of polynomial need only depend on the values of $\mathsf{synt}f(w), \mathsf{synt}f(w')$, for which there are finitely many possibilities. □

For $a \in \mathbf{A}_f$, define $L_a \subseteq \mathsf{T}\Sigma$ to be the inverse image of a under the syntactic morphism. A corollary of the above Claim is that for every element a in $\mathbf{A}_f$ of type τ, its inverse image L_a under $\mathsf{synt}f$ is a Boolean combination of languages

$$[\![p]\!]^{-1} L \qquad \text{with } p \in P_\tau.$$

These languages are all derivatives of L, and by finiteness of P_τ, the Boolean combination is finite. Therefore L_a belongs to $\mathbb{L}$ as a finite Boolean combination of derivatives of L.

Consider a language $K \subseteq \mathsf{T}\Gamma$ that is recognised by a morphism

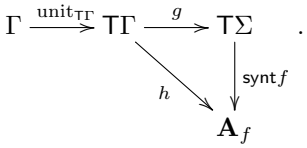
$$h : \mathsf{T}\Gamma \to \mathbf{A}_f.$$

By surjectivity of the syntactic morphism, there is some function from for every $a \in \Gamma$ one can choose some $w_a \in \mathsf{T}\Sigma$ such that h maps (the unit of) a to $\mathsf{synt}f(w_a)$.

The T-algebra $\mathsf{T}\Gamma$ has the property that every mapping from Γ to the universe of some algebra $\mathbf{B}$ extends uniquely to a T-morphism $\mathsf{T}\Gamma \to \mathbf{B}$. Since the syntactic morphism is surjective, for every $a \in \Gamma$ there is some $w_a \in \mathsf{T}\Sigma$ with

$$h(a) = \mathsf{synt}f(w_a).$$

Combining these observations, we see that there is a T-morphism $g : \mathsf{T}\Gamma \to \mathsf{T}\Sigma$ such that the following diagram commutes



The units of Γ are generators in $\mathsf{T}\Gamma$, and if T-morphisms agree on generators then they agree over all elements; therefore $h = \mathsf{synt}f \circ g$. It follows that

$$h^{-1}(a) = g^{-1}(L_a) \qquad \text{for every } a \in \mathbf{A}_f,$$

which is a language in $\mathbb{L}$ by closure under inverse T-morhpisms. $\square$

This completes the proof of the Pseudovariety Theorem.

5 Representing an algebra

In all interesting cases, the monad T produces infinite sets, even on finite arguments. Therefore, the finiteness of the universe of a 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

$$\mathrm{mul}_{\mathbf{A}} : \mathsf{T}A \to A.$$

In this section, we present one such way. The idea is to find a function T_0, such that which chooses for every finite set A a finite subset $\mathsf{T}_0 A \subseteq \mathsf{T}A$ 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 A^2, because multiplication in a monoid is uniquely determined by its binary part. In the example of ∞-algebras, the function T_0 maps A to $A^2 \cup \{a^\omega : a \in A\}$, as per Theorem 1.1. We now describe these notions in more detail.

Subfunctors. In this section, we assume that the monad is in the category of sets, or sorted sets, and therefore the notion of subset can be used. Define a *subfunctor* of a monad T to be a functor T_0 which maps a set X to a subset $\mathsf{T}_0 X \subseteq \mathsf{T}X$, and which maps a function $f : X \to Y$ to the function obtained from $\mathsf{T}f$ by restricting the domain and codomain using T_0.

A subfunctor on its own is not a monad, 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

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

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 $\mathbf{A}$

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

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.

Example 2. Consider the monad ∞ for infinite words. Define

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

The submonad T_0^* maps X to the finite and ultimately periodic words over alphabet X. In particular, T_0 is finitely complete, see Theorem 1.1. □

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:

$$\mathrm{mul}_{\mathbf{A}}|_{\mathsf{T}_0 A} : \mathsf{T}_0 A \to A$$

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. 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.

Example 3. Let us continue the discussion in Example 2. The T_0-representation of a finite ∞-algebra consists of a finite universe A together with two operations

$$\_ \cdot \_ : A \times A \to A \qquad \_^{\omega} : A \to A.$$

This is essentially the same thing as a Wilke semigroup. Note that not every choice of finite universe and two 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 for the case of ∞-algebras in [19]. $\square$

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. Languages recognised by finite recognisers are the same thing as MSO definable languages for finite words, infinite words, finite trees, infinite trees, etc. In Section 6.1 we establish this connection on the abstract level of monads.

6.1 Definition of MSO

To establish the connection between MSO and recognisability, consider the following lemma, see [16], which characterises MSO in a way that does not talk 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 $h : \Sigma^* \to \Gamma^*$, and which contains the languages*

$$0^* \subseteq \{0,1\}^* \qquad \textit{and} \qquad 0^*1^* \subseteq \{0,1\}^*.$$

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 $\mathrm{MSO}_{\mathsf{T}}(\mathcal{L})$ to be the smallest class of T-languages which contains $\mathcal{L}$, is closed under Booolean 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 [2].

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

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 $v \in \mathsf{TP}X$ 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

$$\mathsf{P}\mathbf{A} : \mathsf{TP}A \to \mathsf{P}A$$

where whose multiplication operation maps $w \in \mathsf{TP}A$ to the set

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

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

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 an expression that uses the constructors of MSO formulas, with the predicates being represented by T-morphisms recognising them (see Definition 6.3). 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. This means that there is a finite set of sort names; and the category underlining the monad has sets with these sorts as objects, and sort-preserving functions as morphisms. Finite (sorted) sets are those with finitely many elements, and finite T-algebras are those with finite universes.

Recall the notion of a finitely complete subfunctor $\mathsf{T}_0 \subseteq \mathsf{T}$, which implied that every finite T-algebra is uniquely determined by its T_0-reduct. If T_0 preserves finiteness, then finite T-algebras can be represented in a finite way, and then used in algorithms, by providing the universe and the multiplication operation restricted to T_0. For the representation to be used in deciding MSO, we will use a slightly stronger condition on subfunctor, which is given in the following definition.

Definition 6.3 *A subfunctor T_0 is called* effective *if for every finite set Σ,*

1. *$\mathsf{T}_0 X$ is finite and can be computed up to isomorphism; and*
2. *For every $w \in \mathsf{T}_0\Sigma$, one can compute a representation of a T-morphism*

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

into a finite T*-algebra that recognises* $\{w\}$*. The representation consists of: the universe* A *of* $\mathbf{A}$*, the values of* h *on elements of* Σ*, the image* $h(w)$*, and the multiplication operation of* $\mathbf{A}$ *when restricted to* $\mathsf{T}_0 A$*.*

Example 4. Consider the monad ∞ of infinite words, and the subfunctor

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

which was considered in Examples 2 and 3, and proved to be finitely complete. We claim that T_0 is also effective. Clearly T_0 preserves finiteness and can be computed, as $\mathsf{T}_0 X$ is isomoprhic to $X^2 \sqcup X$. 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 crieterion for decidable MSO satisfiability is having a subfunctor that is finitely complete and effective.

Theorem 6.4 *Let* T *be a monad in the setting of finitely sorted sets. If there is a subfunctor* T_0 *that is effective and finitely complete, then* MSO *satisfiability is decidable.*

Theorem 6.4 is abstract nonsense 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 2, 3 and 4, which show that the monad of infinite words has a subfunctor that is finitely complete and effective, and therefore Theorem 6.4 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 [6], or its algebraic version in [19]. 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 effective, with finite completeness being essential part.

Theorem 6.4 follows immediately from the following lemma.

Lemma 6.5 *Given* T_0*-reducts of a finite* T*-algebras* $\mathbf{A}, \mathbf{B}$*, one can compute the* T_0*-reducts of* $\mathsf{P}\mathbf{A}$ *and* $\mathbf{A} \times \mathbf{B}$*.*

Proof.
The Cartesian product is immediate, the interesting case is the powerset $\mathsf{P}\mathbf{A}$. For $w \in \mathsf{T}_0 \mathsf{P} A$, we need to compute $\mathrm{mul}_{\mathsf{P}\mathbf{A}}(w)$.. By effectivity of T_0, we can compute a T-morphism

$$h : \mathsf{T}(\mathsf{P}A) \to \mathbf{B}$$

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

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

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

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

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

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

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);
- various kinds of trees (Section 9).

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 totally ordered set of *positions*, together with a labelling of these positions by Σ. Chains form a monad. 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.

A famous theorem of Shelah [13] says that MSO is undecidable on the Cantor space, which implies that satisfiability of MSO is undecidable on arbitary chains, or even on chains of cardinality continuum. For the undecidability result it suffices to have the order predicate, which corresponds to the set of chains over alphabet $\{0, 1\}$ where all zeros are before all ones. It follows that the assumptions of Theorem 6.4 cannot be met. These problems go away if one considers countable chains.

Countable chains. A *countable* chain is one where the indexing set 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. 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 [13], see also [7].

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 shuffle$Y$ *is the chain where the positions are rational numbers and where every* $y \in Y$ *labels a dense subset.*

Proof. (of 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 Theorem 1.1. □

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 effective. Therefore, the result follows from Theorem 6.4. □

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 Unary queries

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 U, is used to model unary queries, i.e. sets of words with one distinguished position. The monad is defined by

$$\mathsf{U}A \quad \stackrel{\text{def}}{=} \quad A^* \underline{A} A^*,$$

where $\underline{A}$ is a disjoint copy of the set A. The idea is that a word $w \in \mathsf{U}A$ represents a nonempty word over A where the underlined position is distinguished. 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.

Consider a U-algebra $\mathbf{A}$. For $a \in A$, define its *left transformation* to be the function $A \to A$ defined by

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

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 U-algebra is uniquely specified by its universe A and the the left and right transformations for each $a \in A$.

Let $L \subseteq \mathsf{U}\Sigma$ be a U-language. Define $\mathsf{query}L$ to be the set of pairs (w, x) where $w \in \Sigma^+$ and x is a position in w such that L contains the word obatained from w by underlining position x. Define $\mathsf{lang}L$ to be L when viewed as a language of nonempty words over the extended alphabet $\Sigma \cup \underline{\Sigma}$. For some applications, the two views are essentially the same. For instance $\mathsf{query}L$ is defined by an MSO formula with one free variable if and only if $\mathsf{lang}L$ is defined by an MSO formula without free variables; the same holds for first-order logic and various fragments. For these logics, the monad U does not contribute anything new.

The following example shows that U-algebras are useful for talking about two-variable first-order logic. For concreteness, we assume that 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. This variant of first-order logic is a well-studied fragment for words, see e.g. [15], and it also makes sense for unary queries, as it corresponds to unary queries definable in XPath with only the transitive axis // and its inverse.

Example 5. Let the alphabet be $\{a, b\}$, and consider the unary query "the successor of the selected position has label a", i.e. the U-language

$$L = \{w\underline{\sigma}av : w, v \in \{a, b\}^*, \sigma \in \{a, b\}\}.$$

One can show that $\mathsf{lang}L$ is definable by a formula of two-variable first-order logic (without free variables), while $\mathsf{query}L$ is not definable by any formula of two-variable first-order logic (with one free variable).

Also, one can observe that just looking at the left and right monoids as defined above is not sufficient to understand the query. In this case, the left monoid is trivial, i.e. has one element only, 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. □

Actually, by redoing the proof in [15], we can get an effective characterisation of unary queries definable in two-variable logic.

Theorem 8.1 *Let $L \subseteq \mathsf{U}\Sigma$ be a U-language. Then $\mathsf{query}L$ is definable in two variable first-order logic if and only if its syntactic U-algebra $\mathbf{A}$ satisfies the following condition. There is some $n \in \mathbb{N}$, such that for every $w \in A^*$ which contains all letters from a set $B \subseteq A$, and every $w_1, w_2 \in B^*$ and $a \in B$, the following equality holds.*

$$\mathrm{mul}_{\mathbf{A}}(w^n w_1 \underline{a} w_2 w^n) = \mathrm{mul}_{\mathbf{A}}(w^n \underline{a} w^n)$$

(TODO: prove)

9 Monads for trees

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

9.1 A monad for ranked trees over a fixed 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 T_Σ, which is parametrised by Σ, and which will model ranked trees over Σ. Although the alphabet Σ is ranked, the monad T_Σ itself is in the category of sets, i.e. sets without any special arity structure imposed.

Define 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 $\mathsf{T}_\Sigma\mathsf{T}_\Sigma \to \mathsf{T}_\Sigma$ is term substitution, while the unit maps a variable $a \in \Gamma$ to the term that consists only of this variable. This is the clasical term monad $[]$.

If Γ is a finite alphabet, then a 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 T_Σ-language over the empty alphabet is a set of ranked trees over the alphabet Σ.

Connections with Σ-algebras. Recall that in universal algebra, a Σ-algebra consists of a universe A together, together with an operation $f : A^n \to A$ 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

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

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

$$\Sigma A = \{f(a_1, \ldots, a_n) : f \text{ is an } n\text{-ary symbol in } \Sigma\} \subseteq \mathsf{T}_\Sigma A$$

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

Connection with automata. For a T_Σ algebra $\mathbf{A}$, there is a unique T_Σ-morphism

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

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.

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 A monad for clones

We now define 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 $A^n \to A$, of possibly different arities $n \in \mathbb{N}$, which includes all projection functions of the form $(a_1, \ldots, a_n) \mapsto a_i$ and is closed under composition in the sense that if the clone contains an n-ary operation f and k-ary operations $f_1, \ldots, f_n$, then it also contains the k-ary operation

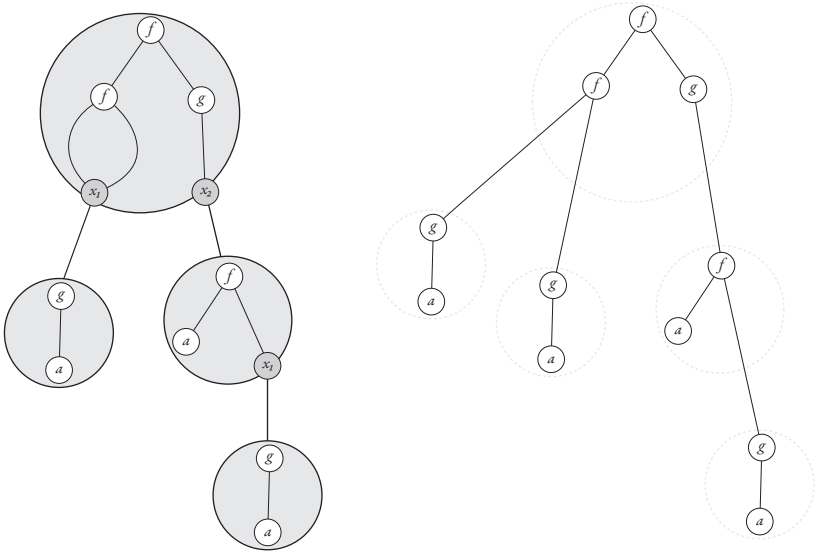
$$\bar{a} \in A^k \qquad \mapsto \qquad f(f_1(\bar{a}), \ldots, f_n(\bar{a})) \in A.$$

The category of ranked sets. To model clones by a monad, we will use a different category than sets. The category is going to be 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 $\mathrm{clo}\Sigma$, where elements of rank n are terms over Σ that use n variables $x_1, \ldots, x_n$ (the sequence of variables $x_1, x_2, \ldots$ 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 composition operation $\mathrm{clo}\mathrm{clo}\Sigma \to \mathrm{clo}\Sigma$ is substitution, as illustrated in Figure 2.

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, 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



$t \in \operatorname{clo}\operatorname{clo}\Sigma$
t has 4 nodes and rank 0

$\operatorname{mul}(t) \in \operatorname{clo}\Sigma$
$\operatorname{mul}(t)$ has 11 nodes and rank 0.

Figure 2: Example of multiplication in a clo-algebra. The ranked alphabet Σ has elements a, g, f of arities $1, 2, 3$ respectively. Variable x_2 is used twice in the label of the root in the left tree t, 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 $\operatorname{mul}(t)$, they just highlight how $\operatorname{mul}(t)$ is obtained from t.

> 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.

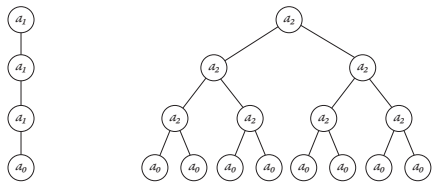
Finitary clones. There is no sense in considering clo-algebras that have a finite universe, because the requirement on projections means that the universe is nonempty on every rank. In clo-algebras, we call a clo-algebra *finite* if it has finitely many elements for every rank, and is finitely generated. The finite generation axiom is natural in the context of recognising languages (every recognisable clo-language over a finite alphabet is recognised by a finitely generated clo-algebra), but it is not superfluous – there exist clones with three elements of rank zero that are not fintiely generated, as shown by Yanov and Muchnik in [20], and this is even the case for polynomial clones [].

Example 6. This is a non-example of a pseudovariety. Consider the following variant of first-order logic on trees over a ranked alphabet. The quantifiers range over nodes of the tree. For every letter $a \in \Sigma$ there is a unary predicate that is true in nodes with label a, and there are binary predicates for the descendant relation, and the i-th child relation for every i. A well-known open problem in is: can one decide if a recognisable language of trees is definable in first-order logic?

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 an example found by Potthoff [12]. Consider letters a_0, a_1, a_2 with ranks $0, 1, 2$ respectively, and consider the clo-morphism

$$h : \mathrm{clo}\{a_0, a_1\} \to \mathrm{clo}\{a_0, a_2\}$$

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:



$t \in \mathrm{clo}\{a_0, a_1\}$ $\qquad$ $h(t) \in \mathrm{clo}\{a_0, a_2\}$

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 is a first-order 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. This problem can be solved by considering a variant of nonduplicating clones, where every variable is used only once. Unfortunately, the theory of nonduplicating clones is less developed than the theory of clones. □

9.3 A monad for forests of unranked trees

We now present another monad for modelling trees. This time the trees are finite (finitely many nodes), labelled (each node comes with a label), unranked (the number of children can be arbitrarily large), and sibling-ordered (the children of a node come with a total order). We assume that the set of labels is partitioned

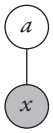
into *forest* labels which are used to label leaves, and *context* labels which are used to label the other nodes. A forest is an ordered sequence of trees in the sense described above.

The forest monad, denoted by F, works in the category of sorted sets, with two sorts "forest" and "context". When applied to a sorted set Σ, it yields:

- on the forest sort, nonempty forests labelled by Σ;
- on the context sort, forests labelled by Σ extended with a fresh leaf letter x, such that x appears in exactly one leaf, which is called the *port*.

The point of the port is to be replaced by forests or contexts. One 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 [5]. 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



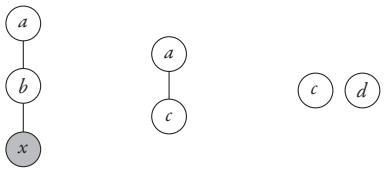
.

The multiplication operation in the monad is illustrated in Figure 3. This is very similar to the multiplication in the clone monad illustrated in Figure 2, with the following differences:

- The forest monad only allows ranks zero (forests) and one (contexts), while the clone monad allows arbitrarily large ranks;
- Rank one elements, i.e. contexts, have exactly one occurrence of the variable, while the clone monad allows multiple use of the same variable;
- Both forests and contexts can have multiple roots.

We call a two-sorted set finite if it is finite on both sorts, and an F-algebra is considered finite if its 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*.*

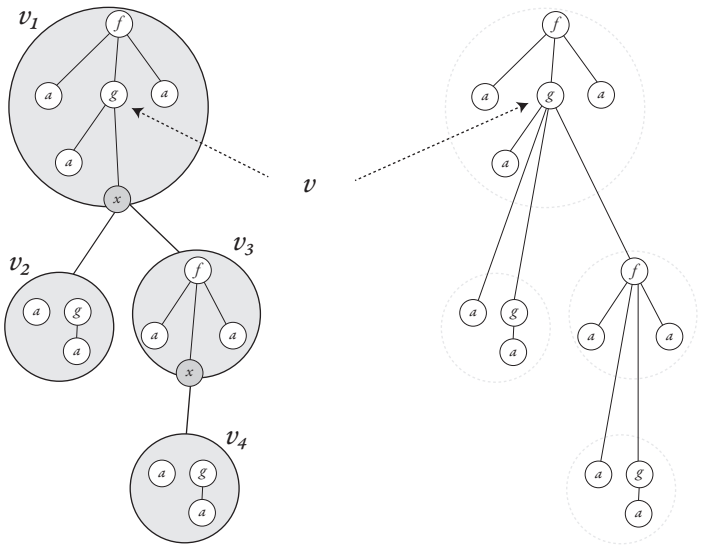


Figure 3: Example of multiplication in the forest monad. Before multiplication, t has two context nodes v_1 and v_3 and two forest nodes v_2 and v_4. After multiplication, t has fourteen nodes, which correspond to the non-variable nodes in the labels of $v_1, \ldots, v_4$. Note how the x in the label 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.

Proof.
The lemma says that 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. □

In other words, F-algebras are the same thing as associative F_0-algebras. Associative F_0-algebras are exactly the same thing as forest algebras from [5], in the variant of forest algebra where there is no empty forest or context.

9.4 A monad for infinite unranked forests.

The ω-forest monad, denoted by $X \mapsto \omega\mathsf{F}X$, 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 7. Consider an alphabet Σ, and let L be an arbitrary set of trees over the alphabet L, not necessarily MSO definable. Define $\mathsf{dense}L$ to be those forests where every node has a some descendant with a subtree in L. We claim that $\mathsf{dense}L$ is recognised by an $\omega\mathsf{F}$-algebra $\mathbf{A}_L$ 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 onde 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 $\mathbf{A}_L$ is seen in the multiplication operation. In particular, there are uncountably many $\omega\mathsf{F}$-algebras with finite universes, and there is no hope of representing them in a finite way. □

As witnessed by the above example, the notion of finite algebra should have some additional requirements. Languages recognised by finite algebras should definitely include those that can be defined in MSO. Recognisable languages should be exactly those definable in MSO; or else there should be a good reason to go beyond MSO. The question of finding an adequate notion of finite $\omega\mathsf{F}$-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 infinite tees. The fact that we use monads, or that the trees are unranked, does not seem to be important.

One, slightly ad hoc, solution is to define an $\omega\mathsf{F}$-algebra $\mathbf{A}$ to finite if its universe is finite on both sorts, and the multiplication operation is MSO definable, in the sense that every language

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

is MSO definable. For this notion of finiteness, [4] shows 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?".

10 Future work

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

- Unranked trees with possibly infintie 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 2. 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 [3]. 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 T, at least in the category of sets or sorted sets, there is a profinite version $\overline{\mathsf{T}}$. This gives immediately definitions, and basic theorems about, things like profinite words, profinite countable chains, profinite trees, etc. Section 14 studies the special case of profinite words, and shows how the generic notion of recognisable language instantiates to an interesting class of languages of profinite words.

11 Profinite monads

Profinite constructions are common in mathematics. In the scope of recognisable languages, the best known profinite construction is the monoid of profinite words. In this section, we show that profinite objects can be defined on the abstract level of monads, in a way that preserves their algebraic structure.

11.1 Definition of the profinite monad

Fix for the rest of this section a monad T, in the category of sets (the generalisation to sorted sets being straightforward). We explain how to convert T into a monad, which we denote by $\overline{\mathsf{T}}$, that describes profinite objects coming from T. The definition is designed so that it generalises the well-known constructions of profinite words, or profinite trees etc.

Types. For a T-algebra $\mathbf{A}$, not necessarily finite, define a T-*morphism type over* $\mathbf{A}$ to be a function τ which maps every surjective T-morphism

$$h : \mathbf{A} \to \mathbf{B} \qquad \text{with } \mathbf{B} \text{ finite} \tag{6}$$

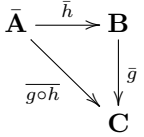
to an element $h^\tau \in \mathbf{B}$, subject to the condition that

$$(g \circ h)^\tau = g(h^\tau) \qquad \text{for every } h : \mathbf{A} \to \mathbf{B} \text{ and } g : \mathbf{B} \to \mathbf{C} \tag{7}$$

where g is a surjective T-morphism between finite T-algebras. The set of of T-morphism types over a T-algebra $\mathbf{A}$ is called its *compactification*, and is denoted by $\bar{\mathbf{A}}$. Define the *profinite extension* of a surjective T-morphism h as in (6) to be the function

$$\bar{h} : \bar{\mathbf{A}} \to \mathbf{B} \qquad \bar{h}(\tau) = h^\tau.$$

In terms of profinite extensions (7) is stated as follows:



(8)

Example 8. Consider the monad $*$ of finite words, where $*$-algebras are monoids, and $*$-morphisms are monoid morphisms. Consider the monoid Σ^* where Σ is a a finite alphabet. A $*$-morphism type (or monoid type) over Σ^* is a function which maps every monoid morphism

$$h : \Sigma^* \to \mathbf{A} \qquad \text{with } \mathbf{A} \text{ a finite monoid}$$

to an element of $\mathbf{A}$, in a way that is consistent with composition. Monoid types over Σ^* are the same thing as profinite words over Σ, see [] for a more thorough introduction to the subject of profinite words.

A simple example of monoid type (i.e. of a profinite word) is one that is induced by a word $w \in \Sigma^*$; and which maps a morphism h to $h(w)$. Stated differently, finite words can be seen as a special case of profinite words.

We now describe a more exciting monoid type (i.e. profinite word), which is constructed using the idempotent power. Recall the well known fact that in every finite monoid $\mathbf{A}$ 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

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

The unique idempotent power of a is usually denoted by a^ω, but we use the notation $a^\#$ to avoid conflict with the ω power in infinite words. Using the idempotent power, we can define for every $w \in \Sigma^*$ a monoid morphism type, denoted by $w^\#$, as follows: a monoid morphism h is mapped to the unique idempotent power of $h(w)$. To show that this is indeed a monoid morphism type, we need to show that for every monoid morphisms

$$h : \Sigma^* \to \mathbf{A} \qquad g : \mathbf{A} \to \mathbf{B}$$

with $\mathbf{A}$ and $\mathbf{B}$ being finite monoids, 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 $\mathbf{B}$.

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

□

The functor of $\overline{\mathsf{T}}$. We now define the profinite monad $\overline{\mathsf{T}}$. An object Σ is mapped by $\overline{\mathsf{T}}$ the compactification of the T-algebra $\mathsf{T}\Sigma$, i.e.

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

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

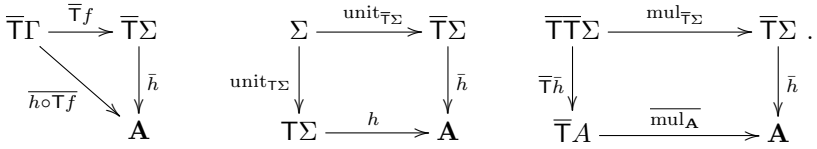
Theorem 11.1 *There are unique operations*

$$\begin{array}{rcll} \overline{\mathsf{T}}f & : & \overline{\mathsf{T}}\Gamma \to \overline{\mathsf{T}}\Sigma & \text{for } f : \Gamma \to \Sigma \\ \mathrm{unit}_{\overline{\mathsf{T}}\Sigma} & : & \Sigma \to \overline{\mathsf{T}}\Sigma & \\ \mathrm{mul}_{\overline{\mathsf{T}}\Sigma} & : & \overline{\mathsf{T}}\overline{\mathsf{T}}\Sigma \to \overline{\mathsf{T}}\Sigma & \end{array}$$

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

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

the following diagrams commute



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

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 $\overline{\mathsf{T}}\Sigma$ is uniquely specified by its values under all possible profinite extensions $\bar{h}$. We need to check that the operations actually produce types, i.e. the values that they produce satisfy (7). Let us first check that $\overline{\mathsf{T}}f$ produces types, i.e. that

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

holds for every finite T-morphisms

$$h : \mathsf{T}\Sigma \to \mathbf{A} \qquad g : \mathbf{A} \to \mathbf{B}$$

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

$$\begin{array}{rcl} \overline{g \circ h} \circ \overline{\mathsf{T}}f & = & \text{(by definition of } \overline{\mathsf{T}}f) \\ \overline{g \circ h \circ \mathsf{T}f} & = & \text{(by (8))} \\ g \circ \overline{h \circ \mathsf{T}f} & = & \text{(by definition of } \overline{\mathsf{T}}f) \\ g \circ \bar{h} \circ \overline{\mathsf{T}}f & & \end{array}$$

For the unit operation, the check is even simpler:

$$\begin{array}{rcl} \overline{g \circ h} \circ \mathrm{unit}_{\overline{\mathsf{T}}\Sigma} & = & \text{(by definition of } \mathrm{unit}_{\overline{\mathsf{T}}\Sigma}) \\ g \circ h \circ \mathrm{unit}_{\mathsf{T}\Sigma} & = & \text{(by definition of } \mathrm{unit}_{\overline{\mathsf{T}}\Sigma}) \\ g \circ \bar{h} \circ \mathrm{unit}_{\overline{\mathsf{T}}\Sigma} & & \end{array}$$

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

$$\overline{\mathsf{T}}(f \circ g) = \overline{\mathsf{T}}f \circ \overline{\mathsf{T}}g \qquad \text{for every } f : \Delta \to \Sigma \text{ and } g : \Gamma \to \Delta.$$

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

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

a T-morphism into a finite T-algebra. This is checked below and illustrated in Figure 4.

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

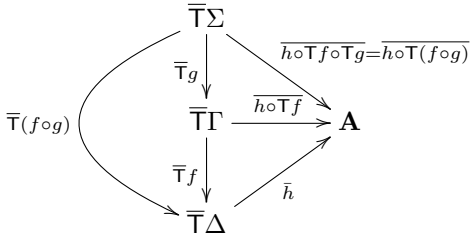


Figure 4: $\overline{\mathsf{T}}$ is a functor.

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

Lemma 11.2 *If $\mathbf{A}$ is a T-algebra, then there is a unique operation*

$$\mathrm{mul}_{\bar{\mathbf{A}}} : \overline{\mathsf{T}}\bar{\mathbf{A}} \to \bar{\mathbf{A}},$$

which makes the following diagram commute

$$\begin{array}{ccc}
\overline{\mathsf{T}}\bar{\mathbf{A}} & \xrightarrow{\mathrm{mul}_{\bar{\mathbf{A}}}} & \bar{\mathbf{A}} \\
\downarrow{\scriptstyle \overline{\mathsf{T}}\bar{h}} & & \downarrow{\scriptstyle \bar{h}} \\
\overline{\mathsf{T}}B & \xrightarrow{\overline{\mathrm{mul}_{\mathbf{A}}}} & B
\end{array} \tag{9}$$

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

Proof.
The diagram (9) leaves no choice in the definition of $\mathrm{mul}_{\bar{\mathbf{A}}}$, 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

$$h : \mathbf{A} \to \mathbf{B} \qquad g : \mathbf{B} \to \mathbf{C}$$

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

$$\overline{g \circ h} \circ \mathrm{mul}_{\overline{\mathsf{T}}\bar{\mathbf{A}}} = g \circ \bar{h} \circ \mathrm{mul}_{\overline{\mathsf{T}}\bar{\mathbf{A}}}.$$

This is done below and illustrated in Figure 5.

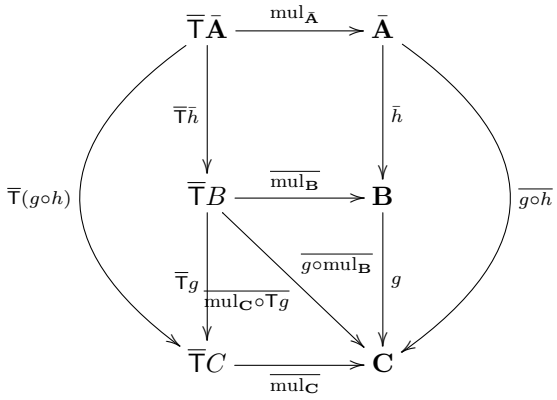


Figure 5: Multiplication in $\overline{\mathsf{T}}$ is well-defined.

$$\begin{array}{rcl}
\overline{g \circ h} \circ \mathrm{mul}_{\overline{\mathsf{T}}\bar{\mathbf{A}}} & = & \text{(by (9))} \\
\overline{\mathrm{mul}_{\mathbf{C}}} \circ \overline{\mathsf{T}}\ \overline{g \circ h} & = & \text{(by (8))} \\
\overline{\mathrm{mul}_{\mathbf{C}}} \circ \overline{\mathsf{T}}(g \circ \bar{h}) & = & \text{(because } \overline{\mathsf{T}} \text{ is a functor)} \\
\overline{\mathrm{mul}_{\mathbf{C}}} \circ \overline{\mathsf{T}}g \circ \overline{\mathsf{T}}\bar{h} & = & \text{(by definition of } \overline{\mathsf{T}}g) \\
\overline{\mathrm{mul}_{\mathbf{C}} \circ \mathsf{T}g} \circ \overline{\mathsf{T}}\bar{h} & = & \text{(because } g \text{ is a } \mathsf{T}\text{-morphism)} \\
\overline{g \circ \mathrm{mul}_{\mathbf{B}}} \circ \overline{\mathsf{T}}\bar{h} & = & \text{(by (8))} \\
g \circ \overline{\mathrm{mul}_{\mathbf{B}}} \circ \overline{\mathsf{T}}\bar{h} & = & \text{(by (9))} \\
g \circ \bar{h} \circ \circ \mathrm{mul}_{\bar{\mathbf{A}}} & &
\end{array}$$

□

So far we have proved that the operations in the statement of Theorem 11.1 are well defined, i.e. they produce T-morphism types, and that $\overline{\mathsf{T}}$ 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 $\mathbf{A} = \mathsf{T}\Sigma$.

Lemma 11.3 *Let* $\mathbf{A}$ *be a* T*-algebra, and let* $\mathrm{mul}_{\bar{\mathbf{A}}}$ *be as in Lemma 11.2. Then the following diagram commutes:*

$$\begin{array}{ccc} \overline{\mathsf{T}}\overline{\mathsf{T}}\bar{\mathbf{A}} & \xrightarrow{\mathrm{mul}_{\overline{\mathsf{T}}\bar{\mathbf{A}}}} & \overline{\mathsf{T}}\bar{\mathbf{A}} \\ \downarrow{\scriptstyle \overline{\mathsf{T}}\mathrm{mul}_{\bar{\mathbf{A}}}} & & \downarrow{\scriptstyle \mathrm{mul}_{\bar{\mathbf{A}}}} \\ \overline{\mathsf{T}}\bar{\mathbf{A}} & \xrightarrow[\mathrm{mul}_{\bar{\mathbf{A}}}]{} & \bar{\mathbf{A}} \end{array}$$

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

$$\bar{h} \circ \mathrm{mul}_{\bar{\mathbf{A}}} \circ \overline{\mathsf{T}}\mathrm{mul}_{\bar{\mathbf{A}}} = \bar{h} \circ \mathrm{mul}_{\bar{\mathbf{A}}} \circ \mathrm{mul}_{\overline{\mathsf{T}}\bar{\mathbf{A}}}.$$

Let us then fix $h : \mathbf{A} \to \mathbf{B}$ and prove the above equality. The calculation is performed below and also illustrated in Figure 6.

$$\begin{array}{rcl} \bar{h} \circ \mathrm{mul}_{\bar{\mathbf{A}}} \circ \overline{\mathsf{T}}\mathrm{mul}_{\bar{\mathbf{A}}} & = & \text{(by (9))} \\ \overline{\mathrm{mul}_{\mathbf{B}}} \circ \overline{\mathsf{T}}\bar{h} \circ \overline{\mathsf{T}}\mathrm{mul}_{\bar{\mathbf{A}}} & = & \text{(by } \overline{\mathsf{T}} \text{ applied to (9))} \\ \overline{\mathrm{mul}_{\mathbf{B}}} \circ \overline{\mathsf{T}}\ \overline{\mathrm{mul}_{\mathbf{B}}} \circ \overline{\mathsf{T}}\overline{\mathsf{T}}\bar{h} & = & \text{(because } \overline{\mathsf{T}} \text{ is a functor)} \\ \overline{\mathrm{mul}_{\mathbf{B}}} \circ \overline{\mathsf{T}}\big(\overline{\mathrm{mul}_{\mathbf{B}}} \circ \overline{\mathsf{T}}\bar{h}\big) & = & \text{(by definition of } \overline{\mathsf{T}}\bar{h}) \\ \overline{\mathrm{mul}_{\mathbf{B}}} \circ \overline{\mathsf{T}}\overline{\mathrm{mul}_{\mathbf{B}} \circ \mathsf{T}\bar{h}} & = & \text{(by (9) with the } \mathsf{T}\text{-morphism being } \mathrm{mul}_{\mathbf{B}} \circ \mathsf{T}\bar{h}) \\ \overline{\mathrm{mul}_{\mathbf{B}} \circ \mathsf{T}\bar{h}} \circ \mathrm{mul}_{\overline{\mathsf{T}}\bar{\mathbf{A}}} & = & \text{(by definition of } \overline{\mathsf{T}}\bar{h}) \\ \overline{\mathrm{mul}_{\mathbf{B}}} \circ \overline{\mathsf{T}}\bar{h} \circ \mathrm{mul}_{\overline{\mathsf{T}}\bar{\mathbf{A}}} & = & \text{(by (9))} \\ \bar{h} \circ \mathrm{mul}_{\bar{\mathbf{A}}} \circ \mathrm{mul}_{\overline{\mathsf{T}}\bar{\mathbf{A}}} & & \end{array}$$

□

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

$$\begin{array}{ccc} \overline{\mathsf{T}}\Sigma & \xrightarrow{\mathrm{unit}_{\overline{\mathsf{T}}\Sigma}} & \overline{\mathsf{T}}\overline{\mathsf{T}}\Sigma \\ \downarrow{\scriptstyle \overline{\mathsf{T}}\mathrm{unit}_{\overline{\mathsf{T}}\Sigma}} & \searrow^{\mathrm{id}_\Sigma} & \downarrow{\scriptstyle \mathrm{mul}_{\overline{\mathsf{T}}\Sigma}} \\ \overline{\mathsf{T}}\overline{\mathsf{T}}\Sigma & \xrightarrow[\mathrm{mul}_{\overline{\mathsf{T}}\Sigma}]{} & \overline{\mathsf{T}}\Sigma \end{array}$$

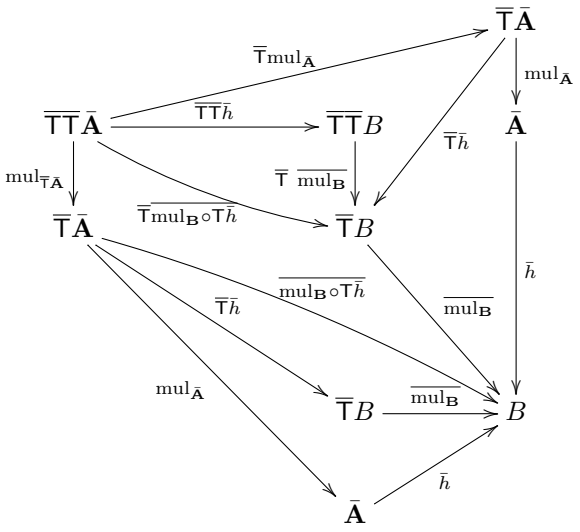


Figure 6: The four-sided faces in the diagram commute by (9), or by $\overline{\mathsf{T}}$ applied to (9). The three-sided faces in the diagram commute by the definition of $\overline{\mathsf{T}}$ on functions, or by $\overline{\mathsf{T}}$ applied to the definition of $\overline{\mathsf{T}}$ on functions.

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

$$\begin{array}{rcl}
\bar{h}\circ\mathrm{mul}_{\overline{\mathsf{T}}\Sigma}\circ\mathrm{unit}_{\overline{\mathsf{T}\mathsf{T}}\Sigma} & = & \text{(by (9))}\\
\overline{\mathrm{mul}_{\mathbf{A}}}\circ\overline{\mathsf{T}}\bar{h}\circ\mathrm{unit}_{\overline{\mathsf{T}\mathsf{T}}\Sigma} & = & \text{(by definition of } \overline{\mathsf{T}}\bar{h}\text{)}\\
\overline{\mathrm{mul}_{\mathbf{A}}\circ\mathsf{T}\bar{h}}\circ\mathrm{unit}_{\overline{\mathsf{T}\mathsf{T}}\Sigma} & = & \text{(by definition of } \mathrm{unit}_{\overline{\mathsf{T}}\Sigma}\text{)}\\
\mathrm{mul}_{\mathbf{A}}\circ\mathsf{T}\bar{h}\circ\mathrm{unit}_{\mathsf{T}\overline{\mathsf{T}}\Sigma} & = & \text{(because } \mathsf{T} \text{ is a monad)}\\
\mathrm{mul}_{\mathbf{A}}\circ\mathrm{unit}_{\mathsf{T}A}\circ\bar{h} & = & \text{(because } \mathrm{mul}_{\mathbf{A}} \text{ is the identity on units)}\\
\bar{h} & &
\end{array}$$

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

$$\begin{array}{rcl}
\bar{h}\circ\mathrm{mul}_{\overline{\mathsf{T}}\Sigma}\circ\overline{\mathsf{T}}\mathrm{unit}_{\overline{\mathsf{T}}\Sigma} & = & \text{(by (9))}\\
\overline{\mathrm{mul}_{\mathbf{A}}}\circ\overline{\mathsf{T}}\bar{h}\circ\overline{\mathsf{T}}\mathrm{unit}_{\overline{\mathsf{T}}\Sigma} & = & \text{(because } \overline{\mathsf{T}} \text{ is a functor)}\\
\overline{\mathrm{mul}_{\mathbf{A}}}\circ\overline{\mathsf{T}}(\bar{h}\circ\mathrm{unit}_{\overline{\mathsf{T}}\Sigma}) & = & \text{(by definition of } \mathrm{unit}_{\overline{\mathsf{T}}\Sigma}\text{)}\\
\overline{\mathrm{mul}_{\mathbf{A}}}\circ\overline{\mathsf{T}}(h\circ\mathrm{unit}_{\mathsf{T}\Sigma}) & = & \text{(because } \overline{\mathsf{T}} \text{ is a functor)}\\
\overline{\mathrm{mul}_{\mathbf{A}}}\circ\overline{\mathsf{T}}h\circ\overline{\mathsf{T}}\mathrm{unit}_{\mathsf{T}\Sigma} & = & \text{(by definition of } \overline{\mathsf{T}}h\text{)}\\
\overline{\mathrm{mul}_{\mathbf{A}}\circ\mathsf{T}h}\circ\overline{\mathsf{T}}\mathrm{unit}_{\mathsf{T}\Sigma} & = & \text{(by definition of } \overline{\mathsf{T}}\mathrm{unit}_{\mathsf{T}\Sigma}\text{)}\\
\overline{\mathrm{mul}_{\mathbf{A}}\circ\mathsf{T}h\circ\mathsf{T}\mathrm{unit}_{\mathsf{T}\Sigma}} & = & \text{(because } h \text{ is a } \mathsf{T}\text{-morphism)}\\
\overline{h\circ\mathrm{mul}_{\mathsf{T}\Sigma}\circ\mathsf{T}\mathrm{unit}_{\mathsf{T}\Sigma}} & = & \text{(because } \mathsf{T} \text{ is a monad)}\\
\bar{h} & &
\end{array}$$

This completes the proof that $\overline{\mathsf{T}}$ is a monad.

11.2 From a T-algebra to a $\overline{\mathsf{T}}$-algebra.

Having defined the monad $\overline{\mathsf{T}}$, it is natural to ask what are finite $\overline{\mathsf{T}}$-algebras, and what are the languages recognised by them. In this section, we discuss how every T-algebra can be transformed into a $\overline{\mathsf{T}}$-algebra. Since this transformation preserves finiteness, it gives a source of examples of finite $\overline{\mathsf{T}}$-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 14.

From A to $\bar{\mathbf{A}}$. An element of $a \in \mathbf{A}$ can 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:

$$\begin{array}{ccc} \mathbf{A} & \xrightarrow{\iota_{\mathbf{A}}} & \bar{\mathbf{A}} \\ & \searrow^{h} & \downarrow{\scriptstyle \bar{h}} \\ & & \mathbf{B} \end{array} \quad . \tag{10}$$

for every T-morphism h into a finite 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 T-morphism into a finite T-algebra. This additional assumption is true in all monads studied in this paper, at least for finitely generated T-algebras, but it can be false, e.g. with a very restrictive notion of finite T-algebra.

An algebraic structure on $\bar{\mathbf{A}}$. In Lemmas 11.2 and 11.3, we have shown that if $\mathbf{A}$ is a T-algebra, then there is a multiplication operation

$$\mathrm{mul}_{\bar{\mathbf{A}}} : \overline{\mathsf{T}}\bar{\mathbf{A}} \to \bar{\mathbf{A}}$$

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

Lemma 11.4 *If $\mathbf{A}$ is a finite T-algebra, then $\bar{\mathbf{A}}$ is isomorphic to the $\overline{\mathsf{T}}$-algebra where the universe is the universe of $\mathbf{A}$, and the multiplication is defined by*

$$\overline{\mathrm{mul}_{\mathbf{A}}} : \overline{\mathsf{T}}A \to A.$$

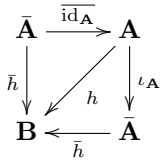
Proof.
We claim that the isomorphism is the profinite extension

$$\overline{\mathrm{id}_{\mathbf{A}}} : \bar{\mathbf{A}} \to \mathbf{A}$$

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

$$\iota_{\mathbf{A}} \circ \overline{\mathrm{id}_{\mathbf{A}}} \qquad \overline{\mathrm{id}_{\mathbf{A}}} \circ \iota_{\mathbf{A}}$$

are the identity functions on $\bar{\mathbf{A}}$ and $\mathbf{A}$ respectively. For the latter, we invoke (10). The former is explained in the following diagram

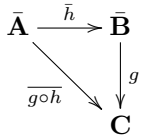


□

Lemma 11.5 *If $h : \mathbf{A} \to \mathbf{B}$ is a T-morphism, then there is a unique function*

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

which makes the following diagram commute



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

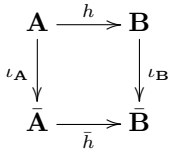
Proof.
Note that the definition of $\overline{\mathsf{T}}f$ is actually a special case of this lemma, because $\overline{\mathsf{T}}f$ is makes the diagram in the lemma commute for $\mathsf{T}f$, i.e. $\overline{\mathsf{T}}f = \overline{\mathsf{T}f}$. The lemma is proved the same way as the we proved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

The above lemma introduces a little clash of notation. If

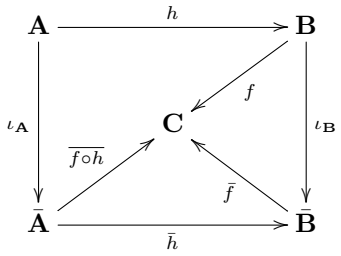
$$h : \mathbf{A} \to \mathbf{B}$$

is a T-morphism such that $\mathbf{B}$ if finite, then $\bar{h}$ has two definitions: namely the profinite extension of $\mathbf{A}$, which is of the type $\bar{\mathbf{A}} \to \mathbf{B}$, and the definition from the above lemma, which is of the type $\bar{\mathbf{A}} \to \mathbf{B}$. However, the two definitions are essentially the same mapping, because they are equal up to the isomorphism from Lemma 11.4.

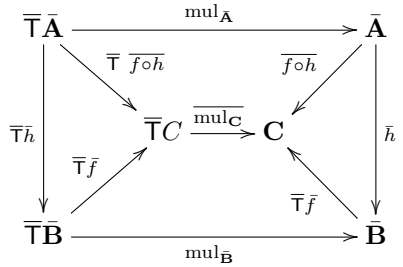
Lemma 11.6 *If $h : \mathbf{A} \to \mathbf{B}$ is a T-morphism, then $\bar{h}$ defined in Lemma 11.5 is a $\overline{\mathsf{T}}$-morphism and makes the following diagram commute:*



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 $\bar{f}$ where $f : \mathbf{B} \to \mathbf{C}$ is a T-morphism into a finite T-algebra. This is shown in the following diagram:



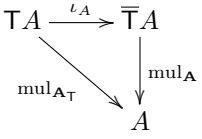
The proof that $\bar{h}$ is a $\overline{\mathsf{T}}$-morphism is in the following diagram.



The upper and lower faces commute by Lemma 11.2, the right face commutes by Lemma 11.5, and the left face commutes by applying the functor $\overline{\mathsf{T}}$ to Lemma 11.5. □

11.3 From a $\overline{\mathsf{T}}$-algebra to a T-algebra.

To go from a $\overline{\mathsf{T}}$-algebra $\mathbf{A}$ to a T-algebra, call it $\mathbf{A}_\mathsf{T}$, one keeps the same universe and defines the multiplication operation by

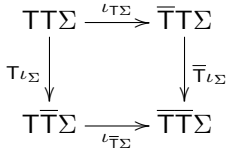


Lemma 11.7 *If $\mathbf{A}$ is a $\overline{\mathsf{T}}$-algebra, then $\mathbf{A}_\mathsf{T}$ is a T-algebra. If $h : \mathbf{A} \to \mathbf{B}$ is a $\overline{\mathsf{T}}$-morphism, then the function underlying h is a T-morphism from $\mathbf{A}_\mathsf{T}$ to $\mathbf{B}_\mathsf{T}$.*

Proof.
By Lemma 11.6 applied to $\bar{h}$ being

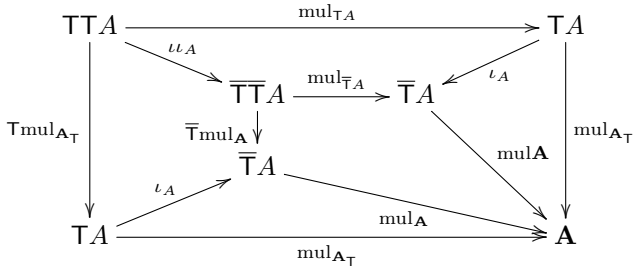
$$\mathsf{T}\iota_{\mathsf{T}\Sigma} : \mathsf{T}\mathsf{T}\Sigma \to \mathsf{T}\overline{\mathsf{T}}\Sigma$$

we see that the following diagram commutes

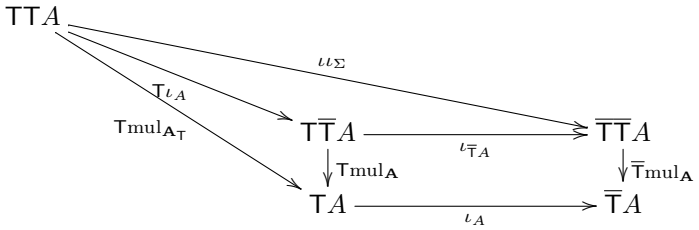


Let us write $\iota\iota_\Sigma$ for the diagonal of the above diagram.

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

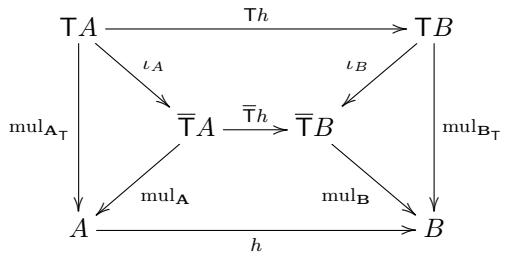


The middle face of the diagram is the assumption that $\mathbf{A}$ is a $\overline{\mathsf{T}}$-algebra. The right and bottom faces are the definition of $\mathrm{mul}_{\mathbf{A}_\mathsf{T}}$. The top face can be shown using the definition of multiplication in $\overline{\mathsf{T}}A$, and does not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



The rectangular face commutes by Lemma 11.6. The lower triangular face commutes by applying the functor T to the definition of $\mathrm{mul}_{\mathbf{A}_\mathsf{T}}$. The upper triangular face commutes because it is the definition of $\iota\iota_A$.

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



The left and right faces commute by definitions of multiplication in $\mathbf{A}_{\mathsf{T}}$ and $\mathbf{B}_{\mathsf{T}}$. The bottom face commutes by assumption that h is a $\overline{\mathsf{T}}$-morphism. The top face commutes by Lemma 11.6. □

12 Topology in the profinite monad

If $\mathbf{A}$ is a T-algebra, then define $\mathsf{rec}\mathbf{A}$ to be the subsets of the universe of $\mathbf{A}$ that are recognised by T-morphisms from $\mathbf{A}$ into finite T-algebras. Define the T-topology on a T-algebra to be the topology where the base open sets are $\mathsf{rec}\mathbf{A}$. Note that since $\mathsf{rec}\mathbf{A}$ is a Boolean algebra, it follows that the base open sets are also closed, i.e. they are clopen.

Stone dual. Consider a Boolean algebra

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

Define an ultrafilter in A to be a subset $U \subsetneq A$ which is closed under intersections, and which contains every element of A or its complement but not both. The *Stone space* associated to A is defined to be the set of all ultrafilters in the Boolean algebra, endowed with a topology where the base open sets are of the form

$$\{U : U \text{ is an ultrafilter containing } a\} \qquad \text{for } a \in A.$$

The following lemma follows straight from the definitions.

Lemma 12.1 *If $\mathbf{A}$ is a T-algebra, then $\bar{\mathbf{A}}$, seen as a topological space with $\overline{\mathsf{T}}$-topology, is homeomorphic with the Stone dual of $\mathsf{rec}\mathbf{A}$.*

12.1 The metric case

If $\mathbf{A}$ is finitely generated, and up to isomorphism there are countably many finite T-algebras, then there $\mathsf{rec}\mathbf{A}$ is countable, and therefore the T-topology can be defined by a distance. To define such a distance, one chooses some enumeration of $\mathsf{rec}\mathbf{A}$, and defines the distance between elements of $\mathbf{A}$ to be $1/n$ where n is the number of the first language in $\mathsf{rec}\mathbf{A}$ that distinguishes the two elements.

Lemma 12.2 *If* $\mathsf{rec}\mathbf{A}$ *is countable, then for every*

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

12.2 Stone algebras

Having defined the monad $\overline{\mathsf{T}}$, it is natural to ask what are finite $\overline{\mathsf{T}}$-algebras, and what are the languages recognised by them. In this section, we study a special case of finite $\overline{\mathsf{T}}$-algebras, which are obtained by taking a natural closure of a T-algebra. For this reason, these closures can be considered as the uninteresting examples of finite $\overline{\mathsf{T}}$-algebras.

Back and forth between finite T-algebras and $\overline{\mathsf{T}}$-algebras. 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. In Lemma 12.3 we show that there is a one-to-one correspondence between finite T-algebras and finite $\overline{\mathsf{T}}$-algebras that are Stone. As we illustrate in Section 14, there are interesting examples of finite $\overline{\mathsf{T}}$-algebras that are not Stone.

To go from a finite T-algebra $\mathbf{A}$ to a finite $\overline{\mathsf{T}}$-algebra, call it $\overline{\mathbf{A}}$, one keeps the same universe and applies Lemma **??** to its multiplication operation.

Lemma 12.3 *The mapping* $\mathbf{A} \mapsto \overline{\mathbf{A}}$ *is a one-to-one correspondence between finite* T*-algebras and finite* $\overline{\mathsf{T}}$*-algebras that are Stone.*

Proof.
To show that $\overline{\mathbf{A}}$ is a $\overline{\mathsf{T}}$-algebra, one needs to show that the following diagram commutes.

$$\begin{array}{ccc} \overline{\mathsf{T}}\overline{\mathsf{T}}A & \xrightarrow{\mathrm{mul}_{\overline{\mathsf{T}}A}} & \overline{\mathsf{T}}A \\ {\scriptstyle \overline{\mathsf{T}}\ \overline{\mathrm{mul}_{\mathbf{A}}}}\downarrow & & \downarrow{\scriptstyle \overline{\mathrm{mul}_{\mathbf{A}}}} \\ \overline{\mathsf{T}}A & \xrightarrow[\overline{\mathrm{mul}_{\mathbf{A}}}]{} & A \end{array} .$$

This follows immediately from (9). The multiplication operation is easily seen to be continuous.

To prove that the correspondence is one-to-one, it suffices to show that if $\mathbf{A}$ is Stone, then $\mathbf{A} = \overline{\mathbf{A}_{\mathsf{T}}}$. This is true by Lemma **??**. □

Lemma 12.4 *The mapping*

$$h : \mathsf{T}\Sigma \to \mathbf{A} \qquad \mapsto \qquad \bar{h} : \overline{\mathsf{T}}\Sigma \to \overline{\mathbf{A}}$$

is a one-to-one correspondence between T*-morphisms into finite* T*-algebras and continuous* $\overline{\mathsf{T}}$*-morphisms into finite* $\overline{\mathsf{T}}$*-algebras that are Stone.*

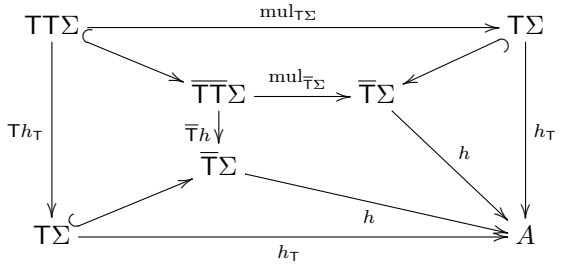
Proof.
That $\bar{h}$ is a $\overline{\mathsf{T}}$-morphism follows from the definition of $\overline{\mathbf{A}}$ and the diagram in Theorem **??**. For the converse implication, consider a continuous $\overline{\mathsf{T}}$-morphism

$$h : \overline{\mathsf{T}}\Sigma \to \mathbf{A}.$$

and define

$$h_{\mathsf{T}} : \mathsf{T}\Sigma \to \mathbf{A}_{\mathsf{T}}.$$

To prove that h_{T} is a T-morphism, we use the same diagram as in the proof of Lemma 12.3:



Since h is continuous and extends h_{T}, it follows that $h = \overline{h_{\mathsf{T}}}$ by Lemma **??**. □

Clopen languages. As shown in Lemma 12.3, there is a one-to-one correspondence between finite T-algebras and $\overline{\mathsf{T}}$-algebras which are Stone. This correspondence extends to languages, as stated in the following theorem. In the lemma, for $L \subseteq \mathbf{A}$, we write $\bar{L} \subseteq \bar{\mathbf{A}}$ for the closure, in the $\overline{\mathsf{T}}$-topology, of the image of L under the $\iota_{\mathbf{A}}$.

Theorem 12.5 *Let* $\mathbf{A}$ *be a* T*-algebra. The following conditions are equivalent for a subset* $L \subseteq \bar{\mathbf{A}}$*:*

1. *L is clopen in the $\overline{\mathsf{T}}$-topology of $\bar{\mathbf{A}}$;*
2. *L is recognised by a continuous $\overline{\mathsf{T}}$-morphism into a finite Stone $\overline{\mathsf{T}}$-algebra.*
3. *L is equal to $\bar{K}$ for some $K \in \mathsf{rec}\mathbf{A}$.*

(**Proof.**
1 implies 2. Like any clopen set in a compact space, L is a finite union of base open sets, i.e. sets of the form

$$\bar{h}^{-1}(a) \qquad \text{for some } \mathsf{T}\text{-morphism } h : \mathsf{T}\Sigma \to \mathbf{A}.$$

By Lemma 12.4, every base open set is recognised by a continuous morphisms into finite Stone $\overline{\mathsf{T}}$-algebras. Sets recognised by such morphisms are closed under finite Boolean combinations.

2 implies 3. By Lemma 12.4, if L is recognised by a continuous $\overline{\mathsf{T}}$-morphism into a finite Stone $\overline{\mathsf{T}}$-algebra, then L is recognised by

$$\bar{h} : \overline{\mathsf{T}}\Sigma \to \overline{\mathbf{A}}$$

for some T-morphism $h : \mathsf{T}\Sigma \to \mathbf{A}$. To prove the implication, it suffices to show

$$\overline{h^{-1}(A_0)} = \bar{h}^{-1}(A_0) \qquad \text{for every } A_0 \subseteq A, \tag{11}$$

which follows from the denseness of the (injective image) of $\mathsf{T}\Sigma$ in $\overline{\mathsf{T}}\Sigma$.

3 implies 1. Follows from (11). □

13 Two applications of profinite monads

In this section we give two applications of profinite objects. The application result says that profinite objects can be used to describe lattices of languages, although the theorem is not effective in any way. The second application says that although the profinite object makes sense for any class of languages, e.g. for decidable languages, only recognisable languages can be used if one wants the algebraic structure of the monad to be consistent with the topology.

13.1 Defining classes by implications and identities

One of the advantages of profinite objects is that they can be used to describe lattices of languages, including the special case of pseudovarieties of languages. The results of this section a generalisation of Theorem 5.2 in [9] from monoids to arbitrary monads over sets.

Profinite implications. A *profinite implication* over an alphabet Σ is an expression of the form $w \to v$, where $w, v \in \overline{\mathsf{T}}\Sigma$. A T-recognisable T-language $L \subseteq \mathsf{T}\Sigma$ satisfies the implication if $w \in \bar{L}$ implies $v \in \bar{L}$. A set of T-recognisable subsets of $\mathsf{T}\Sigma$ is said to be defined by a set of profinite implications if it contains exactly the languages that satisfy all implications from the set. In the following lemma, a lattice of languages is a family of languages that contains the empty and full languages, and which is closed under finite unions and finite intersections.

Theorem 13.1 *Let Σ be a finite alphabet, and let $\mathcal{L}$ be a family of T-recognisable subsets of $\mathsf{T}\Sigma$. Then $\mathcal{L}$ is a lattice if and only if it is definable by profinite implications.*

Proof.
This theorem is true for Stone duals of Boolean algebras in general, and does not need that assumptions that profinite objects are obtained from recognisable subsets of an algebra. The more general result is as follows. Consider a Boolean algebra A and its Stone dual $\bar{A}$, i.e. the topological space of ultrafilters on the

Boolean algebra. Then a subset of A is a lattice if and only if it is definable by implications in $\bar{A}$. The result is well known [], but we give a proof below to make the text self-contained.

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 $\mathcal{L} \subseteq A$. We claim that $\mathcal{L}$ is defined by the set of implications:

$$\{x \to y : \text{for every } a \in \mathcal{L}, \text{ if } x \in \bar{a} \text{ then } y \in \bar{a}\}. \tag{12}$$

By definition, every element of $\mathcal{L}$ satisfies the implications above. Let then $a \in A$ be such that a satisfies all the implications above. To finish the proof of the theorem, we need to show $a \in \mathcal{L}$.

We first claim that every $x \in \bar{a}$ satisfies

$$x \in \bar{b} \subseteq \bar{a} \qquad \text{for some } b \in \mathcal{L} \tag{13}$$

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

$$\bigcap_{x \in b \in \mathcal{L}} \bar{b}.$$

It is easy to see that $[x]$ is the set of all $y \in \bar{A}$ such that the implication $x \to y$ belongs to the set (12). By assumption that a satisfies all these 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 $x \in b \in \mathcal{L}$. Furthermore, the intersection is nonempty, because $\mathcal{L}$ contains the greatest element of the Boolean algebra, being a lattice. Because $\mathcal{L}$ is closed under finite intersections, it follows that $[x] = \bar{b}$ for some b with $x \in b \in \mathcal{L}$. Together with $[x] \subseteq \bar{a}$, this proves (13).

From (13) it follows that $\bar{a}$ is the union of all $\bar{b}$ ranging over $b \in \mathcal{L}$ such that $\bar{b} \subseteq \bar{a}$. By compactness, the union can be made finite, and by closure of $\mathcal{L}$ under finite union, it follows that there is some $b \in \mathcal{L}$ such that $\bar{a} = \bar{b}$. Finally, since a, b are in the Boolean algebra, it follows that $a = b$. $\square$

Defining pseudovarieties by identities As mentioned in its proof, Theorem 13.1 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.

Define a *profinite identity* to be a pair of profinite objects $w, v \in \overline{\mathsf{T}}X$, where X is a finite set of variables. We say that a T-language $L \subseteq \mathsf{T}\Sigma$ satisfies such an identity if

$$\mathsf{T}f(w) \in \overline{p^{-1}L} \qquad \text{iff} \qquad \mathsf{T}f(v) \in \overline{p^{-1}L}$$

holds for every unary polynomial p over $\mathsf{T}\Sigma$ and every valuation $f : X \to \overline{\mathsf{T}}\Sigma$.

Corollary 13.2 *Let $\mathcal{L}$ be class of recognisable T-languages. Then $\mathcal{L}$ is a pseudovariety if and only if it is defined by a set of profinite identities.*

13.2 Uniform continuity of term operations

As mentioned in its proof, Theorem 13.1 does not use any assumptions on recognisability. For instance, if the topology in $\overline{\mathsf{T}}\Sigma$ were defined using a different Boolean algebra of subsets, such as those recognised by some monad variant of Turing machines, then the theorem would still be true. In this section, we show that recognisable languages are special in their topology makes all term operations uniformly continuous. The result in this section is a generalisation of Theorem 4.1 in [10] from monoids to a certain class of monads over sets.

Uniformly continuous operations. We begin by defining the notion of a uniformly continuous operation in a Stone dual. Let A be a set and let

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

be a countable family of subsets of A, along with some enumeration. We assume countability so that the topology of the Stone dual is definable by a distance, and therefore uniform continuity can be discussed. 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

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

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

$$f : A^X \to A$$

with X finite 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

$$v_1 \equiv_{\mathcal{M}} v_2 \quad \text{implies} \quad f(v_1) \equiv_{\mathcal{K}} f(v_2) \qquad \text{for every } v_1, v_2 \in A^X,$$

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.

Example 9. Let Σ be a finite alphabet, let X be a set of finite monoids (e.g. all finite monoids, or all aperiodic monoids), and consider the $\mathcal{L}$-distance on the monoid Σ^* where $\mathcal{L}$ is all subsets of Σ^* recognised by monoids in X. We claim that concatenation 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

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

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 monoids that are used to recognise the

languages from $\mathcal{K}$. The family $\mathcal{M}$ is finite because there are finitely many possible monoid morphisms from Σ^* to a finite set of finite monoids. The same solution works for other operations in Σ^* that can be built using concatenation. □

Example 10. 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 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 monoid.

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

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

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 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

$$w_1 \equiv_{\mathcal{K}} v_2 \qquad \text{and} \qquad w_1^{-1}L \neq w_2^{-1}L.$$

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. □

The two examples above are essentially Theorem 4.1 of [10]. The goal of this section is to generalise that result to algebras over abstract monads. The role of concatenation will be played by an arbitrary term operation on the universe of a T-algebra $\mathbf{A}$, i.e. an operation of the form

$$f : A^X \qquad \mapsto \qquad [\![p]\!](f) \in A$$

where $p \in \mathsf{T}X$ for some finite set of variables. 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 is straightforward. There is one additional assumption in our generalisation, which will require some more definitions. In the special case of monoids, this additional assumption boils down to the observation that the syntactic congruence is obtained already by looking at unary polynomials of the form $u \mapsto wuv$.

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. Let $\mathbf{A}$ be a T-algebra. We write $\mathsf{pol}_1\mathbf{A}$ for the unary polynomials in an algebra $\mathbf{A}$; this notation makes sense because we use unsorted sets. A set $P \subseteq \mathsf{pol}_1\mathbf{A}$ is called *observationally complete for* $\mathbf{A}$ if the following conditions are equivalent for every $a, b \in A$ and every subset $L \subseteq A$ of the universe:

$$w \in p^{-1}L \quad \text{iff} \quad w' \in p^{-1}L \qquad \text{for every } p \in \mathsf{pol}_1\mathbf{A} \tag{14}$$

$$w \in p^{-1}L \quad \text{iff} \quad w' \in p^{-1}L \qquad \text{for every } p \in \mathsf{pol}_1\mathbf{A} \cap P. \tag{15}$$

Recall that the condition in (14) is the equivalence relation defined in the proof of the Syntactic Morphism Theorem.

Example 11. Consider the monad of finite words where algebras are monoids. Call a unary polynomial nonduplicating if it uses its variable exactly once. It is not difficult to show that the unary nonduplicating polynomials are observationally complete in every monoids. This holds in all finitary monads mentioned in this paper, in particular in for the monad of ultimately periodic ∞-words. □

Finite covers. We say that $w \in \mathsf{T}X$ is *covered* by $v \in \mathsf{T}Y$ if

$$w = \mathrm{mul}_{\mathsf{T}X} f(v) \qquad \text{for some } f : Y \to \mathsf{T}X.$$

A set W is said to have a finite cover, if there is a finite set V such that every $w \in W$ is covered by some $v \in V$. We assume the convention that the variable in unary polynomials is called 1. We will be interested in finite covers for subsets

$$W \subseteq \mathsf{pol}_1\mathbf{A} = \mathsf{T}(A \sqcup \{1\})$$

Example 12. Consider the monad of finite words. Using the convention that the variable in unary polynomials is called 1 (not to be confused with the monoid neutral element), a nonduplicating unary polynomial in a monoid $\mathbf{A}$ is a word of the form

$$w1v \qquad \text{with } w, v \in A^*$$

Therefore, every nonduplicating unary polynomial is covered by $a1b \in \{a, 1, b\}^*$.

Consider the monad of ultimately periodic ∞-words. A nonduplicating unary polynomial in this monad is a word of one of the forms

$$v1w \qquad \text{or} \qquad v(1w)^\omega \qquad \text{with } v, w \in A^*.$$

Therefore, every nonduplicating unary polynomial in this monad is covered by one of $axb, a(xb)^\omega \in \{a, x, b\}^*$.

Summing up, in both monads from this example, every algebra has a finite cover for the set of nonduplicating unary polynomials. □

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.

Theorem 13.3 *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* $\mathsf{der}\mathcal{L}$*-distance, where* $\mathsf{der}\mathcal{L}$ *is all derivatives of* $\mathcal{L}$*.*
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 12, 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 9, 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 V, 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 (14) has infinite index, and therefore the equivalence relation defined in (15) as applied to P has infinite index. For v in the finite cover V, define $\sim_v$ to be the relation as in (15) but with polynomials restricted to those that are covered by v, i.e. $\sim_v$ identifies $a, b \in A$ if

$$a \in p^{-1}L \qquad \text{iff} \qquad b \in p^{-1}L \qquad \text{for every } p \in \mathsf{pol}_1 \text{ covered by } v.$$

Since the relation (15) has infinite index and is the intersection of the finitely many relations $\sim_v$, there must be some $v \in V$ such that $\sim_v$ has infinite index. Recall that v is an element of $\mathsf{T}Y$ for some finite set of variables Y. We claim that the term operation corresponding to v, i.e.

$$f \in A^Y \qquad \mapsto \qquad \mathrm{mul}_{\mathbf{A}}(\mathsf{T}f(v)) \in A$$

is not uniformly continuous. To prove this, consider some finite set $\mathcal{K} \subseteq \mathcal{L}$. Because the index of $\sim_v$ is infinite and the index of $\equiv_{\mathcal{K}}$ is finite, there must be some $a, b \in A$ such that

$$a \not\sim_v b \qquad \text{and} \qquad a \equiv_{\mathcal{K}} b.$$

Unraveling the definition of $\sim_v$, this means that there is some $p \in \mathsf{pol}_1\mathbf{A}$ that is covered by v such that

$$[\![p]\!](a) \not\equiv_{\{L\}} [\![p]\!](b).$$

Unraveling the definition of covering, this means that

$$[\![p]\!](a) = [\![v]\!](f_a) \qquad \text{and} \qquad [\![p]\!](b) = [\![v]\!](f_b)$$

holds for some $f_a, f_b : Y \to A$ which agree on all arguments except some $y_1 \in Y$, and which map y_1 to the $\equiv_{\mathcal{K}}$-equivalent elements a, b respectively. Summing up, for every finite set $\mathcal{K} \subseteq \mathcal{L}$ we have found valuations f_a, f_b such that

$$f_a \equiv_{\mathcal{K}} f_b \qquad \text{but} \qquad [\![v]\!](a) \not\equiv_{\{L\}} [\![v]\!](b),$$

and therefore the term operation corresponding to v is not uniformly continuous. □

14 Profinite words

In this section, we illustrate the profinite monad construction from Section 11 in the special case of words. Consider the monad $\Sigma \mapsto \Sigma^*$ that was used for finite words. Let us denote by $\Sigma \mapsto \Sigma^{\bar{*}}$ the profinite version of this monad, as defined in Section 11. In particular, $\Sigma^{\bar{*}}$ is a monoid thanks to Lemma 11.7, and $\Sigma^{\bar{*}}$ is has a topology which makes it a Stone space. An element of $\Sigma^{\bar{*}}$ is called a *profinite word* over the alphabet Σ.

As discussed in Section 12.2, for every finite monoid M, one can extend its multiplication operation to an operation

$$\text{mul} : M^{\bar{*}} \to M$$

which makes it into a finite $\bar{*}$-algebra, and which is continuous assuming the discrete topology on M. This construction is one source of finite $\bar{*}$-algebras; but it does not really introduce anything new with respect to finite monoids. In this section we give an example of a finite $\bar{*}$-algebra that is not obtained this way.

14.1 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

$$h : \Sigma^* \to \{0, 1, \ldots, n\}$$

is the monoid morphism which counts the number of letters in Γ up to threshold n. The notion of exactly n letters is defined by saying at least n 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 14.1 *The set of profinite words in $2^{\bar{*}}$ which have unboundedly many ones is $\bar{*}$-recognisable.*

Proof.
The set is recognised by a $\bar{*}$-morphism

$$h : 2^{\bar{*}} \to \mathbf{A}$$

where $\mathbf{A}$ is the finite $\bar{*}$-algebra defined as follows. The universe A has three elements, call them $0, 1$ and ∞, which represent profinite words that have no ones, a bounded number of ones, and an unboundedly number of ones respectively. The multiplication operation

$$\mathrm{mul}_{\mathbf{A}} : A^{\bar{*}} \to A$$

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:

$$\begin{array}{ccc} (A^{\bar{*}})^{\bar{*}} & \xrightarrow{\mathrm{mul}_{A^{\bar{*}}}} & A^{\bar{*}} \\ {\scriptstyle (\mathrm{mul}_{\mathbf{A}})^{\bar{*}}}\downarrow & & \downarrow{\scriptstyle \mathrm{mul}_{\mathbf{A}}} \\ A^{\bar{*}} & \xrightarrow[\mathrm{mul}_{\mathbf{A}}]{} & A \end{array}$$

Consider a word $w \in (A^{\bar{*}})^{\bar{*}}$. Our goal is to show that

$$\mathrm{mul}_{\mathbf{A}}((\mathrm{mul}_{\mathbf{A}})^{\bar{*}}(w)) = \mathrm{mul}_{\mathbf{A}}(\mathrm{mul}_{A^{\bar{*}}}(w)). \tag{16}$$

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 (16) holds, because both sides are equal to ∞. Consider first the left side. Let

  $$h_n : A^* \to \{0, \ldots, n\}$$

  be the monoid morphism that counts the number of nonzero letters. Our assumption on w says that $(\overline{h}_1)^{\bar{*}}(w)$ has unboundedly many ones. Since the image of h_1 is a subset of A, it makes sense to compare values of $\overline{h}_1$ with values of $\mathrm{mul}_{\mathbf{A}}$, in particular the following observation is easy to get:

  $$\overline{h}_1(v) \ \leq \ \mathrm{mul}_{\mathbf{A}}(v) \qquad \text{for every } v \in A^{\bar{*}}.$$

  As in the proof of Lemma 6.2, a binary relation $R \subseteq X \times Y$ lifts to a relation $R^{\mathsf{T}} \subseteq \mathsf{T}X \times \mathsf{T}Y$. Apply this construction to the natural ordering on A, and call $\leq$ the resulting relation on $A^{\bar{*}}$. As we have observed,

  $$(\overline{h}_1)^{\bar{*}}(w) \ \leq^{\bar{*}} \ (\mathrm{mul}_{\mathbf{A}})^{\bar{*}}(w).$$

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 $\mathrm{mul}_{\mathbf{A}}$ to ∞. It is not difficult to see that the mapping $\mathrm{mul}_{\mathbf{A}}$ is monotone with respect to $\leq$, and therefore the left side of the equality in (16) is ∞.

To prove that the right side of the equality in (16) is also ∞, by definition of $\mathrm{mul}_{\mathbf{A}}$ we need to show that every n satisfies

$$\overline{h_n}(\mathrm{mul}_{A^{\bar{*}}}(w)) = n.$$

Let mul_n be the multiplication operation in the monoid $\{0, \ldots, n\}$. Theorem **??** says that

$$\begin{array}{ccc} A^{\bar{*}\bar{*}} & \xrightarrow{\mathrm{mul}_{A^{\bar{*}}}} & \overline{\mathsf{T}}A \\ \Big\downarrow{\scriptstyle \overline{h_n}^{\bar{*}}} & & \Big\downarrow{\scriptstyle \overline{h_n}} \\ \overline{\mathsf{T}}\{0,\ldots,n\} & \xrightarrow{\overline{\mathrm{mul}_n}} & \{0,\ldots,n\} \end{array}$$

To prove that the right side of the equality in (16) is also ∞, from the definition of $\mathrm{mul}_{\mathbf{A}}$ 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 $(\overline{h_n})^{\bar{*}}(w)$ 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 $0^{\bar{*}}$. We begin with a straightforward lemma, which uses the monoid structure of profinite words that was described in Lemma 11.7. If $a \in \Sigma$, then let us write $[a]_\Sigma \in \Sigma^{\bar{*}}$ for the corresponding profinite word.

Lemma 14.2 *If $w \in \Sigma^{\bar{*}}$ has a bounded number of letters in $\Gamma \subseteq \Sigma$ then it admits a finite decomposition*

$$w = w_0 \cdot [a_1]_\Sigma \cdot w_1 \cdots w_{n-1} \cdot [a_n]_\Sigma \cdot w_n \qquad \text{with } w_i \in (\Sigma - \Gamma)^{\bar{*}} \text{ and } a_i \in \Gamma.$$

By applying Lemma 14.2, there is a decomposition

$$w = w_0 \cdot [a_1]_{A^{\bar{*}}} \cdot w_1 \cdots w_{n-1} \cdot [a_n]_{A^{\bar{*}}} \cdot w_n$$

where $w_i \in (0^{\bar{*}})^{\bar{*}}$, $a_i \in A^{\bar{*}} - 0^{\bar{*}}$, and the $\cdot$ operation is in the monoid $(A^{\bar{*}})^{\bar{*}}$. Lemma 11.7 implies that

$$\mathrm{mul}_{A^{\bar{*}}}(w) = \mathrm{mul}_{A^{\bar{*}}}(w_0) \cdot a_1 \cdot \mathrm{mul}_{A^{\bar{*}}}(w_1) \cdots \mathrm{mul}_{A^{\bar{*}}}(w_{n-1}) \cdot a_n \cdot \mathrm{mul}_{A^{\bar{*}}}(w_n)$$

where the $\cdot$ operation is in the monoid $A^{\bar{*}}$. Since $\mathrm{mul}_{\mathbf{A}}$ is a monoid morphism, and it maps words in $0^{\bar{*}}$ to the identity in A, it follows that

$$\mathrm{mul}_{\mathbf{A}}(\mathrm{mul}_{A^{\bar{*}}}(w)) = \mathrm{mul}_{\mathbf{A}}(a_1) \cdots \mathrm{mul}_{\mathbf{A}}(a_n).$$

Let us now consider $\mathrm{mul}_{\mathbf{A}}((\mathrm{mul}_{\mathbf{A}})^{\bar{*}}(w))$. Lemma 11.7 says that $(\mathrm{mul}_{\mathbf{A}})^{\bar{*}}$ is a monoid morphism, and therefore

$$\mathrm{mul}_{\mathbf{A}}{}^{\bar{*}}(w) = \mathrm{mul}_{\mathbf{A}}{}^{\bar{*}}(w_0) \cdot \mathrm{mul}_{\mathbf{A}}{}^{\bar{*}}([a_1]_{A^{\bar{*}}}) \cdot \mathrm{mul}_{\mathbf{A}}{}^{\bar{*}}(w_1) \cdots \mathrm{mul}_{\mathbf{A}}{}^{\bar{*}}(w_{n-1}) \cdot \mathrm{mul}_{\mathbf{A}}{}^{\bar{*}}([a_n]_{A^{\bar{*}}}) \cdot \mathrm{mul}_{\mathbf{A}}{}^{\bar{*}}(w_n).$$

By the axioms of a monad, we have

$$\mathrm{mul}_{\mathbf{A}}{}^{\bar{*}}([a_i]_{A^{\bar{*}}}) = [\mathrm{mul}_{\mathbf{A}}(a_i)]_A.$$

Each word $\mathrm{mul}_{\mathbf{A}}{}^{\bar{*}}(w_i)$ in the decomposition of $\mathrm{mul}_{\mathbf{A}}{}^{\bar{*}}(w)$ belongs to $0^{\bar{*}}$. Therefore, because $\mathrm{mul}_{\mathbf{A}}$ is a monoid morphism that maps $0^{\bar{*}}$ to the identity, we get

$$\mathrm{mul}_{\mathbf{A}}(\mathrm{mul}_{\mathbf{A}}{}^{\bar{*}}(w)) = \mathrm{mul}_{\mathbf{A}}([\mathrm{mul}_{\mathbf{A}}(a_i)]_A) \cdots \mathrm{mul}_{\mathbf{A}}([\mathrm{mul}_{\mathbf{A}}(a_n)]_A).$$

The result follows because $\mathrm{mul}_{\mathbf{A}}([a]_A) = a$ holds for every $a \in A$.

This completes the proof that $\mathrm{mul}_{\mathbf{A}} : A^{\bar{*}} \to A$ is a $\bar{*}$-morphism. □

Let us define MSO+inf to by applying the abstract notion of MSO defined in Section 6.1, with the unique predicate being the language of unboundedly many ones. This class of languages was considered in [17]. From Lemma 6.2 it follows that MSO+inf contains only $\bar{*}$-recognisable languages. It is not clear if it contains all $\bar{*}$-recognisable languages. It is an open problem whether or not emptiness is decidable for MSO+inf. As shown in [17], decidability of MSO+inf on profinite words would imply decidability of MSO+U, which is an extension of MSO on infinite words, and whose decidability is an open problem.

References

[1] Nicolas Bedon and Chloé Rispal. Schützenberger and eilenberg theorems for words on linear orderings. *J. Comput. Syst. Sci.*, 78(2):517–536, 2012.

[2] Mikolaj Bojanczyk. Nominal monoids. *Theory Comput. Syst.*, 53(2):194–222, 2013.

[3] 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.

[4] Mikołaj Bojańczyk and Tomasz Idziaszek. Algebra for infinite forests with an application to the temporal logic EF. In *CONCUR*, pages 131–145, 2009.

[5] 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.

[6] 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.

[7] 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.

[8] S. Eilenberg. *Automata, languages, and machines. Vol. A.* 1974.

[9] 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.

[10] Mai Gehrke, Serge Grigorieff, and Jean-Éric Pin. A topological approach to recognition. In *Automata, Languages and Programming*, pages 151–162. Springer, 2010.

[11] Dominique Perrin and Jean-Éric Pin. *Infinite Words: Automata, Semigroups, Logic and Games*. Elsevier, 2004.

[12] A. Potthoff. First-order logic on finite trees. *Lecture Notes in Computer Science*, 915:125–139, 1995.

[13] Saharon Shelah. The monadic theory of order. *The Annals of Mathematics*, 102(3):379–419, 1975.

[14] Magnus Steinby. A theory of tree language varieties. In *Tree Automata and Languages*, pages 57–82. 1992.

[15] 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.

[16] Wolfgang Thomas. Languages, automata and logics. Technical Report 9607, Institut für Informatik und Praktische Mathematik, Christian-Albsechts-Universität, Kiel, Germany, 1996.

[17] Szymon Toruńczyk. *Languages of profinite words and the limitedness problem*. PhD thesis, University of Warsaw, 2011.

[18] 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.

[19] Thomas Wilke. An algebraic theory for regular languages of finite and infinite words. *Int. J. Alg. Comput.*, 3:447–489, 1993.

[20] Y. I. Yanov and A. A. Muchnik. Existence of k-valued closed classes without a finite basis. *Dokl. Akad. Nauk.*, 127:44–46, 1959.