

Algebraic language theory

Mikołaj Bojańczyk

April 15, 2020

The latest version can be downloaded from:

<https://www.mimuw.edu.pl/bojan/2019-2020/algebraic-language-theory-2020>

Contents

<i>Preface</i>	<i>page v</i>
Part I Words	1
1 Semigroups, monoids and their structure	3
1.1 Recognising languages	8
1.2 Green's relations and the structure of finite semigroups	11
1.3 The Factorisation Forest Theorem	19
2 Logics on finite words, and the corresponding monoids	29
2.1 All monoids and monadic second-order logic	30
2.2 Aperiodic semigroups and first-order logic	37
2.3 Suffix trivial semigroups and linear logic with F only	47
2.4 Infix trivial semigroups and piecewise testable languages	50
2.5 Two-variable first-order logic	57
3 Infinite words	62
3.1 Determinisation of Büchi automata for ω -words	62
3.2 Countable words and \circ -semigroups	72
3.3 Finite representation of \circ -semigroups	79
3.4 From \circ -semigroups to MSO	89
Part II Beyond words	99
4 Monads	101
4.1 Monads and their Eilenberg-Moore algebras	102
4.2 A zillion examples	110
4.3 Syntactic algebras	120
4.4 The Eilenberg Pseudo-Variety Theorem	131

<i>Bibliography</i>	141
<i>Author index</i>	145
<i>Subject index</i>	146

Preface

These are lecture notes on the algebraic approach to regular languages. The classical algebraic approach is for finite words; it uses semigroups instead of automata. However, the algebraic approach can be extended to structures beyond words, e.g. infinite words, or trees or graphs.

PART ONE

WORDS

1

Semigroups, monoids and their structure

In this chapter, we define semigroups and monoids, and show how they can be used to recognise languages of finite words.

Definition 1.1 (Semigroup). A *semigroup* consists of an underlying set S together with a binary product operation

$$(a, b) \mapsto ab,$$

which is associative in the sense that

$$a(bc) = (ab)c \quad \text{for all } a, b, c \in S.$$

The definition says that the order of evaluation in a semigroup is not important, i.e. that different ways of bracketing a sequence of elements in the semigroup will yield the same result as far as the semigroup product is concerned. For example,

$$((ab)c)(d(e f)) = (((ab)c)d)e)f.$$

Therefore, it makes sense to omit the brackets and write simply

$$abcdef.$$

This means that the product operation in the semigroup can be seen as an operation of type $S^+ \rightarrow S$, i.e. it is defined not just on pairs of semigroup elements, but also on finite nonempty words consisting of semigroup elements.

A *semigroup homomorphism* is a function between semigroups that preserves the structure of semigroups, i.e. a function

$$h : \underbrace{S}_{\text{semigroup}} \rightarrow \underbrace{T}_{\text{semigroup}}$$

which is consistent with the product operation in the sense that

$$h(a \cdot b) = h(a) \cdot h(b),$$

where the semigroup product on the left is in S , and the semigroup product on the right is in T . An equivalent definition of a semigroup homomorphism is requiring the following diagram to commute:

$$\begin{array}{ccc} S^+ & \xrightarrow{h^+} & T^+ \\ \text{product in } S \downarrow & & \downarrow \text{product in } T \\ S & \xrightarrow{h} & T \end{array}$$

In the above, h^+ is the natural lifting of h to words.

A *monoid* is the special case of a semigroup where there is an identity element, denoted by $1 \in S$, which satisfies

$$1a = a1 \quad \text{for all } a \in S.$$

The identity element, if it exists, must be unique. This is because if there are two candidates for the identity, then taking their product reveals the true identity. The product operation in a monoid can be thought of as having type $S^* \rightarrow S$, since with the empty word ε being mapped to 1. A *monoid homomorphism* is a semigroup homomorphism that preserves the identity element. In terms of commuting diagrams, a monoid homomorphism is a function which makes the following diagram commute:

$$\begin{array}{ccc} S^* & \xrightarrow{h^*} & T^* \\ \text{product in } S \downarrow & & \downarrow \text{product in } T \\ S & \xrightarrow{h} & T \end{array}$$

Clearly there is a pattern behind the diagrams. This pattern will be explored in the second part of this book, when talking about monads.

Example 1.2. Here are some examples of monoids and semigroups.

- (1) If Σ is a set, then the set Σ^+ of nonempty words over Σ , equipped with concatenation, is a semigroup, called the *free¹ semigroup over generators Σ* . The *free monoid* is the set Σ^* of possibly empty words.

¹ The reason for this name is the following universality property. The free semigroup is generated by Σ , and it is the biggest semigroup generated by Σ in the following sense. For every semigroup S that is generated by Σ , there exists a (unique) surjective semigroup homomorphism $h : \Sigma^+ \rightarrow S$ which is the identity on the Σ generators.

- (2) Every group is a monoid.
 (3) For every set Q , the set of all functions $Q \rightarrow Q$, equipped with function composition, is a monoid. The monoid identity is the identity function.
 (4) For every set Q , the set of all binary relations on Q is a monoid, when equipped with relational composition

$$a \circ b = \{(p, q) : \text{there is some } r \in Q \text{ such that } (p, r) \in a \text{ and } (r, q) \in b\}.$$

The monoid identity is the identity function. The monoid from the previous item is a sub-monoid of this one, i.e. the inclusion map is a monoid homomorphism.

- (5) Here are all semigroups of size two, up to semigroup isomorphism:

$$\underbrace{(\{0, 1\}, +)}_{\text{addition mod 2}} \quad (\{0, 1\}, \min) \quad \underbrace{(\{0, 1\}, \pi_1)}_{\text{product } ab \text{ is } a} \quad \underbrace{(\{0, 1\}, \pi_2)}_{\text{product } ab \text{ is } b} \quad \underbrace{(\{0, 1\}, (a, b) \mapsto 1)}_{\text{all products are 1}}$$

The first two are monoids.

Compositional functions. Semigroup homomorphisms are closely related with functions that are compositional in the sense defined below. Let S be a semigroup, and let X be a set (without a semigroup structure). A function

$$h : S \rightarrow X$$

is called *compositional* if for every $a, b \in S$, the value $h(a \cdot b)$ is uniquely determined by the values $h(a)$ and $h(b)$. If X has a semigroup structure, then every semigroup homomorphism $S \rightarrow X$ is a compositional function. The following lemma shows that the converse is also true for surjective functions.

Lemma 1.3. *Let S be a semigroup, let X be a set, and let $h : S \rightarrow X$ be a surjective compositional function. Then there exists (a unique) semigroup structure on X which makes h into a semigroup homomorphism.*

Proof Saying that $h(a \cdot b)$ is uniquely determined by $h(a)$ and $h(b)$, as in the definition of compositionality, means that there is a binary operation \circ on X , which is not yet known to be associative, that satisfies

$$h(a \cdot b) = h(a) \circ h(b) \quad \text{for all } a, b \in S. \quad (1.1)$$

The semigroup structure on X uses \circ as the semigroup operation. It remains to prove associativity of \circ . Consider three elements of X , which can be written as $h(a), h(b), h(c)$ thanks to the assumption on surjectivity of h . We have

$$(h(a) \circ h(b)) \circ h(c) \stackrel{(1.1)}{=} (h(ab)) \circ h(c) \stackrel{(1.1)}{=} h(abc).$$

The same reasoning shows that $h(a) \circ (h(b) \circ h(c))$ is equal to $h(abc)$, thus establishing associativity. \square

Commuting diagrams. We finish this section with by an alternative description of semigroups which uses commuting diagrams. We include this description, because similar descriptions will be frequently used in this book, e.g. for generalisations of semigroups for infinite words, so we want to start using it as early as possible.

The binary product operation in a semigroup S can be extended to a general operation of type $S^+ \rightarrow S$. The following lemma explains, using commuting diagrams, which operations arise this way.

Lemma 1.4. *An operation $\pi : S^+ \rightarrow S$ arises from some semigroup operation on S if and only if the following two diagrams commute:*

$$\begin{array}{ccc}
 S & & (S^+)^+ \xrightarrow{\text{product in free semigroup } S^+} S^+ \\
 \text{view a letter as} \downarrow & \searrow \text{identity} & \downarrow \pi^+ \\
 \text{a one-letter word} & & S^+ \xrightarrow{\pi} S \\
 S^+ & \xrightarrow{\pi} & S
 \end{array}$$

In the above, π^+ stands for the coordinate-wise lifting of π to words of words.

For monoids, the same lemma holds, with $+$ replaced by $*$. There is no need to add an extra diagram for the monoid identity, since the monoid identity can be defined as the image under π of the empty word ε . The axioms $1 \cdot a$ and $a \cdot 1$ then follow from

$$1 \cdot a = \pi(\varepsilon) \cdot \pi(a) = \pi(\varepsilon a) = \pi(a) = a,$$

and a symmetric reasoning for $a \cdot 1$.

Also homomorphisms can be defined using commuting diagrams. A function $h : S \rightarrow T$ is a semigroup homomorphism if and only if the following diagram commutes

$$\begin{array}{ccc}
 S^+ & \xrightarrow{h^+} & T^+ \\
 \text{product in } S \downarrow & & \downarrow \text{product in } T \\
 S & \xrightarrow{h} & T
 \end{array}$$

By replacing $+$ with $*$ we get the definition of a monoid homomorphism.

Exercises

Exercise 1. (1) Show a function between two monoids that is a semigroup homomorphism, but not a monoid homomorphism.

Exercise 2. (1) Show that there are exponentially many semigroups of size n .

Exercise 3. Show that for every semigroup homomorphism $h : \Sigma^+ \rightarrow S$, with S finite, there exists some $N \in \{1, 2, \dots\}$ such that every word of length at least N can be factorised as $w = w_1 w_2 w_3$ where $h(w_2)$ is an idempotent².

Exercise 4. (1) Show that if S is a semigroup, then the same is true for the *powerset semigroup*, whose elements are possibly empty subsets of S , and where the product is defined coordinate-wise:

$$A \cdot B = \{a \cdot b : a \in A, b \in B\} \quad \text{for } A, B \subseteq S.$$

Exercise 5. (1) Let us view semigroups as a category, where the objects are semigroups and the morphisms are semigroup homomorphisms. What are the product and co-products of this category?

Exercise 6. (2) Let Σ be an alphabet, and let

$$X \subseteq \Sigma^+ \times \Sigma^+$$

be a set of words pairs. Define \sim_X to be least congruence on Σ^+ which contains all pairs from X . This is the same as the symmetric transitive closure of

$$\{(wxy, wxy) : w, y \in \Sigma^+, (x, y) \in X\}.$$

Show that the following problem – which is called the *word problem for semigroups* – is undecidable: given finite Σ, X and $w, v \in \Sigma^+$, decide if $w \sim_X v$.

Exercise 7. (2) Define the *theory of semigroups* to be the set of first-order sentences, which use one ternary relation $x = y \cdot z$, that are true in every semigroup. Show that the theory of semigroups is undecidable, i.e. it is undecidable if a first-order sentence is true in all semigroups.

Exercise 8. (2) Show that the theory of finite semigroups is different from the theory of (all) semigroups, but still undecidable.

² This exercise can be seen as the semigroup version of the pumping lemma.

1.1 Recognising languages

In this book, we are interested in monoids and semigroups as an alternative to finite automata for the purpose of recognising languages. Since languages are usually defined for possibly empty words, we use monoids and not semigroups when recognising languages.

Definition 1.5. Let Σ be a finite alphabet. A language $L \subseteq \Sigma^*$ is *recognised* by a monoid homomorphism

$$h : \Sigma^* \rightarrow M$$

if membership in $w \in L$ is determined uniquely by $h(w)$. In other words, there is a subset $F \subseteq M$ such that

$$w \in L \quad \text{iff} \quad h(w) \in F \quad \text{for every } w \in \Sigma^*.$$

We say that a language is recognised by a monoid if it is recognised by some monoid homomorphism into that monoid. The following theorem shows that, for the purpose of recognising languages, finite monoids and finite automata are equivalent.

Theorem 1.6. *The following conditions are equivalent for every $L \subseteq \Sigma^*$:*

- (1) *L is recognised by a finite nondeterministic automaton;*
- (2) *L is recognised by a finite monoid.*

Proof

2 \Rightarrow 1 From a monoid homomorphism one creates a deterministic automaton, whose states are elements of the monoid, the initial state is the identity, and the transition function is

$$(m, a) \mapsto m \cdot (\text{homomorphic image of } a).$$

After reading an input word, the state of the automaton is its homomorphic image, and therefore the accepting state from the monoid homomorphisms can be used. This automaton computes the monoid product according to the choice of parentheses illustrated in this example:

$$((((ab)c)d)e)f)g.$$

1 \Rightarrow 2 Let Q be the states of the nondeterministic automaton recognising L . Define a function³

$$\delta : \Sigma^* \rightarrow \text{monoid of binary relations on } Q$$

³ This transformation from a nondeterministic (or deterministic) finite automaton to a monoid incurs an exponential blow-up, which is unavoidable in the worst case.

which sends a word w to the binary relation

$$\{(p, q) \in Q^2 : \text{some run over } w \text{ goes from } p \text{ to } q\}.$$

This is a monoid homomorphism. It recognises the language: a word is in the language if and only if its image under the homomorphism contains at least one (initial, accepting) pair.

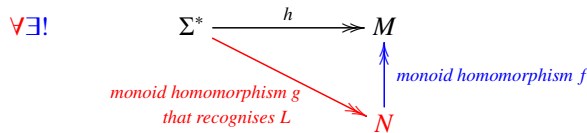
□

The syntactic monoid of a language. Deterministic finite automata have minimisation, i.e. for every language there is a minimal deterministic automaton, which can be found inside every other deterministic automaton that recognises the language. The same is true for monoids, as proved in the following theorem.

Theorem 1.7. *For every language⁴ $L \subseteq \Sigma^*$ there is a surjective monoid homomorphism*

$$h : \Sigma^* \rightarrow M,$$

called the syntactic homomorphism of L , which recognises it and is minimal in the sense explained in the following quantified diagram⁵



Proof The proof is the same as for the Myhill-Nerode theorem about minimal automata, except that the corresponding congruence is two-sided. Define the *syntactic congruence* of L to be the equivalence relation \sim on Σ^* which identifies two words $w, w' \in \Sigma^*$ if

$$uwv \in L \quad \text{iff} \quad uw'v \in L \quad \text{for all } u, v \in \Sigma^*.$$

Define h to be the function that maps a word to its equivalence class under syntactic congruence. It is not hard to see that h is compositional, and therefore by (the monoid version of) Lemma 1.3, one can equip the set of equivalence classes of syntactic congruences with a monoid structure – call M the resulting monoid – which turns h into a monoid homomorphism.

⁴ The language need not be regular, and the alphabet need not be finite.

⁵ Here is how to read the diagram. **For every red extension** of the black diagram **there exists a unique blue extension** which makes the diagram commute. Double headed arrows denote surjective homomorphisms, which means that \forall quantifies over surjective homomorphisms, and the same is true for $\exists!$.

It remains to show minimality of h , as expressed by the diagram in the lemma. Let then g be as in the diagram. Because g recognises the language L , we have

$$g(w) = g(w') \quad \text{implies} \quad w \sim w',$$

which, thanks to surjectivity of g , yields some function f from N to M , which makes the diagram commute, i.e. $h = f \circ g$. Furthermore, f must be a monoid homomorphism, because

$$\begin{aligned} f(a_1 \cdot a_2) &= \text{(by surjectivity of } g, \text{ each } a_i \text{ can be presented as } g(w_i) \text{ for some } w_i) \\ f(g(w_1) \cdot g(w_2)) &= \text{(} g \text{ is a monoid homomorphism)} \\ f(g(w_1 w_2)) &= \text{(the diagram commutes)} \\ h(w_1 w_2) &= \text{(} h \text{ is a monoid homomorphism)} \\ h(w_1) \cdot h(w_2) &= \text{(the diagram commutes)} \\ f(g(w_1)) \cdot f(g(w_2)) &= \\ f(a_1) \cdot f(a_2). & \end{aligned}$$

□

Exercise 9. (1) Show that the translation from deterministic finite automata to monoids is exponential in the worst case.

Exercise 10. (1) Show that the translation from (left-to-right) deterministic finite automata to monoids is exponential in the worst case, even if there is a right-to-left deterministic automaton of same size.

Exercise 11. (1) Which languages are recognised by finite commutative monoids?

Exercise 12. (1) Prove that surjectivity of g is important in Theorem 1.7.

Exercise 13. (1) Show that for every language, not necessarily regular, its syntactic homomorphism is the function

$$w \in \Sigma^* \quad \mapsto \quad \underbrace{(q \mapsto qw)}_{\substack{\text{state transformation} \\ \text{in the syntactic automaton}}}$$

where the syntactic automaton is the deterministic finite automaton from the Myhill-Nerode theorem.

Exercise 14. (2) Let \mathcal{L} be a class of regular languages with the following closure properties:

- \mathcal{L} is closed under Boolean combinations;
- \mathcal{L} is closed under inverse images of homomorphisms $h : \Sigma^\circ \rightarrow \Gamma^\circ$;
- Let $L \subseteq \Sigma^*$ be a language in \mathcal{L} . For every $w \in \Sigma^*$, \mathcal{L} contains the inverse image of L under the following operations:

$$v \mapsto wv \quad v \mapsto vw$$

Show that if L belongs to \mathcal{L} , then the same is true for every language recognised by its syntactic monoid.

1.2 Green's relations and the structure of finite semigroups

In this section, we describe some of the structural theory of finite semigroups. This theory is based on Green's relations, which are pre-orders in a semigroup that are based on prefixes, suffixes and infixes.

We begin with idempotents, which are ubiquitous in the analysis of finite semigroups. A semigroup element e is called *idempotent* if it satisfies

$$ee = e.$$

Example 1.8. In a group, there is a unique idempotent element, namely the group identity. There can be several idempotent elements, for example all elements are idempotent in the semigroup

$$(\{1, \dots, n\}, \max).$$

One can think of idempotents as being a relaxed version of identity elements.

Lemma 1.9 (Idempotent Power Lemma). *Let S be a finite semigroup. For every $a \in S$, there is exactly one idempotent in the set*

$$\{a^1, a^2, a^3, \dots\} \subseteq S.$$

Proof Because the semigroup is finite, the sequence a^1, a^2, \dots must contain a repetition, i.e. there must exist $n, k \in \{1, 2, \dots\}$ such that

$$a^n = a^{n+k} = a^{n+2k} = \dots.$$

After multiplying both sides of the above equation by a^{n-k} we get

$$a^{nk} = a^{n(k+k)} = a^{n(k+2k)} = \dots,$$

and therefore $a^{nk} = a^{nk+nk}$ is an idempotent. To prove uniqueness of the idempotent, suppose $n_1, n_2 \in \{1, 2, \dots\}$ are powers such that a^{n_1} and a^{n_2} are idempotent. Then we have

$$\underbrace{a^{n_1} = (a^{n_1})^{n_2}}_{\substack{\text{because } a^{n_1} \\ \text{is idempotent}}} = a^{n_1 n_2} = \underbrace{(a^{n_1})^{n_2} = a^{n_2}}_{\substack{\text{because } a^{n_2} \\ \text{is idempotent}}}$$

□

Finiteness is crucial for the above lemma, for example the infinite semigroup

$$(\{1, 2, \dots\}, +)$$

contains no idempotents. For $a \in S$, we use the name *idempotent power* for the element a^n , and we use the name *idempotent exponent* for the number n . The idempotent power is unique, but the idempotent exponent is not. It is easy to see that there is always an idempotent exponent which is at most the size of the semigroup, and idempotent exponents are closed under multiplication. Therefore, if a semigroup has n elements, then the factorial $n!$ is an idempotent exponent for every element of the semigroup. This motivates the following notation: we write $a^!$ for the idempotent power of a . The notation usually used in the semigroup literature is a^ω , but we will use ω for infinite words.

The analysis presented in the rest of this chapter will hold in any semigroup which satisfies the conclusion of the Idempotent Power Lemma.

Green's relations

We now give the main definition of this chapter.

Definition 1.10 (Green's relations). Let a, b be elements of a semigroup S . We say that a is a *prefix* of b if there exists a solution x of

$$ax = b.$$

The solution x can be an element of the semigroup, or empty (i.e. $a = b$). Likewise we define the suffix and infix relations, but with the equations

$$\underbrace{xa = b}_{\text{suffix}} \quad \underbrace{xay = b}_{\text{infix}}.$$

In the case of the infix relation, one or both of x and y can be empty.

Figure 1.1 shows a monoid along with the accompanying Green's relations. The prefix, suffix and infix relations are pre-orders, i.e. they are transitive and

reflexive⁶. They need not be anti-symmetric, for example in a group every element is an prefix (suffix, infix) of every other element. We say that two elements of a semigroup are in the same *prefix class* if they are prefixes of each other. Likewise we define *suffix classes* and *infix classes*.

Clearly every prefix class is contained in some infix class, because prefixes are special cases of infixes. Therefore, every infix class is partitioned into prefix classes. For the same reasons, every infix class is partitioned into suffix classes. The following lemma describes the structure of these partitions.

Lemma 1.11 (Egg-box lemma). *The following hold in every finite semigroup.*

- (1) *all distinct prefix classes in a given infix class are incomparable:*

a, b are infix equivalent, and a is a prefix of $b \Rightarrow a, b$ are prefix equivalent

- (2) *if a prefix class and a suffix class are contained in the same infix class, then they have nonempty intersection;*
 (3) *all prefix classes in the same infix class have the same size.*

Of course, by symmetry, the lemma remains true after swapping infixes with suffixes.

Proof

- (1) This item says that distinct prefix classes in the same infix class are incomparable, with respect to the prefix relation. This item of the Egg-box Lemma is the one that will be used most often.

Suppose that a, b are infix equivalent and a is a prefix of b , as witnessed by solutions x, y, z to the equations

$$b = ax \quad a = ybz.$$

⁶ Another description of the prefix pre-order is that a is a prefix of b if

$$aS^1 \supseteq bS^1. \tag{1.2}$$

In the above, S^1 is the monoid obtained from S by adding an identity element, unless it was already there. The sets aS^1, bS^1 are called *right ideals*. Because of the description in terms of inclusion of right ideals, the semigroup literature uses the notation

$$a \geq_{\mathcal{R}} b \stackrel{\text{def}}{=} aS^1 \supseteq bS^1$$

for the prefix relation. Likewise, $a \geq_{\mathcal{L}} b$ is used for the prefix relation, which is defined in terms of left ideals. Also, for some mysterious reason, $a \geq_{\mathcal{J}} b$ is used for the infix relation. We avoid this notation, because it makes longer words smaller.

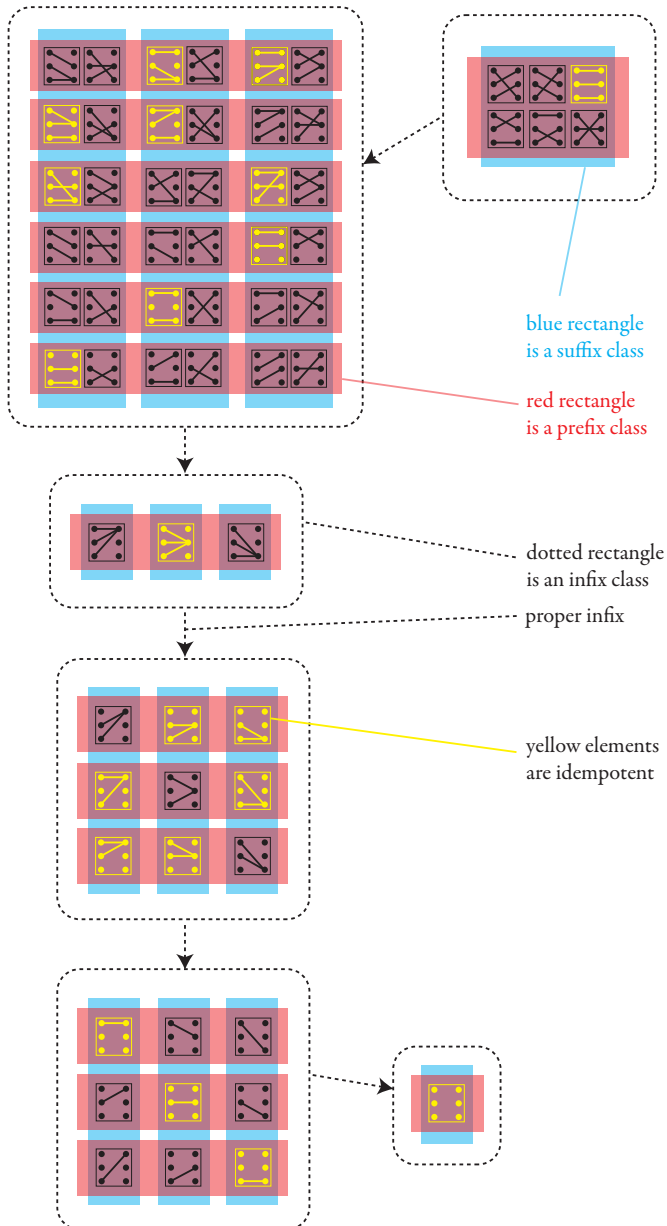
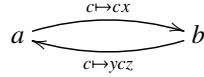


Figure 1.1 The semigroup of partial functions from a three element set to itself, partitioned into prefix, suffix and infix classes. In this particular example, the infix classes are totally ordered, which need not be the case in general.

As usual, each of x, y, z could be empty. This can be illustrated as



Consider the idempotent exponent $! \in \{1, 2, \dots\}$ which arises from Idempotent Power Lemma. We have:

$$\begin{aligned}
 b &= \text{(follow } !+! \text{ times the loop around } a, \text{ then go to } b) \\
 y^{!+!}a(xz)^{!+!}x &= \text{(} y^! \text{ is an idempotent)} \\
 y^!a(xz)^{!+!}x &= \text{(follow } ! \text{ times the loop around } a) \\
 & a(xz)^!x,
 \end{aligned}$$

which establishes that b is a prefix of a , and therefore a, b are in the same prefix class.

- (2) We now show that prefix and suffix classes in the same infix class must intersect. Suppose that a, b are in the same infix class, as witnessed by

$$a = xby.$$

With respect to the infix relation, by is between b and a , and therefore it must be in the same infix class as both of them. We have

$$\begin{array}{c}
 by \text{ is a suffix of } xby = a \\
 \underbrace{x \quad b \quad y} \\
 b \text{ is a prefix of } by
 \end{array}
 ,$$

and therefore, thanks to the previous item, by is prefix equivalent to b and suffix equivalent to a . This witnesses that the prefix class of b and the suffix class of a have nonempty intersection.

- (3) We now show that all prefix classes in the same infix class have the same size. Take some two prefix classes in the same infix class, given by representatives a, b . We can assume that a, b are in the same suffix class, thanks to the previous item. Let

$$a = xb \quad b = ya$$

be witnesses for the fact that a, b are in the same suffix class. The following claim implies that the two prefix classes under consideration have the same size.

Claim 1.12. *The following maps are mutually inverse bijections*

$$\text{prefix class of } a \begin{array}{c} \xrightarrow{c \rightarrow yc} \\ \xleftarrow{c \rightarrow xc} \end{array} \text{prefix class of } b$$

Proof Suppose that c is in the prefix class of a , as witnessed by a decomposition $c = az$. If we apply sequentially both maps in the statement of the claim to c , then we get

$$xyc = xyaz \stackrel{ya=b}{=} xbz \stackrel{xb=a}{=} az \stackrel{az=c}{=} c.$$

This, and a symmetric argument for the case when c is in the prefix class of b , establishes that the maps in the statement of the claim are mutually inverse. It remains to justify that the images of the maps are as in the statement of the claim, i.e. the image of the top map is the prefix class of b , and the image of the bottom map is the prefix class of a . Because the two maps are mutually inverse, and they prepend elements to their inputs, it follows that each of the maps has its image contained in the infix class of a, b . To show that the image of the top map is in the prefix class of b (a symmetric argument works for the bottom map), we observe that every element of this image is of the form yaz , and therefore it has $b = ya$ as a prefix, but it is still in the same infix class as a, b as we have observed before, and therefore it must be prefix equivalent to b thanks to the item (1) of the lemma. \square

\square

The Egg-box Lemma establishes that each infix class has the structure of a rectangular grid (which apparently is reminiscent of a box of eggs), with the rows being prefix classes and the columns being suffix classes. Let us now look at the eggs in the box: define an \mathcal{H} -class to be a nonempty intersection of some prefix class and some suffix class. By item (2) of the Egg-box Lemma, every pair of prefix and suffix classes in the same infix class lead to some \mathcal{H} -class. The following lemma shows that all \mathcal{H} -classes in the same infix class have the same size.

Lemma 1.13. *If a, b are in the same infix class, then there exist possibly empty x, y such that the following is a bijection*

$$\mathcal{H}\text{-class of } a \xrightarrow{c \mapsto xcy} \mathcal{H}\text{-class of } b$$

Proof Consider first the special case of the lemma, when a and b are in the same suffix class. Take the map from Claim 1.12, which maps bijectively the prefix class of a to the prefix class of b . Since this map preserves suffix classes, it maps bijectively the \mathcal{H} -class of a to the \mathcal{H} -class of b . By a symmetric argument, the lemma is also true when a and b are in the same prefix class.

For the general case, we use item (2) of the Egg-box Lemma, which says that there must be some intermediate element that is in the same prefix class

as a and in the same suffix class as b , and we can apply the previously proved special cases to go from the \mathcal{H} -class of a to the \mathcal{H} -class of the intermediate element, and then to the \mathcal{H} -class of b . \square

The following lemma shows a dichotomy for an \mathcal{H} -class: either it is a group, or the the product of every two elements in that \mathcal{H} -class falls outside the infix class.

Lemma 1.14 (*\mathcal{H} -class Lemma*). *The following conditions are equivalent for every \mathcal{H} -class G in a finite semigroup:*

- (1) G contains an idempotent;
- (2) ab is in the same infix class as a and b , for some $a, b \in G$;
- (3) $ab \in G$ for some $a, b \in G$;
- (4) $ab \in G$ for all $a, b \in G$;
- (5) G is a group (with product inherited from the semigroup)

Proof Implications (5) \Rightarrow (1) \Rightarrow (2) in the lemma are obvious, so we focus on the remaining implications.

(2) \Rightarrow (3) Suppose that ab is in the same infix class as a and b . Since a is a prefix of ab , and the two elements are in the same infix class, item (1) of the Egg-box Lemma implies that ab is in the prefix class of a , which is the same as the prefix class of b . For similar reasons, ab is in the same suffix class as a and b , and therefore $ab \in G$.

(3) \Rightarrow (4) Suppose that there exist $a, b \in G$ with $ab \in G$. We need to show that G contains the product of every elements $c, d \in G$. Since c is prefix equivalent to a there is a decomposition $a = xc$, and for similar reasons there is a decomposition $b = dy$. Therefore, cd is an infix of

$$\underbrace{a}_{xc} \underbrace{b}_{dy} \in G,$$

and therefore it is in the same infix class as G . Since c is a prefix of cd , and both are in the same infix class, the Egg-box Lemma implies that cd is in the prefix class of c . For similar reasons cd is in the suffix class of d . Therefore, $cd \in G$.

(4) \Rightarrow (5) Suppose that G is closed under products, i.e. it is a subsemigroup. We will show that it is a group. By the Idempotent Power Lemma, G contains some idempotent, call it e . We claim that e is an identity element in G , in particular it is unique. Indeed, let $a \in G$. Because a and e are in the same suffix class, it follows that a can be written as xe , and therefore

$$ae = xee = xe = a.$$

For similar reasons, $ea = a$, and therefore e is an identity element in G . The group inverse is defined as follows. Take $! \in \{1, 2, \dots\}$ to be the idempotent exponent which arises from the Idempotent Power Lemma. For every $a \in G$, the power $a^!$ is an idempotent. Since there is only one idempotent in G , we have $a^! = e$. Therefore, $a^{!-1}$ is a group inverse of a .

□

Exercise 15. (1) Show that for every finite monoid, the infix class of the monoid identity is a group.

Exercise 16. (2) Consider a finite semigroup. Show that an infix class contains an idempotent if and only if it is *regular*, which means that there exist a, b in the infix class such that ab is also in the infix class.

Exercise 17. (2) Show that if G_1, G_2 are two \mathcal{H} -classes in the same infix class of a finite semigroup, and they are both groups, then they are isomorphic as groups⁷.

Exercise 18. (2) We say that semigroup is *prefix trivial* if its prefix classes are singletons. Show that a finite semigroup S is prefix trivial if and only if it satisfies the identity

$$(xy)^! = (xy)^!x \quad \text{for all } x, y \in S.$$

Exercise 19. (2) Define the *syntactic semigroup* of a language to be the subset of the syntactic monoid which is the image of the nonempty words under the syntactic homomorphism. The syntactic semigroup may be equal to the syntactic monoid. We say that a language $L \subseteq \Sigma^*$ is definite if it is a finite Boolean combination of languages of the form $w\Sigma^*$, for $w \in \Sigma^*$. Show that a language is definite if and only if its syntactic semigroup S satisfies the identity

$$x^! = x^!y \quad \text{for all } x, y \in S.$$

Exercise 20. (2) Show two regular languages such that one is definite and the other is not, but both have isomorphic syntactic monoids.

⁷ Let us combine Exercises 16 and 17. By Exercises (16) and the \mathcal{H} -class lemma, an infix class is regular if and only if it contains an \mathcal{H} -class which is a group. By Exercise (17), the corresponding group is unique up to isomorphism. This group is called the *Shützenberger group* of the regular infix class.

Exercise 21. (1) Consider semigroups S which satisfy the following property: (*) that there is an infix class $J \subseteq S$ such that every $a \in S$ is an infix of J , or an absorbing zero element. Show that every finite semigroup is sub-semigroup of a product of finite semigroups that satisfy (*).

Exercise 22. (2) Show that every finite semigroup satisfies

$$\forall x_1 \forall x_2 \exists y_1 \exists y_2 \underbrace{z_1 = z_1 z_1 = z_1 z_2 \wedge z_2 = z_2 z_2 = z_2 z_1,}_{\text{where } z_i = x_i y_i}$$

where quantifiers range over elements of the finite semigroup.

Exercise 23. (2) Show that the following problem is decidable:

- **Input.** Two disjoint sets of variables

$$X = \{x_1, \dots, x_n\} \quad Y = \{y_1, \dots, y_m\}$$

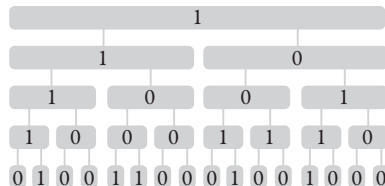
and two words $w, w' \in (X \cup Y)^+$.

- **Question.** Is the following true in all finite semigroups:

$$\forall x_1 \dots \forall x_n \exists y_1 \dots \exists y_m \underbrace{w = w'}_{\text{same product}}$$

1.3 The Factorisation Forest Theorem

In this section, we show how products in a semigroup can be organised in trees, so that in each node of the tree the products are very simple. The most natural way to do this is to have binary tree, as in the following example, which uses the two semigroup $\{0, 1\}$ with addition modulo 2:



We use the name *factorisation tree* for structures as in the above picture: a *factorisation tree over a semigroup S* is a tree, where nodes are labelled by semigroup elements, such that every node is either a leaf, or is labelled by the

semigroup product of the labels of its children. A binary factorisation tree is one where every node has zero or two children. For every word in S^+ , one can find a corresponding factorisation tree (i.e. one where the word is obtained by reading the leaves left-to-right) whose height (i.e. the maximal number of edges on a root-to-leaf path) is logarithmic in the length of the word.

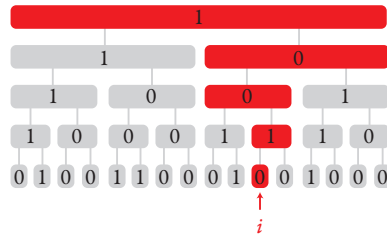
Binary factorisation trees are a natural data structure for several problems about regular languages.

Example 1. Fix a regular language $L \subseteq \Sigma^*$. Consider the following dynamic problem. We begin with some word in Σ^* . We want to build a data structure that handles efficiently the following updates and queries:

Query. Is the current word in L ?

Update. Change the label of position i to $a \in \Sigma$.

To solve this problem, as the data structure we can use a binary factorisation tree with respect to some semigroup that recognises the language. If we assume that the regular language is fixed and not part of the input, then the queries are processed in constant time, by checking if the root label of the factorisation tree. The updates are processed in time proportional to the height of the factorisation tree, by updating all of the nodes on the path from the updates position to the root, as in the following picture:



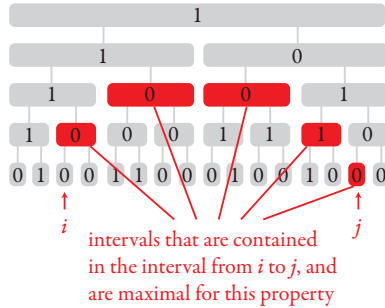
If the factorisation tree is chosen to be balanced, then the updates are processed in logarithmic time. \square

Example 2. Fix a regular language $L \subseteq \Sigma^*$. Consider the following dynamic problem. We begin with some word in Σ^* . We want to build a data structure that handles efficiently the following queries (there are no updates):

Query. Given positions $i \leq j$, does L contain the infix from i to j ?

As in the previous example, we solve the problem using a binary factorisation tree, with respect to some semigroup recognising the language. Suppose that the tree has height k . Each node of the factorisation tree corresponds to an infix

of the underlying word. The infix from i to j can be partitioned into at most $2k$ intervals, each of which corresponds to a node of the tree, as in the following picture:



Therefore, the queries can be processed in time proportional to the height of the tree, which can be assumed to be logarithmic in the length of the underlying word. \square

We will now show a data structure which will allow constant time query processing in the problem from Example 2. We will also use a variant of factorisation trees, except that non-binary nodes will need to be used. The problem in Example 1 cannot be solved in constant time⁸.

Simon trees

A Simon tree is a factorisation tree which allows nodes of degree higher than 2, but these nodes must have idempotent children. The data structure is named after Imre Simon, who introduced it⁹.

Definition 1.15 (Simon Tree). Define a *Simon tree* to be a factorisation tree where every non-leaf node has one (or both) of the following types:

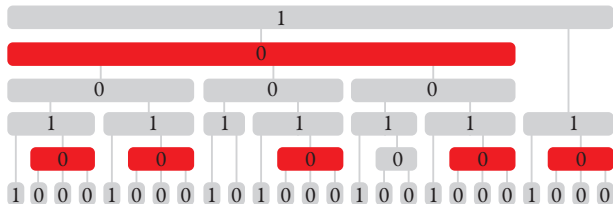
binary: has two children; or

idempotent: all children have the same label, which is an idempotent.

Here is a picture of a Simon tree for the semigroup $\{0, 1\}$ with addition modulo 2, with idempotent nodes drawn in red:

⁸ Lower bounds for this problem can be seen in [14] Frandsen, Miltersen, and Skyum, "Dynamic Word Problems", 1997, Fig. 1

⁹ Under the name Ramseyan factorisation forests, in [31] Simon, "Factorization Forests of Finite Height", 1990, 69



The main result about Simon trees is that their height can be bounded by a constant that depends only on the semigroup, and not the underlying word.

Theorem 1.16 (Factorisation Forest Theorem). *Let S be a finite semigroup. Every word in S^+ admits a Simon tree of height¹⁰ $< 5|S|$.*

The rest of this chapter is devoted to proving the theorem.

Groups. We begin with the special case of groups.

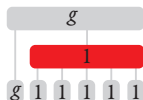
Lemma 1.17. *Let G be a finite group. Every word in G^+ admits a Simon tree of height $< 3|G|$.*

Proof Define the *prefix set* of a word $w \in G^+$ to be the set of group elements that can be obtained by taking a product of some nonempty prefix of w . By induction on the size of the prefix set, we show that every $w \in G^+$ has a Simon tree of height strictly less than 3 times the size of the prefix set. Since the prefix set has maximal size $|G|$, this proves the lemma.

The induction base is when the prefix set is a singleton $\{g\}$. This means that the first letter is g , and every other letter h satisfies $gh = g$. In a group, only the group identity $h = 1$ can satisfy $gh = g$, and therefore h is the group identity. In other words, if the prefix set is $\{g\}$, then the word is of the form

$$g \underbrace{1 \cdots 1}_{\text{a certain number of times}}.$$

Such a word admits a Simon tree as in the following picture:



¹⁰ The first version of this theorem was proved in [31, Theorem 6.1], with a bound of $9|S|$. The optimal bound is $3|S|$, which was shown in [18] Kufleitner, “The Height of Factorization Forests”, 2008, Theorem 1. The proof here is based on Kufleitner, with some optimisations removed.

The height of this tree is 2, which is strictly less than three times the size of the prefix set.

To prove the induction step, we show that every $w \in G^+$ admits a Simon tree, whose height is at most 3 plus the size from the induction assumption. Choose some g in the prefix set of w . Decompose w into factors as

$$w = \underbrace{w_1 w_2 \cdots w_{n-1}}_{\substack{\text{nonempty} \\ \text{factors}}} \underbrace{w_n}_{\substack{\text{could} \\ \text{be empty}}}$$

by cutting along all prefixes with product g . For the same reasons as in the induction base, every factor w_i with $1 < i < n$ has a product which is the group identity.

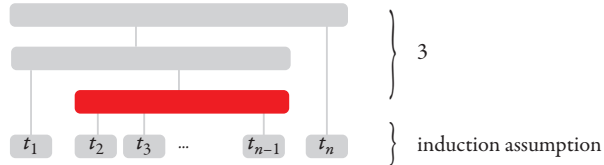
Claim 1.18. *The induction assumption applies to all of w_1, \dots, w_n .*

Proof For the first factor w_1 , the induction assumption applies, because its prefix set omits g . For the remaining blocks, we have a similar situation, namely

$$g \cdot (\text{prefix set of } w_i) \subseteq (\text{prefix set of } w) - \{g\} \quad \text{for } i \in \{2, 3, \dots, n\},$$

where the left side of the inclusion is the image of the prefix set under the operation $x \mapsto gx$. Since this operation is a permutation of the group, it follows that the left side of the inclusion has smaller size than the prefix set of w , and therefore the induction assumption applies. \square

By the above claim, we can apply the induction assumption to compute Simon trees t_1, \dots, t_n for the factors w_1, \dots, w_n . To get a Simon tree for the whole word, we join these trees as follows:

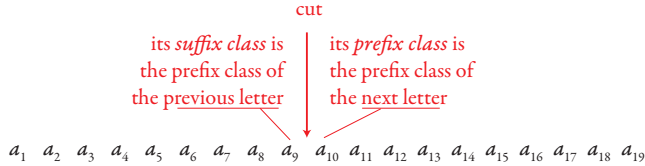


The gray nodes are binary, and the red node is idempotent because every w_i with $1 < i < n$ evaluates to the group identity. \square

Smooth words. In the next step, we prove the theorem for words where all infixes come from the same infix class. We say that a word $w \in S^+$ is *smooth* if all of its nonempty infixes have product in the same infix class. The following lemma constructs Simon trees for smooth words.

Lemma 1.19. *If a word is smooth, and the corresponding infix class is $J \subseteq S$, then it admits a Simon tree of height $< 4|J|$.*

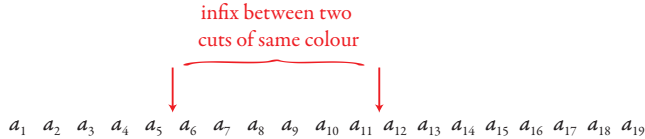
Proof Define a *cut* in a word to be the space between two consecutive letters; in other words this is a decomposition of the word into a nonempty prefix and a nonempty suffix. For a cut, define its *prefix* and *suffix* classes as in the following picture:



For every cut, both the prefix and suffix classes are contained in J , and therefore they have nonempty intersection thanks to item (2) of the Egg-box Lemma. This nonempty intersection is an \mathcal{H} -class, which is defined to be the *colour* of the cut. The following claim gives the crucial property of cuts and their colours.

Claim 1.20. *If two cuts have the same colour H , then the infix between them has product in H .*

Proof Here is a picture of the situation:



The infix begins with a letter from the prefix class containing H . Since the infix is still in the infix class J , by assumption on smoothness, it follows from item (1) of the Egg-box Lemma that the product of the infix is in the prefix class of H . For the same reason, the product of the infix is in the suffix class of H . Therefore, it is in H . \square

Define the *colour set* of a word to be the set of colours of its cuts; this is a subset of the \mathcal{H} -classes in J . Thanks to Lemma 1.12, all \mathcal{H} -classes contained in J have the same size, and therefore it makes sense to talk about the \mathcal{H} -class size in J , without specifying which \mathcal{H} -class is concerned.

Claim 1.21. *Every J -smooth word has a Simon tree of height at most*

$$|\text{colour set of } w| \cdot (3 \cdot \mathcal{H}\text{-class size} + 1).$$

Before proving the claim, we show how it implies the lemma. Since the number of possible colours is the number of \mathcal{H} -classes, the maximal height that can arise from the claim is

$$3 \cdot |J| + (\text{maximal size of colour set}) < 4|J|.$$

It remains to prove the claim.

Proof Induction on the size of the colour set. The induction base is when the colour set is empty. In this case the word has no cuts, and therefore it is a single letter, which is a Simon tree of height zero.

Consider the induction step. Let w be a smooth word. To prove the induction step, we will find a Simon tree whose height is at most the height from the induction assumption, plus

$$3 \cdot (\mathcal{H}\text{-class size}) + 1.$$

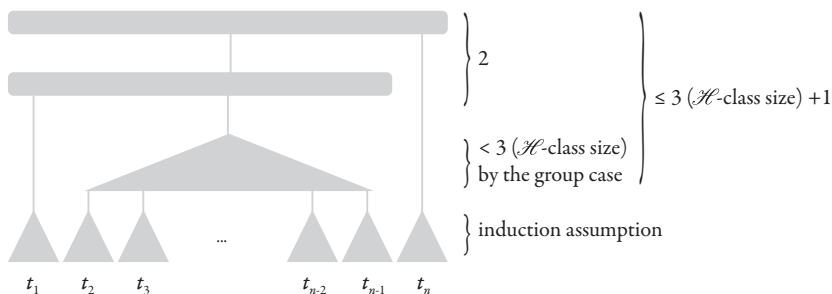
Choose some colour in the colour set of w , which is an \mathcal{H} -class H . Cut the word w along all cuts with colour H , yielding a decomposition

$$w = w_1 \cdots w_n.$$

None of the words w_1, \dots, w_n contain a cut with colour H , so the induction assumption can be applied to yield corresponding Simon trees t_1, \dots, t_n .

If $n \leq 3$, then the Simon trees from the induction assumption can be combined using binary nodes, increasing the height by at most 2, and thus staying within the bounds of the claim.

Suppose now that $n \geq 4$. By Claim 1.20, any infix between two cuts of colour H has product in H . In particular, all w_2, \dots, w_{n-1} have product in H , and the same is true for $w_2 w_3$. It follows that H contains at least one product of two elements from H , and therefore H is a group thanks to item (3) of the \mathcal{H} -class Lemma. Therefore, we can apply the group case from Lemma 1.17 to join the trees t_2, \dots, t_{n-1} . The final Simon tree looks like this:



□

□

General case. We now complete the proof of the Factorisation Forest Theorem. The proof is by induction on the *infix height* of the semigroup, which is defined to be the longest chain that is strictly increasing in the infix ordering. If the infix height is one, then the semigroup is a single infix class, and we can apply Lemma 1.19 since all words in S^+ are smooth. For the induction step, suppose that S has infix height at least two, and let $T \subseteq S$ be the elements which have a proper infix. It is not hard to see that T is a subsemigroup, and its induction parameter is smaller.

Consider a word $w \in S^+$. As in Lemma 1.19, define a cut to be a space between two letters. We say that a cut is *smooth* if the letters preceding and following the cut give a two-letter word that is smooth.

Claim 1.22. *A word in S^+ is smooth if and only if all of its cuts are smooth.*

Proof Clearly if a word is smooth, then all of its cuts must be smooth. We prove the converse implication by induction on the length of the word. Words of length one or two are vacuously smooth. For the induction step, consider a word $w \in S^+$ with all cuts being smooth. Since all cuts are smooth, all letters are in the same infix class. We will show that w is also in this infix class. Decompose the word as $w = vab$ where $a, b \in S$ are the last two letters. By induction assumption, va is smooth. Since the last cut is smooth, a and ab are in the same infix class, and therefore they are in the same prefix class by the Egg-box Lemma. This means that there is some x such that $abx = a$. We have

$$va = vabx = wx$$

which establishes that w is in the same infix class as va , and therefore in the same infix class as all the letters in w . □

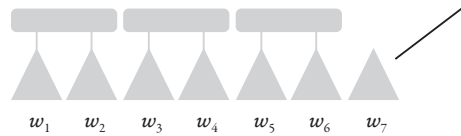
Take a word $w \in S^+$, and cut it along all cuts which are not smooth, yielding a factorisation

$$w = w_1 \cdots w_n.$$

By Claim 1.22, all of the words w_1, \dots, w_n are smooth, and therefore Lemma 1.19 can be applied to construct corresponding Simon trees of height strictly smaller than

$$4 \cdot (\text{maximal size of an infix class in } S - T).$$

Using binary nodes, group these trees into pairs, as in the following picture:



Each pair corresponds to a word with a non-smooth cut, and therefore each pair has product in T . Therefore, we can combine the paired trees into a single tree, using the induction assumption on a smaller semigroup. The resulting height is the height from the induction assumption on T , plus at most

$$1 + 4 \cdot (\text{maximal size of an infix class in } S - T) < 5|S - T|,$$

thus proving the induction step.

Exercises

Exercise 24. (1) Show that for every semigroup homomorphism

$$h : \Sigma^+ \rightarrow S \quad \text{with } S \text{ finite}$$

there is some $k \in \{1, 2, \dots\}$ such that for every $n \in \{3, 4, \dots\}$, every word of length bigger than n^k can be decomposed as

$$w_0 w_1 \cdots w_n w_{n+1}$$

such that all of the words w_1, \dots, w_n are mapped by h to the same idempotent.

Exercise 25. (2) Show optimality for the previous exercise, in the following sense. Show that for every $k \in \{1, 2, \dots\}$ there is some semigroup homomorphism

$$h : \Sigma^+ \rightarrow S \quad \text{with } S \text{ finite}$$

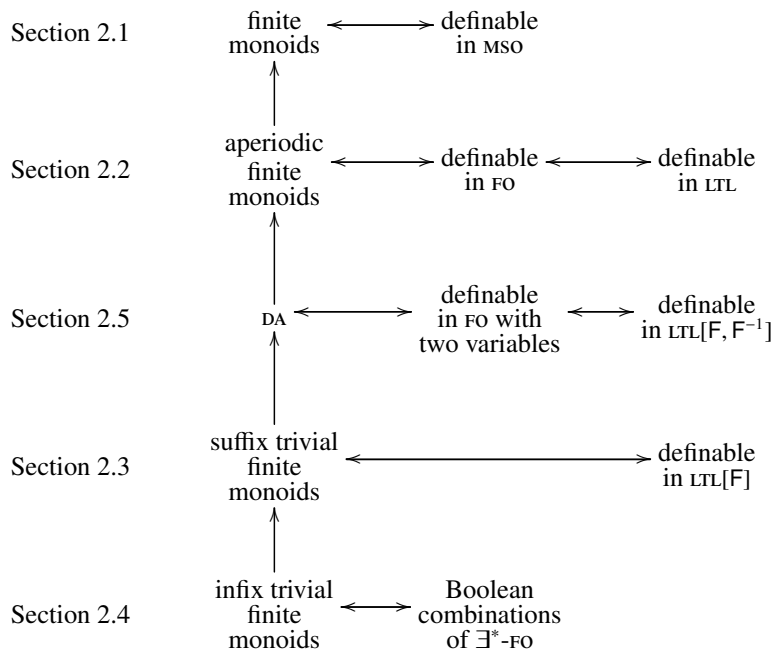
such that for every $n \in \{1, 2, \dots\}$ there is a word of length at least n^k which does not admit a factorisation $w_0 \cdots w_{n+1}$ where all of w_1, \dots, w_n are mapped by h to the same idempotent.

Exercise 26. (2) Let $h : \Sigma^* \rightarrow M$ be a monoid homomorphism. Consider a regular expression over Σ , which does not use Kleene star L^* but only Kleene plus L^+ . Such a regular expression is called h -typed if every subexpression has singleton image under h , and furthermore subexpressions with Kleene plus have idempotent image. Show that every language recognised by h is defined by finite union of h -typed expressions.

2

Logics on finite words, and the corresponding monoids

In this chapter, we show how structural properties of a monoid correspond to the logical power needed to define languages recognised by this monoid. We consider two kinds of logic: monadic second-order logic MSO and its fragments (notably first-order logic FO), as well as linear temporal logic LTL and its fragments. Here is a map of the results from this chapter, with horizontal arrows being equivalences, and the vertical arrows being strict inclusions.



2.1 All monoids and monadic second-order logic

We begin with monadic second-order logic (MSO), which is the logic that captures exactly the class of regular languages.

Logic on words. We assume that the reader is familiar with the basic notions of logic. The following descriptions are meant to fix notation. We use the name *vocabulary* for a set of relation names, each one with associated arity in $\{1, 2, \dots\}$. A *model* over a vocabulary consists of an underlying set (called the *universe* of the model), together with an interpretation of the vocabulary, which maps each relation name to a relation over the universe of corresponding arity. We allow the universe to be empty. For example, a directed graph is the same thing as a model over the vocabulary which contains one binary relation $E(x, y)$.

To express properties of models, we use first-order logic FO and MSO. Formulas of first-order logic over a given vocabulary are constructed as follows:

$$\underbrace{\forall x \quad \exists x}_{\substack{\text{quantification over} \\ \text{elements of the universe}}} \quad
 \underbrace{\varphi \wedge \psi \quad \varphi \vee \psi \quad \neg \varphi}_{\text{Boolean operations}} \quad
 \underbrace{R(x_1, \dots, x_n)}_{\substack{\text{an } n\text{-ary relation name from} \\ \text{the vocabulary applied} \\ \text{to a tuple of variables}}} \quad
 \underbrace{x = y}_{\text{equality}}$$

For the semantics of first-order formulas, we use the notation

$$\mathbb{A}, a_1, \dots, a_n \models \varphi(x_1, \dots, x_n)$$

to say that φ is true in the model \mathbb{A} , assuming that free variable x_i is mapped to $a_i \in \mathbb{A}$. A *sentence* is a formula without free variables.

The logic MSO extends first-order logic by allowing quantification over subsets of the universe (in other words, monadic relations over the universe, hence the name). The syntax of the logic has two kinds of variables: lower case variables x, y, z, \dots describe elements of the universe as in first-order logic, while upper case variables X, Y, Z, \dots describe subsets of the universe. Apart from the syntactic constructions of first-order logic, MSO also allows:

$$\underbrace{\forall X \quad \exists Y}_{\substack{\text{quantification over} \\ \text{subsets of the universe}}} \quad
 \underbrace{x \in X}_{\text{membership}}$$

We do not use more powerful logics (e.g. full second-order logic, which can also quantify over binary relations, ternary relations, etc.).

The following definition associates to each word a corresponding model. With this correspondence, we can use logic to define properties of words.

Definition 2.1 (Languages definable in first-order logic and mso). For a word $w \in \Sigma^*$, define its *ordered model* as follows. The universe is the set of positions in the word. The vocabulary is

$$\underbrace{x \leq y}_{\text{arity 2}} \quad \{ \underbrace{a(x)}_{\text{arity 1}} \}_{a \in \Sigma},$$

where $x \leq y$ is interpreted as the natural order on positions, and with $a(x)$ is interpreted as the set of positions with label a . For a sentence φ of mso over this vocabulary, we define its *language* to be

$$\{w \in \Sigma^* : \text{the ordered model of } w \text{ satisfies } \varphi\}.$$

A language is called *mso definable* if it is of this form. If φ is in first-order logic, then the language is called *first-order definable*.

Example 3. The language $a^*bc^* \subseteq \{a, b, c\}^*$ is first-order definable, as witnessed by the sentence:

$$\underbrace{\exists x}_{\text{there is a position}} \quad \underbrace{b(x)}_{\text{which has label } b} \wedge \underbrace{\forall y \overset{x \leq y \wedge x \neq y}{y < x} \Rightarrow a(y)}_{\text{and every earlier position has label } a} \wedge \underbrace{\forall y y > x \Rightarrow c(y)}_{\text{and every later position has label } b}.$$

□

Example 4. The language $(aa)^*a \subseteq a^*$ of words of odd length is mso definable, as witnessed by the sentence:

$$\underbrace{\exists X}_{\text{there is a set of positions}} \quad \underbrace{\forall x \overset{\forall y y \geq x}{\text{first}(x)} \vee \overset{\forall y y \leq x}{\text{last}(x)} \Rightarrow x \in X}_{\text{which contains the first and last positions,}} \wedge \underbrace{\forall x \forall y \overset{x < y \wedge \forall z z \leq x \vee y \leq z}{x = y + 1} \Rightarrow (x \in X \Leftrightarrow y \notin X)}_{\text{and contains every second position.}}$$

As we will see in Section 2.2, this language is not first-order definable. □

One could imagine other ways of describing a word via a model, e.g. a *successor model* where $x \leq y$ is replaced by a successor relation $x + 1 = y$. The successor relation can be defined in first-order logic in terms of order, but the converse is not true: there are languages that are first-order definable in the order model but not in the successor model, see Exercise 38. For the logic mso, there is no difference between successor and order, since the order can be defined in mso as follows

$$x \leq y \quad \text{iff} \quad \underbrace{\forall X (x \in X \wedge (\forall y \forall z y \in X \wedge y + 1 = z \Rightarrow z \in X))}_{X \text{ contains } x \text{ and is closed under successors}} \Rightarrow y \in X.$$

We now present the seminal Trakhtenbrot-Büchi-Elgot Theorem, which says that mso describes exactly the regular languages.

Theorem 2.2 (Trakhtenbrot-Büchi-Elgot). *A language $L \subseteq \Sigma^*$ is mso definable if and only if it is regular¹.*

This result is seminal for two reasons.

The first reason is that it motivates the search for other correspondences

$$\text{machine model} \quad \sim \quad \text{logic},$$

which can concern either restrictions or generalisations of the regular languages. In the case of restrictions, important examples are first-order logic and its fragments, which will be described later in this chapter. In this book, we do not study the generalisations; we are only interested in regular languages. Nevertheless, it is worth mentioning Fagin’s Theorem, which says that NP describes exactly the languages definable in existential second-order logic².

The second reason is that the Trakhtenbrot-Büchi-Elgot theorem generalises well to structures beyond finite words. For example, there are natural notions of mso definable languages for: infinite words, finite trees, infinite trees, graphs, etc. It therefore makes sense to search for notions of regularity – e.g. based on generalisations of semigroups – which have the same expressive power as mso. This line of research will also be followed in this book.

The rest of Section 2.1 proves the Trakhtenbrot-Büchi-Elgot Theorem.

The easy part is that every regular language is mso definable. Using the same idea as for the parity language in Example 4, the existence of a run of nondeterministic finite automaton can be formalised in mso. If the automaton has n states, then the formula looks like this:

$$\underbrace{\exists X_1 \exists X_2 \cdots \exists X_n}_{\text{existential set quantification}} \quad \underbrace{\text{“the sets } X_1, \dots, X_n \text{ describe an accepting run”}}_{\text{first-order formula}}$$

A corollary is that if we take any mso definable language, turn it into an automaton using the hard implication, and come back to mso using the easy implication, then we get an mso sentence of the form described above.

The “easy part” will become hard part in some generalisations – e.g. for

¹ This result was proved, independently, in the following papers:
 [34] Trakhtenbrot, “The synthesis of logical nets whose operators are described in terms of one-place predicate calculus (Russian)”, 1958, Theorems 1 and 2
 [7] Büchi, “Weak second-order arithmetic and finite automata”, 1960, Theorems 1 and 2
 [12] Elgot, “Decision problems of finite automata design and related arithmetics”, 1961, Theorem 5.3
² [13] Fagin, “Generalized first-order spectra and polynomial-time recognizable sets”, 1974, Theorem 6

some kinds of infinite words or for graphs – because these generalisations lack a suitable automaton model.

We now turn to the hard part, which says that every mso definable language is regular. This implication is proved in the rest of Section 2.1. For the definition of regularity, we use finite monoids. In other words, we will show that every mso definable language is recognised by a finite monoid. The monoid will consist of “mso types” of some fixed quantifier rank, as described later in this section.

To avoid talking about the two kinds of variables in mso, instead of mso over ordered models, we will use first-order logic over models with second-order information, as defined below.

Definition 2.3. Define the *set model* of a word $w \in \Sigma^*$ as follows. The universe is sets of positions, and it is equipped with the following relations:

$$\begin{array}{cccc}
 x \subseteq y & \underbrace{x < y}_{\text{every position in } x \text{ is strictly before every position in } y} & \underbrace{x = \emptyset}_{x \text{ is empty}} & \underbrace{x \subseteq a}_{\text{all positions in } x \text{ have label } a}
 \end{array}$$

Lemma 2.4. *A language $L \subseteq \Sigma^*$ is mso definable in the ordered model if and only if it is first-order definable in the set model.*

Proof Set quantification in the ordered model corresponds to first-order quantification in the set model. To recover first-order quantification from the ordered model, we use singleton sets in the set model; these can be defined using the inclusion relation. \square

Thanks to the above lemma, instead of mso over the ordered model, we can work with first-order logic over the set model. Recall that the *quantifier rank* of a first-order formula is the maximal number of quantifiers that can be found on some branch of the syntax tree in the formula. Here is an example:

$$\underbrace{\forall x (x \neq \emptyset \wedge x \subseteq a \Rightarrow \underbrace{(\exists y y < x \wedge y \neq \emptyset)}_{\text{quantifier rank 1}} \wedge \underbrace{(\exists y x < y \wedge y \neq \emptyset)}_{\text{quantifier rank 1}})}_{\text{quantifier rank 2}}$$

An important property of this size measure, unlike e.g. formula size, is that formulas of quantifier rank at most $k \in \{0, 1, \dots\}$ are closed under Boolean combinations. For words $w, w' \in \Sigma^*$, we write

$$w \equiv_k w'$$

if the corresponding set models satisfy the same first-order sentences of quan-

tifier rank at most k . The following lemma is the key to the equivalence of finite semigroups and mso.

Lemma 2.5 (Compositionality of mso). *Let Σ be a finite alphabet and $k \in \{0, 1, \dots\}$. Then \equiv_k is an equivalence relation on Σ^* with the following properties:*

- Finite index: \equiv_k has finitely many equivalence classes.
- Saturation: if $L \subseteq \Sigma^*$ is defined by a first-order sentence of quantifier rank k over the set model, then

$$w \equiv_k w' \quad \text{implies} \quad w \in L \Leftrightarrow w' \in L.$$

- Congruence: \equiv_k is a semigroup congruence, i.e.

$$\bigwedge_{i \in \{1,2\}} w_i \equiv_k w'_i \quad \text{implies} \quad w_1 w_2 \equiv_k w'_1 w'_2.$$

Before we prove the lemma, we use it to finish the proof of the Trakhtenbrot-Büchi-Elgot Theorem. Suppose that $L \subseteq \Sigma^*$ is definable in mso. By Lemma 2.4, L is first-order definable using set models; let $k \in \{0, 1, \dots\}$ be the quantifier rank of the corresponding first-order sentence. Consider the function

$$w \in \Sigma^* \quad \mapsto \quad \underbrace{\text{equivalence class of } w \text{ under } \equiv_k}_{\text{we call this the rank } k \text{ mso type of } w}$$

By the congruence property from Lemma 2.5, the function is compositional. Therefore, thanks to Lemma 1.3, the image can be equipped with a monoid product so that the function becomes a monoid homomorphism. By the finite index property, the monoid is finite. By the saturation property, the monoid homomorphism recognises the language. We have thus established that L is recognised by a finite monoid, and therefore it is regular.

It remains to prove the lemma.

Proof of Lemma 2.5. The saturation property is an immediate consequence of the definition. The finite index property is also easy: once the vocabulary and the free variables are fixed, then up to logical equivalence there are finitely many formulas of quantifier rank at most k .

We are left with the congruence property. We assume that the reader is familiar with Ehrenfeucht-Fraïssé games³. Let $w_1, w_2, w'_1, w'_2 \in \Sigma^*$. The main reason behind the congruence property is that a set of positions in a concatenation of

³ For an introduction to Ehrenfeucht-Fraïssé games, see [16] Hodges, *Model Theory*, 1993, Section 3.2

two words is the same thing as a pair (set of positions in the first word, set of positions in the second word)⁴. Define

$$(\text{set model of } w_1 w_2) \xrightarrow{f} (\text{set model of } w_1) \times (\text{set model of } w_2)$$

to be the corresponding bijection; likewise define a bijection f' for w'_1 and w'_2 . Using these bijections, we will combine Duplicator's winning strategies for the Ehrenfeucht-Fraïssé games corresponding to the assumption

$$\underbrace{w_1 \equiv_k w'_1 \quad w_2 \equiv_k w'_2}_{\text{small games}},$$

to get a winning strategy for Duplicator in the Ehrenfeucht-Fraïssé game corresponding to the conclusion

$$\underbrace{w_1 w_2 \equiv_k w'_1 w'_2}_{\text{big game}}.$$

This is done as follows. Whenever Spoiler makes a move in the big game, then Duplicator uses the bijections to transform this move into a pair of Spoiler moves in the small games. Using Duplicator's winning strategies in the small games, Duplicator gets a pair of responses, and combines them using the bijection into a response in big game.

We now argue that Duplicator's strategy in the big game, as described above, is winning. This argument will depend on the set properties (inclusion, emptiness, etc.) that are in the vocabulary of the set model, and it would fail if the relation $x = \emptyset$ would be removed from the vocabulary (even though this relation is first-order definable in terms of the remaining ones). Suppose that all k rounds have been played, resulting in tuples

$$\underbrace{a_1, \dots, a_k}_{\text{sets of positions in } w_1 w_2} \quad \text{and} \quad \underbrace{a'_1, \dots, a'_k}_{\text{sets of positions in } w'_1 w'_2}.$$

We need to show that the tuples satisfy the same quantifier-free formulas in the corresponding set models. We only prove

$$\underbrace{a_i < a_j}_{\text{in } w_1 w_2} \quad \text{iff} \quad \underbrace{a'_i < a'_j}_{\text{in } w'_1 w'_2} \quad \text{for every } i, j \in \{1, \dots, k\}, \quad (2.1)$$

the other relations from the vocabulary are left as an exercise. For $i \in \{1, \dots, k\}$

⁴ This explains why monadic second-order logic, and not general second-order logic, is used. In general second-order logic, one can use relations of higher arities, e.g. binary relations. A binary relation over positions in $w_1 w_2$ is not the same thing as two binary relations over positions, in w_1 and w_2 , respectively. See Exercise 30.

and $\ell \in \{1, 2\}$, define $a_{i,\ell}$ to be the ℓ -th coordinate of $f(a_i)$. Likewise we define $a'_{i,\ell}$, using f' . The main observation is that the relation $<$ in $w_1 w_2$ reduces to a quantifier free properties of the corresponding elements in w_1 and w_2 , as explained below:

$$\underbrace{a_i < a_j}_{\text{in } w_1 w_2} \quad \text{iff} \quad \left(\underbrace{a_{i,1} < a_{j,1}}_{\text{in } w_1} \text{ and } \underbrace{a_{j,2} = \emptyset}_{\text{in } w_2} \right) \text{ or } \left(\underbrace{a_{j,1} = \emptyset}_{\text{in } w_1} \text{ and } \underbrace{a_{i,2} < a_{j,2}}_{\text{in } w_2} \right).$$

Since Duplicator's strategy in the big game was obtained by combining winnngs strategies for Duplicator in the small games, we know that the truth value of the right side of the equivalence does not change when we go from $a_{i,\ell}$ to $a'_{i,\ell}$, and therefore the same is true for the left side of the equivalence⁵. \square

Exercises

Exercise 27. (3) Define \mathcal{U}_2 to be the monoid with elements $\{a, b, 1\}$ and product

$$xy = \begin{cases} y & \text{if } x = 1 \\ x & \text{otherwise.} \end{cases}$$

Show that every finite monoid can be obtained from \mathcal{U}_2 by applying Cartesian products, quotients (under semigroup congruences), sub-semigroups, and the powerset construction from Exercise 4.

Exercise 28. (2) For an alphabet Σ , consider the model where the universe is Σ^* , and which is equipped with the following relations:

$$\underbrace{x \text{ is a prefix of } y}_{\text{binary relation}} \quad \underbrace{\text{the last letter of } x \text{ is } a \in \Sigma}_{\text{one unary relation for each } a \in \Sigma}$$

Show that a language $L \subseteq \Sigma^*$ is regular if and only if there is a first-order formula $\varphi(x)$ over the above vocabulary such that L is exactly the words that satisfy $\varphi(x)$ in the above structure.

Exercise 29. (1) What happens if the prefix relation in Exercise 28 is replaced by the infix relation?

⁵ As mentioned previously, this part of the argument depends on the choice of relations in the definition of the set model, since the relation $x = \emptyset$ is used when talking about $x < y$.

Exercise 30. (1) Consider the fragment of second-order logic where one can quantify over: elements, unary relations, and binary relations. (This fragment is expressively complete.) Define \equiv_k to be the equivalence on Σ^* which identifies two words if they satisfy the same sentences from the above fragment of second-order logic, up to quantifier rank k . Show that this equivalence relation has finite index, but it is not a semigroup congruence.

2.2 Aperiodic semigroups and first-order logic

We now begin the study of fragments of mso logic. We show that such fragments correspond to structural restrictions on finite monoids. The first – and arguably most important – fragment is first-order logic. This fragment will be described in the Shützenberger-McNaughton-Papert-Kamp Theorem. One part of the theorem says that a language is first-order definable if and only if it is recognised by a finite monoid M which satisfies

$$\underbrace{a^! = a^! a}_{\text{for all } a \in M},$$

a monoid or semigroup which satisfies this is called *aperiodic*

where $! \in \{1, 2, \dots\}$ is the idempotent exponent from the Idempotent Power Lemma. In other words, in an aperiodic monoid the sequence a, a^2, a^3, \dots is eventually constant, as opposed to having some non-trivial periodic behaviour.

Example 5. Consider the parity language $(aa)^* \subseteq a^*$. We claim that this language is not recognised by any aperiodic monoid, and therefore it is not first-order definable. Of course the same is true for the complement of the language, which was discussed in Example 4.

Suppose that the parity language is recognised by a homomorphism h into some finite monoid M . By Theorem 1.7, there is a surjective homomorphism from the image of h , which is a sub-monoid of M , into the syntactic monoid. In other words, the syntactic monoid is a quotient (i.e. image under a surjective homomorphism) of a sub-monoid of M . Since the syntactic monoid is the two-element group, which is not aperiodic, and since aperiodic monoids are closed under taking quotients and sub-monoids, it follows that M cannot be aperiodic.

The above argument shows that a regular language is first-order definable if and only if its syntactic monoid is aperiodic. Since the syntactic monoid can be computed, and aperiodicity is clearly decidable, it follows that there is an algorithm which decides if a regular language is first-order definable. \square

Apart from first-order logic and aperiodic monoids, the Shützenberger-McNaughton-Papert-Kamp Theorem considers also linear temporal logic and star-free regular expressions, so we begin by defining those.

Linear temporal logic Linear temporal logic (LTL) is an alternative to first-order logic which does not use quantifiers. The logic LTL only makes sense for structures equipped with a linear order; hence the name.

Definition 2.6 (Linear temporal logic). Let Σ be a finite alphabet. Formulas of *linear temporal logic* (LTL) over Σ are defined by the following grammar:

$$\underbrace{a \in \Sigma}_{\substack{\text{the current} \\ \text{position has} \\ \text{label } a}} \quad \varphi \wedge \psi \quad \varphi \vee \psi \quad \neg \varphi \quad \underbrace{\varphi \mathbf{U} \psi}_{\varphi \text{ until } \psi} .$$

The semantics for LTL formulas⁶ is a ternary relation, denoted by

$$\underbrace{w,}_{\substack{\text{word} \\ \text{in } \Sigma^+}} \underbrace{x}_{\substack{\text{position} \\ \text{in } w}} \models \underbrace{\varphi}_{\text{LTL formula}} ,$$

which is defined as follows. A formula $a \in \Sigma$ is true in positions with label a . The semantics of Boolean combinations are defined as usual. For formulas of the form $\varphi \mathbf{U} \psi$, the semantics⁷ are

$$w, x \models \varphi \mathbf{U} \psi \stackrel{\text{def}}{=} \underbrace{\exists y \ x < y \wedge}_{\substack{\text{there is some} \\ \text{position strictly} \\ \text{after } x}} \underbrace{w, y \models \psi}_{\text{which satisfies } \psi} \wedge \underbrace{\forall z \ x < z < y \Rightarrow w, z \models \varphi.}_{\text{and such that all intermediate positions satisfy } \varphi}$$

We say that an LTL formula is true in a word, without specifying a position, if the formula is true in the first position of that word; this only makes sense for nonempty words. A language $L \subseteq \Sigma^*$ is called LTL *definable* if there is an LTL formula φ that defines the language on nonempty words:

$$w \in L \quad \text{iff} \quad w \models \varphi \quad \text{for every } w \in \Sigma^+ .$$

For example, the formula $a \mathbf{U} b$ defines the language $\Sigma^* a \Sigma^* b \Sigma^*$.

Example 6. To get a better feeling for LTL, we discuss some extra operators that can be defined using until. We write \perp for any vacuously false formula,

⁶ The semantics make sense for any linear order, possibly infinite, with positions coloured by Σ .

⁷ We use a variant of the until operator which is sometimes called *strict until*.

e.g. $a \wedge \neg a$, likewise \top denotes any vacuously true formula. Here are some commonly used extra operators:

$$\underbrace{X\varphi \stackrel{\text{def}}{=} \perp U\varphi}_{\text{the next position satisfies } \varphi}, \quad \underbrace{F\varphi \stackrel{\text{def}}{=} \top U\varphi}_{\text{some strictly later position satisfies } \varphi}, \quad \underbrace{\varphi U^*\psi \stackrel{\text{def}}{=} \psi \vee (\varphi U\psi)}_{\text{non-strict until}}$$

Similarly, we define a non-strict version of the operator F , with $F^*\varphi = \varphi \vee F\varphi$. For example, the formula

$$F^*(a \wedge \underbrace{\neg F\top}_{\text{last position}})$$

says that the last position in the word has label a . \square

Almost by definition, every LTL definable language is also first-order definable. Indeed, by unfolding the definition, one sees that for every LTL formula there is a first-order formula $\varphi(x)$ that is true in the same positions.

Star-free languages. Apart from first-order logic, aperiodic monoids, and LTL, another equivalent formalism is going to be star-free expressions. As the name implies, star-free expressions cannot use Kleene star. However, in exchange they are allowed to use complementation (without star and complementation one could only define finite languages). For an alphabet Σ , the star-free expressions are those that can be defined using the following operations on languages:

$$\underbrace{a \in \Sigma}_{\substack{\text{the language that} \\ \text{contains only} \\ \text{the word } a}}, \quad \underbrace{\emptyset}_{\substack{\text{empty} \\ \text{language}}}, \quad \underbrace{LK}_{\text{concatenation}}, \quad \underbrace{L + K}_{\text{union}}, \quad \underbrace{\bar{L}}_{\substack{\text{complementation} \\ \text{with respect} \\ \text{to } \Sigma^*}}$$

Note that the alphabet needs to be specified to give meaning to the complementation operation. A language is called *star-free* if it can be defined by a star-free expression.

Example 7. Assume that the alphabet is $\{a, b\}$. The expression $\bar{\emptyset}$ describes the full language $\{a, b\}^*$. Therefore

$$\bar{\emptyset} \cdot a \cdot \bar{\emptyset}$$

describes all words with at least one a . Taking the complement of the above expression, we get a star-free expression for the language b^* . \square

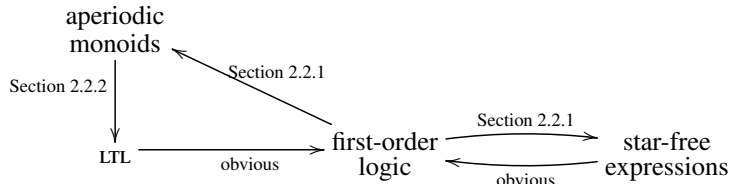
Like for LTL formulas, almost by definition every star-free expression describes a first-order definable language. This is because to every star-free expression one can associate a first-order formula $\varphi(x, y)$ which selects a pair of positions $x \leq y$ if and only if the corresponding infix (including x and y) belongs to the language described by the expression.

Equivalence of the models. The Shützenberger-McNaughton-Papert-Kamp Theorem says that all of the models discussed so far in this section are equivalent.

Theorem 2.7 (Shützenberger-McNaughton-Papert-Kamp). *The following are equivalent⁸ for every $L \subseteq \Sigma^*$:*

- (1) *recognised by a finite aperiodic monoid;*
- (2) *star-free;*
- (3) *first-order definable;*
- (4) *LTL definable.*

The rest of Section 2.2 is devoted to proving the theorem, according to the following plan:



2.2.1 From first-order logic to aperiodic monoids and star-free expressions

In this section, we prove two inclusions: first-order logic is contained in both aperiodic monoids and star-free expressions.

The proof uses a compositionality analysis of Ehrenfeucht-Fraïssé games, similar to the mso-to-regular implication of the Trakhtenbrot-Büchi-Elgot Theorem. For $k \in \{0, 1, 2, \dots\}$ and $w, w' \in \Sigma^*$, we write

$$w \equiv_k w'$$

if the ordered models of w and w' satisfy the same first-order formulas of quantifier rank at most k . We use the blue colour to distinguish the equivalence from

⁸ This theorem combines three equivalences.

The equivalence aperiodic monoids and star-free expressions was shown in [28] Schützenberger, “On finite monoids having only trivial subgroups”, 1965, p. 190

The equivalence of star-free expressions and first-order logic was shown in [21] McNaughton and Papert, *Counter-free automata*, 1971, Theorem 10.5

The equivalence of first-order logic and LTL, not just for finite words, was shown in [17] Kamp, “Tense Logic and the Theory of Linear Order”, 1968

the one for mso that was used in Section 2.1. By the same argument as in Section 2.1, one shows that

$$w \in \Sigma^* \quad \mapsto \quad \underbrace{\text{equivalence class of } w \text{ under } \equiv_k}_{\text{we call this the rank } k \text{ FO type of } w}$$

is a monoid homomorphism, into a finite monoid, which recognises every language that can be defined by a first-order sentence of quantifier rank k .

We will now show that every equivalence class of \equiv_k is defined by a star-free expression, and that the monoid of equivalence classes is aperiodic. In both cases, we use the following lemma, which characterises equivalence classes of \equiv_{k+1} in terms of equivalence classes of \equiv_k by using only Boolean combinations and concatenation.

Lemma 2.8. *Let $w, w' \in \Sigma^*$. Then $w \equiv_{k+1} w'$ if and only if*

$$w \in LaK \Leftrightarrow w' \in LaK$$

holds for every $a \in \Sigma$ and every $L, K \subseteq \Sigma^$ which are equivalence classes of \equiv_k .*

Proof A simple Ehrenfeucht-Fraïssé argument. The letter a corresponds to the first move of player Spoiler. \square

We now use the lemma to prove the inclusion of first-order logic in star-free expressions and aperiodic monoids.

From first-order logic to star-free. It is enough to show that every equivalence class of \equiv_k is star-free. This is proved by induction on k . For the induction base, there is only one equivalence class, namely all words, which is clearly star-free. Consider now the induction step. Consider an equivalence class M of \equiv_{k+1} . Let be X the set of triples

$$\underbrace{L}_{\substack{\text{equivalence} \\ \text{class of } \equiv_k}} \quad \underbrace{a}_{\substack{\text{letter in } \Sigma}} \quad \underbrace{K}_{\substack{\text{equivalence} \\ \text{class of } \equiv_k}}.$$

By Lemma 2.8, if L, a, K are as above, then M is either contained in LaK or disjoint with LaK . Therefore, the equivalence class M is equal to the following Boolean combination of concatenations

$$\bigcap_{\substack{(L,a,K) \in X \\ M \subseteq LaK}} LaK \quad \cap \quad \bigcap_{\substack{(L,a,K) \in X \\ LaK \cap M = \emptyset}} \overline{LaK}.$$

This is a star-free expression, if we assume that L and K are described by star-free expressions from the induction assumption. Since every first-order definable language is a finite union of equivalence classes of \equiv_k for some k , the result follows.

From first-order logic to aperiodic monoids. As we have argued before, every first-order definable language is recognised by the finite monoid of equivalence classes of \equiv_k , for some k . It remains to show that this monoid is aperiodic. To prove this, we use Lemma 2.8 and induction on k to show that

$$w^{2^k-1} \equiv_k w^{2^k} \quad \text{for every } w \in \Sigma^* \text{ and } k \in \{1, 2, \dots\}.$$

2.2.2 From aperiodic monoids to LTL

The last, and most important, step in the proof is constructing an LTL formula based on an aperiodic monoid⁹. In this part of the proof, semigroups will be more convenient than monoids. We will use LTL to define colourings, which are like languages but with possibly more than two values: a function from Σ^+ to a finite set of colours is called LTL definable if for every colour, the words sent that colour are an LTL definable language. For example, a semigroup homomorphism into a finite semigroup is a colouring.

Lemma 2.9. *Let S be a semigroup, and let $\Sigma \subseteq S$. The colouring*

$$w \in \Sigma^+ \quad \mapsto \quad \text{product of } w$$

is LTL definable.

By applying the lemma to the special case of S being a monoid, and substituting each monoid element for the letters that get mapped to it in the recognising homomorphism, we immediately get the implication from finite aperiodic monoids to LTL.

It remains to prove the lemma. The proof is by induction on two parameters: the size of the semigroup S , and the size of the subset Σ . These parameters are ordered lexicographically, with the size of S being more important. Without loss of generality, we assume that Σ generates S , i.e. every element of S is the product of some word in Σ^+ .

The induction base is treated in the following claim.

Claim 2.10. *If either S or Σ has size one, then Lemma 2.9 holds.*

Proof If the semigroup has one element, there is nothing to do, since colourings with one possible colour are clearly LTL definable. Consider the case when the Σ contains only one element $a \in S$. By aperiodicity, the sequence

$$a, a^2, a^3, \dots$$

⁹ The proof in this section is based on [35] Wilke, ‘‘Classifying Discrete Temporal Properties’’, 1999, Section 2

is eventually constant, because all powers bigger than the threshold ! give the same result. The product is therefore easily seen to be an LTL definable colouring. \square

We are left with the induction step. For $c \in S$, consider the function

$$a \in S \mapsto ca \in S.$$

Claim 2.11. *If $a \mapsto ca$ is a permutation of S , then it is the identity.*

Proof Suppose that $a \mapsto ca$ is a permutation of S , call it π . By aperiodicity,

$$\pi^! \circ \pi = \pi^!.$$

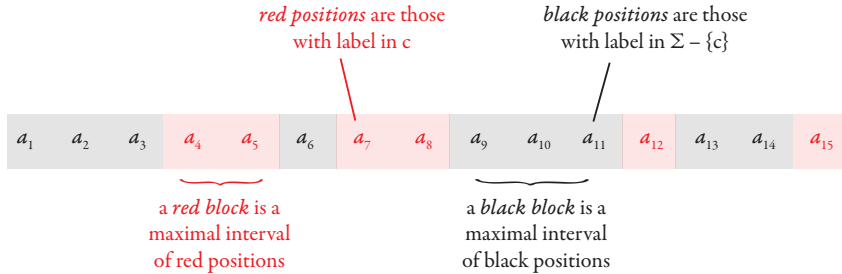
Since permutations form a group, it follows that π is the identity. \square

If the function $a \mapsto ca$ is the identity for every $c \in \Sigma$, then the product of a word is the same as its last letter; and such a colouring is clearly LTL definable. We are left with the case when there is some $c \in \Sigma$ such that $a \mapsto ca$ is not the identity. Fix this c for the rest of the proof. Define T to be the image of the function $a \mapsto ca$, this is a proper subset of S by assumption on c .

Claim 2.12. *T is a sub-semigroup of S .*

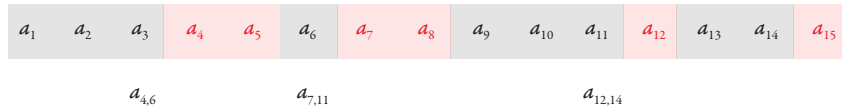
Proof If two semigroup elements have prefix c , then the same is true for their product. \square

In the rest of the proof, we use the following terminology for a word $w \in \Sigma^+$:



We first describe the proof strategy. For each black block, its product can be computed in LTL using the induction assumption on a smaller set of generators. The same is true for red blocks. Define a *red-black block* to be any union of a red block plus the following (non-empty) black block; as illustrated below:

- (2) Take the output of the function in the previous step, and for each red letter (the product of a red block), multiply it with the next letter (which is the product of a black block). As a result, we get the values of all red-black blocks which do not begin in the first position. Here is a picture:



The function in this step is clearly an LTL transduction.

By induction assumption on a smaller semigroup, the product operation $T^+ \rightarrow T$ is an LTL colouring. By composing the functions described above with the semigroup product in T , we see that

$$w \in \Sigma^+ \quad \mapsto \quad \text{value of the union of red-black blocks}$$

is an LTL colouring. The values of the (at most two) blocks that do not participate in above union can also be computed using LTL colourings, and therefore the product of the entire word can be computed.

Exercises

Exercise 31. (2) Show that for every sentence of first-order logic, there is a sentence that is equivalent on finite words, and which uses at most three variables (but these variables can be repeatedly quantified).

Exercise 32. (2) Show that the following are equivalent for a finite semigroup:

- (1) aperiodic;
- (2) \mathcal{H} -trivial, which means that all \mathcal{H} -classes are singletons;
- (3) no sub-semigroup is a non-trivial group.

Exercise 33. (1) Consider the successor model of a word $w \in \Sigma^*$, which is defined like the ordered model, except that instead of $x < y$ we have $x + 1 = y$. Given an example of a regular language that is first-order definable using the ordered model, but not using the successor model.

Exercise 34. (1) Show two languages which have the same syntactic monoid, and such that only one of them is first-order definable in the successor model. In particular, one of the closure properties from Exercise 14 must fail for this logic.

Exercise 35. (3) Let Σ be a finite alphabet and let \vdash, \dashv be fresh symbols. For $k, \ell \in \{0, 1, \dots\}$, we say that $w, w' \in \Sigma^*$ are (k, ℓ) -locally equivalent if

$$\vdash w \dashv \text{ has at least } i \text{ occurrences of infix } v \quad \text{iff} \quad \vdash w' \dashv \text{ has at least } i \text{ occurrences of infix } v$$

holds for every $i \in \{0, \dots, k\}$ and every $v \in \Sigma^*$ of length at most ℓ . Show that $L \subseteq \Sigma^*$ is first-order definable in the successor model if and only if it is a union of equivalence classes of (k, ℓ) -local equivalence, for some k, ℓ .

Exercise 36. (1) Let $\Gamma \subseteq \Sigma$ and let $L \subseteq \Gamma^*$. If L is definable in LTL, then the same is true for

$$\{w \in \Sigma^* : L \text{ contains the maximal prefix of } w \text{ which uses only letters from } \Sigma\}.$$

Exercise 37. (2) Consider $\text{LTL}[X]$, i.e. the fragment of LTL where the only operator is X . Show that this fragment is equal to the definite languages from Exercise 19.

Exercise 38. (1) Show that if a language is first-order definable in the successor model, then the syntactic semigroup satisfies the following equality

$$eafbecf = ecfbeaf \quad \text{for all } \underbrace{e, f, a, b, c}_{\text{idempotents}}$$

Exercise 39. (3) Show that the identity in Exercise 38, together with aperiodicity, is equivalent to first-order definability in the successor model.

Exercise 40. (3) Consider the following extension of LTL with group operators. Suppose that G is a finite group, and let

$$\{\varphi_g\}_{g \in G},$$

be a family of already defined formulas such that every position in an input

word is selected by exactly one formula φ_g . Then we can create a new formula, which is true in a word of length n if

$$1 = g_1 \cdots g_n,$$

where $g_i \in G$ is the unique group element whose corresponding formula selects position i . Show that this logic defines all regular languages.

2.3 Suffix trivial semigroups and linear logic with F only

In the previous section, we showed that first-order logic corresponds to the monoids without groups, which is the same thing as monoids with trivial \mathcal{H} -classes (Exercise 32). What about monoids with trivial suffix classes, prefix classes, or infix classes? Trivial infix classes will be described in Section 2.4. In this section, we give a logical characterisation of trivial suffix classes (of course, a symmetric statement holds for trivial prefix classes).

In the characterisation, we use the fragment of LTL where until is replaced by the following operators

$$\underbrace{\top \cup \varphi}_{F\varphi} \quad \underbrace{\neg F \neg \varphi}_{G\varphi} \quad \underbrace{\varphi \vee F\varphi}_{F^*\varphi} \quad \underbrace{\neg F^* \neg \varphi}_{G^*\varphi}.$$

Since all of the above operators can be defined in terms of F, we write LTL[F] for the resulting logic.

Theorem 2.13. *The following conditions are equivalent for $L \subseteq \Sigma^*$:*

- (1) *Recognised by a finite suffix trivial monoid.*
- (2) *Defined by a finite union of regular expressions of the form*

$$\underbrace{\Sigma_0^* a_1 \Sigma_1^* a_2 \cdots a_n \Sigma_n^*}_{\text{we call such an expression suffix unambiguous}} \quad \text{where } a_i \in \Sigma - \Sigma_i \text{ for } i \in \{1, \dots, n\}.$$

In the above, some of the sets $\Sigma_i \subseteq \Sigma$ might be empty, in which case $\Sigma_i^ = \{\varepsilon\}$.*

- (3) *Defined by a Boolean combination of LTL[F] formulas of the form $F^*\varphi$.*

To see why the formulas in item (3) need to be guarded by F^* , consider the LTL[F] formula a which defines the language “words beginning with a ”. This language is not recognised by any finite suffix trivial monoid.

Proof

(1) \Rightarrow (2) We will show that for every finite suffix trivial monoid M , and every $F \subseteq M$, the language

$$\{w \in M^* : F \text{ contains the product of } w\}$$

is defined by a finite union of suffix unambiguous expressions. It will follow that for every monoid homomorphism into M , the recognised language is defined by a similar expression, with monoid elements substituted by the letters that map to them (such a substitution preserves suffix unambiguity).

It is of course enough to consider the case when F contains only one element, call it $a \in M$. The proof is by induction on the position of a in the suffix ordering.

The induction base is when a is the monoid identity. By Exercise 15, the suffix class of the identity is a group, and a group must be trivial in a suffix trivial monoid. It follows that a word has product a if and only if it belongs to a^* , which is a suffix unambiguous expression.

We now prove the induction step. Consider a word with product a . This word must be nonempty, since otherwise its product would be the identity. Let i be the maximal position in the word such that the suffix starting in i also has product a . By suffix triviality, every position $< i$ is labelled by a letter in

$$\Sigma_0 = \{b \in M : ba = a\}.$$

Let b be the product of the suffix that starts after i , not including i , and let c be the label of position i . By choice of i , b is a proper suffix of a and $a = cb$. Summing up, words with product a are defined by the expression

$$\bigcup_{\substack{b, c \in M \\ b \text{ is a proper suffix of } c \\ a = cb}} \Sigma_0^* c \cdot (\text{words with product } b),$$

Apply the induction assumption to b , yielding a finite union of suffix unambiguous expressions, and distribute the finite union across concatenation. It remains to justify that the resulting expressions are also suffix unambiguous. This is because none of the expressions that define words with product b can begin with Σ_1^* with $c \in \Sigma_1$, since otherwise we would contradict the assumption that $cb = a \neq b$.

(2) \Rightarrow (3) Since the formulas from item (3) are closed under union, it is enough to show that every suffix unambiguous expression

$$\Sigma_0^* a_1 \Sigma_1^* a_2 \cdots a_n \Sigma_n^*$$

can be defined by a formula as in (3). For $i \in \{0, \dots, n\}$, define L_i to be

the suffix of the above expression that begins with Σ_i^* . By induction on i , starting with n and progressing down to 0, we show that L_i can be defined by a formula φ_i as in item (3). In the induction base, we use the formula

$$\varphi_n = \underbrace{\mathbf{G}^* \bigvee_{a \in \Sigma_n} a}_{\text{all positions have label in } \Sigma_n}.$$

For the induction step, we first define the language $a_i L_i$, using a formula of $\text{LTL}[F]$ (which is not in the shape from item (3)):

$$\psi_i = a_i \wedge (\mathbf{F}\varphi_i) \wedge \mathbf{G} \bigvee_{j>i} \varphi_j.$$

Because the expression is suffix unambiguous, the formula ψ_i selects at most one position in a given input word; this property will be used below. The language L_{i-1} is then defined by

$$\varphi_{i-1} = \underbrace{\mathbf{F}^* \psi_i \wedge \mathbf{G}^* ((\mathbf{F}\psi_i) \Rightarrow \bigwedge_{a \in \Sigma_0} a)}_{\substack{\text{if a position is to the left} \\ \text{of the unique position} \\ \text{satisfying } \psi_i, \text{ then} \\ \text{it has label in } \Sigma_0.}}$$

(3) \Rightarrow (1) Define the rank of a formula in $\text{LTL}[F]$ to be the nesting depth of the operator F . For $k \in \{0, 1, \dots\}$, define \approx_k to be the equivalence relation on Σ^* which identifies two words if they satisfy the same formulas of the form $\mathbf{F}^* \varphi$, where φ has rank at most k . The key observation is the following pumping lemma.

Claim 2.14. *For every $k \in \{0, 1, 2, \dots\}$ we have*

$$w(xy)^i u \approx_k wy(xy)^j u \quad \text{for every } w, x, y, u \in \Sigma^* \text{ and } i, j \geq k.$$

Proof Induction on k . For $k = 0$, we observe that $v \approx_0 v'$ holds if and only if v and v' use the same letters, because the truth value of a rank 0 formulas depends only on the first letter in a word. For the induction step, when going from k to $k + 1$, we need to show that if $i, j \geq k + 1$, then for every suffix of the word on one side of equivalence

$$w(xy)^i u \approx_{k+1} wy(xy)^j u$$

there is a suffix on the other side of the equivalence, such that the two suffixes are equivalent under \approx_k . Suppose first that v is a suffix of the left side. If v is a suffix of $(xy)^{k+1} u$, then the same v is a suffix of the right side. Otherwise, we can use the induction assumption. Consider now a suffix v of the right

side. Here we argue in the same way as previously, except that there is one extra case, when

$$v = z(xy)^i u \quad \text{for some } z \text{ that is a suffix of } y.$$

In this case, the matching suffix on the left side is $z(xy)^k u$. □

By unravelling the definition of the syntactic monoid, in terms of two-sided congruences, we infer from the above claim that for every rank k formula φ of $\text{LTL}[F]$, the syntactic monoid M of $F^*\varphi$ satisfies

$$(xy)^! = y(xy)^! \quad \text{for all } x, y \in M. \quad (2.2)$$

The same is also true for syntactic monoids of Boolean combinations of such formulas. To finish the proof, we observe that property (2.2) is true in a finite monoid if and only if it is suffix trivial. Indeed, if a monoid is suffix trivial, then $(xy)^!$ and $y(xy)^!$ must be in the same suffix class, and hence equal. Conversely, if a, b are in the same suffix class, then there must be some x, y such that $b = xa$ and $a = yb$; it follows that

$$a = y(xy)^! b \stackrel{(2.2)}{=} (xy)^! b = b.$$

□

Exercises

Exercise 41. (1) Let Σ be an alphabet and let $c \notin \Sigma$ be a fresh letter. Show that $L \subseteq \Sigma^+$ satisfies the conditions of Theorem 2.13 if and only if cL is definable in $\text{LTL}[F]$.

2.4 Infix trivial semigroups and piecewise testable languages

In this section, we describe the languages recognised by finite monoids that are infix trivial. For languages recognised by finite infix trivial monoids, a prominent role will be played embeddings (also known as the Higman ordering).

Definition 2.15 (Embedding). We say that $w \in \Sigma^*$ *embeds* in $v \in \Sigma^*$, denoted by $w \hookrightarrow v$, if there is an injective function from positions in w to positions in v , which preserves the order on positions and the labels.

In other words, w embeds in v if and only if w can be obtained from v by removing zero or more positions. For example “ape” embeds into “example”. It is easy to see that embedding is an ordering: it is reflexive, transitive and anti-symmetric (although it will cease to be anti-symmetric for infinite words). We say that a language $L \subseteq \Sigma^*$ is *upward closed* if

$$v \hookrightarrow w \wedge v \in L \Rightarrow w \in L.$$

Symmetrically, we define downward closed languages. The main result about embedding is that it is a well-quasi order, as explained in the following lemma.

Lemma 2.16 (Higman’s Lemma). *For every upward closed $L \subseteq \Sigma^*$ there is a finite subset $U \subseteq L$ such that*

$$L = \underbrace{\{w \in \Sigma^* : v \hookrightarrow w \text{ for some } v \in U\}}_{\text{we call this the upward closure of } U}$$

Here is a logical corollary of Higman’s lemma.

Theorem 2.17. *A language is upward closed if and only if it can be defined in the ordered model by an \exists^* -sentence, i.e. a sentence of the form*

$$\underbrace{\exists x_1 \exists x_2 \cdots \exists x_n}_{\text{only existential quantifiers}} \underbrace{\varphi(x_1, \dots, x_n)}_{\text{quantifier-free}}.$$

Proof Clearly every \exists^* -sentence defines an upward closed language. Higman’s Lemma gives the converse implication, because the upward closure of every finite set is definable by an \exists^* -sentence. \square

Embeddings will also play an important role in the characterisation of languages recognised by monoids that are infix trivial. Before stating the characterisation, we introduce one more definition, namely zigzags¹¹. For languages $L, K \subseteq \Sigma^*$, define a *zigzag between L and K* to be a sequence

$$\underbrace{w_1}_{\in L} \hookrightarrow \underbrace{w_2}_{\in K} \hookrightarrow \underbrace{w_3}_{\in L} \hookrightarrow \underbrace{w_4}_{\in K} \hookrightarrow \underbrace{w_5}_{\in L} \hookrightarrow \underbrace{w_6}_{\in K} \hookrightarrow \cdots.$$

In other words, this is a sequence that is growing with respect to embeddings, and such that odd-numbered elements are in L and even-numbered elements are in K . The zigzag does not need to be strictly growing, but it will be if L and K are disjoint.

We are now ready for the characterisation of infix trivial monoids.

¹¹ Zigzags and the Zigzag Lemma are inspired by Czerwinski et al., “A Characterization for Decidable Separability by Piecewise Testable Languages”, 2017 [10]

Theorem 2.18. *The following conditions are equivalent¹² for every $L \subseteq \Sigma^*$:*

- (1) *recognised by a finite monoid that is infix trivial;*
- (2) *is a finite Boolean combination of upward closed languages;*
- (3) *there is no infinite zigzag between L and its complement.*

We use the name *piecewise testable* for languages as in item (2) of the above theorem. Equivalence of items (2) and (3) is a corollary of the following lemma, when applied to $K = \Sigma^* - L$.

Lemma 2.19 (Zigzag Lemma). *Let $L, K \subseteq \Sigma^*$. The following are equivalent:*

- (1) *there are zigzags between L and K of every finite length;*
- (2) *there is an infinite zigzag between L and K ;*
- (3) *there is no piecewise testable language $M \subseteq \Sigma^*$ such that*

$$\underbrace{L \subseteq M \quad M \cap K = \emptyset.}$$

we say that M separates L and K

Proof

- (1) \Rightarrow (2) Assume that zigzags between L and K can have arbitrarily long finite lengths. Define a directed acyclic graph G as follows. Vertices are words in L , and there is an edge $w \rightarrow v$ if

$$w \leftrightarrow u \leftrightarrow v \quad \text{for some } u \in K.$$

For a vertex $v \in L$ of this graph, define its *potential*

$$\alpha(v) \in \{0, 1, \dots, \omega\}$$

to be the least upper bound on the lengths of paths in the graph that start in v . This can be either a finite number, or ω if the paths have unbounded length.

We first show that some vertex must have potential ω . By assumption on arbitrarily long zigzags, potentials have arbitrarily high values. By definition of the graph, α is monotone with respect to (the opposite of the) embedding, in the following sense:

$$v \hookrightarrow v' \quad \text{implies} \quad \alpha(v) \geq \alpha(v') \quad \text{for every } v, v' \in L.$$

By Higman's Lemma, the language L , like any set of words, has finitely many minimal elements with respect to embedding. By monotonicity, one of these minimal words must therefore have potential ω .

¹² Equivalence of items (1) and (2) was first proved in [32] Simon, "Piecewise testable events", 1975

For the same reason as above, if a word has potential ω , then one of its successors (words reachable in one step in the graph) must also have potential ω ; this is because there are finitely many successors that are minimal with respect to embedding. This way, we can construct an infinite path in the graph which only sees potential ω , as in the König Lemma.

(2) \Rightarrow (3) Suppose that there is a zigzag between L and K of infinite length. Every upward closed set selects either no elements of the zigzag, or all but finitely many elements of the zigzag. It follows that every finite Boolean combination of upward closed sets must contain, or be disjoint with, two consecutive elements of the zigzag. Therefore, such a Boolean combination cannot separate L from K .

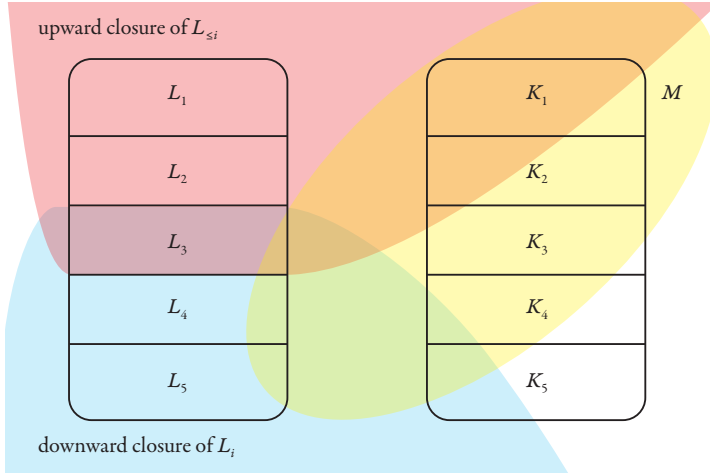
(3) \Rightarrow (1) We prove the contra-positive: if zigzags between L and K have bounded length, then L and K can be separated by a piecewise testable language. For $w \in L$ define its *potential* to be the maximal length of a zigzag between L and K that starts in w ; likewise we define the potential for $w \in K$, but using zigzags between K and L . Define $L_i \subseteq L$ to be the words in L with potential exactly $i \in \{1, 2, \dots\}$, likewise define $K_i \subseteq K$. Our assumption is that the potential is bounded, and therefore L is a finite union of the languages L_i , likewise for K . By induction on $i \in \{0, 1, \dots\}$, we will show that

$$\underbrace{L_1 \cup \dots \cup L_i}_{L_{\leq i}} \quad \text{and} \quad \underbrace{K_1 \cup \dots \cup K_i}_{L_{\leq i}}.$$

can be separated by a piecewise testable language, call it M_i . In the induction base, both languages are empty, and can therefore be separated by the empty language, which is clearly piecewise testable. Consider the induction step, where we find the separator M_i . We will use the following sets

$$\underbrace{L_{\leq i} \uparrow}_{\text{upward closure of } L_{\leq i}} \quad \underbrace{L_i \downarrow}_{\text{downward closure of } L_i} \quad \underbrace{M_i}_{\text{a separator of } K_{< i} \text{ and } L_{< i} \text{ from the induction assumption}}.$$

All of these sets are piecewise testable: the first one is upward closed, the second one is the complement of an upward closed set, and the third one is obtained from the induction assumption. These sets are depicted in the following picture, with $i = 3$:



The set $L_{\leq i} \uparrow$ contains $L_{\leq i}$ by definition. It is also disjoint with K_i , because otherwise there would be some words

$$\underbrace{w}_{L_{\leq i}} \leftrightarrow \underbrace{v}_{K_i},$$

and therefore w would need to have potential $i + 1$. For similar reasons, the set $L_i \downarrow$ is disjoint with K_i and $L_{\leq i-1}$. Putting these facts together, we see that

$$M_i = L_{\leq i} \uparrow - (M - L_i \downarrow)$$

separates $L_{\leq i}$ from $K_{\leq i}$.

□

The Zigzag Lemma proves that equivalence of the conditions about infinite zigzags and piecewise testability in Theorem 2.18. To finish the proof of the Theorem, we show that the syntactic monoid of L is finite and infix trivial (which is the same as saying that some recognising monoid is finite and infix trivial) if and only if there is no infinite zigzag between L and its complement.

Suppose first that the syntactic monoid of L is not finite or infix trivial. If the syntactic monoid is not finite, then the language cannot be piecewise testable, since piecewise testable languages are necessarily regular. Assume therefore that the syntactic monoid is finite but not infix trivial. This means that the syntactic monoid is either not prefix trivial, or not suffix trivial. By symmetry, we only consider the case where the syntactic monoid is not suffix trivial. This means that there exist a, b in the syntactic monoid such that

$$(ab)^! \neq b(ab)^!.$$

By unravelling the definition of the syntactic monoid, the above disequality can be easily used to create an infinite zigzag between L and its complement.

It remains to show that if the syntactic monoid of L is finite and infix trivial, then there is no zigzag between L and its complement. Let M be the syntactic monoid. For $a, b \in M$, define a zigzag between a and b to be a zigzag, over alphabet M , between the words with product a and the words with product b . If M recognises L , then a zigzag between L and its complement can be used, by extraction, to obtain a zigzag between $a \neq b \in M$. The following lemma shows that this cannot happen, thus completing the proof of Theorem 2.18.

Lemma 2.20. *Let M be finite and infix trivial, and let $a, b \in M$. If there is an infinite zigzag between a and b , then $a = b$.*

Proof The proof is by induction on the infix ordering lifted to pairs:

$$(x, y) \leq (a, b) \stackrel{\text{def}}{=} x \text{ is an infix of } a \text{ and } y \text{ is an infix of } b.$$

The induction base is proved the same way as the induction step. Suppose that we have proved the lemma for all pairs $(x, y) < (a, b)$.

Claim 2.21. *If there is an infinite zigzag between a and b , then there exists $n \in \{0, 1, \dots\}$ and monoid elements $\{a_i, b_i, c_i\}_i$ such that*

$$\begin{aligned} a &= c_1 c_2 \cdots c_n \\ b &= b_0 c_1 b_1 c_2 b_2 \cdots b_{n-1} c_n b_n \\ a &= a_0 c_1 a_1 c_2 a_2 \cdots a_{n-1} c_n a_n \end{aligned}$$

and for every $i \in \{0, \dots, n\}$ there is an infinite zigzag between a_i and b_i .

Proof Consider an infinite zigzag between a and b of the form

$$w_1 \hookrightarrow w_2 \hookrightarrow \cdots$$

Let the letters in w_1 be $c_1, \dots, c_n \in M$. For $j \geq 2$, define an *important position* in w_j to be any position that arises by starting in some position of w_1 , and then following the embeddings

$$w_1 \hookrightarrow w_2 \hookrightarrow \cdots \hookrightarrow w_j.$$

By distinguishing the important positions in w_j , we get a factorisation

$$w_j = \underbrace{w_{j,0}}_{M^*} c_1 \underbrace{w_{j,1}}_{M^*} c_2 \cdots c_{n-1} \underbrace{w_{j,n-1}}_{M^*} c_n \underbrace{w_{j,n}}_{M^*}.$$

By definition of important positions, for every $i \in \{0, \dots, n\}$ the following sequence is growing with respect to embedding:

$$w_{2,i} \hookrightarrow w_{3,i} \hookrightarrow \cdots$$

By extracting a subsequence, we can assume that for every $i \in \{0, 1, \dots, n\}$, the above chain is a zigzag between b_i and a_i , for some $b_i, a_i \in M$. This proves the conclusion of the claim. \square

Claim 2.22. *If there is an infinite zigzag between a and b , then there exist $c, c' \in M$ such that $a = cc'$ and $cb = b = bc'$.*

Proof Apply Claim 2.21, yielding monoid elements with

$$\begin{array}{rcccccccc} a & = & & c_1 & & c_2 & & \cdots & & c_n \\ b & = & b_0 & c_1 & b_1 & c_2 & b_2 & \cdots & b_{n-1} & c_n & b_n \\ a & = & a_0 & c_1 & a_1 & c_2 & a_2 & \cdots & a_{n-1} & c_n & a_n \end{array}$$

For every $i \in \{0, \dots, n\}$, we can see that $(b_i, a_i) \leq (b, a)$. If the inclusion is strict, then the induction assumption of the lemma yields $b_i = a_i$. Otherwise, the inclusion is not strict, and therefore

$$(a_i, b_i) = (a, b).$$

If the inclusion is strict for all i , then the third and second rows in the conclusion of Claim 2.21 are equal, thus proving $a = b$, and we are done. Otherwise, there is some $i \in \{0, \dots, n\}$ such that $(b_i, a_i) = (b, a)$. By infix triviality, every interval in the second row that contains i will have product b . It follows that

$$\underbrace{c_j b = b}_{\text{for all } j \leq i} \quad \underbrace{b c_j = b}_{\text{for all } j > i}$$

It is now easy to see that the conclusion of the claim holds if we define c to be the prefix of $a = c_1 \cdots c_n$ that ends with c_i , and define c' to be the remaining suffix. \square

Apply the above claim, and a symmetric one with the roles of a and b swapped, yielding elements c, c', d, d' such that

$$a = cc' \quad cb = b = bc' \quad b = dd' \quad da = a = d'. \quad (2.3)$$

We can now prove the conclusion of the lemma:

$$a \stackrel{(2.3)}{=} (dc)^! (c'd')^! \underbrace{=}_{\text{infix triviality}} (cd)^! (d'c')^! \stackrel{(2.3)}{=} b.$$

\square

Exercises

Exercise 42. (2) Prove Higman's Lemma.

Exercise 43. (2) Give a polynomial time algorithm, which inputs two non-deterministic automata, and decides if their languages can be separated by a piecewise testable language.

Exercise 44. (1) Consider ω -words, i.e. infinite words of the form

$$a_1 a_2 \cdots \quad \text{where } a_1, a_2, \dots \in \Sigma.$$

Embedding naturally extends to ω -words (in fact, any labelled orders). Show that the embedding on ω -words is also a well-quasi order, i.e. every upward closed set is the upward closure of finitely many elements.

2.5 Two-variable first-order logic

We finish this chapter with one more monoid characterisation of a fragment of first-order logic. A corollary of the equivalence of first-order logic and LTL (or of the equivalence of first-order logic and star-free expressions) is that, over finite words, first-order logic is equivalent to its three variable fragment. What about one or two variables?

It is an easy exercise to show that first-order logic with one variable defines exactly the languages that are recognised by monoids that are aperiodic and commutative:

$$a^! = a^{!+1} \quad ab = ba \quad \text{for all } a, b.$$

The more interesting case is first-order logic with two variables, which we denote by FO². This logic is characterised in the following theorem.

Theorem 2.23. *For a language $L \subseteq \Sigma^*$, the following are equivalent:*

- (1) *Definable in two variable first-order logic;*
- (2) *Recognised by a finite monoid M with the following property¹³: M is a aperiodic, and if an infix class $J \subseteq M$ contains an idempotent, then J is a sub-semigroup of M .*

¹³ This property appears in [29] Schützenberger, "Sur Le Produit De Concatenation Non Ambigu", 1976 where it is used to characterise certain unambiguous regular expressions, see Exercise 49.

We use the name DA for the monoids (more generally, finite semigroups) that satisfy the property in item (2). In the exercises, we add several other equivalent conditions for the above theorem, including the temporal logic $\text{LTL}[F, F^{-1}]$ and the following fragment of first-order logic:

(definable by a $\exists^*\forall^*$ -sentence) \cap (definable by a $\forall^*\exists^*$ -sentence).

The rest of Section 2.5 is devoted to proving the theorem. We begin with an equational description of DA . (A stronger equational description is given in Exercise 45.)

Lemma 2.24. *A finite semigroup S is in DA if and only if it satisfies:*

$$\underbrace{w^! = w^!vw^!}_{\text{the equality means that}} \quad \text{for all } w, v \in M^* \text{ with } v \leftrightarrow w.$$

the two words have the same product

Proof We first prove that the identity implies that S is DA . Let e be an idempotent. We need to show that if a, b are infix equivalent to e , then the same is true for ab . Because a, b are infixes of e , and e is an idempotent, one can find word w in S^+ which has product e , and where both a and b appear. In particular, $ab \leftrightarrow w$. By the identity in the lemma, we know that $e = eabe$, and therefore ab is an infix of e .

We now show that if S is in DA , then the identity is satisfied. Let $v \leftrightarrow w$ be as in the identity. Let e be the product of $w^!$, and let J be the infix class of e . This infix class is a semigroup, by definition of DA . For every letter a that appears in the word w , there is a suffix of $w^!w^!$ which begins with a and has product in J . Let $a' \in J$ be the product of this suffix. Since J is a semigroup, it follows that $ea' \in J$ and therefore also $ea \in J$. Since $ea \in J$ holds for every letter that appears in w , it follows that $eve \in J$. This means that eve is in the \mathcal{H} -class of e , and therefore $e = eve$ by aperiodicity (which is part of the definition of DA), thus establishing the identity. \square

We now prove the theorem.

To prove the implication (1) \Rightarrow (2), we show that for every language definable in FO^2 , its syntactic monoid belongs to DA . By Lemma 2.24 and unravelling the definition of the syntactic monoid, it is enough to show that for every $w_1, w_2, v, w \in \Sigma^*$ and $n \in \{0, 1, \dots\}$, if $v \leftrightarrow w$ then the words

$$w_1w^nw_2 \quad w_1w^nvw^nw_2$$

satisfy the same FO^2 sentences of quantifier rank at most n . This is shown using a simple Ehrenfeucht-Fraïssé argument.

For the implication (2) \Rightarrow (1), we use the following lemma.

Lemma 2.25. *Let M be a monoid in DA , and let $a_1, a_2 \in M$. Then*

$$w \in M^* \quad \mapsto \quad \underbrace{a_1 \cdot (\text{product of } w) \cdot a_2}_{\in M}$$

is a colouring definable in FO^2 , i.e. every inverse image is definable in FO^2 .

If we apply the above lemma to a_1 and a_2 being the monoid identity, we conclude that the product operation is definable in FO^2 . This implies that every language recognised by the monoid is definable in FO^2 , thus proving the implication (1) \Leftarrow (2) in the theorem.

Proof Induction on the following parameters, ordered lexicographically:

- (1) size of M ;
- (2) number of elements that properly extend a_1 in the prefix ordering;
- (3) number of elements that properly extend a_2 in the suffix ordering.

The induction base is when M has one element, in which case the colouring in the lemma is constant, and therefore definable in FO^2 .

Let us also prove another variant of the induction base, namely when induction parameters (2) and (3) are zero, which means that a_1 is maximal in the prefix ordering and a_2 is maximal in the suffix ordering. It follows that, for

$$\mathcal{H}\text{-class of } a_1 a a_2 = \mathcal{H}\text{-class of } a_1 b a_2 \quad \text{for all } a, b \in M.$$

Since DA implies aperiodicity, which implies \mathcal{H} -triviality, the colouring in the statement of the lemma is constant, and therefore definable in FO^2 .

It remains to prove the induction step. Because of the two kinds of induction base that were considered above, we can assume that one of the parameters (2) or (3) is nonzero. By symmetry, assume that a_1 is not maximal in the prefix ordering.

Claim 2.26. *For every $a \in M$, the following is a sub-monoid of M :*

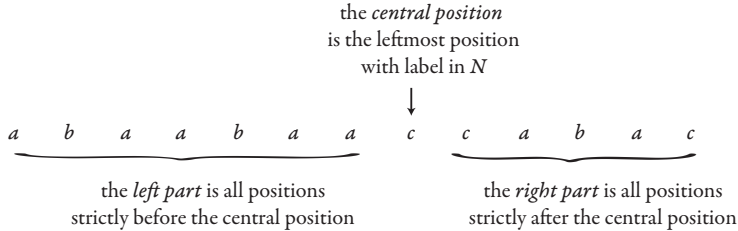
$$\underbrace{\{b \in M : ab \text{ is prefix equivalent to } a\}}_{\text{we call this set the prefix stabiliser of } a}$$

Proof The prefix stabiliser clearly contains the monoid identity. It remains to show that it is closed under composition. Let b, c be in the prefix stabiliser of a . Using the definition of the prefix stabiliser, it is easy to construct a word $w \in M^*$, such that $bc \hookrightarrow w$ and $aw = a$. By Lemma 2.24, it follows that

$$a = aw = aw^! = aw^!bcw^! = abcw^!,$$

which establishes that bc is in the prefix stabiliser of a . □

Let $N \subseteq M$ be the prefix stabiliser of a_1 ; our assumption says that N is a proper subset of M , and by the above claim it is also a sub-monoid. We decompose a word $w \in M^*$ into three parts, as explained in the following picture:



There is an FO^2 formula which selects the central position. Since all labels in the left part are from N , we can use the induction assumption on a smaller monoid to prove that the colouring

$$w \quad \mapsto \quad \text{product of left part}$$

is definable in FO^2 . (When using the induction assumption, we restrict all quantifiers of the formulas from the induction assumption so that they quantify over positions that are to the left of the central position.) Let c be the product of the prefix up to and including the central position; as we have shown above, this product can be computed in FO^2 . By definition of the central position, we know that a_1 is a proper prefix of a_1c , and therefore we can use the induction assumption to prove that

$$w \quad \mapsto \quad a_1c \cdot (\text{product of right part}) \cdot a_2$$

is a colouring definable in FO^2 . The conclusion of the lemma follows. \square

Exercises

Exercise 45. (2) Show that a semigroup belongs to DA if and only if it satisfies the identity

$$(ab)^! = (ab)^!a(ab)^! \quad \text{for all } a, b.$$

Exercise 46. (2) Show that FO^2 has the same expressive power as $\text{LTL}[F, F^{-1}]$, which is the extension of $\text{LTL}[F]$ with the following past operator:

$$w, x \models F^{-1}\varphi \stackrel{\text{def}}{=} \exists y y < x \wedge w, y \models \varphi.$$

Exercise 47. (3) Define the *syntactic ordering* on the syntactic monoid, which depends on the accepting set F , as follows:

$$a \leq b \stackrel{\text{def}}{=} \forall x, y \in M \ xay \in F \Rightarrow xby \in F.$$

Show that a language can be defined by a first-order sentence of the form

$$\underbrace{\exists x_1 \cdots \exists x_n \forall y_1 \cdots \forall y_m}_{\text{such a formula is called an } \exists^* \forall^* \text{-sentence}} \overbrace{\varphi(x_1, \dots, x_n, y_1, \dots, y_m)}^{\text{quantifier-free}}$$

if and only if

$$w^! \leq w^! v w^! \quad \text{for all } \underbrace{v \hookrightarrow w}_{\text{Higman ordering}}$$

Hint¹⁴: use Exercise 26.

Exercise 48. (3) Show that L is definable in FO^2 if and only both L and its complement can be defined using $\exists^* \forall^*$ -sentences.

Exercise 49. (2) We say that a regular expression

$$\Sigma_0^* a_1 \Sigma_1^* \cdots \Sigma_{n-1}^* a_n \Sigma_n^*$$

is *unambiguous* if every word w admits at most one factorisation

$$w = w_0 a_1 w_1 \cdots w_{n-1} a_n w_n \quad \text{where } w_i \in \Sigma_i^* \text{ for all } i \in \{1, \dots, n\}.$$

Show that a language is a finite disjoint union of unambiguous expressions if and only if its syntactic monoid of L is in DA ¹⁵.

¹⁴ An effective characterisation of $\exists^* \forall^*$ -sentence was first given in [2] Arfi, “Polynomial Operations on Rational Languages”, 1987, Theorem 3. The proof was simplified in [22] Pin and Weil, “Polynomial closure and unambiguous product”, 1997, Theorem 5.8 The solution which uses Exercise 26 is based on [22]. Characterisations of fragments of first-order logic such as $\exists^* \forall^*$ are widely studied, see [23] Place and Zeitoun, “Going Higher in First-Order Quantifier Alternation Hierarchies on Words”, 2019

¹⁵ This exercise is based on [29] Schützenberger, “Sur Le Produit De Concatenation Non Ambigu”, 1976

3

Infinite words

In this chapter, we study infinite words.

In Section 3.1, we begin with the classical model of infinite words, namely ω -words. In ω -word, the positions are ordered like the natural numbers. We show how the structure of finite semigroups, which was developed in Section 1.2, can be applied to prove McNaughton's Theorem about determinisation of ω -automata.

In Section 3.2, we move to more general infinite words, where the positions can be any countable linear order, e.g. the rational numbers. For this kind of infinite words, we define a suitable generalisation of semigroups, and show that it has the same expressive power as monadic second-order logic.

3.1 Determinisation of Büchi automata for ω -words

An ω -word is a function from the natural numbers to some alphabet Σ . We write Σ^ω for the set of all ω -words over alphabet Σ . To recognise properties of ω -words, we use Büchi automata.

Definition 3.1 (Büchi automata). The syntax of a *nondeterministic Büchi automaton* is the same as the syntax of a nondeterministic finite automaton for finite words, namely it consists of:

$$\underbrace{Q}_{\text{states}} \quad \underbrace{\Sigma}_{\substack{\text{input} \\ \text{alphabet}}} \quad \underbrace{I, F \subseteq Q}_{\substack{\text{initial and} \\ \text{final states}}} \quad \underbrace{\delta \subseteq Q \times \Sigma \times Q}_{\text{transition relation}}$$

The difference, with respect to nondeterministic automata on finite words, is in the semantics: a word in Σ^ω is accepted by the automaton if there exists a run which begins in an initial state, and which satisfies the *Büchi condition*:

some accepting state appears infinitely often in the run.

A *deterministic Büchi automaton* is the special case when there is one initial state, and the transition relation is a function from $Q \times \Sigma$ to Q .

The following example shows that deterministic Büchi automata are weaker than than nondeterministic ones.

Example 8. Consider the language of ω -words over alphabet $\{a, b\}$ where letter a appears finitely often. This language is recognised by a nondeterministic Büchi automaton as in the following picture:



The idea is that the automaton nondeterministically guesses some position which will not be followed by any a letters; this guess corresponds to the horizontal transition with label b in the picture.

This language is not recognised by any deterministic Büchi automaton. Toward a contradiction, imagine a hypothetical deterministic Büchi automaton which recognises the language. Run this automaton on b^ω . Since a appears finitely often in this ω -word, the corresponding run (unique by determinism) must use an accepting state in some finite prefix. Extend that finite prefix by appending ab^ω . Again, the word must be accepted, so an accepting state must be eventually visited after the first a . By repeating this argument, we get a word which has infinitely many a 's and where the (unique) run of the deterministic automaton sees accepting states infinitely often; a contradiction. \square

The above shows that languages recognised by deterministic Büchi automata are not closed under Boolean combinations. This turns out to be the only limitation of the model, as shown in the following theorem.

Theorem 3.2. *If a language of ω -words is recognised by a nondeterministic Büchi automaton, then it is a Boolean combination of languages recognised by deterministic Büchi automata¹.*

¹ A Boolean combination of deterministic Büchi automata is the same thing as what is known as a *deterministic Muller automaton*. Therefore, the theorem is the same McNaughton's Theorem,

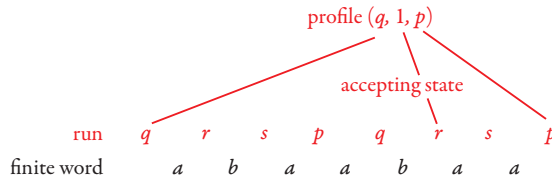
The converse implication in the above theorem is also true, which is left as an exercise for the reader (Exercise 51). A language is called ω -regular if it is recognised by a nondeterministic Büchi automaton, or equivalently, by a Boolean combination of deterministic Büchi automata. The ω -regular languages are closed under Boolean combination thanks to the deterministic characterisation. The original application of Büchi automata was Büchi's proof² that they recognise exactly the same languages of ω -words as monadic second-order logic; this application is a simple corollary of Theorem 3.2, see Exercise 55.

There are several combinatorial proofs for the determinisation result in Theorem 3.2. In this section, we present an algebraic proof, which leverages the structural theory of finite semigroups developed in Section 1.2.

Let \mathcal{A} be a nondeterministic Büchi automaton, with states Q and input alphabet Σ . For an ω -word, define its ω -type to be the set of states from which the word is accepted. We also define the type for finite words, but here we need to store a bit more information. For a run of the automaton over a finite word, define the *profile* of the run to be the triple (q, i, p) where q is the source state of the run, p is the target state of the run, and

$$i = \begin{cases} 0 & \text{if the run does not use any accepting state} \\ 1 & \text{if the run uses some accepting state.} \end{cases}$$

Here is a picture of a run with its profile:



Define the *type* of a finite word $w \in \Sigma^+$ to be the set of profiles of runs over this word. It is not hard to see that the function

$$w \in \Sigma^+ \quad \mapsto \quad \text{type of } w \in \underbrace{\mathbf{P}(Q \times \{0, 1\} \times Q)}_S$$

[20] McNaughton, "Testing and generating infinite sequences by a finite automaton", 1966, p. 524

which says that nondeterministic Büchi automata can be determinised into deterministic Muller automata.

² [6] Büchi, "On a decision method in restricted second order arithmetic", 1962

is a semigroup homomorphism, with a naturally defined semigroup structure on S . The following lemma shows that types and ω -types are compatible.

Lemma 3.3. *If $w_i \in \Sigma^+$ and $v_i \in \Sigma^+$ have the same type for every $i \in \{1, 2, \dots\}$, then $w_1 w_2 \dots \in \Sigma^\omega$ and $v_1 v_2 \dots \in \Sigma^\omega$ have the same ω -type.*

Proof By substituting parts of an accepting run, while preserving the Büchi condition. \square

Thanks to the above lemma, it makes sense to talk about the ω -type of a word $w \in S^\omega$ built out of types. In particular, it makes sense to say whether or not a word $w \in S^\omega$ is accepted by \mathcal{A} , since this information is stored in the type. A special case of this notation is ae^ω , where $a, e \in S$, which is the ω -type of the ω -word that begins with letter a and has all other letters equal to e . The importance of this special case is explained by the following lemma about factorisations of ω -words³

Lemma 3.4. *For every $w \in S^\omega$ there exist $a, e \in S$, such that e is an idempotent, $ae = e$, and there is a factorisation*

$$w = \underbrace{\quad}_{w_0}^{\text{type } a} \underbrace{\quad}_{w_1}^{\text{type } e} \underbrace{\quad}_{w_2}^{\text{type } e} \underbrace{\quad}_{w_3}^{\text{type } e} \dots$$

Proof Define a *cut* in w to be the space between two positions. Consider an undirected edge-labelled graph, defined as follows. Vertices are cuts. For every two distinct cuts, there is an undirected edge, labelled by the type of the finite word that connects the two cuts. By Ramsey's Theorem A, see Exercise 50, there exists a type $a \in S$ and an infinite set X of vertices, such every two distinct vertices from X are connected by an edge with label e . Define the decomposition from the lemma to be the result of cutting w along all cuts from X . By assumption on X , every word w_i with $i > 0$ has type e . Idempotence of e follows from

$$\underbrace{\underbrace{w_i}_{e} \underbrace{w_{i+1}}_{e}}_e.$$

Finally, we can assure that $ae = a$ by joining the first two groups. \square

A corollary of Lemmas 3.3 and 3.4 is that $w \in L$ if and only if

(*) there is a factorisation as in Lemma 3.4 such that $ae^\omega \in L$.

³ This lemma was first observed by Büchi in [6, Lemma 1] where it was used to prove that nondeterministic Büchi automata are closed under complementation, without passing through a deterministic model.

So far, we are doing the same argument as in Büchi's original complementation proof from [6]. In his proof, Büchi observed that variant of (*) with $ae^\omega \notin L$, which characterises the complement of L , can be expressed by a nondeterministic Büchi automaton, and therefore nondeterministic Büchi automata are closed under complementation.

Now, we diverge from Büchi's proof, since we are interested in determinisation and not complementation. Here, semigroups will be helpful. Since it is not clear how to express condition (*) using a deterministic Büchi automaton, we will reformulate it. This is done in the following lemma. In the lemma, we say that type $a \in S$ *appears infinitely often* in an ω -word $w \in \Sigma^\omega$ if every suffix (necessarily infinite) of w has finite infix of type a .

Lemma 3.5. *An ω -word $w \in \Sigma^\omega$ is accepted by \mathcal{A} if and only if*

(**) *there exist $a, e \in S$, with e idempotent, $ae = e$, and $ae^\omega \in L$, such that all of the following conditions are satisfied:*

- (1) *infinitely many prefixes of w have type a ; and*
- (2) *type e appears infinitely often in w ; and*
- (3) *if type b appears infinitely often in w , then b is an infix of e (in the infix ordering of the finite semigroup S).*

Proof The top-down implication, which says that every word accepted by \mathcal{A} must satisfy (**), is an immediate consequence of Lemma 3.4. We are left with the bottom-up implication. Suppose that w satisfies (**), as witnessed by $a, e \in S$. Cut the word along the infinitely many cuts which induce a prefix of type a , which exist by condition (1), and group the pieces using Lemma 3.4, yielding some idempotent f such that $af = f$ and a decomposition

$$w = \underbrace{\text{type } a}_{w_0} \underbrace{\text{type } f}_{w_1} \underbrace{\text{type } f}_{w_2} \underbrace{\text{type } f}_{w_3} \dots$$

Although we will not prove $e = f$, we will prove that ae^ω and af^ω have the same ω -type, which will be sufficient to prove that w is accepted. By condition (2) we know that e is an infix of f , and therefore f can be obtained as a product $f = xey$ for some $x, y \in S$. If \sim denotes having the same ω -type, then

$$af^\omega \sim \underbrace{ae}_a \underbrace{(xey)}_f^\omega \sim a \underbrace{exe}_{\text{call this } g} \underbrace{(eyxe)}_{\text{call this } h}^\omega.$$

By condition (3), both g and h are infixes of e , and therefore they are in the

same \mathcal{H} -class as e . This \mathcal{H} -class is a group because it contains the idempotent e , by the \mathcal{H} -class Lemma. For every a, b in this group we have

$$a^\omega \sim (a^1)^\omega = e^\omega = (b^1)^\omega \sim b^\omega.$$

Therefore:

$$agh^\omega \sim agg^\omega \sim ae^\omega.$$

Since the last word has a type that is accepted by \mathcal{A} , we conclude that the word w is accepted. \square

To finish the determinisation construction in Theorem 3.2, it remains to show that condition (***) from the above lemma is a finite Boolean combination of languages recognised by deterministic Büchi automata. Condition (***) is a finite Boolean combination of statements of the form “infinitely many prefixes have type a ” or “type a appears infinitely often in w ”. Therefore, it suffices that statements of this are recognised by deterministic Büchi automata.

We begin with “infinitely many prefixes have type a .” Let $L_a \subseteq \Sigma^+$ be the words of type a . This is a regular language, because it is recognised by the finite semigroup of types. Consider a deterministic automaton which recognises this language (as a language of finite words), and view the automaton as a deterministic Büchi automaton. The Büchi condition is satisfied by exactly by those ω -words which have infinitely many prefixes of type a .

Consider now “type a appears infinitely often”. Let $L_a \subseteq \Sigma^+$ be the regular language of words of type a . Consider a deterministic automaton on finite words which recognises the language $\Sigma^* L_a$, i.e. words with some suffix of type a . View this automaton as a deterministic Büchi automaton, but with the following modification: whenever it sees an accepting state, it returns to the initial state. The resulting deterministic Büchi automaton recognises the language $(\Sigma^* L_a)^\omega$, which is exactly the words where type a appears infinitely often.

This completes the proof of Theorem 3.2.

Semigroups for ω -words. There is an implicit algebraic structure in the proof of Theorem 3.2, which is formalised in the following definition.

Definition 3.6. An ω -semigroup consists consists of:

- two sets S_+ and S_ω , called the *finite sort* and the *ω -sort*, respectively.
- a finite product $\pi_+ : (S_+)^+ \rightarrow S_+$, associative as in the sense of semigroups;
- an ω -product $\pi_\omega : (S_+)^{\omega} \rightarrow S_\omega$, associative in the following sense:

$$\pi_\omega(w_1 w_2 \cdots) = \pi_\omega(\pi_+(w_1) \pi_+(w_2) \cdots) \quad \text{for every } w_1, w_2, \dots \in S^+.$$

An example of an ω -semigroup is the automaton types that were used in the proof of Theorem 3.2. Another example is the *free ω -semigroup over a set Σ* , where the finite sort is Σ^+ , the ω -sort is Σ^ω , and the two products are defined in the natural way. The same proof as in Theorem 3.2 shows that a language is ω -regular if and only if it is recognised by a homomorphism into an ω -semigroup which is finite (on both sorts). This is discussed in more detail in some of the exercises at the end of this section.

The associativity axiom on the ω -product can be represented using a commuting diagram, in the same spirit as for Lemma 1.4:

$$\begin{array}{ccc}
 ((S_+)^+)^{\omega} & \xrightarrow{\omega\text{-product in free } \omega\text{-semigroup over } S_+} & (S_+)^{\omega} \\
 (\pi_+)^{\omega} \downarrow & & \downarrow \pi_{\omega} \\
 (S_+)^{\omega} & \xrightarrow{\pi_{\omega}} & S_{\omega}
 \end{array}$$

In the above diagram, $(\pi_+)^{\omega}$ denotes the coordinate-wise lifting of π_+ to ω -words of finite words.

Exercises

Exercise 50. (2) Prove the following result, called *Ramsey's Theorem A*⁴. Consider an infinite undirected graph, where every two distinct vertices are connected by an edge that is labelled by one of finitely many colours. Then the graph contains an infinite monochromatic clique, which means that there exists a colour e and an infinite set X of vertices, such that every two distinct vertices from X are connected by an edge with colour e .

Exercise 51. (1) Prove the converse implication in Theorem 3.2.

Exercise 52. (1) We say that an ω -word is *ultimately periodic* if it has the form wu^ω , for some finite words $w, u \in \Sigma^\omega$. Show that every nonempty ω -regular language contains an ultimately periodic ω -word.

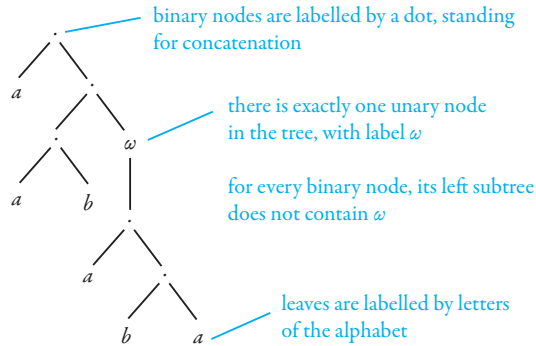
Exercise 53. (1) Show that two ω -regular languages are equal if and only if they contain the same ultimately periodic ω -words.

⁴ [24] Ramsey, "On a problem of formal logic", 1929, Theorem A

Exercise 54. (1) Show that an ω -word w is ultimately periodic if and only if $\{w\}$ is an ω -regular language.

Exercise 55. (2) To an ω -word we associate an ordered model, in the same way as for finite words. Show that a language is mso definable (using the ordered model) if and only if it is ω -regular.

Exercise 56. (2) Define an ω -term to be any tree as in the following picture:



Every ω -term represents some ultimately periodic ω -word, but several ω -terms might represent the same ultimately periodic ω -word. Show that two ω -terms represent the same ultimately periodic ω -word if and only if one can be transformed into the other using the equations:

$$(xy)z = x(yz) \quad (xy)^\omega = x(yx)^\omega \quad \underbrace{(x^n)^\omega = x^\omega}_{\text{for every } n \in \{1, 2, \dots\}}$$

where x, y, z stand for ω -terms.

Exercise 57. (1) Let $L \subseteq \Sigma^\omega$. Consider the following equivalence relations on Σ^+ .

- Right equivalence is defined by

$$w \sim w' \stackrel{\text{def}}{=} wv \in L \Leftrightarrow w'v \in L \text{ for every } v \in \Sigma^\omega.$$

- Two-sided congruence is defined by

$$w \sim w' \stackrel{\text{def}}{=} uwv \in L \Leftrightarrow uw'v \in L \text{ for every } u \in \Sigma^*, v \in \Sigma^\omega.$$

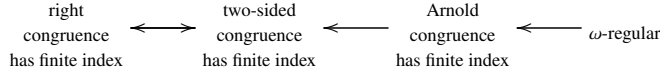
- Arnold congruence is defined by

$$w \sim w' \stackrel{\text{def}}{=} \bigwedge \begin{cases} u(wv)^\omega \in L \Leftrightarrow u(w'v)^\omega \in L & \text{for every } u, v \in \Sigma^*. \\ uwv \in L \Leftrightarrow uw'v \in L & \text{for every } u \in \Sigma^*, v \in \Sigma^\omega. \end{cases}$$

Show that the latter two, but not necessarily the first one, are semigroup congruences, i.e. they satisfy

$$\bigwedge_{i \in \{1,2\}} w_i \sim w'_i \quad \text{implies} \quad w_1 w_2 \sim w'_1 w'_2.$$

Exercise 58. (2) Consider the equivalence relations defined in Exercise 57. Prove that the arrows in the following diagram are true implications, and provide counter-examples the missing arrows:



Exercise 59. (2) Define the *Arnold semigroup* of a language $L \subseteq \Sigma^\omega$ to be the quotient of Σ^+ under Arnold congruence. Let $L \subseteq \Sigma^\omega$ be a ω -regular. Show that L is definable in first-order logic if and only if its Arnold semigroup is aperiodic.

Exercise 60. (2) The temporal logic $\text{LTL}[F]$ can also be used to define languages of ω -words. Let $L \subseteq \Sigma^\omega$ be a ω -regular. Show that L is definable in LTL if and only if its Arnold semigroup is suffix-trivial.

Exercise 61. (1) Show an ω -regular language where the Arnold semigroup is infix trivial, but which cannot be defined by a Boolean combination of \exists^* -sentences.

Exercise 62. (1) Define a *safety automaton* to be an automaton on ω -words with the following acceptance condition: all states in the run are accepting. Show that deterministic and nondeterministic safety automata recognise the same languages.

Exercise 63. (1) Show that an ω -regular language of ω -words is recognised by a safety automaton (deterministic or nondeterministic, does not matter by Exercise 62) if and only if

$$uw^1v \in L \Leftrightarrow u(w^1)^\omega \in L \quad \text{for every } u, w \in \Sigma^+ \text{ and } v \in \Sigma^\omega,$$

where $! \in \{1, 2, \dots\}$ is the exponent obtained from the Idempotent Power Lemma as applied to the Arnold semigroup of L .

Exercise 64. (2) For a finite alphabet Σ , we can view Σ^ω as metric space, where the distance between two different ω -words is defined to be

$$\frac{1}{2^{(\text{length of longest common prefix})}}$$

This is indeed a distance, i.e. it satisfies the triangle inequality. Let $L \subseteq \Sigma^\omega$ be ω -regular. Show that L is recognised by a safety automaton if and only if it is a closed set with respect to this distance.

Exercise 65. (2) Find a condition on the Arnold semigroup of an ω -regular language which characterises the clopen languages (i.e. languages which are both closed and open with respect to the distance from Exercise 64)

Exercise 66. (1) We use the topology from Exercise 64. Define a G_δ set to be any countable intersection of open sets. Show that every ω -regular language is a finite Boolean combination of G_δ sets.

Exercise 67. (2) Let $L \subseteq \Sigma^\omega$ be an ω -regular language, and define $!$ as in Exercise 62. Show that L is recognised by a deterministic Büchi automaton if and only if:

$$u(wv^!)^!v^\omega \in L \Rightarrow u(wv^!)^\omega \in L \quad \text{for every } u, w, v \in \Sigma^+.$$

Exercise 68. (2) Let $L \subseteq \Sigma^\omega$. Define an ω -congruence to be any equivalence relation \sim on Σ^+ which is a semigroup congruence and which satisfies

$$\bigwedge_{i \in \{1, 2, \dots\}} w_i \sim w'_i \quad \text{implies} \quad w_1 w_2 \cdots \in L \Leftrightarrow w'_1 w'_2 \cdots \in L. \quad (3.1)$$

Show that a language is ω -regular if and only if it has an ω -congruence of finite index.

Exercise 69. (2) Define *semi- ω -congruence* for a language $L \subseteq \Sigma^\omega$ to be an equivalence relation on finite words which satisfies (3.1), but which is not necessarily a semigroup congruence. Show that if there is a semi- ω -congruence of finite index, then there is an ω -congruence of finite index.

Exercise 70. (2) We say that \sim is the *syntactic ω -congruence* of $L \subseteq \Sigma^\omega$ if it is an ω -congruence, and every other ω -congruence for L refines \sim . Show that if a language is ω -regular, then it has a syntactic ω -congruence, which is equal to the Arnold congruence.

Exercise 71. (2) Show a language of ω -words which does not have a syntactic ω -congruence.

3.2 Countable words and \circ -semigroups

In this section, we move to \circ -words. These are words where the set positions for some countable linear order. The positions could be some finite linear order, as in finite words, or the natural numbers, as in ω -words, but the positions could also be dense, as in the rational numbers. One advantage of \circ -words, as compared to ω -words, is that they can be concatenated, which is useful when defining the corresponding generalisation of semigroups.

For finite words, as well as for ω -words, the approach via semigroups can be seen as an alternative to existing automata models. This is no longer the case for \circ -words. There is no known corresponding automaton model, and therefore \circ -semigroups are the only known model of recognisability.

Definition 3.7 (\circ -words). A Σ -labelled linear order consists of a set X of *positions*, equipped with a total order and a labelling of type $X \rightarrow \Sigma$. Two such objects are considered *isomorphic* if there is a bijection between their positions, which preserves the order and labelling. Define a *\circ -word over Σ* to be any isomorphism class of countable⁵ Σ -labelled linear orders. We write Σ° for the set of \circ -words⁶.

Every finite word is a \circ -word, likewise for every ω -word. Another example

⁵ The reader might wonder why we assume countability. The reason is that the decidability theory that will be described in this section breaks down for uncountable linear orders. In fact, the mso theory of the order of real numbers $(\mathbb{R}, <)$ is undecidable, as shown [30] Shelah, “The Monadic Theory of Order”, 1975, Theorem 7

The description of \circ -semigroups in this section is based on

[8] Carton, Colcombet, and Puppis, “An algebraic approach to MSO-definability on countable linear orders”, 2018

which itself is based on Shelah’s approach to countable linear orders from [30].

⁶ Formally speaking, this is not a set, because the linear orders form a class not a set. However, without loss of generality we can use some fixed countably infinite set, e.g. the natural numbers, for the positions (but the order need not be the same as in the natural numbers). Under this restriction, the labelled linear orders become a set, and no isomorphism types are lost. For this reason, we can refer to Σ° as a set.

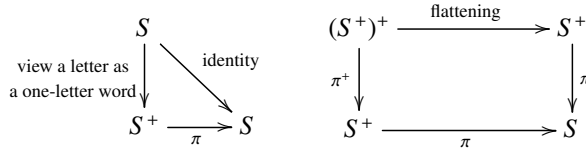
is labelled countable ordinals, e.g. any \circ -word where the positions are $\omega + \omega$. Below is a more fancy example, which uses a dense set of positions.

Example 3.8 (Shuffles). A classical exercise on linear orders is that the rational numbers are the unique – up to isomorphism – countable linear order which is dense and has no endpoints (i.e. neither a least nor greatest element). This is proved by constructing, using the back-and-forth method, an isomorphism between any two such orders. The same argument shows that for every countable Σ there is a \circ -word over Σ which has no endpoints, and which satisfies

$$\bigwedge_{a \in \Sigma} \underbrace{\forall x \forall y \exists z \quad x < z < y \wedge a(x)}_{\text{label } a \text{ is dense}}.$$

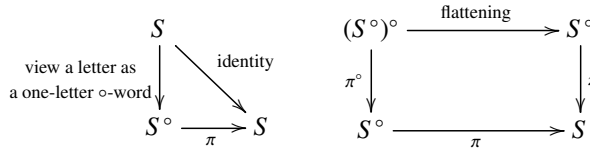
We use the name *shuffle* of Σ for the above \circ -word. Shuffles will play an important role in semigroups for \circ -words.

We now define the generalisation of semigroups for \circ -words. We use the approach to associativity via commuting diagrams that was described in Lemma 1.4. Recall from that lemma that a semigroup product on a set S could be defined as any operation π which makes the following diagram commute:



For \circ -semigroups, we take the same approach. For a set S , the flattening operation $(S^\circ)^\circ \rightarrow S^\circ$ is defined in the natural way, by replacing each position with the \circ -word that is in its label (a formal definition uses a lexicographic product of labelled linear orders).

Definition 3.9. A \circ -semigroup consists of an underlying set S equipped with a product operation $\pi : S^\circ \rightarrow S$, which is associative in the sense that the following two diagrams commute:



In the above diagram, π° denotes the coordinate-wise lifting of π to \circ -words of \circ -words.

Example 9. The *free* \circ -semigroup over alphabet Σ has Σ° as its underlying set, and its product operation is flattening. To check that this product operation is associative, one needs to prove that the following diagram commutes:

$$\begin{array}{ccc}
 ((\Sigma^\circ)^\circ)^\circ & \xrightarrow{\text{flattening for alphabet } \Sigma^\circ} & (\Sigma^\circ)^\circ \\
 \downarrow \text{(flattening for alphabet } \Sigma^\circ) & & \downarrow \text{flattening for alphabet } \Sigma \\
 (\Sigma^\circ)^\circ & \xrightarrow{\text{flattening for alphabet } \Sigma} & \Sigma^\circ
 \end{array}$$

To prove this formally, one uses the formal definition of flattening, in terms of lexicographic products of linear orders (see Example 19). This \circ -semigroup is called *free* for the usual reasons; a more formal description of these usual reasons will appear later in the book, when discussing monads. \square

Example 10. Recall the semigroups of size two that were discussed in Example 1.2:

$$\begin{array}{ccccc}
 \underbrace{(\{0, 1\}, +)} & \underbrace{(\{0, 1\}, \min)} & \underbrace{(\{0, 1\}, \pi_1)} & \underbrace{(\{0, 1\}, \pi_2)} & \underbrace{(\{0, 1\}, (a, b) \mapsto 1)} \\
 \text{addition mod 2} & & \text{product } ab \text{ is } a & \text{product } ab \text{ is } b & \text{all products are 1}
 \end{array}$$

Which ones can be extended to \circ -semigroups in at least one way?

The first example, i.e. the two-element group, cannot be extended in any way, because the product a of the ω -word 1^ω would need satisfy

$$a = \pi(1^\omega) = \pi(\pi(1)\pi(1^\omega)) = \pi(1a) = 1 + a.$$

The remaining semigroups can be extended to \circ -semigroups. As we will see in Example 11, the extensions are not necessarily unique. \square

We use \circ -semigroups to recognise languages of \circ -words. Define a *homomorphism of \circ -semigroups* to be a function h which makes the following diagram commute:

$$\begin{array}{ccc}
 S^\circ & \xrightarrow{h^\circ} & T^\circ \\
 \downarrow \text{product in } S & & \downarrow \text{product in } T \\
 S & \xrightarrow{h} & T
 \end{array}$$

Like for semigroups, homomorphisms of \circ -semigroup can be described in terms of compositional functions. Suppose that S is a \circ -semigroup and T is a set. We say that a function $h : S \rightarrow T$ is *compositional* if there exists a

function $\pi : T^\circ \rightarrow T$ which makes the following diagram commute

$$\begin{array}{ccc} S^\circ & \xrightarrow{h^\circ} & T^\circ \\ \text{product in } S \downarrow & & \downarrow \pi \\ S & \xrightarrow{h} & T \end{array}$$

Using the same proof as for Lemma 1.3, one shows that if h is a compositional and surjective, then π is necessarily associative, thus turning T into a \circ -semigroup, and furthermore h is a homomorphism.

We say that a language $L \subseteq \Sigma^\circ$ is *recognised* by a \circ -semigroup S if there is a homomorphism $h : \Sigma^\circ \rightarrow S$ which recognises it, i.e.

$$h(w) = h(w') \quad \text{implies} \quad w \in L \Leftrightarrow w' \in L \quad \text{for every } w, w' \in L.$$

We are mainly interested in languages recognised by finite \circ -semigroups.

Example 11. Consider un-labelled countable linear orders, which can be viewed as \circ -words over a one letter alphabet $\{a\}$. Consider the function

$$h : \{a\}^\circ \rightarrow \{0, 1\}$$

which sends well-founded \circ -words to 1, and the remaining \circ -words to 0. We claim that h compositional (and therefore the language of well-founded \circ -words is recognised by a finite \circ -semigroup). Indeed, take some $v \in (\{a\}^\circ)^\circ$ which flattens to $w \in \{a\}^\circ$. To prove compositionality, need to show that $h^\circ(v)$ uniquely determines $h(w)$. This is because $h(w) = 1$ if and only if the positions of v are well-founded, and every such a position is labelled by a well-founded order. All of this information can be recovered from $h^\circ(v)$. The compositional function h induces an underlying structure of a \circ -semigroup on $\{0, 1\}$. When restricted to finite products, this \circ -semigroup is the same as $(\{0, 1\}, \min)$. Note that a symmetric \circ -semigroup can be constructed, for orders which are well-founded after reversing. The symmetric \circ -semigroup also coincides with $(\{0, 1\}, \min)$ on finite words. \square

Example 12. Consider the language $L \subseteq \{a, b, 1\}^\circ$, which contains \circ -words where some position with label a is to the left of some position with label b . Consider the following function

$$w \in \{a, b, 1\}^\circ \quad \mapsto \quad \begin{cases} 0 & \text{if } w \in L \\ 1 & \text{if all letters are 1} \\ b & \text{if all letters are } b \text{ or } c, \text{ and there is some } b \\ a & \text{otherwise} \end{cases}$$

This function is easily seen to be compositional, and therefore its image is a \circ -semigroup. The element 0 is absorbing, and 1 is a monoid identity. The language L is therefore recognised by the corresponding \circ -semigroup. \square

Monadic second-order logic on \circ -words. The *ordered model* of a \circ -word is defined in the same way as for finite words: the universe is the positions, and the relations and their meaning are the same as for finite words. We say that a language $L \subseteq \Sigma^\circ$ is definable in mso if there is an mso sentence φ , using the vocabulary of the ordered model, such that

$$w \in L \iff \text{the ordered model of } w \text{ satisfies } \varphi \quad \text{for every } w \in \Sigma^\circ.$$

Example 13. Consider the language of well-founded \circ -words that was discussed in Example 11. This language is definable in mso, by simply writing in mso the definition of well-foundedness:

$$\underbrace{\forall X}_{\substack{\text{for every} \\ \text{set of} \\ \text{positions}}} \quad \underbrace{(\exists x \in X)}_{\text{which is nonempty}} \Rightarrow \underbrace{(\exists x \in X \forall y \in X x \leq y))}_{\text{there is a least position}}).$$

Another example is the \circ -words which contain a sub-order that is dense:

$$\underbrace{\exists X}_{\substack{\text{exists a} \\ \text{set of} \\ \text{positions}}} \quad \underbrace{(\exists x \in X)}_{\text{which is nonempty}} \wedge \underbrace{(\forall x \in X \forall y \in Y x < y \Rightarrow \forall z \in X x < z < y))}_{\text{and dense in itself}}).$$

An \circ -word which violates the second property, i.e. it does not have any dense sub-order, is called *scattered*. \square

Once we have built up all the necessary ideas in the Trakhtenbrot-Büchi-Elgot Theorem for finite words, it is very easy to get the extension for \circ -words.

Lemma 3.10. *If a language $L \subseteq \Sigma^\circ$ is definable in mso, then it is recognised by a finite \circ -semigroup.*

Proof We only give a brief sketch using Ehrenfeucht-Fraïssé games – a more detailed proof using a powerset construction on \circ -semigroups will be given later in Lemma 3.21s. We use the same argument as in Section 2.1. The set model is defined the same way as for finite words – its universe is the subsets of the positions. For $k \in \{0, 1, \dots\}$, define \equiv_k to be the equivalence relation on Σ° which identifies \circ -words if their set models satisfy the same first-order sentences of quantifier rank at most k . Using the same proof as for Lemma 2.5,

except that now a \circ -word can be divided into more than two parts, one shows that

$$w \in \Sigma^\circ \quad \mapsto \quad \underbrace{\text{equivalence class of } w \text{ under } \equiv_k}_{\text{we call this the rank } k \text{ mso type of } w}$$

is a compositional function into a finite set. Therefore, the function is a homomorphism of \circ -semigroups. Every language definable in mso is recognised by such a homomorphism, for suitably chosen k . \square

The above lemma seems too easy. Is there a catch? Yes: the lemma does not give any algorithm for deciding if an mso definable language is empty, or any answering any other computational problems. In fact, the lemma would remain true for uncountable words, and satisfiability of mso sentences for such words is undecidable. The issue of finite representation for \circ -semigroups will be addressed in the following section; and countability will play a crucial role.

Another interesting question is about the converse of the lemma: can one define in mso every language that is recognised by a finite \circ -semigroup? For finite words and ω -words, the answer was “obviously yes”, because one can use mso to formalise the acceptance by an automaton. Since we have no automata for \circ -words, the question is harder. However, the answer is still “yes”, and it will be given in Section 3.4.

Exercises

Exercise 72. (1) Give a formula of mso which is true in some uncountable well-founded linear order, but is false in all countable well-founded linear orders.

Exercise 73. (1) Find two countable ordinals (viewed as \circ -words over a one letter alphabet), which have the same mso theory.

Exercise 74. (1) We write ω^* for the reverse of ω . An $(\omega^* + \omega)$ -word is a \circ -word where the underlying order is the same as for the integers. Show that the following problem is decidable: given an mso sentence, decide if it is true in some bi-infinite word.

Exercise 75. (2) We say that a $(\omega^* + \omega)$ -word v is *recurrent* if every finite word $w \in \Sigma^+$ appears as an infix in every prefix of v and in every suffix of v . Show that all recurrent $(\omega^* + \omega)$ -words have the same mso theory.

Exercise 76. (2) Let Σ be an alphabet, and let $x \notin \Sigma$ be a fresh letter. For $w \in \Sigma^\circ$ and $u \in (\Sigma \cup \{x\})^\circ$, define $u[x := w] \in \Sigma^\circ$ to be the result of substituting each occurrence of variable x in u by the argument w . For a language $L \subseteq \Sigma^\circ$, define *contextual equivalence* to be the equivalence relation on Σ° defined by

$$w \sim w' \quad \text{iff} \quad u[x := w] \in L \Leftrightarrow u[x := w'] \in L \text{ for every } u \in (\Sigma \cup \{x\})^\circ.$$

Show that \sim is a \circ -congruence (which means that the function that maps w to its equivalence class is compositional) for every language recognised by some finite \circ -semigroup.

Exercise 77. (1) Give an example of a language $L \subseteq \Sigma^\circ$ where contextual equivalence is not a \circ -congruence.

Exercise 78. (1) Show that every language recognised by a finite \circ -semigroup has syntactic \circ -semigroup, but there are some languages (not recognised by finite \circ -semigroups), which do not have a syntactic \circ -semigroup.

Exercise 79. (1) Consider a binary tree (every node has either zero or two children, and we distinguish left and right children), where leaves are labelled by an alphabet Σ . The tree might have infinite branches. Define the *yield* of such a tree to be the \circ -word where the positions are leaves of the tree, the labels are inherited from the tree, and the ordering on leaves is lexicographic (for every node, its left subtree is before its right subtree). Show that every \circ -word can be obtained as the yield of some tree.

Exercise 80. (1) Show that the following problems are equi-decidable:

- given an mso sentence, decide if it is true in some \circ -word $w \in \Sigma^\circ$
- given an mso sentence, decide if its true in $(\mathbb{Q}, <)$.

Exercise 81. (1) Assume Rabin's Theorem, which says that the mso theory of the complete binary tree

$$(\{0, 1\}^*, \underbrace{x = y0}_{\substack{\text{left} \\ \text{child}}}, \underbrace{x = y1}_{\substack{\text{right} \\ \text{child}}})$$

is decidable. Show that the problems from Exercise 80 are decidable. (We will also prove this in the next section, without assuming Rabin's theorem.)

3.3 Finite representation of \circ -semigroups

The product operation in a finite semigroup can be seen as an operation of type $S^+ \rightarrow S$, or as a binary operation of type $S^2 \rightarrow S$. The binary operation has the advantage that a finite semigroup can be represented in a finite way, by giving a multiplication table. In this section, we show that a similar finite representation is also possible for \circ -semigroups. Apart from binary product, we will use two types of ω -iteration – one forward and one backward – and a shuffle operation (which inputs a set of elements, and not a tuple of fixed length).

Definition 3.11 (Läuchli-Leonard operations). For a \circ -semigroup, define its *Läuchli-Leonard operations*⁷ to be the following four operations (with their types written in red).

$$\begin{array}{cccc}
 \underbrace{ab} & \underbrace{a^\omega} & \underbrace{a^{\omega*}} & \underbrace{\{a_1, \dots, a_n\}^\eta} \\
 \text{binary} & \text{product of} & \text{product of} & \text{product of the} \\
 \text{product} & aaa \dots & \dots aaa & \text{shuffle of } a_1, \dots, a_n \\
 S^2 \rightarrow S & S \rightarrow S & S \rightarrow S & PS \rightarrow S
 \end{array}$$

Theorem 3.12. *The product operation in a finite \circ -semigroup is uniquely determined by its Läuchli-Leonard operations.*

Another way of stating the theorem is that if S is a finite set S equipped with Läuchli-Leonard operations, then there is at most one way of extending these operations to an associative product $S^\circ \rightarrow S$. We say at most one instead of exactly one, because the Läuchli-Leonard operations need to satisfy certain associativity laws, such as:

$$aa^\omega = a^\omega \quad (ab)^\omega = a(ba)^\omega \quad \{a_1, \dots, a_n\}^\eta = \{\{a_1, \dots, a_n\}^\eta\}^\eta$$

We do not worry too much about giving the full set of axioms⁸, because we will only consider product operations that arise from compositional functions – e.g. the product operation on mso types of given quantifier rank k – and such product operations are guaranteed to be associative.

3.3.1 Proof of Theorem 3.12

The key idea in the proof of Theorem 3.12 is that the Läuchli-Leonard operations are enough to generate all sub-algebras, as stated in the following lemma.

⁷ [19] Läuchli and Leonard, “On the elementary theory of linear order”, 1966 , p. 109

⁸ It can be found in
 [3] Bloom and Ésik, “The equational theory of regular words”, 2005 , Section 7

Lemma 3.13. *Let S be a \circ -semigroup, and let $\Sigma \subseteq S$. Then*

$$\underbrace{\{\text{product of } w : w \in \Sigma^\circ\}} \subseteq S$$

this is called the sub-algebra generated by Σ

is equal to the smallest subset of S which contains Σ and is closed under the L\"auchli-Leonard operations.

Before proving the lemma, we use it to prove Theorem 3.12.

Proof of Theorem 3.12, assuming Lemma 3.13. Suppose that S_1 and S_2 are two \circ -semigroups, which have the same underlying set, and product operations which agree on the L\"auchli-Leonard operations. We will show that the product operations are the same. Consider the product \circ -semigroup $S_1 \times S_2$, defined in the usual coordinate-wise way. Apply Lemma 3.13 to the diagonal

$$\Sigma = \{(a, a) : a \in S\} \subseteq S_1 \times S_2.$$

Since the L\"auchli-Leonard operations agree for S_1 and S_2 , it follows from the lemma that the sub-algebra generated by Σ is also on the diagonal, which shows that the product operations of S_1 and S_2 are equal. \square

The rest of Section 3.3.1 is devoted to proving Lemma 3.13. Define $L \subseteq \Sigma^\circ$ to be the \circ -words whose product can be obtained from Σ by applying the L\"auchli-Leonard operations. The lemma says that $L = \Sigma^\circ$. This follows immediately from the following lemma (which is stated slightly more generally, because it will be used again later), by taking λ to be the product operation of the \circ -semigroup.

Lemma 3.14. *Assume that $L \subseteq \Sigma^\circ$ contains Σ and is closed under binary concatenation. A sufficient condition for $L = \Sigma^\circ$ is that there exists a colouring $\lambda : \Sigma^\circ \rightarrow C$, with C a finite set of colours, such that:*

(*) *Let $w \in (\Sigma^\circ)^\circ$ be such that every position has label in L , and*

$$\underbrace{\lambda^\circ(w) = c^\omega}_{\text{for some } c \in C} \quad \text{or} \quad \underbrace{\lambda^\circ(w) = c^{\omega*}}_{\text{for some } c \in C} \quad \text{or} \quad \underbrace{\lambda^\circ(w) = \text{shuffle of } D.}_{\text{for some } D \subseteq C}$$

Then the flattening of w belongs to L .

Before proving the lemma, we fix some notation for \circ -words. Define an *interval* in a \circ -word to be any set of positions X such that is connected in the following sense:

$$\forall x \in X \forall y \in Y \forall z \quad x < z < y \Rightarrow z \in X.$$

An infix of a \circ -word is any \circ -word obtained by restricting the positions to

some interval. For example, the rationals – viewed as a \circ -word over a one letter alphabet – have uncountably many intervals, but five possible infixes. If $x < y$ are positions in a \circ -word w , then we write $w(x..y)$ for the infix corresponding to the interval $\{z : x < z < y\}$.

Proof Suppose then that L satisfies (*), as witnessed by a colouring λ . We say that $w \in \Sigma^\circ$ is *simple* if all of its infixes are in L . We will show that all of Σ° is simple, thus proving $L = \Sigma^\circ$. For the sake of contradiction, suppose that $w \in \Sigma^\circ$ is not simple. Define \sim to be the binary relation on positions in w , which identifies positions if they are equal, or $w(x..y)$ is simple (where x is the smaller position and y is the bigger position).

Claim 3.15. *The relation \sim is an equivalence relation, every equivalence class is an interval, and this interval induces a simple \circ -word.*

Proof The relation \sim is symmetric and reflexive by definition. Transitivity is because simple words are closed under binary concatenation (which itself easily follows from the fact that L contains all letters and is closed under binary concatenation). This establishes that \sim is an equivalence relation. Because simple \circ -words are closed under infixes by definition, every equivalence class of \sim is an interval.

It remains to show that every (infix induced by an) equivalence class is simple. Here we use countability and the assumption (*). Consider an equivalence class X . Choose some $x \in X$. We will show that the suffix of X that begins in x is simple. A symmetric argument will establish that the prefix leading up to x is simple, and thus X itself is simple by binary concatenation.

If X has a last position, then the suffix that begins in x is simple by definition of \sim . Suppose then that there is no last position in X . By countability, choose some sequence of positions

$$x = x_0 < x_1 < x_2 < \dots \in X$$

which is co-final, i.e. every position in X is smaller than some x_i . By the Ramsey Theorem, we can assume without loss of generality that there is some $c \in C$ such that for every $i \in \{1, 2, \dots\}$, the infix obtained from $w(x_i..x_{i+1})$ by appending position x_{i+1} has colour c under λ . Furthermore, this infix is simple by definition of \sim . Therefore, thanks to assumption (*), we know that the concatenation of all of these infixes is simple. \square

Since the equivalence classes of \sim are intervals, they can be viewed as an ordered set, with the order inherited from the original order on positions in w . Because simple words are closed under binary concatenation, the order on equivalence classes is dense, since otherwise two consecutive equivalence classes

would need to be merged into a single one. Define $w_{\sim} \in C^{\circ}$ to be the result of replacing every equivalence class of \sim by its colour under λ . By assumption that w is not simple, \sim has more than one equivalence class, and therefore the positions of w_{\sim} are an infinite dense linear order.

Claim 3.16. *Some infix of w_{\sim} is a shuffle.*

Proof Take some colour $c \in C$. If there is some infinite infix of w_{\sim} where c does not appear at all, then we can continue working in that infix (its positions are still an infinite dense linear order). Otherwise, positions with colour c are dense. By iterating this argument for all finitely many colours in C , we find an infinite infix where every colour either does not appear at all, or is dense. This infix is a shuffle. \square

By (*), the flattening of the infix from the above claim is simple. It follows that the corresponding interval should have been a single equivalence class of \sim , contradicting the assumption. \square

3.3.2 Decidability of MSO

By Theorem 3.12, a finite \circ -semigroup can be represented in a finite way, by giving its underlying set and the multiplication tables for its Lauchli-Leonard operations. We will use this representation to give decision procedure for mso on \circ -words.

A powerset construction. To decide mso, we will use a powerset construction for finite \circ -semigroups, which will correspond to set quantification in mso. We begin by describing this construction.

Definition 3.17. For a \circ -semigroup S , define the *powerset \circ -semigroup* PS as follows. The underlying set of PS is the powerset of the underlying set of S , including the empty set⁹. The product operation is defined by

$$w \in (PS)^{\circ} \mapsto \underbrace{\{\text{product of } v : v \in^{\circ} w\}}_{\text{in } S},$$

where $v \in^{\circ} w$ means that $v \in S^{\circ}$ can be obtained from $w \in (PS)^{\circ}$ by choosing

⁹ Whether or not we allow the empty set is not important for the construction.

for each position an element of its label¹⁰. We leave it as an exercise to check that the product operation defined this way is associative¹¹.

The following lemma shows that the powerset construction is computable. What is not obvious is finding the multiplication tables for the L\"auchli-Leonard operations.

Lemma 3.18. *Given the multiplication tables for the L\"auchli-Leonard operations in a finite \circ -semigroup S , one can compute the multiplication tables for the L\"auchli-Leonard operations in the powerset \circ -semigroup \mathbf{PS} .*

Proof In the proof, we adopt the convention that elements of S are denoted by lower-case letters a, b, c , while elements of \mathbf{PS} are denoted by upper-case letters A, B, C . The multiplication table for binary product of \mathbf{PS} , namely

$$A, B \in \mathbf{PS} \quad \mapsto \quad \underbrace{\{ab\}}_{\text{product in } S} : a \in A, b \in B,$$

is easily computable using the binary product in S . The hard part is the multiplication tables for the infinitary operations, i.e. ω -power, ω^* -power and shuffles.

ω -power. We begin by clarifying some notation. For a $A \in \mathbf{PS}$, the expression A^ω can be understood in three different ways:

- (1) $A^\omega \subseteq S^\circ$ is the set of ω -words where all letters are from A ;
- (2) $A^\omega \in (\mathbf{PS})^\circ$ is the ω -word where all letters have label equal to A ;
- (3) $A^\omega \in \mathbf{PS}$ is the product of the word from item (2) in the \circ -semigroup \mathbf{PS} .

To avoid confusion, we use the red type annotation below. The L\"auchli-Leonard operation in the powerset \circ -semigroup \mathbf{PS} that we are discussing in this lemma uses the third meaning of A^ω :

$$A \in \mathbf{PT} \quad \mapsto \quad A^\omega \in \mathbf{PS}.$$

To compute this operation, we will use, apart from S , one other \circ -semigroup. This other \circ -semigroup, call it T , is used to recognise the singleton language

$$\{A^\omega \in (\mathbf{PS})^\circ\} \subseteq (\mathbf{PS})^\circ.$$

¹⁰ The relation $v \in^\circ w$ can be formalised by saying that there exists a \circ -word u over alphabet

$$\{(a, A) : a \in A \subseteq S\}$$

such that v is the projection of u to the first coordinate, and w is the projection of u to the second coordinate.

¹¹ One has to a bit careful. For example, there is no such thing as a powerset group.

The elements of T are $\{\omega, +, 0\}$ and the product operation is the unique product which makes the following function h into a homomorphism:

$$w \in (\text{PS})^\circ \mapsto \begin{cases} \omega & \text{if } w = A^\omega \in (\text{PS})^\circ \\ + & \text{if } w \text{ is a finite word and all letters have label } A \\ 0 & \text{otherwise} \end{cases}$$

For the Lauchli-Leonard operations in T , all outputs are 0 with the following exceptions:

$$++ = + \quad +\omega = \omega \quad +^\omega = \omega.$$

Claim 3.19. *Let $a \in S$. Then $a \in A^\omega \in \text{PS}$ if and only if (a, ω) belongs to the sub-algebra of the \circ -semigroup $S \times T$ that is generated by $\{(b, +) : b \in A\}$.*

Proof By unravelling the definitions, (a, ω) belongs to the sub-algebra from the claim if and only if it is the product of some \circ -word u where every letter is of the form $(b, +)$ for some $b \in A$. Since the product of u has ω on the second coordinate, then u must be an ω -word. Therefore, if we project u onto the first coordinate, we get a word in $A^\omega \subseteq S^\circ$ whose product is a , thus proving the right-to-left implication. The left-to-right implication reverses this reasoning. \square

By the above claim, in order to compute $A^\omega \in \text{PS}$, it is enough to compute the sub-algebra from the claim. Thanks to Lemma 3.13, this sub-algebra is the closure of $\{(b, +) : b \in A\}$ under the Lauchli-Leonard operations of $S \times T$, which are simply the coordinate-wise liftings of the Lauchli-Leonard operations in S and T . Therefore, $A^\omega \in \text{PS}$ can be computed.

Shuffle power. The argument for ω^* -power is symmetric to the one above, so we are left with the shuffle power in the \circ -semigroup PS :

$$\{A_1, \dots, A_n\} \mapsto \{A_1, \dots, A_n\}^\sharp$$

We use a similar argument as for the ω -power, except that we need to define a \circ -semigroup which describes the singleton language $\{w\}$ where

$$w \stackrel{\text{def}}{=} \underbrace{\text{shuffle of } \{A_1, \dots, A_n\}}_{\text{a } \circ\text{-word over alphabet PS}}.$$

Such a \circ -semigroup T and a recognising homomorphism

$$h : (\text{PS})^\circ \rightarrow T$$

are left as an exercise for the reader (see Exercise 82). By definition of the powerset \circ -semigroup,

$$\{A_1, \dots, A_n\}^\eta = \underbrace{\{\text{product of } v : v \in^\circ w\}}_{\text{in } S}.$$

Let $a \in S$. The same proof as for Claim 3.19 shows that

$$a \in \{A_1, \dots, A_n\}^\eta$$

holds if and only if $(a, h(w))$ belongs to the sub-algebra of $S \times T$ that is generated by elements of the form

$$\{(b, h(A_i)) : i \in \{1, \dots, n\}, b \in A_i\}.$$

This sub-algebra can be computed, as in the case of ω -power. □

Decidability of MSO. Using the powerset construction on \circ -semigroups, we can now prove decidability of MSO over \circ -words.

Theorem 3.20. *The following problem is decidable:*

Input. An MSO sentence φ , which defines a language $L \subseteq \Sigma^\circ$.

Question. Is the language L nonempty?

The rest of Section 3.3.2 is devoted to proving the above theorem. As in Section 2.1, instead of using MSO in the ordered model, it will be easier to use first-order logic over (the extension for \circ -words of) the set model, see Definition 2.3. By induction on formula size, for every first-order formula over the set model we will construct a homomorphism into a finite \circ -semigroup that recognises the language of the formula. The finite \circ -semigroup will be represented, according to Theorem 3.12, by giving the underlying set and the multiplication tables for its L\"auchli-Leonard operations.

For the induction, we need to deal with formulas with free variables. Consider a first-order formula

$$\varphi(\underbrace{x_1, \dots, x_n}_{\substack{\text{the free variables range} \\ \text{over sets of positions}}})$$

over the vocabulary of the set model (over some fixed input alphabet Σ .) We define the *language of φ* to be the set of \circ -words over alphabet $\Sigma \times \{0, 1\}^n$, such that φ is true in the projection to the Σ coordinate, assuming that variable x_i is set to the set of positions that have 1 on the i -th coordinate from $\{0, 1\}$.

Lemma 3.21. *Let Σ be an input alphabet. Given a formula $\varphi(x_1, \dots, x_n)$ of first-order logic over the vocabulary of the set model, we can compute a finite \circ -semigroup S , a (not necessarily surjective) homomorphism*

$$h : (\Sigma \times \{0, 1\}^n)^\circ \rightarrow S,$$

and an accepting set $F \subseteq S$ such that the language of φ is exactly $h^{-1}(F)$. The \circ -semigroup is represented by multiplication tables for its Lauchli-Leonard operations, and the homomorphism is represented by its restriction to one-letter words.

Once we have proved the lemma, Theorem 3.20 follows immediately. We simply need to check if the image of h contains some accepting element. There is a slightly subtle point: since h is not necessarily surjective, the accepting set could be nonempty but disjoint with the image of h . Therefore, we need to compute the image of h , which is done by closing the images of the one-letter words under the Lauchli-Leonard operations.

It remains to prove the lemma.

Proof of Lemma 3.21 Induction on the size of $\varphi(x_1, \dots, x_n)$.

Atomic formulas. The atomic formulas are $x \subseteq y$, $x < y$, $x = \emptyset$ and $x \subseteq a$.

With the exception of $x < y$, all of the atomic formulas are of the form “the label of every position has some property”. Therefore, for every atomic formula except for $x < y$, the corresponding language is recognised by a homomorphism into the semigroup $\{0, 1\}$ with product defined by

$$w \in \{0, 1\}^\circ \mapsto \begin{cases} 0 & \text{if some position has label 0} \\ 1 & \text{if all positions have label 1.} \end{cases}$$

For $x < y$, the appropriate \circ -semigroup is the one from Example 12.

Boolean combinations. For negation $\neg\varphi$, we use the same homomorphism as for φ , and we complement the accepting set. For conjunction $\varphi_1 \wedge \varphi_2$ and disjunction $\varphi_1 \vee \varphi_2$, we use the product $S_1 \times S_2$ of the inductively obtained \circ -semigroups, with a naturally defined homomorphism¹². Since $S_1 \times S_2$ is defined coordinate-wise, the multiplication tables for its Lauchli-Leonard operations can be computed using those from S_1 and S_2 .

Quantification. We are left with quantification. Consider an existentially quantified formula (for universal quantification, the reasoning is the same):

$$\exists x_{n+1} \varphi(x_1, \dots, x_{n+1}).$$

¹² The homomorphism needs to account for the possibility that φ_1 and φ_2 use different subsets of the free variables in $\varphi_1 \wedge \varphi_2$.

Apply the induction assumption, yielding a homomorphism

$$h : (\Sigma \times \{0, 1\}^{n+1})^\circ \rightarrow S$$

which recognises the language of φ . Consider the function

$$H : (\Sigma \times \{0, 1\}^n)^\circ \rightarrow \underbrace{PS}_{\text{powerset } \circ\text{-semigroup}},$$

such that $H(w)$ is the set of all values $h(v) \in S$, where v ranges over \circ -words over alphabet $\Sigma \times \{0, 1\}^{n+1}$ such that w can be obtained from v by erasing the last bit from each letter. It is not hard to see that H is a homomorphism. The homomorphism H recognises the language of the existentially quantified formula; the accepting set consists of those subsets of S which intersect the accepting set for φ . The multiplication tables for the L\"auchli-Leonard operations in PS can be computed thanks to Lemma 3.18.

□

Exercises

Exercise 82. (1) Let Σ be a finite alphabet, and let w be the shuffle of all letters in Σ . Show a finite \circ -semigroup which recognises the singleton language $\{w\}$.

Exercise 83. (2) A \circ -word w is called *regular* if the singleton language $\{w\}$ is recognised by a finite \circ -semigroup. Show that w is regular if and only if it can be constructed from the letters by using the L\"auchli-Leonard operations.

Exercise 84. (1) Show that every nonempty mso definable language $L \subseteq \Sigma^\circ$ contains some regular \circ -word.

Exercise 85. (1) Show that if w is a regular \circ -word, then $\{w\}$ is mso definable (without invoking Theorem 3.22).

Exercise 86. (2) Show that for every finite alphabet Σ there exists a \circ -word $w \in \Sigma^\circ$ such that

$$h(wvw) = h(w) \quad \text{for every } \underbrace{h : \Sigma^\circ \rightarrow S}_{\substack{\text{homomorphism into} \\ \text{a finite } \circ\text{-semigroup}}} \text{ and } v \in \Sigma^\circ.$$

Exercise 87. (2) For a countable linear order X , let $\{a, b\}^X \subseteq \{a, b\}^\circ$ be the set of \circ -words with positions X . We can equip this set with a probabilistic measure, where for each position $x \in X$, the label is selected independently, with a and b both having probability half. We say that X has a zero-one law if for every mso definable language L , the probability of $\varphi \cap \{a, b\}^X$ is either zero or one. For which of the following $X = \mathbb{N}, \mathbb{Z}, \mathbb{Q}$ is there a zero-one law?

Exercise 88. (2) A countable linear order can be viewed as a \circ -word over a one-letter alphabet. Among these, we can distinguish the countable linear orders that are regular, i.e. generated by the L\"auchli-Leonard operations, see Exercise 83. Give an algorithm, which inputs a countable linear order that is regular in the above sense, and decides if it has a zero-one law (in the sense of Exercise 87).

Exercise 89. (2) Show that every mso definable language of \circ -words belongs to the least class of languages which:

- contains the following two languages over alphabet $\{a, b, c\}$:

$$\underbrace{\exists x a(x)}_{\text{some } a} \quad \underbrace{\exists x \exists y a(x) \wedge b(y) \wedge x < y}_{\text{a before b}}$$

- is closed under Boolean combinations;
- is closed under images and inverse images of letter-to-letter homomorphisms.

Exercise 90. (2) We say that a binary tree (possibly infinite) is *regular* if it has finitely many non-isomorphic sub-trees. Show that a \circ -word is regular (in the sense of Exercise 83) if and only if it is the yield (in the sense of Exercise 79) of some regular tree.

Exercise 91. (91) Consider the embedding ordering (Higman ordering) $w \hookrightarrow v$ on \circ -words. Show that for every \circ -word w there is a regular \circ -word v such that $w \hookrightarrow v$ and $v \hookrightarrow w$. Hint: use Lemma 3.14.

Exercise 92. (1) Suppose that we are given a language $L \subseteq \Sigma^\circ$, represented by a finite \circ -semigroup S , a homomorphism $h : \Sigma^\circ \rightarrow S$, and an accepting set $F \subseteq S$. Give an algorithm which computes the syntactic \circ -semigroup (which exists by Exercise 78).

Exercise 93. (2) Let \mathcal{L} be a class of languages, such that \mathcal{L} satisfies the following conditions:

- every language in \mathcal{L} is recognised by a finite \circ -semigroup;
- \mathcal{L} is closed under Boolean combinations;
- \mathcal{L} is closed under inverse images of homomorphisms $h : \Sigma^\circ \rightarrow \Gamma^\circ$;
- Let $L \subseteq \Sigma^\circ$ be a language in \mathcal{L} . For every $w, w_1, \dots, w_n \in \Sigma^\circ$, \mathcal{L} contains the inverse image of L under the following operations:

$$v \mapsto wv \quad v \mapsto vw \quad v \mapsto v^\omega \quad v \mapsto v^{\omega*} \quad v \mapsto \text{shuffle of } \{w_1, \dots, w_n, v\}.$$

Show that if L belongs to \mathcal{L} , then the same is true for every language recognised by its syntactic \circ -semigroup.

Exercise 94. (2) Let Σ be an alphabet and let $c \notin \Sigma$ be a fresh letter. We say that $L \subseteq \Sigma^\circ$ is definable in $\text{LTL}[F]$ if there is a formula of $\text{LTL}[F]$ which defines the language cL , see Exercise 41. Give an algorithm which inputs the finite syntactic \circ -semigroup of a language $L \subseteq \Sigma^\circ$, and answers if the language is definable in $\text{LTL}[F]$. Hint: the \circ -semigroup must be suffix trivial, but this is not sufficient.

Exercise 95. (3) Give an algorithm which inputs the finite syntactic \circ -semigroup of a language $L \subseteq \Sigma^\circ$, and answers if the language is definable in two-variable first-order logic FO^2 . Hint: the \circ -semigroup must be in DA , but this is not sufficient.

Exercise 96. (2) Show that aperiodicity is not sufficient for first-order definability for \circ -words: give an example of a language $L \subseteq \Sigma^\circ$ that is recognised by a finite aperiodic \circ -semigroup, but which is not definable in first-order logic.

3.4 From \circ -semigroups to MSO

In Theorem 3.10, and again in Lemma 3.21, we have shown that if a language of \circ -words is definable in MSO , then it is recognised by a finite \circ -semigroup. We now show that the converse implication is also true.

Theorem 3.22. *If a language of \circ -words is recognised by a finite \circ -semigroup, then it is definable in MSO ¹³ which says that*

¹³ This theorem was first shown in
 [8] Carton, Colcombet, and Puppis, “An algebraic approach to MSO-definability on countable linear orders”, 2018, Theorem 5.1.
 The proof presented here is different, and it is based on the proof in
 [28] Schützenberger, “On finite monoids having only trivial subgroups”, 1965, p. 192
 which shows that every aperiodic monoid recognises a star-free language. We use the different

As mentioned before in this chapter, the theorem would be easy if there was an automaton model, which would assign states to positions, and where the acceptance condition could be formalised in mso. Unfortunately, no such automaton model is known. Therefore, we need a different proof for the theorem. The rest of Section 3.4 is devoted to such a proof.

We begin by defining regular expressions for \circ -words. For a finite family \mathcal{L} of languages of \circ -words, define the shuffle of \mathcal{L} to be the \circ -words which can be partitioned into intervals so that: (a) every interval induces a word from L for some $L \in \mathcal{L}$; (b) the order type on the intervals is that of the rational numbers; and (c) for every $L \in \mathcal{L}$, the intervals from L are dense.

Lemma 3.23. *Languages definable in mso are closed under Boolean combinations and the following kinds of concatenation:*

$$LK \quad L^+ \quad L^\omega \quad L^{\omega*} \quad \text{shuffle of } \underbrace{\mathcal{L}}_{\substack{\text{a finite family} \\ \text{of languages}}}$$

Proof For the Boolean operations, there is nothing to do, since Boolean operations are part of the logical syntax. For the concatenations, we observe that mso can quantify over factorisations, as described below.

Define a *factorisation* of a \circ -word to be a partition of its positions into intervals, which are called *blocks*. For a factorisation, define a *compatible colouring* to be any colouring of positions that uses two colours, such that all blocks are monochromatic, and if a block has a successor, then the successor has a different colour. A compatible colouring always exists (there could be uncountably many choices). A factorisation can be recovered from any compatible colouring: two positions are in the same block if and only if the interval connecting them is monochromatic. A compatible colouring can be represented using a single set – namely the positions with one of the two colours. This representation can be formalised in mso, i.e. one can write an mso formula $\varphi(x, y, X)$ which says that positions x and y are in the same block of the factorisation which arises from the compatible colouring represented by set X .

Using the above representation, we show closure of mso under the concatenations in the lemma. For LK , we simply say that there exists a factorisation with two blocks, where the first block is in L and the second block is in K . (To say that a block is in L or K , we observe that mso sentences can be relativised to a given interval.) For L^+ , we say that there exists a factorisation with finitely many blocks, where all blocks are in L . Here is how we express that there are finitely many blocks: there are first and last blocks, and there is

proof because, after suitable modifications, it allows us to characterise star-free languages of \circ -words, see Exercise 102.

no proper subset of positions that contains the first block and is closed under adding successor blocks. For L^ω , we do the same, except that there is no last block. For $L^{\omega*}$, we use a symmetric approach. For the shuffle, we say that the blocks are dense and there is no first and last block. \square

In the proof of Theorem 3.22, we will only use the closure properties of mso from the above lemma. In particular, it will follow that every language recognised by a finite \circ -semigroup can be defined by a regular expression which uses single letters and the closure operations from the lemma (which include intersection and complementation).

To prove Theorem 3.22, we will show that the product operation of every finite \circ -semigroup can be defined in mso, in the following sense. Let S be a finite \circ -semigroup. We will show that for every $a \in S$, the language

$$L_a = \{w \in S^\circ : w \text{ has product } a\}$$

is mso definable. This will immediately imply that every language recognised by a homomorphism into S is mso definable, thus proving the theorem.

The proof is by induction on the infix ordering. Fix for the rest of this section an infix class $J \subseteq S$. We partition S into two parts:

$$\underbrace{\text{easy elements}}_{\text{proper prefixes of } J} \cup \underbrace{\text{hard elements}}_{\text{the rest}}.$$

The induction hypothesis says L_a is mso definable for every easy $a \in S$. We need to prove the same thing for every $a \in J$.

We begin with an observation about smooth products, which follows from the Ramsey argument that was used in Theorem 3.2. We say that $w \in S^\circ$ is J -smooth if every finite infix of w has a product in J . This is a lifting to infinite words of the notion of smoothness that was used in Section 1.3. In particular, by Claim 1.22 from that section, a \circ -word is J -smooth if and only if all of its infixes of length at most two are J -smooth. The following lemma describes the products of certain J -smooth words.

Lemma 3.24. *For every idempotent $e \in J$ the following holds. Let $w \in J^\circ$ be J -smooth. If w is an ω -word, then its product is ae^ω , where a depends only on e and the first letter in w . If w is an $(\omega* + \omega)$ -word, then its product is $e^{\omega*}e^\omega$.*

Since the lemma is true for every choice of idempotent e , it follows that $e^{\omega*}e^\omega$ does not depend on the choice of e .

Proof The main observation is the following claim.

Claim 3.25. *If w is J -smooth and has first letter e , then its product is e^ω .*

Proof By Lemma 3.4, the product of w is equal to af^ω , for some a, f . Since w is J -smooth, a and f belong to J . Since the first letter of w is e , we have $ea = a$. Since f is infix equivalent to e , it admits a decomposition $f = xee y$. Therefore

$$af^\omega = ea(xee y)^\omega = \underbrace{eaxe}_g (\underbrace{eyxe}_h)^\omega.$$

We now continue as in the proof of Lemma 3.5: because g, h, e are in the same group, then $g^\omega = e^\omega = h^\omega$, and therefore $gh^\omega = e^\omega$. \square

The claim immediately proves the lemma. Indeed, consider a J -smooth ω -word with first letter a . The first letter admits a decomposition axe for some x , because every prefix class intersects the suffix class of e . By the above claim, the product of every J -smooth ω -word that begins with a is equal to axe^ω . A similar argument works when the positions are ordered as the integers: every J -smooth $(\omega^* + \omega)$ -word has the same product as a smooth $(\omega^* + \omega)$ -word with an infix ee , and the latter has product $e^{\omega^*} e^\omega$ thanks to the claim and its symmetric version for ω^* . \square

In the rest of the proof, we will use the following terminology. We say that a colouring (a function from \circ -words to a finite set of colours) is mso definable if for every colour, its inverse image is an mso definable language. We say that a colouring λ is mso definable on a subset L of inputs if there exists an mso definable colouring that agrees with λ on inputs from L .

The strategy for the rest of the proof is as follows. Define $L_J \subseteq S^\circ$ to be the \circ -words that have product in J . We first show in Lemma 3.26 that the colouring

$$w \in S^\circ \quad \mapsto \quad \text{prefix class of the product of } w$$

is mso definable on L_J . Next, we use this result about prefixes and a symmetric one for suffixes, to show in Lemma 3.28 that the product operation is mso definable on L_J . Finally, in Lemma 3.31 we show that the language L_J is definable in mso. We can then conclude as follows: a \circ -word has product $a \in J$ if and only if it belongs to L_J , and the colouring from Lemma 3.28 step maps it to a . It remains to prove the lemmas.

Lemma 3.26. *The following colouring is mso definable on L_J :*

$$w \in S^\circ \quad \mapsto \quad \text{prefix class of the product of } w.$$

Proof We write $H \subseteq S^\circ$ for the \circ -words with a hard product. This language is definable in mso, as the complement of the easy products that are definable by induction assumption. For an interval in w , define its product to be the product

of the infix of w that is induced by the interval. An interval is called *easy* if its product is easy, otherwise it is called *hard*. By the induction assumption, we can check in MSO if an interval is easy or hard. An interval is called *almost easy* if all of its proper sub-intervals are easy.

Claim 3.27. *The product operation is MSO definable on easy intervals.*

Proof If there is a last position, then the product can be easily computed: remove the last position, compute the product, and then add the last position. Otherwise, if there is no last position, then we can use Lemma 3.4 to see that an almost easy interval has product $b \in S$ if and only if it belongs to

$$L_a(L_e)^\omega \quad \text{for some easy } a, e \text{ such that } ae^\omega = a.$$

The above condition can be formalised in MSO thanks to the induction assumption and Lemma 3.23. \square

Define a *prefix interval* to be a nonempty downward closed interval. We will compute in MSO the prefix class of some hard prefix interval; if the \circ -word is in L_J then this hard prefix has the same prefix class as w . We do a case disjunction, depending on whether or not there is an easy prefix interval (which can clearly be checked in MSO).

- Suppose first that there is no easy prefix interval. This means that either w has a first letter which is in J , or $w \in H^{\omega^*}$. In the first case, the first letter uniquely determines the prefix class of the product of w . In the second case, when $w \in H^{\omega^*}$, then under the assumption of $w \in L_J$, we can use Lemma 3.24 to conclude that the product of w is in the prefix class of e^{ω^*} , where e is some arbitrarily chosen idempotent from J .
- Suppose next that there is some easy prefix interval. Let X be the union of all easy prefix intervals. This is an almost easy interval, and therefore its product $a \in S$ can be computed thanks to Claim 3.27. If a is hard, then we know the prefix class of w . Otherwise, if a is easy, it follows that after removing X , we get a \circ -word as in the first case, and we can use that case to compute the prefix class.

\square

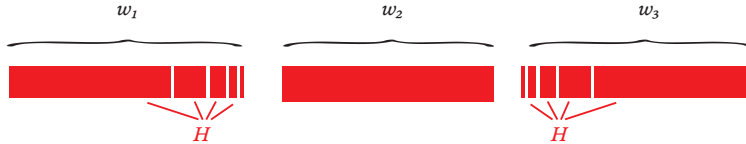
Lemma 3.28. *The product operation of S is MSO definable on L_J .*

Proof Let $w \in L_J$. We use the terminology about intervals from the proof of Lemma 3.26.

Claim 3.29. *There exists a factorisation $w = w_1w_2w_3$ such that:*

- w_1 is either empty or in H^ω ;
- w_2 is a finite concatenation of almost easy \circ -words;
- w_3 is either empty or in $H^{\omega*}$.

Here is a picture of the factorisation:



Proof Define a *limit prefix* of w to be any prefix interval which induces a \circ -word in H^ω . Limit prefixes are closed under (possibly infinite) unions. If there is a limit prefix, then there is a maximal one, namely the union of all limit prefixes (if there are not limit prefixes, we define the maximal limit prefix to be empty). Define $w_1 \in H^\omega$ to be the maximal limit prefix of w (if no limit prefix exists, then w_1 is empty). Remove the prefix w_1 , and to the remaining part of the word apply a symmetric process, yielding a suffix $w_3 \in H^{\omega*}$ and a remaining part w_2 . This is the factorisation in the statement of the claim.

It remains to show that w_2 is a finite concatenation of almost easy \circ -words. By construction, the remaining part w_2 does not have any prefix in H^ω , nor does it have any suffix in $H^{\omega*}$. Take the union of all easy prefixes of w_2 (this union exists, because w_2 has no suffix in $H^{\omega*}$, and it is almost easy), and cut it off. After repeating this process a finite number of times, we must exhaust all of w_2 , since otherwise there would be a prefix in H^ω . Therefore, w_2 is a finite concatenation of almost easy \circ -words. \square

Let w_1, w_2, w_3 be as in the above claim. By Lemma 3.24, the product of w_1 is uniquely determined by its prefix class (under the assumption that the entire \circ -word belongs to L_J). Therefore, thanks to Lemma 3.26, we can compute in mso the product of the w_1 . Symmetrically, we can compute the product of w_3 . It remains to compute the product of w_2 . This is done in the following claim.

Claim 3.30. *If a \circ -word is a finite concatenation almost easy \circ -words, then its product can be computed in mso.*

Proof By the Kleene theorem about regular expressions being equivalent to finite automata, the set of finite concatenations of almost easy intervals can be described using a regular expressions, where the atomic expressions describe almost easy words of given product. Such a regular expression can be formalised in mso thanks to Lemma 3.23 \square

□

Lemma 3.31. *The language L_J is mso definable.*

Proof Define $I \subseteq S$ to be the hard elements which are not in J . This is an ideal in the \circ -semigroup S , i.e. if $w \in S^\circ$ has at least one letter in I , then its product is in I . Define L_I to be the \circ -words with product in I . Again, this is an ideal, this time in the free \circ -semigroup S° . We will show how to define L_I in mso; it will follow that L_J is mso definable as

$$L_J = H - L_I.$$

The key is the following characterisation of L_I . Define an *error* to be a \circ -word in S° which satisfies at least one of the following conditions:

- binary error: belongs to $L_a L_b$ for some $a, b \in S - I$ such that $ab \in I$;
- ω -error: belongs to $(L_a)^\omega$, for some $a \in S - I$ such that $a^\omega \in I$;
- ω^* -error: belongs to $(L_a)^{\omega^*}$, for some $a \in S - I$ such that $a^{\omega^*} \in I$;
- shuffle error: is in the shuffle of $\{L_a\}_{a \in A}$ for some $A \subseteq S - I$ such that $A^\eta \in I$.

Note that in the above definition, we can use languages L_a for $a \in J$. These languages are not yet known to be definable in mso.

Claim 3.32. *A \circ -word belongs to L_I if and only if it has an error infix.*

Proof Clearly every error is in L_I , and since L_I is an ideal, it follows that L_I contains every \circ -word with an error infix. We are left with the converse implication: every \circ -word in L_I contains an error infix. To prove this implication, we will show that the language

$$L = \{w \in S^\circ : \text{if } w \in L_I \text{ then } w \text{ has an error infix}\}$$

satisfies the assumptions of Lemma 3.14, with λ being the product operation in S . The conclusion of Lemma 3.14 will then say that L is equal to S° , thus showing that every \circ -word in L_I has an error infix.

The first assumption of Lemma 3.14 says that L is closed under binary concatenation. Suppose that $u, v \in L$. We need to show that $uv \in L$. Suppose that $uv \in L_I$. If $u \in L_I$, then it has an error infix by assumption on $u \in L$, and therefore also uv has an error infix. We argue similarly if $v \in L_I$. Finally, if both u, v have products in $S - I$, then uv is a binary error.

The remaining assumptions of Lemma 3.14 are checked the same way. □

As remarked before Claim 3.32, the definition of errors refers to languages L_a with $a \in J$, which are not yet known to be definable in mso. We deal with this issue now. By Lemma 3.28, for every $a \in J$ there an mso definable language

which contains all \circ -words that have product a , and does not contain any \circ -words that have product in $J - \{a\}$. By removing the \circ -words with easy products from that language, we get an mso definable language K_a with

$$L_a \subseteq K_a \subseteq L_a \cup L_I.$$

Define a *weak error* in the same way as an error, except that K_a is used instead of L_a for $a \in J$. Since K_a is obtained from L_a by adding some words from the ideal L_I , it follows from Claim 3.32 that a \circ -word is in L_I if and only if it has an infix that is a weak error. Finally, weak errors can be defined by an expression which uses mso definable languages and the closure operators from Lemma 3.23, and therefore weak errors are mso definable. It follows that L_I is mso definable, and therefore L_J is mso definable. \square

As we have already remarked when describing the proof strategy, the above lemma completes the proof of the induction step in Theorem 3.22. Indeed, a \circ -word has product $a \in J$ if and only if it belongs to L_J and it is assigned a by the colouring from Lemma 3.28.

Exercises

Exercise 97. (2) The syntax of star-free expression for \circ -words is the same as for finite words, except that the complementation operation is interpreted as $\Sigma^\circ - L$ instead of $\Sigma^* - L$. Define a \circ -star-free language to be a language $L \subseteq \Sigma^\circ$ that is defined by a star-free expression. Show that if L is \circ -star-free, then its syntactic \circ -semigroup is aperiodic, but the converse implication fails.

Exercise 98. (1) What is the modification for \circ -star-free expressions that is needed to get first-order logic (over the ordered model)?

Exercise 99. (2) Show that if $L \subseteq \Sigma^\circ$ is \circ -star-free, then the same is true for every language recognised by its syntactic \circ -semigroup.

Exercise 100. (2) Show that if $L \subseteq \Sigma^\circ$ is \circ -star-free, then the same is true for L^ω .

Exercise 101. (2) Show that if S is aperiodic, then the constructions from Lemmas 3.26 and 3.28 can be done using \circ -star-free expressions.

Exercise 102. (2) Show that $L \subseteq \Sigma^\circ$ is \circ -star-free if and only if its syntactic \circ -semigroup is finite, aperiodic and satisfies¹⁴:

$$e^{\omega^*} = e = e^\omega \quad \Rightarrow \quad e = \{e\}^\eta \quad \text{for every idempotent } e.$$

Hint: use Exercises 100 and 101.

Exercise 103. (1) Show that languages of \circ -words definable in first-order logic (in the ordered model) are not closed under concatenation LK .

Exercise 104. (2) We say that a product operation $\pi : S^\circ \rightarrow S$ is regular-associative if it satisfies the associativity condition from Definition 3.9, but with the diagrams restricted so that only

$$S^\bullet = \{w \in S^\circ : w \text{ is regular}\}$$

is used instead of S° . Show that if S finite and $\pi : S^\circ \rightarrow S$ is mso definable and regular-associative, then π is associative.

Exercise 105. (2) Show that if S is finite and $\pi : S^\bullet \rightarrow S$ is regular associative, then it can be extended uniquely to an associative product $\bar{\pi} : S^\circ \rightarrow S$. Hint: the mso formulas defined in the proof of Theorem 3.22 depend only on the Lauchli-Leonard operations of the \circ -semigroup S .

¹⁴ This exercise is based on
 [9] Colcombet and Sreejith, "Limited Set Quantifiers over Countable Linear Orderings",
 2015, Theorem 2, item 2.

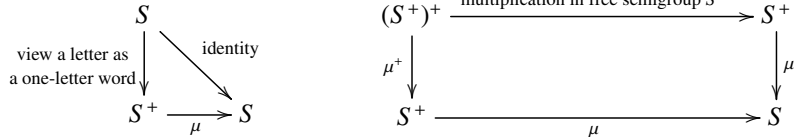
PART TWO

BEYOND WORDS

4

Monads

As discussed in Chapter 1, instead of viewing a semigroup as having a binary multiplication operation, one could think of a semigroup as a set S equipped with a multiplication operation $\mu : S^+ \rightarrow S$, which is associative in the sense that the following two diagrams commute:



The same is true for monoids, with $*$ used instead of $+$, and for \circ -semigroups, with \circ used instead of $+$. In this chapter, we examine the common pattern behind this constructions, which is that they are the Eilenberg-Moore algebras for the monads of $+$ -words, $*$ -words and \circ -words, respectively.

From the perspective of this book, the idea behind monads is the following. Instead of first defining not necessarily free algebras (e.g. semigroups) and then defining free algebras (e.g. the free semigroup) as a special case, an opposite approach is used. We begin with the free algebra (which is the monad), and then other, not necessarily free, algebras are defined as a derived notion (which is the Eilenberg-Moore algebras of the monad). This opposite approach is useful for less standard algebras such as graphs: axiomatising the not necessarily free algebras is possible but tedious and not intuitive, while the free algebra is very natural, because it consists of graphs with a certain substitution structure.

4.1 Monads and their Eilenberg-Moore algebras

This section, presents the basic definitions for monads and their algebras. These notions make sense for arbitrary categories. However, for simplicity we use the category of sets and functions, because this is where most of our examples live. In later chapters we will consider multi-sorted sets (e.g. sets with sorts $\{+, \omega\}$ for ω -semigroups, or sets with sorts $\{0, 1, \dots\}$ for hypergraphs), but this is as far as we go with respect to the choice of categories.

Definition 4.1 (Monad). A *monad* in the category of sets¹ consists of the following ingredients:

- *Structures*: for every set X , a set TX ;
- *Substitution*: for every function $f : X \rightarrow Y$, a function $Tf : TX \rightarrow TY$;
- *Unit and free multiplication*: for every set X , two functions

$$\underbrace{\text{unit}_X : X \rightarrow TX}_{\text{the unit of } X} \quad \underbrace{\text{mult}_X : TTX \rightarrow TX}_{\text{free multiplication on } X}.$$

These ingredients are subject to six axioms

$$\begin{array}{ccc} \underbrace{(4.1)} & \underbrace{(4.2) \quad (4.3)} & \underbrace{(4.4) \quad (4.5) \quad (4.6)} \\ \text{T is a functor} & \text{unit and multiplication} & \text{TX with free multiplication} \\ & \text{are natural transformations} & \text{is an Eilenberg-Moore algebra,} \\ & & \text{and one more associativity axiom} \end{array}$$

which will be described later in this section.

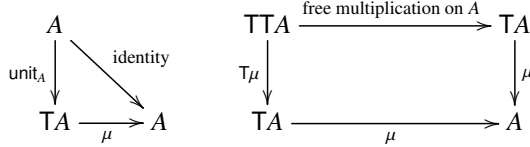
Before describing the monad axioms, we discuss some examples, and define Eilenberg-Moore algebras. The purpose of the monad axioms is to ensure that Eilenberg-Moore algebras are well-behaved. Therefore it is easier to see the monad axioms after the definition of Eilenberg-Moore algebras.

Example 4.2 (Monad of finite words). The monad of finite words is defined as follows. The structures are defined by $TX = X^*$. For a function f , its corresponding substitution Tf is defined by applying f to every letter in the input word. The unit operation maps a letter to the one-letter word consisting of this letter. Free multiplication flattens a word of words into a word. The monad of \circ -words is defined as in the same way, except that one uses \circ -words.

For this book, the key notion about monads is their Eilenberg-Moore algebras. The idea is that TX describes the free algebra, while the Eilenberg-Moore algebras are the algebras that are not necessarily free.

¹ The same definition can be applied to any other category, by using “object” instead of “set”, and “morphism” instead of “function”.

Definition 4.3 (Eilenberg-Moore algebras). An Eilenberg-Moore algebra in a monad T , also called a T -algebra, consists of an *underlying set* A and a *multiplication operation* $\mu : TA \rightarrow A$, subject to the following associativity axioms:



The reader will recognise, of course, the diagrammatic definitions of semi-groups, monoids and ω -semigroups. Also, the diagrammatic definition of ω -semigroup will fall under the scope of the above definition, if we think about ω -semigroups as living in the category of sets with two sorts $\{+, \omega\}$.

By abuse of notation, we use the same letter to denote a T -algebra and its underlying set, assuming that the multiplication operation is clear from the context. Also, if the monad T is clear from the context, we will say algebra instead of T -algebra.

Example 14. [Group monad] The *free group over a set X* is defined to be

$$\underbrace{(X + X)^*}_{\substack{\text{two copies of } X, \\ \text{one blue, and one red}}}$$

modulo the identities

$$xx = xx = \varepsilon \quad \text{for every } x \in X,$$

where ε represents the empty word, while x and x represent the blue and red copies of x . The identities can be applied in any context. For example,

$$zx \quad zyx \quad zzzxyy,$$

represent the same element of the free group. Define T to be the monad where TX is the free group over X , and the remaining monad structure is defined similarly as for finite words, except that we have the two copies of the alphabet, and the identities. The unit operation maps an element to its blue copy.

An algebra over this monad is the same thing as a group. Indeed, if G is an algebra over this monad, with multiplication μ , then the group structure is recovered as follows:

$$\underbrace{1 \stackrel{\text{def}}{=} \mu(\varepsilon)}_{\text{group identity}} \quad \underbrace{x^{-1} \stackrel{\text{def}}{=} \mu(x)}_{\text{group inverse}} \quad \underbrace{x \cdot y \stackrel{\text{def}}{=} \mu(xy)}_{\text{group operation}}$$

The axioms of a group are easily checked, e.g.

$$\begin{aligned}
 x \cdot x^{-1} &= && \text{(definition of inverse)} \\
 x \cdot \mu(x) &= && \text{(unit followed by multiplication is the identity, i.e. axiom 4.4)} \\
 \mu(x) \cdot \mu(x) &= && \text{(definition of the group operation)} \\
 \mu(\mu(x)\mu(x)) &= && \text{(associativity of multiplication, i.e. axiom 4.5)} \\
 \mu(xx) &= && \text{(equality in the free group)} \\
 \mu(\varepsilon) &= && \text{(definition of group identity)} \\
 &1
 \end{aligned}$$

For the converse, we observe that for every group G , its group multiplication can be extended uniquely to an operation of type $TG \rightarrow G$, and the resulting operation will be associative in the sense required by Eilenberg-Moore algebras. \square

4.1.1 Axioms of a monad

We now describe the axioms of a monad.

Functoriality. The first group of axioms says that the first two ingredients (the structures and substitutions) of a monad are a functor in the sense of category theory. This means that substitutions preserve the identity and composition of functions. Preserving identity means that if we apply T to the identity function on X , then the result is the identity function on TX . Preserving composition means that the composition of substitutions is the same as the substitution of their composition, i.e. for every functions $f : X \rightarrow Y$ and $g : Y \rightarrow Z$, the following diagram commutes

$$\begin{array}{ccc}
 TX & \xrightarrow{Tf} & TY \\
 & \searrow T(g \circ f) & \downarrow Tg \\
 & & TZ
 \end{array} \tag{4.1}$$

Naturality. The naturality axioms say that for every function $f : X \rightarrow Y$, the following diagrams commute.

$$\begin{array}{ccc}
 X & \xrightarrow{f} & Y \\
 \text{unit}_X \downarrow & & \downarrow \text{unit}_Y \\
 TX & \xrightarrow{Tf} & TY
 \end{array} \tag{4.2}$$

$$\begin{array}{ccc}
 \mathbb{T}\mathbb{T}X & \xrightarrow{\mathbb{T}\tau_f} & \mathbb{T}\mathbb{T}Y \\
 \downarrow \text{free multiplication on } X & & \downarrow \text{free multiplication on } Y \\
 \mathbb{T}X & \xrightarrow{\tau_f} & \mathbb{T}Y
 \end{array} \quad (4.3)$$

In the language of category theory, this means that the unit and free multiplication are natural transformations. Also, as we will see later on, the second naturality axiom (naturality of free multiplication) says that the substitution τ_f is a homomorphism between the free algebras $\mathbb{T}X$ and $\mathbb{T}Y$.

Associativity. We now turn to the most important monad axioms, which ensure the the Eilenberg-Moore algebras are well behaved. The main associativity axiom says that for every set X , the set $\mathbb{T}X$ equipped with free multiplication on X is a \mathbb{T} -algebra (we call this the free \mathbb{T} -algebra over X , or simply free algebra if the monad is clear from the context). By unravelling the definitions, this means that the following two diagrams commute:

$$\begin{array}{ccc}
 \mathbb{T}X & & \\
 \text{unit}_{\mathbb{T}X} \downarrow & \searrow \text{identity} & \\
 \mathbb{T}\mathbb{T}X & \xrightarrow{\text{free multiplication on } X} & \mathbb{T}X
 \end{array} \quad (4.4)$$

$$\begin{array}{ccc}
 \mathbb{T}\mathbb{T}\mathbb{T}X & \xrightarrow{\text{free multiplication on } \mathbb{T}X} & \mathbb{T}\mathbb{T}X \\
 \mathbb{T}(\text{free multiplication on } X) \downarrow & & \downarrow \text{free multiplication on } X \\
 \mathbb{T}\mathbb{T}X & \xrightarrow{\text{free multiplication on } X} & \mathbb{T}X
 \end{array} \quad (4.5)$$

Apart from the above two, there is one more associativity axiom, namely:

$$\begin{array}{ccc}
 \mathbb{T}X & & \\
 \mathbb{T}(\text{unit}_X) \downarrow & \searrow \text{identity} & \\
 \mathbb{T}\mathbb{T}X & \xrightarrow{\text{free multiplication on } X} & \mathbb{T}X
 \end{array} \quad (4.6)$$

This completes the axioms of a monad, and the definition of a monad.

4.1.2 Homomorphisms and recognisable languages

A homomorphism between two \mathbb{T} -algebras is any function between their underlying sets which is consistent with the multiplication operation.

Definition 4.4 (Homomorphism). Let T be a monad. A T -homomorphism is a function $h : A \rightarrow B$ on the underlying sets of two T -algebras A and B , which makes the following diagram commute

$$\begin{array}{ccc}
 TA & \xrightarrow{\tau_h} & TB \\
 \text{multiplication in } A \downarrow & & \downarrow \text{multiplication in } B \\
 A & \xrightarrow{h} & B
 \end{array}$$

When the monad is clear from the context, we simply write homomorphism, instead of T -homomorphism. Again, the reader will recognise the notion of homomorphism for semigroups, monoids and \circ -semigroups.

In the rest of this section, we describe some basic properties of homomorphisms.

Lemma 4.5. *Homomorphisms are closed under composition.*

Proof Consider two homomorphisms

$$A \xrightarrow{g} B \xrightarrow{h} C.$$

Saying that the composition $h \circ g$ is a homomorphism is the same as saying that the perimeter of the following diagram commutes:

$$\begin{array}{ccccc}
 TA & & \xrightarrow{\tau(h \circ g)} & & TC \\
 \downarrow \text{multiplication in } A & & \searrow \tau_g & & \nearrow \tau_h \\
 & & TB & & \\
 & & \downarrow \text{multiplication in } B & & \\
 & & B & & \\
 \nearrow g & & & & \searrow h \\
 A & & \xrightarrow{h \circ g} & & C \\
 \downarrow \text{multiplication in } C & & & & \\
 & & & &
 \end{array}$$

The upper face commutes because of the functoriality axioms (substitutions are compatible with composition). The left and right faces commute by assumption that g and h are homomorphisms, and the lower face commutes by definition. \square

Recall that we defined the *free algebra over X* to be X equipped with free multiplication on X . The monad axioms say that this is indeed an algebra. It is called free because of the universal property given in the following lemma.

Lemma 4.6 (Free Algebra Lemma). *For every set X , the free algebra $\mathbb{T}X$ has the following universal property:*

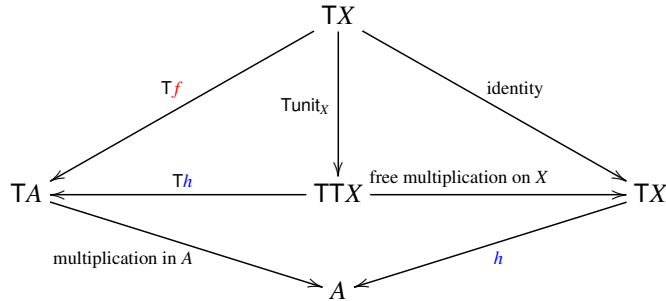
$$\forall \exists! \quad X \begin{array}{c} \xrightarrow{\text{function on sets}} A \\ \searrow \text{unit}_X \\ \rightarrow \mathbb{T}X \end{array} \begin{array}{c} \uparrow \text{homomorphism of } \mathbb{T}\text{-algebras} \\ \mathbb{T}X \end{array} \quad (4.7)$$

Proof We begin by showing that there is at least one blue homomorphism h for each red function f ; later we show that this homomorphism is unique. Let $\mu : \mathbb{T}A \rightarrow A$ be multiplication in the algebra A . Define h to be the composition of the following functions:

$$\mathbb{T}X \xrightarrow{\mathbb{T}f} \mathbb{T}A \xrightarrow{\mu} A.$$

The axiom on naturality of free multiplication says that $\mathbb{T}f$ is a homomorphism from the free algebra $\mathbb{T}X$ to the free algebra $\mathbb{T}A$. The associativity axiom in the definition of an Eilenberg-Moore algebra says that multiplication μ is a homomorphism from the free algebra $\mathbb{T}A$ to the algebra A . Therefore, h is a homomorphism, as the composition of two homomorphisms $\mathbb{T}f$ and μ .

We now show uniqueness – every homomorphism h which makes the diagram must be equal to the one described above. Consider the following diagram:



The upper left face commutes by applying \mathbb{T} to the assumption that h extends f . (Applying \mathbb{T} preserves commuting diagrams, because of the functoriality axioms.) The upper right face commutes by the first associativity axiom. The lower face commutes, because it says that h is a homomorphism. Therefore, the perimeter of the diagram commutes. The perimeter says that h must be equal to $\mathbb{T}f$ followed by multiplication in A , and therefore h is unique. \square

Compositional functions. Fix a monad \mathbb{T} . Suppose that A is an algebra, while B is a set, which is not (yet) equipped with a multiplication operation. We say

that a function $h : A \rightarrow B$ is *compositional* if

$$\exists \mu \quad \begin{array}{ccc} \mathbb{T}A & \xrightarrow{\mathbb{T}h} & \mathbb{T}B \\ \text{multiplication in } A \downarrow & & \downarrow \mu \\ A & \xrightarrow{h} & B \end{array}$$

This is the same notion of compositionality as was used for monoids, semi-groups and \circ -semigroups in part I of the book. For the same reason as before, surjective compositional functions are equivalent to surjective homomorphisms, as stated in the following lemma.

Lemma 4.7. *If A is an algebra, B is a set, and $h : A \rightarrow B$ is compositional and surjective, then there is a (unique) multiplication operation on B which turns it into an algebra and h into a homomorphism.*

Proof The multiplication operation – no surprises here – is μ from the definition of a compositional function. The diagram in the definition of a compositional function is the same diagram as in the definition of a homomorphism, and therefore if B equipped with μ is an algebra, then h is a homomorphism. It remains to show that B equipped with μ is indeed an algebra. We only prove the more interesting of the two associativity diagrams.

We first observe that \mathbb{T} preserves surjectivity of functions. Indeed, if a function $h : A \rightarrow B$ is surjective, then it has a one-sided inverse, i.e. a function $h^{-1} : B \rightarrow A$ such that $h \circ h^{-1}$ is the identity on B . By the functoriality axioms, $\mathbb{T}h^{-1}$ is a one-sided inverse for $\mathbb{T}h$, and therefore $\mathbb{T}h$ is also surjective. This argument justifies the surjectivity annotation (double-headed arrows) in the following diagram.

$$\begin{array}{ccccc} \mathbb{T}\mathbb{T}B & \xrightarrow{\text{free multiplication on } B} & & \xrightarrow{\mathbb{T}h} & \mathbb{T}B \\ & \swarrow \mathbb{T}\mathbb{T}h & \mathbb{T}\mathbb{T}A & \xrightarrow{\text{free multiplication on } A} & \mathbb{T}A & \searrow \mathbb{T}h & \\ & & \downarrow \mathbb{T}(\text{multiplication in } A) & & \downarrow \text{multiplication in } A & & \\ \mathbb{T}B & \xrightarrow{\mathbb{T}h} & \mathbb{T}A & \xrightarrow{\text{multiplication in } A} & A & \xrightarrow{h} & B \\ & & & & \downarrow h & & \\ & & & & B & \xrightarrow{\mu} & B \end{array}$$

The central face commutes by the assumption that A is an algebra. The upper face commutes by naturality of free multiplication. The right and lower faces

commute by definition of a compositional function, and the left face commutes by the same definition with \mathbb{T} applied to it. It follows that all paths that begin in $\mathbb{T}A$ and end in B denote the same function. Since h is surjective, it follows that the perimeter of the diagram commutes. This proves the second of the associativity diagrams in the definition of an Eilenberg-Moore algebra. \square

Recognisable colourings and languages. In this book, we are most interested in the Eilenberg-Moore algebras as recognisers of languages. A language is a subset L of a free algebra $\mathbb{T}\Sigma$. (Typically we are interested in the case where the alphabet Σ is finite, but this assumption does not seem to play a role in the results we care about, so we omit it.) A language is called *recognisable* if it is recognised by a finite algebra, as explained in the following definition (which uses a slightly more general notion of language, called colourings).

Definition 4.8 (Recognisable colourings). Fix a monad \mathbb{T} . An *algebra colouring* is defined to be any function from an algebra to a set of colours². A *finite algebra* is an algebra where the underlying set is finite³. An algebra colouring $\lambda : A \rightarrow U$ is called *recognisable* if it factors through a homomorphism into a finite algebra, as expressed in the following diagram:

$$\exists \quad \begin{array}{ccc} A & \xrightarrow{\lambda} & U \\ & \searrow \text{homomorphism} & \uparrow \text{colouring} \\ & \text{into a finite algebra} & B \end{array}$$

Note that a recognisable colouring will necessarily use finitely many colours.

A language can be viewed as the special case of an algebra where the algebra is a free algebra and there are two colours “yes” and “no”. For languages, we prefer set notation, e.g. we can talk about the complement of a language, or order languages by inclusion. The above definition is easily seen to coincide with the notions of recognisability for semigroups, monoids and \circ -semigroups that were discussed in the first part of this book. In the next section, we give more examples.

² For some monads, it will be more useful to deviate from this definition. For example, in the monad from Example 23 that deals with vector spaces, a more useful notion of colouring is a linear map to the underlying field. Therefore, one could think of a parametrised notion of recognisability, where the notion of “algebra colouring” is taken as a parameter.

³ Like for algebra colourings, sometimes this notion of finite algebra is not the right one. In the monad from Example 23, the more useful notion is that a finite algebra is one where the underlying set is a vector space of finite dimension. Again, one could think of the notion of “finite algebra” as being a parameter.

Exercises

Exercise 106. Consider a monad \mathbb{T} in the category of sets. For a binary relation R on a set X , define

$$R^X \subseteq (\mathbb{T}X) \times (\mathbb{T}X)$$

to be the binary relation on $\mathbb{T}X$ that is defined by

$$R^X = \{((\mathbb{T}\pi_1)(t), (\mathbb{T}\pi_2)(t)) : t \in \mathbb{T}R\}.$$

Does transitivity of R imply transitivity of $R^{\mathbb{T}}$?

Exercise 107. For an algebra A with multiplication operation $\mu : \mathbb{T}A \rightarrow A$, define its powerset as follows: the underlying set is the powerset $\mathbb{P}A$, and multiplication is defined by

$$t \in \mathbb{P}A \mapsto \{\mu(s) : s \in^{\mathbb{T}} t\},$$

where $\in^{\mathbb{T}}$ is defined as in Exercise 106. Show an example of a monad \mathbb{T} where this construction does not yield an algebra.

Exercise 108. Find a monad \mathbb{T} in the category of sets where the following implication is not true: if $L \subseteq \mathbb{T}\Sigma$ is recognisable, and $h : \mathbb{T}\Sigma \rightarrow \mathbb{T}\Gamma$ is a homomorphism, then $h(L)$ is recognisable.

Exercise 109. Show that the monad of groups satisfies the implication from Exercise 108.

Exercise 110. Consider the implication in the previous exercise. Show that even if we restrict h to substitutions, i.e. homomorphisms of the form $\mathbb{T}f$ for some $f : \Sigma \rightarrow \Gamma$, then the implication can still be false in some monads.

4.2 A zillion examples

Monads have an abundance of interesting examples. This section is devoted to a collection of such examples, with an emphasis on the algebras arising from the monads, and the languages recognised by the finite algebras.

Example 15. [Finite multisets] Define $\mathbb{T}X$ to be the finite multisets over X ,

i.e. functions of type $X \rightarrow \mathbb{N}$ which have finite support (all but finitely many elements are mapped to 0). Functions are lifted to multisets point-wise, e.g.

$$\underbrace{\{x_1, \dots, x_n\}}_{\text{red brackets indicate multisets}} \xrightarrow{\mathbb{T}f} \{f(x_1), \dots, f(x_n)\}.$$

Another perspective on finite multisets is that they are finite words modulo commutativity $xy = yx$. The unit is $x \mapsto \{x\}$, and free multiplication is simply removing nested brackets, e.g.

$$\{\{x, y\}, \{z\}\} \mapsto \{x, y, z\}.$$

This is a monad. An algebra over this monad is the same thing as commutative monoid. Recognisable languages over this monad are the same things are regular languages – in the usual sense – which are commutative, see Exercise 11.

If we lift the restriction on finite supports, then we do not get a monad. The problem is with the substitutions: if $f : X \rightarrow \{a\}$ is the constant function with an infinite domain, then there is no way to define

$$(\mathbb{T}f)\{\underbrace{x_1, x_2, \dots}_{\text{infinitely many distinct elements}}\}.$$

The problem is that the above multiset should contain a infinitely many times. To overcome this problem, we would need to allow the multisets to have infinitely many copies of an element. \square

Example 16. [Idempotent finite words] Define $\mathbb{T}X$ to be finite words X^* , modulo the equation $ww = w$. This equation can be applied to arbitrary words, e.g.

$$abcabc = abc.$$

The remaining ingredients of the monad are defined in the natural way. An algebra over this monad is the same thing as an idempotent monoid, i.e. a monoid where all elements are idempotent. Green and Rees show that if X is a finite finite set, then $\mathbb{T}X$ is finite⁴. It follows that for every finite alphabet, there are finitely many languages over this alphabet, and all of them are recognisable. \square

Example 17. [Powersets] The *powerset monad*, and its variant the *finite powerset monad*, are defined in the same way as the multiset monad, except that we use sets (or finite sets) instead of multisets. The substitutions are defined

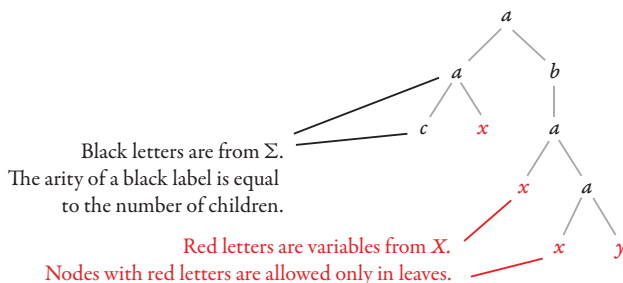
⁴ [15] Green and Rees, “On semi-groups in which $x' = x$ ”, 1952, p. 35

via images (in the language of category theory, we use the covariant powerset functor):

$$A \subseteq X \quad \xrightarrow{\mathbb{T}f} \quad \{f(x) : x \in A\} \subseteq Y.$$

The algebras over the finite powerset monad are the same thing as monoids that are commutative and idempotent. If X is a finite set, then both powerset monads generate finite sets; and therefore all languages over finite alphabets are recognisable. \square

Example 18. [Terms] Fix a ranked set Σ , i.e. a set where every element has an associated arity in $\{0, 1, \dots\}$. Define $\mathbb{T}X$ to be the terms over Σ with variables from X , i.e. an element of $\mathbb{T}X$ is a tree that looks like this:



The unit operation maps $x \in X$ to a term which consists only of x . The substitution $\mathbb{T}f$ is defined in the natural way, by applying f to the variables and leaving the remaining part of the term unchanged. Finally, free multiplication replaces each variable with the corresponding term. It is a simple exercise to check that a \mathbb{T} -algebra is the same thing as an *algebra of type Σ* , in the sense of universal algebra⁵. In the terminology of automata theory, both of these notions are the same as deterministic bottom-up tree automata over finite trees, where Σ is the input alphabet⁶. From the above observation it follows that a language $L \subseteq \mathbb{T}X$ is recognisable in the sense of Definition 4.8 if and only if it is a regular tree language in the sense of automata theory⁷, where the input

⁵ An algebra of type Σ is defined to be an underlying set A , together with an interpretation which maps each n -ary symbol from Σ to an operation of type $A^n \rightarrow A$. See

[27] Sankappanavar and Burris, "A course in universal algebra", 1981, Definition 1.3

⁶ [33] Thatcher and Wright, "Generalized Finite Automata Theory with an Application to a Decision Problem of Second-Order Logic", 1968, Section 2

⁷ This monad describes finite trees. Finding an algebraic account of languages of infinite trees remains an open problem This problem is discussed in the following papers:

[4] Blumensath, "Regular Tree Algebras", 2018

alphabet is obtained from Σ by adding one letter of arity 0 for each element of X . If the ranked set Σ contains only letters of arity exactly one, then a T -algebra can be seen as a deterministic word automaton with input alphabet Σ , without distinguished initial and final states. \square

Example 19. [Linear orders] Define a *chain* over a set X to be a linear order with positions labelled by X , modulo isomorphism of labelled linear orders. For an infinite cardinal κ , a monad T_κ is defined as follows. The set $T_\kappa X$ consists of chains over X , which have cardinality strictly less than κ . The monad structure is defined in the same way as for \circ -words (which are the special case of this construction for where κ is the first uncountable cardinal, which is the same as \mathfrak{c} assuming the Continuum Hypothesis). Nevertheless, we give a more exact description below.

For a function f , the corresponding substitution $T_\kappa f$ is defined in the natural way, by applying f to the labels in a chain and leaving the positions and ordering unchanged. The unit maps a letter to the one letter chain with that letter. The free multiplication operation is defined using lexicographic products, as follows. Suppose that $w \in T_\kappa T_\kappa X$. The positions in the free multiplication of w are pairs (i, j) such that i is a position of w and j is a position in the label $w(i) \in T_\kappa X$ of position i in the chain w . The label of such a position is inherited from j , and the ordering is lexicographic. The cardinality of the resulting chain is at most κ , since every infinite cardinal satisfies $\kappa = \kappa^2$.

We only prove one of the monad axioms:

$$\begin{array}{ccc}
 T_\kappa T_\kappa T_\kappa X & \xrightarrow{\text{free multiplication on } T_\kappa X} & T_\kappa T_\kappa X \\
 \downarrow T_\kappa(\text{free multiplication on } X) & & \downarrow \text{free multiplication on } X \\
 T_\kappa T_\kappa X & \xrightarrow{\text{free multiplication on } X} & T_\kappa X
 \end{array}$$

Let $w \in T_\kappa T_\kappa T_\kappa X$. If, in the diagram above, we first go right and then down, then the resulting linear order will have positions of the form $((i, j), k)$, where i is a position in w , j is a position in $w(i)$, and k is a position in $w(i)(j)$. If, in the diagram, we first go down and then right, then we get positions of the form $(i, (j, k))$, where i, j, k satisfy the same conditions as above. In both cases, the tuples of positions are ordered lexicographically, and the label is inherited from k . Therefore

$$((i, j), k) \mapsto (i, (j, k))$$

[5] Bojańczyk and Klin, “A non-regular language of infinite trees that is recognizable by a sort-wise finite algebra”, 2019

is an isomorphism of labelled linear orders, and hence the two outcomes are equal as chains.

If we take $\kappa = \aleph_0$, then we get the monad of finite words. If we take κ to be the first uncountable cardinal, then we get the monad corresponding to ω -words.

If we take κ to be the first cardinal bigger than c , then T_κ describes chains of cardinality at most c . In this case, we have the following phenomenon. Recall the powerset construction that was described in Section 3.3. This construction also makes sense for chains of size at most c . Define \mathcal{X} to be the least class of T_κ -algebras which contains the syntactic algebra of the language “every a is before every b ”, and which is closed under products and the powerset construction. Using the same proof as for Lemma 3.21, one can show that every mso definable language $L \subseteq T_\kappa \Sigma$ is recognised by an algebra from \mathcal{X} . Since satisfiability for mso over the reals is undecidable⁸, it follows that there is no finite way of representing algebras from \mathcal{X} . This means that the powerset construction over finite T_κ -algebras is not computable. \square

Example 20. Consider a class \mathcal{X} of linear orders which have size bounded by some cardinal κ , and which are closed under free multiplication as defined in the previous example, when viewed as chains over a one letter alphabet. If we restrict the monad from the previous example to chains where the underlying linear order is in \mathcal{X} , then we also get a monad. This construction yields the following monads (in all cases, we assume some fixed upper bound on κ on the cardinality, e.g. we can require countability):

- well-founded words (the class of well-founded linear orders);
- scattered words (the class of scattered orders, i.e. those into which one cannot embed the rational numbers)⁹.
- dense words (the class which contains two orders: a singleton order for units, and the rational numbers).

\square

Example 21. [ω -semigroups] We now describe a monad that corresponds to ω -semigroups, see Definition 3.6. Since an ω -semigroup has two sorts, we leave

⁸ [30] Shelah, “The Monadic Theory of Order”, 1975, Theorem 7

⁹ Algebras for the monad of countable scattered words are studied in
 [26] Rispoli and Carton, “Complementation of Rational Sets on Countable Scattered Linear Orderings”, 2005

the category of sets, and use instead the category

$$\text{Set}^{(+, \omega)}$$

of sets with two sorts $+$ and ω . An object in this category is a set, where every element is assigned exactly one of two sorts, called $+$ and ω . A morphism in this category is any sort-preserving function between sorted sets. We use the name *sorted set* for the objects. We write the objects and morphisms of this category in red, to distinguish them from usual sets and functions. We also use the following notation for sorts:

$$\underbrace{X}_{\text{a sorted set}} = \underbrace{X[+]_{\text{elements of sort } +}} \cup \underbrace{X[\omega]_{\text{elements of sort } \omega}}.$$

We define a monad \mathbf{T} over this category as follows. For a sorted set X , the sorted set \mathbf{TX} is defined by:

$$(\mathbf{TX})[+] = X[+]^+ \quad \cup \quad (\mathbf{TX})[\omega] = (X[+])^*(X[\omega]) \cup (X[+])^\omega.$$

For a morphism $f : X \rightarrow Y$, the substitution morphism $\mathbf{T}f$ is defined in the natural way, by applying f to every letter. The unit and free multiplication are defined in the natural way as well. (An element of sort $+$ in \mathbf{TTX} is simply a finite nonempty word of finite nonempty words over $X[+]$, and we can use free multiplication from the monad of finite nonempty words. On sort ω , there are more cases to consider, but the definition is natural as well.) An Eilenberg-Moore algebra over this monad is the same thing as an ω -semigroup, as defined at the end of Section 3.1. \square

Example 22. [Vector spaces] Define \mathbf{TX} to be the vector space, over the field of rational numbers, where the basis is X . In other words, elements of \mathbf{TX} are finite linear combinations of elements from X with rational coefficients. For example,

$$3x + 7y - 0.5z \in \mathbf{T}\{x, y, z\}.$$

The action of \mathbf{T} on functions is defined by

$$q_1x_1 + \cdots + q_nx_n \quad \xrightarrow{\mathbf{T}f} \quad q_1f(x_1) + \cdots + q_nf(x_n).$$

The unit operation maps $x \in X$ to the corresponding basis vector, and free multiplication is defined in the natural way, as illustrated in the following example:

$$3(4x + 0.5y) - 0.2(5x - 0.1y) \quad \mapsto \quad 12x + 1.5y - x + 0.02y = 11x + 1.48y.$$

An algebra A over this monad, with multiplication μ , is also equipped with the structure of a vector space, because we can add elements

$$a + b \stackrel{\text{def}}{=} \mu(a + b)$$

and multiply them by scalars $q \in \mathbb{Q}$

$$qa \stackrel{\text{def}}{=} \mu(qa).$$

If $B \subseteq A$ is a basis for the vector space A , then the algebra A is isomorphic to TB . Therefore, over this monad, every algebra is isomorphic to a free algebra.

□

Example 23. [Algebra over a field] Define TX to be finite linear combinations of words in X^* , with rational coefficients. For example,

$$2xyx + -2xx + 0.5xyz \in T\{x, y, z\}.$$

In other words, $TX = T_{\text{vec}}X^*$, where T_{vec} is the monad of vector spaces from Example 22 and X^* is the monad of finite words¹⁰. On functions, the monad acts as follows

$$q_1x_1 + \cdots + q_nx_n \quad \xrightarrow{Tf} \quad q_1f^*(x_1) + \cdots + q_nf^*(x_n),$$

where f^* is the substitutions in the monad of finite words. The unit maps x to the linear combination which has the one-letter word x with coefficient 1. The free multiplication is defined like for polynomials, but the variables are not commuting, e.g.:

$$3(4xy - 2y)(2x - yx) \quad \mapsto \quad 24xyx - \underbrace{12xyx - 12xyx + 2yyx}_{\text{this is not 0}}.$$

Every algebra over this monad has the structure of a vector space over the rationals, but there is more structure (e.g. one can multiply two elements of the algebra)¹¹.

What is a recognisable colouring over this monad? In the context of this monad (and also the simpler monad of vector spaces from Example 22), it is more useful to work with different notions of “finite algebra” and “algebra colouring”: instead of finite algebras, one should consider finite dimensional

¹⁰ This is an example of a composite monad that arises via a distributive law of two monads.

This type of construction was first described in

[1] Appelgate et al., *Seminar on triples and categorical homology theory*, 1969, Chapter on distributive laws

¹¹ Algebras over this monad are known as “algebras over the field of rational numbers”, but we avoid this terminology due to the over-loading of “algebra over”.

algebras (i.e. those where the underlying vector space has finite dimension), and instead of algebra colourings one should consider linear maps to vector spaces. Under these adapted definitions, the algebra colourings recognised by finite algebras are exactly those which are recognised by weighted automata, see Exercise 123. \square

Exercises

Exercise 111. Consider the monad T_Σ from Example 18, where Σ is some ranked set (possibly infinite). Let X be some possibly infinite set of variables, and consider a set

$$\mathcal{E} \subseteq \underbrace{(TX) \times (TX)}.$$

elements of this set will be called identities

For a set Y , define \sim to be the least congruence on $T\Sigma$ that satisfies

$$(Tf)(t_1) \sim (Tf)(t_2) \quad \text{for every } (t_1, t_2) \in \mathcal{E} \text{ and } f : X \rightarrow Y.$$

(This congruence can be obtained by intersecting all congruences with the above property.) Define a new monad as follows: TY is equal to $T_\Sigma Y$ modulo \sim , and the remaining components of the monad are defined in the natural way. Show that this is a monad.

Exercise 112. For monads S and T , define a monad morphism from S to T to be a family of functions

$$\{\delta_X : SX \rightarrow TX\}_{X \text{ is a set}}$$

which is subject to the axioms in Figure 4.1. Using the monads from Section 4.2, give five examples of monad morphisms, and five examples of pairs of monads which do not allow a monad morphism.

Exercise 113. We say that $w \in T\Sigma$ is *regular* if $\{w\}$ is a recognisable language. Find a monad where there are no regular elements. (Hint: it appears in this section.)

Exercise 114. What is the monad for rings (commutative and non-commutative)? Semirings?

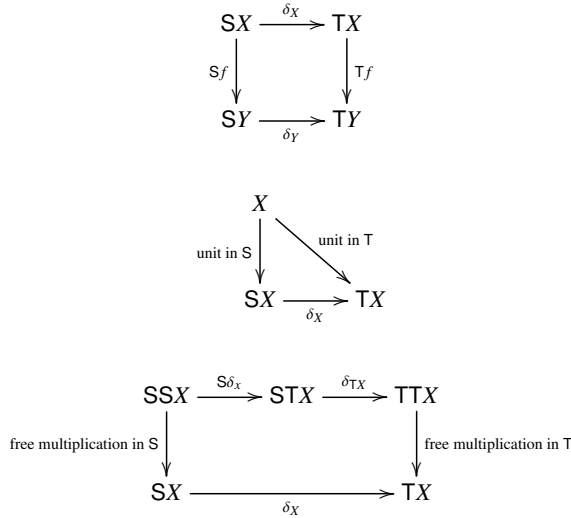


Figure 4.1 These three diagrams should commute for all sets X and all functions $f : X \rightarrow Y$. The top diagram says that $\{\delta_X\}_X$ is a natural transformation, while the bottom two diagrams say that it is compatible with unit and free multiplication.

Exercise 115. Consider the following monad T . The set TX is the set of ω -words X^ω , and the substitution $Tf : TX \rightarrow TY$ is defined coordinate-wise. The unit is $x \mapsto x^\omega$, and free multiplication is defined by

$$w \in TTX \quad \mapsto \quad (i \mapsto \underbrace{(w[i])[i]}_{\substack{i\text{-th letter of} \\ \text{of } i\text{-th letter of } w}}).$$

Show that this is a monad. Also, show that a language $L \subseteq T\Sigma$ is recognisable if and only if it is clopen in the sense of Exercise 65.

Exercise 116. Let T be the monad of countable well founded. Show that a finite algebra with universe S is uniquely determined by the operations

$$\underbrace{ab}_{\substack{\text{binary} \\ \text{product} \\ S^2 \rightarrow S}} \quad \underbrace{a^\omega}_{\substack{\text{product of} \\ aaa \dots \\ S \rightarrow S}}$$

Exercise 117. Consider the monad from Example 116. Show that a language $L \subseteq T\Sigma$ is definable in first-order logic (in the ordered model) if and only

if it is recognised by a finite T-algebra S where the underlying semigroup is aperiodic.

Exercise 118. Consider the monad from Example 116. Consider regular expressions defined by the usual operators, plus L^ω . Show that these expressions do not describe all recognisable languages.

Exercise 119. Consider the monad and regular expressions from Exercise 118. Given an effective condition on finite algebras which corresponds exactly to the Boolean combinations of regular expressions.

Exercise 120. Let T be the monad of countable scattered chains. Show that a finite algebra with universe S is uniquely determined by the operations

$$\begin{array}{ccc}
 \underbrace{ab} & \underbrace{a^\omega} & \underbrace{a^{\omega*}} \\
 \text{binary} & \text{product of} & \text{product of} \\
 \text{product} & \text{aaa}\cdots & \cdots\text{aaa} \\
 S^2 \rightarrow S & S \rightarrow S & S \rightarrow S
 \end{array}$$

Exercise 121. Consider the monads from Examples 116 and 120. In which of these monads is first-order logic (over ordered models) equivalent to star-free expressions?

Exercise 122. Consider the monad from Example 120. Which class of languages corresponds to aperiodicity (of the semigroup underlying the T-algebra)?

Exercise 123. A weighted automaton over the rationals consists of:

$$\begin{array}{ccccc}
 \underbrace{\Sigma} & \underbrace{S} & \underbrace{s_0 \in S} & \underbrace{\{\delta_a : S \rightarrow S\}_{a \in \Sigma}} & \underbrace{F : S \rightarrow \mathbb{Q}} \\
 \text{input alphabet,} & \text{state space,} & \text{initial state} & \text{state updates,} & \text{final function,} \\
 \text{which is a} & \text{which is a} & & \text{which are} & \text{which is a} \\
 \text{finite set} & \text{vector space of} & & \text{linear maps} & \text{linear map} \\
 & \text{finite dimension} & & &
 \end{array}$$

The semantics of this automaton is a function of type $\Sigma^* \rightarrow \mathbb{Q}$ defined as follows. Given an input word, do the following: start with the initial state, apply the state update for the first letter, then the state update for the second letter, and so on for all letters, and at the end apply the final function. The semantics of a weighted automaton can also be naturally extended to finite linear combinations of words over Σ , i.e. to elements of the $T\Sigma$ as in Example 23.

Show that $L : T\Sigma \rightarrow \mathbb{Q}$ is recognised by a weighted automaton if and only if it is recognised by a finite dimensional T-algebra.

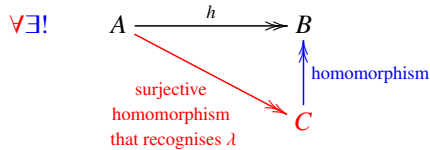
4.3 Syntactic algebras

In this section, we show that every recognisable algebra colouring has a syntactic homomorphism, i.e. a recognising homomorphism that stores the minimal amount of information. This can be viewed as a monad version of the Myhill-Nerode theorem.

Definition 4.9 (Syntactic homomorphism). The *syntactic homomorphism* of an algebra colouring $\lambda : A \rightarrow U$ is any surjective homomorphism

$$h : A \rightarrow B$$

which recognises λ and which is minimal in the sense explained in the following quantified diagram



The algebra used by the syntactic homomorphism is called the *syntactic algebra*. The syntactic algebra, if it exists, is unique up to isomorphism of algebras. Also the syntactic homomorphism is unique in the following sense: every two syntactic homomorphisms will have the same kernel (equivalence relation on A that identifies two elements with the same homomorphic image). This kernel is called the *syntactic congruence* of λ .

We are mainly interested in the case where λ describes a language, i.e. A is a free algebra, and there are two colours.

There are two main results in this section. The first one, Theorem 4.11, says that if an algebra colouring is recognisable, then it has a syntactic homomorphism. The second one, Theorem 4.16, says that a monad is finitary (roughly speaking, this means that all structures described by the monad are finite) if and only if every (not necessarily recognisable) algebra colouring has a syntactic homomorphism. To illustrate these theorems, we begin with an example of an algebra colouring that does not have a syntactic homomorphism. By Theorem 4.11, this colouring is necessarily not recognisable.

Example 24. Consider the monad of \circ -words. Define L to be the set of \circ -words over a one letter alphabet which contain every finite word as an infix. More formally,

$$L = \{w \in \{a\}^\circ : a^n \text{ is an infix of } w \text{ for every } n \in \{0, 1, \dots\}\},$$

which is a language over alphabet a , and therefore a special case of an algebra colouring. This language is not recognisable, because all finite words must have different images under any recognising homomorphism (we leave this as an exercise for the reader). We will show that L does not have a syntactic homomorphism. For $n \in \{1, 2, \dots\}$ define

$$w_n = (\text{shuffle of } \{a\}) \cdot a^n \cdot (\text{shuffle of } \{a\})$$

and define h_n to be the function

$$w \in \{a\}^\circ \quad \mapsto \quad \begin{cases} w_n & \text{if } w = w_{n+1} \\ w & \text{otherwise.} \end{cases}$$

This function is compositional for every n , and therefore it can be viewed as a homomorphism. If there would be a syntactic homomorphism, then it would need to factor through h_n . Since h_n gives the same result for w_n and w_{n+1} , therefore the same would have to be true for the syntactic homomorphism. Therefore, the syntactic homomorphism h , if it existed, would need to satisfy

$$h(w_1) = h(w_2) = \dots$$

By associativity, we would have

$$h(\underbrace{w_1 w_1 w_1 \dots}_{\notin L}) = h(\underbrace{w_1 w_2 w_3 \dots}_{\in L}).$$

Therefore the syntactic homomorphism does not exist. \square

As we will see later in this section, the monad of \circ -words from the above example is not finitary.

4.3.1 Terms, polynomials and congruences

To prove the existence of syntactic homomorphisms, we will use classical concepts from universal algebra such as a terms, polynomials and congruences. We begin by defining these notions.

Terms and polynomials. If X is a set, then a *term over variables* X is defined to be any element of TX . Given an algebra A , a term is interpreted as the following operation

$$\eta \in A^X \quad \mapsto \quad \text{multiply } (\mathsf{T}\eta)(t) \text{ in } A.$$

We write t^A for the above operation; and we use the name *variable valuation* for η . An operation of this form is called a *term operation* in the algebra A . We distinguish between a term (which can be viewed as syntax) and the term operation that it generates in a given algebra (which can be viewed as semantics). If a term uses a finite set of variables $\{x_1, \dots, x_n\}$, then for $a_1, \dots, a_n \in A$ we write

$$t^A(a_1, \dots, a_n)$$

for the result of applying t^A to the variable valuation $x_i \mapsto a_i$.

Example 25. Consider the monad of finite words. The word xy is a term, and the term operation induced it in an algebra (which is the same thing as a monoid) is binary multiplication. The operation induced by the term ε , which has an empty set of variables, is the constant that represents the monoid identity. Another example of a term operation is squaring, which is given by the term xx . A non-example is the idempotent power operation $a \mapsto a^!$. This is not a term operation, because the number $!$ depends on the algebra at hand (also, this number does not exist in some infinite algebras).

In the monad of \circ -chains, the Lauchli-Leonard operations x^ω and $x^{\omega*}$ are also term operations. To model the remaining Lauchli-Leonard operation, namely shuffling, we use an infinite family of terms, with the n -th one being the shuffle $\{x_1, \dots, x_n\}$. \square

A *polynomial* in an algebra is like a term except that some variables are fixed elements of the algebra. More formally, a *polynomial with variables* X in an algebra A is defined to be any element $t \in \mathsf{T}(A + X)$. The semantics of such a polynomial is the operation

$$\eta \in A^X \quad \mapsto \quad t^A(\underbrace{\text{id}_A + \eta}_{\eta \text{ extended by the identity on } A}).$$

Such an operation is called a *polynomial operation* in A . We will be mainly interested in unary polynomial operations, which are functions of type $A \rightarrow A$ that arise from polynomials with one variable.

Congruences. Consider an equivalence relation \sim on the underlying set in an algebra A . It is not hard to see that the following conditions are equivalent:

- \sim is the kernel of some homomorphism from A to some algebra B ;
- the function which maps $a \in A$ to its equivalence class is compositional;
- \sim is compatible with every term operation $f : A^X \rightarrow A$, which means:

$$\underbrace{\eta_1 \sim \eta_2}_{\substack{\eta_1(x) \sim \eta_2(x) \\ \text{for every } x \in X}} \Rightarrow f(\eta_1) \sim f(\eta_2) \quad \text{for every } \eta_1, \eta_2 \in A^X.$$

Equivalence of the first two conditions is Lemma 4.7, and equivalence of the second two conditions follows by unravelling the definition of compositionality. We say that \sim is *congruence* if it satisfies any of the above conditions. Every congruence induces a quotient algebra, where the universe is equivalence classes.

Contextual equivalence. To construct the syntactic homomorphism, we will use contextual equivalence, which is defined similarly as in the usual proof of the Myhill-Nerode Theorem, see e.g. Theorem 1.7. The only difference is that instead of two-sided contexts (as in monoids and semigroups), we use unary polynomial operations.

Definition 4.10 (Contextual equivalence). Define *contextual equivalence* of an algebra colouring $\lambda : A \rightarrow U$ to be the equivalence relation on A , which identifies $a, a' \in A$ if they have the same image under $\lambda \circ f$ for every unary polynomial operation $f : A \rightarrow A$.

The following example shows that, in general, contextual equivalence is not a congruence.

Example 26. Consider the language L from Example 24. We show that contextual equivalence for this language (viewed as a colouring with values “yes” and “no”) is not a congruence. A unary polynomial operation in $\{a\}^\circ$ corresponds to a \circ -word over alphabet $\{a, x\}$. From the point of view of L , there are two kinds of unary polynomial operations. If f is a unary polynomial such that the corresponding \circ -word over $\{a, x\}$ contains a^n as an infix for every n , then $f(w) \in L$ for every $w \in \{a\}^\circ$. Otherwise, if the corresponding \circ -word does not contain a^n as an infix for some n , then $f(w) \in L \Leftrightarrow w \in L$. These observations imply that contextual equivalence for L has two equivalence classes: namely L and its complement. In particular, contextual equivalence is not a congruence, since otherwise L would be recognisable. \square

4.3.2 Syntactic homomorphisms for recognisable colourings

We now state the first result about syntactic homomorphisms, which says that they always exist for algebra colourings that are recognisable.

Theorem 4.11. *Let \mathbb{T} be a monad in the category of sets. Every recognisable algebra colouring has a syntactic homomorphism.*

If \sim is an equivalence relation on the underlying set of an algebra A , then we say that \sim recognises an algebra colouring $\lambda : A \rightarrow U$ if all equivalence classes are monochromatic, i.e.

$$a_1 \sim a_2 \quad \text{implies} \quad \lambda(a_1) = \lambda(a_2).$$

An algebra colouring is recognisable if and only if there is an equivalence relation that recognises it, is a congruence, and has finitely many equivalence classes.

We say that an equivalence relation \approx refines an equivalence relation \sim if $a \approx b$ implies $a \sim b$. If we view equivalence relations as sets of pairs, this is the same as set inclusion. The following lemma shows that contextual equivalence refines every recognising congruence. (Note that the lemma does not say that contextual equivalence is a congruence. This will be proved later, using the assumption on recognisable.)

Lemma 4.12. *For every algebra colouring λ , contextual equivalence of λ is an equivalence relation that refines every congruence that recognises λ .*

Proof Let f be a unary polynomial operation in the underlying algebra. Let \approx be a congruence that recognises λ . As a congruence, \approx is compatible with every term operation, and therefore it is compatible with every unary polynomial operation (see Exercise 124). This means that if $a \approx b$, then also $f(a) \approx f(b)$, and therefore $f(a)$ has the same colour under λ as $f(b)$, by assumption that \approx is a recognising congruence. By arbitrary choice of f , we have established that $a \approx b$ implies $a \sim b$. \square

We now use the assumption on recognisability to prove that contextual equivalence is a congruence.

Lemma 4.13. *If an algebra colouring λ is recognisable, then its contextual equivalence is a congruence.*

Proof Let $\lambda : A \rightarrow U$ be an algebra colouring, and let \sim be its contextual equivalence relation. We will show that \sim is a congruence in three steps.

- (1) In the first step, we show that \sim is compatible with all unary polynomial operations. The key observation is that unary polynomial operations are closed

under composition. Indeed, consider two unary polynomial operations

$$A \xrightarrow{f} A \xrightarrow{g} A.$$

Take the unary polynomial in $\mathsf{T}(A + \{x\})$ that defines f , and for the variable x substitute the unary polynomial in $\mathsf{T}(A + \{x\})$ that defines g . The resulting unary polynomial defines the composition $g \circ f$.

Equipped with this observation, we prove that \sim is compatible with all unary polynomial operations. Let f be a unary polynomial operation. We need to show

$$a_1 \sim a_2 \quad \text{implies} \quad f(a_1) \sim f(a_2).$$

The assumption of the implication says that a_1, a_2 have the same image under all functions of the form $\lambda \circ g$, where g is a unary polynomial operation, while the conclusion of the implication says that a_1, a_2 have the same image under all functions of the form $\lambda \circ g \circ f$, where g is a unary polynomial operation. Because $g \circ f$ is a unary polynomial operation, the implication follows.

- (2) We now show that \sim is compatible with all term operations that have finitely many variables. Consider an n -ary term operation $f : A^n \rightarrow A$. We need to show

$$\bigwedge_{i \in \{1, \dots, n\}} a_i \sim b_i \quad \text{implies} \quad f(a_1, \dots, a_n) \sim f(b_1, \dots, b_n). \quad (4.8)$$

Assume the assumption of the above implication. For $i \in \{1, \dots, n\}$ consider the following unary polynomial operation

$$f_i(x) = f(a_1, \dots, a_{i-1}, x, b_{i+1}, \dots, b_n).$$

Because $a_i \sim b_i$ and \sim is compatible with all unary polynomial operations,

$$f(a_1, \dots, a_{i-1}, b_i, \dots, b_n) = f_i(a_i) \sim f_i(b_i) = f(a_1, \dots, a_i, b_{i+1}, \dots, b_n).$$

By iterating the above for $i = 1, \dots, n$, we get the conclusion of (4.8).

- (3) Finally, we show that \sim is compatible with all term operations, possibly with infinitely many variables. In this step, we use the assumption that λ is recognisable. Consider a term $t \in \mathsf{TX}$. We need to show that every valuations $\eta_1, \eta_2 \in A^X$ satisfy

$$\eta_1 \sim \eta_2 \quad \text{implies} \quad t^A(\eta_1) \sim t^A(\eta_2).$$

Assume the assumption of the above implication. Because λ is recognisable, there is some congruence \approx on A which recognises λ and has finitely many equivalence classes. Consider a choice function $\tau : A \rightarrow A$ which maps each

\approx -equivalence class in A to a chosen element in that equivalence class. Let $i \in \{1, 2\}$. Because the choice function respects \approx -equivalence, we have

$$\eta_i \approx \tau \circ \eta_i. \quad (4.9)$$

Since \approx is a congruence,

$$t^A(\eta_i) \approx t^A(\tau \circ \eta_i). \quad (4.10)$$

Since each of the valuations $\tau \circ \eta_i$ for $i = 1, 2$ has finite image, we can find a finite set of variables Y (think of pairs of equivalence classes of \approx) and functions δ and γ_i which make the following diagram commute:

$$\begin{array}{ccc} & X & \\ \tau \circ \eta_1 \swarrow & \downarrow \sigma & \searrow \tau \circ \eta_2 \\ A & Y & A \\ \gamma_1 \longleftarrow & & \longrightarrow \gamma_2 \end{array}$$

Define $s \in TY$ to be the term that results from t by the substitution $\top\sigma$. By definition,

$$s^A(\gamma_i) = t^A(\gamma_i \circ \sigma) = t^A(\tau \circ \eta_i) \stackrel{(4.10)}{\approx} t^A(\eta_i).$$

By the first step of the proof, \approx refines \sim . Therefore, to finish the proof, it remains to show

$$s^A(\gamma_1) \sim s^A(\gamma_2).$$

Since s is a term with finitely many variables, and we have already established that \sim is compatible with terms that have finitely many variables, it is enough to show $\gamma_1 \sim \gamma_2$. This is because

$$\tau \circ \eta_1 \sim \eta_1 \sim \eta_2 \sim \tau \circ \eta_2.$$

In the above, the first and last equivalence are due to (4.9) and the fact that \approx refines \sim . Since the first valuation in the above is equal to $\gamma_1 \circ \sigma$, and the last one is equal to $\gamma_2 \circ \sigma$, it follows that $\gamma_1 \sim \gamma_2$.

We have proved that \sim is a congruence. \square

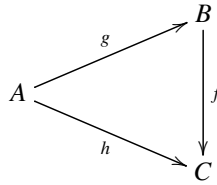
We now complete the proof of Theorem 4.11. Let \sim be contextual equivalence of a recognisable algebra colouring $\lambda : A \rightarrow U$. By Lemma 4.13, \sim is a congruence. Because \sim is a congruence, the quotient function, call it h , is a homomorphism. Since the identity function is a unary polynomial operation, it follows that equivalence classes of \sim are monochromatic, and therefore h recognises λ .

It remains to show that every other recognising homomorphism factors through h . Suppose that

$$g : A \rightarrow B$$

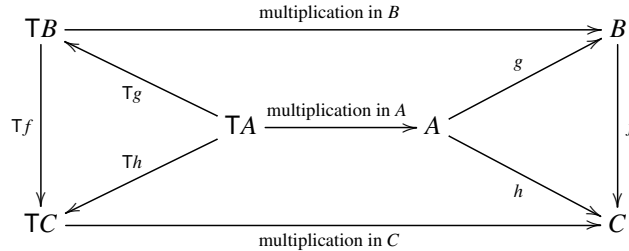
is a surjective homomorphism that recognises λ . The kernel of g is a congruence that recognises λ , and therefore the kernel of g refines \sim . In other words, g factors through h , i.e. there is some function f such that $g = f \circ h$. The last thing to show is that f is in fact a homomorphism. This is shown in the following lemma, with C being the quotient A/\sim .

Lemma 4.14. *Let A, B, C be algebras, let g, h be surjective homomorphisms, and let f be a function which makes the following diagram commute.*



Then f is a homomorphism.

Proof Consider the following diagram:



The upper and lower faces commute because g and h are homomorphisms, and the left and right faces commute by definition of f . Since all arrows in the diagram are surjective, it follows that the perimeter of the diagram commutes, which means that f is a homomorphism. \square

4.3.3 Finitary monads

In the monad of finite words, every language – even not necessarily recognisable – has a syntactic homomorphism. In the monad of \circ -words, this is no longer true, as witnessed by Example 24. What is the difference?

The difference is that every finite word uses only a finite subset of the alphabet, which is no longer true for \circ -words. This is made precise by the following definition.

Definition 4.15 (Finitary elements and monads.). Let \mathbb{T} be a monad in the category of sets. We say that an element $t \in \mathbb{T}X$ is *finitary*

$$t = (\mathbb{T}f)(t) \quad \text{for some } f : X \rightarrow X \text{ with finite image.}$$

We say that \mathbb{T} is finitary if for every X , all elements of $\mathbb{T}X$ are finitary.

For example, the monad of finite words is finitary, while the monads of \circ -words is not. The following theorem shows that finitary monads are exactly those monads where all algebra colourings have syntactic homomorphisms.

Theorem 4.16. *Let \mathbb{T} be a monad in the category of sets. Then \mathbb{T} is finitary if and only if every algebra colouring has a syntactic homomorphism.*

Proof We begin with the left-to-right implication. Suppose that \mathbb{T} is finitary, and let $\lambda : A \rightarrow U$ be some algebra colouring, not necessarily recognisable. We use the same proof as in Theorem 4.11. The only place where the proof used the assumption on recognisability was in step (3) of Lemma 4.13, where contextual equivalence was shown to be compatible with all term operations. If the monad is finitary, then every term operation uses finitely many variables, and therefore step (3) of the proof follows immediately from step (2), without any need for invoking recognisability.

We now prove the converse implication. Fix some infinite set X . We will show that all elements of X are finitary. Let \mathbf{X} be a disjoint copy of X . For a finite subset $Y \subseteq \mathbf{X}$, define

$$f_Y : X + \mathbf{X} \rightarrow X + \mathbf{X}$$

to be the function which maps all elements of Y to their corresponding black copies, and which is the identity on the remaining elements. Define \sim_Y to be the equivalence relation on $\mathbb{T}(X + \mathbf{X})$ which is the kernel of the homomorphism

$$\mathbb{T}f_Y : \mathbb{T}(X + \mathbf{X}) \rightarrow \mathbb{T}(X + \mathbf{X})$$

The mapping $Y \mapsto \sim_Y$ is monotone with respect to inclusion:

$$Y \subseteq Z \quad \text{implies} \quad \sim_Y \subseteq \sim_Z .$$

In the conclusion of the above implication, we view equivalence relations as sets of pairs; under this view smaller equivalence relations refine bigger ones. Viewing equivalence relations as sets of pairs, define \sim to be the union of the

equivalences \sim_Y , ranging over all finite subsets $Y \subseteq X$. This is an equivalence relation; transitivity follows from the monotonicity property above.

Consider the two natural injections

$$X + X \xleftarrow{f} X \xrightarrow{f} X + X,$$

such that f maps all elements to their black copies, and f maps all elements to their red copies. By assumption on the monad, there is a syntactic homomorphism for the quotient function of \sim , call it h . For every $x \in X$, the units of x and x are equivalent under the congruence $\sim_{\{x\}}$. Since the latter is a congruence that recognises \sim , it follows that the units of x and x must have the same image under h . By arbitrary choice of x , we see that

$$h \circ \mathbb{T}f = h \circ \mathbb{T}f. \quad (4.11)$$

We now prove that every $t \in TX$ is finitary. Fix t . Since h recognises \sim , the equality (4.11) implies that

$$(\mathbb{T}f)(t) \sim (\mathbb{T}f)(t).$$

By definition of \sim , there must be some finite Y , which depends on t , such that

$$(\mathbb{T}f)(t) \sim_Y (\mathbb{T}f)(t) \quad (4.12)$$

Choose an element of Y and let

$$g : X + X \rightarrow X$$

be the function which is the identity on X and sends all red elements to the chosen element of Y . We have

$$\begin{aligned} t &= && \text{(because } g \circ f \text{ is the identity on } X) \\ (\mathbb{T}(g \circ f))(t) &= && \text{(because } f_Y \circ f = f) \\ (\mathbb{T}(g \circ f_Y \circ f))(t) &= && \text{((4.12))} \\ (\mathbb{T}(g \circ f_Y \circ f))(t) &= && \end{aligned}$$

The image of the function $g \circ f_Y \circ f$ is contained in Y , and therefore we have established that t is finitary. \square

Exercises

Exercise 124. Fix a monad. Let A be an algebra, and let $p : A^X \rightarrow A$ be a polynomial operation. Show that for every homomorphism $h : A \rightarrow B$ there is a polynomial operation $p^h : B^X \rightarrow B$ which commutes with the homomorphism in the following sense:

$$\begin{array}{ccc} A^X & \xrightarrow{p} & A \\ h^X \downarrow & & \downarrow h \\ B^X & \xrightarrow{p^h} & B \end{array}$$

Exercise 125. Consider an algebra colouring $\lambda : A \rightarrow U$. Consider the family of congruences on A that recognise U , ordered by inclusion. Show that this family is a lattice, i.e. every two elements have a greatest lower bound and also a least upper bound.

Exercise 126. Consider the monad from Example 23. Show that if $\lambda : T\Sigma \rightarrow \mathbb{Q}$ is recognised by a weighted automaton, then the same is true for the syntactic homomorphism.

Exercise 127. Let T be a monad in the category of sets. Show that if every algebra colouring with two colours has a syntactic homomorphism, then every algebra colouring with an arbitrary number of colours has a syntactic homomorphism.

Exercise 128. Give an example of a monad T which is not finitary, but such that all elements of TX are finitary for countable X .

Exercise 129. Let S be a finite set of sort names, and consider the category

$$\text{Set}^S$$

of S -sorted sets with sort-preserving functions. Prove Theorem 4.11 for monads over this category.

Exercise 130. Consider a category of sorted sets, as in the previous exercise, but with infinitely many sort names. Define a finite algebra to be one that is finite on every sort. Show that Theorem 4.11 fails.

Exercise 131. Show that a monad in the category of sets is finitary if and only if it arises as a result of the construction described in Exercise 111.

Exercise 132. Let T be a monad in the category of sets, and consider a set of terms \mathcal{B} , each one using finitely many variables. We say that \mathcal{B} is a *term basis* if for every finite algebra A and subset $\Gamma \subseteq A$, the sub-algebra generated by Γ is equal to the least subset of A that contains Γ and which is closed under applying terms from \mathcal{B} . Show that if \mathcal{B} is a term basis, then a finite algebra A is uniquely determined by its \mathcal{B} -multiplication tables, which is the family of term operations $\{t^A\}_{t \in \mathcal{B}}$.

Exercise 133. Let T be a monad which has a finite term basis \mathcal{B} . Show that given the \mathcal{B} -multiplication tables in an algebra A , one can compute the \mathcal{B} -multiplication tables of the powerset algebra PA (as defined in Exercise 107, assuming that PA is indeed an algebra).

Exercise 134. Find a notion of *computable term basis* which generalises the previous exercise so as to capture the Lauchli-Leonard operations in the monad of \circ -words.

Exercise 135. Recall the notion of *regular elements* from Exercise 113. Show that if t is regular, then it is finitary.

Exercise 136. Assume that the regular elements, as considered in the previous exercise, are closed under free multiplication in the following sense: if $t \in TX$ is a regular term operation, and $\eta : X \rightarrow TY$ is a valuation of its variables that uses only regular elements, then $t^{TY}(\eta)$ is a regular element. Under these assumptions, define a monad of regular elements.

4.4 The Eilenberg Pseudo-Variety Theorem

In Chapter 2, we proved several theorems of the kind

$$\text{class of languages} \quad \sim \quad \text{class of semigroups.}$$

For example, a language of finite words is definable in first-order logic if and only if it is recognised by an aperiodic semigroup. In this section, we prove a result, originally due to Eilenberg, which says that every class of languages with good closure properties will correspond to a class of algebras with good

closure properties. Eilenberg proved the theorem for monoids and semigroups, but the proof, as presented below, is the same in the abstract setting of monads.

The classes with good closure properties will be called pseudo-varieties¹². There will be two kinds of pseudo-varieties: for algebras and for languages. We begin with the algebras.

Definition 4.17 (Algebra pseudo-variety). Let T be a monad in the category of sets. An *algebra pseudo-variety* is a class \mathcal{A} of finite T -algebras with the following closure properties:

- *Quotients*. If \mathcal{A} contains A , then it contains every quotient of A .
- *Sub-algebras*. If \mathcal{A} contains A , then it contains every sub-algebra of A .
- *Products*. If \mathcal{A} contains A and B , then it contains $A \times B$.

Example 27. Consider the monad of finite words, where algebras are monoids. Examples algebra pseudo-varieties include: finite groups, finite aperiodic monoids, finite infix trivial monoids, or finite prefix trivial monoids. \square

Example 28. Here is a non-example. Consider the monad of nonempty finite words, where algebras are semigroups. The class of monoids (i.e. semigroups which have an identity element) is not an algebra pseudo-variety, because it is not closed under sub-algebras. \square

Example 29. Consider a monad T . Define an *identity* to be a pair of terms $s, t \in TX$ over a common set of variables. An algebra A is said to satisfy the identity if

$$s^A(\eta) = t^A(\eta) \quad \text{for every } \eta \in A^X.$$

The class of finite algebras that satisfy a given identity (more generally, all identities in a given set of identities) is easily seen to be an algebra pseudo-variety. For example, the algebra pseudo-variety of commutative semigroups arises from the identity

$$xy = yx.$$

Unlike in Birkhoff's theorem, some algebra pseudo-varieties do not arise this

¹² The name is supposed to be similar to the varieties that appear in Birkhoff's theorem from universal algebra; the "pseudo" in the name signifies that we are talking about classes of finite objects.

way. For example, pseudo-varieties discussed in Example 27 do not arise from (even possibly infinite sets of) identities¹³ \square

We now describe language pseudo-varieties. In Eilenberg's original formulations, this is a class of regular languages that is closed under Boolean combinations, inverse images of homomorphisms, and inverse images of operations of the form

$$w \in \Sigma^+ \mapsto v_1 w v_2 \in \Sigma^+ \quad \text{for fixed } v_1, v_2 \in \Sigma^*.$$

In the more abstract setting of monads, the role of these operations will be played by unary polynomial operations, as described in the following definition.

In the following definition, by recognisable languages we mean recognisable subsets of free algebras.

Definition 4.18 (Language pseudo-variety). Let T be a monad in the category of sets. A *language pseudo-variety* is a class of recognisable languages with the following closure properties:

- *Boolean combinations.* \mathcal{L} is closed under Boolean combinations, including complementation.
- *Inverses of homomorphisms.* If $h : T\Sigma \rightarrow T\Gamma$ is a homomorphism of free algebras, then \mathcal{L} is closed under inverse images of h .
- *Inverses of unary polynomials.* If $f : T\Sigma \rightarrow T\Sigma$ is a unary polynomial operation in a free algebra $T\Sigma$, then \mathcal{L} is closed under inverse images of f .

Example 30. Consider the monad of finite words, where algebras are monoids. The languages definable in first-order logic are a language pseudo-variety. Closure under Boolean combinations is immediate, because we are dealing with a logic. Closure under inverse images of homomorphism or unary polynomials can be proved using Ehrenfeucht-Fraïssé games: if f is either a homomorphism or a unary polynomial operation, then a strategy copying argument shows

$$\begin{array}{ccc} \text{Duplicator wins the} & \text{implies} & \text{Duplicator wins the} \\ k \text{ round game on } w \text{ and } w' & & k \text{ round game on } f(w) \text{ and } f(w'). \end{array}$$

¹³ However, if we use the more general form of *profinite identities*, then every pseudo-variety can be axiomatised equationally.

[25] Reiterman, "The Birkhoff theorem for finite algebras", 1982, Theorem 3.1.

Another answer is given in

[11] Eilenberg and Schützenberger, *On pseudovarieties*, 1975, Theorem 1,

where it is shown that every pseudo-variety can be axiomatised as follows: one gives an infinite set of identities, but requires only that all but finitely many identities are satisfied.

This implies that first-order definable languages are closed under inverse images of homomorphisms and unary polynomial operations. The same is true for first-order logic on \circ -words. \square

Example 31. Consider again the monad of finite words, where algebras are monoids. The definite languages from Example 19 are not a pseudo-variety, because it is not closed under inverse images of the homomorphisms. Indeed, the language

$$a\{a, b\}^* \subseteq \{a, b\}^*$$

is definite. If we take the inverse image under the homomorphism

$$h : \{a, b, c\}^* \rightarrow \{a, b\}^*,$$

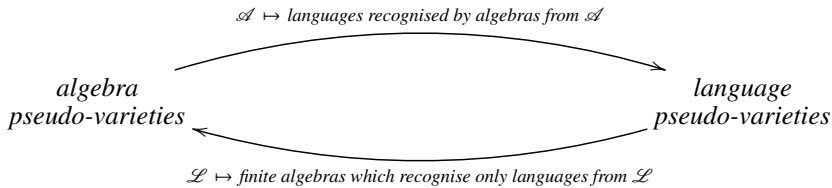
which erases the c letters, then we get the language

$$c^*a\{a, b, c\}^* \subseteq \{a, b, c\}^*,$$

which is not definite. The problem is with homomorphism that erase letters. If we would consider the same class of languages but in the monad of nonempty finite words, where algebras are semigroups, then we would get a pseudo-variety. \square

The Eilenberg Pseudo-Variety Theorem says that the two notions of pseudo-variety are two sides of the same coin.

Theorem 4.19 (Eilenberg Pseudo-Variety Theorem). *Let T be a monad in the category of sets. The following maps are mutually inverse bijections*



Proof Let us write L for the left-to-right map, and A for the right-to-left map. We first show that each of these two maps take pseudo-varieties to pseudo-varieties, and then we show that the two maps are mutual inverses.

- (1) We first show that if \mathcal{A} is a class of finite algebras closed under products (which covers the special case of algebra pseudo-varieties), then $L\mathcal{A}$ is a language pseudo-variety. We begin with Boolean combinations. If L is recognised by an algebra $A \in \mathcal{A}$, then its complement is recognised by the same algebra. If furthermore K is recognised by $B \in \mathcal{A}$, then then $L \cup K$ and

$L \cap K$ are both recognised by the product $A \times B$, which belongs to \mathcal{A} by closure under products. Consider now the inverse images. Let L be recognised by a homomorphism

$$h : T\Sigma \rightarrow A \in \mathcal{A}.$$

If $g : T\Sigma \rightarrow T\Gamma$ is a homomorphism, then the inverse image $g^{-1}(L)$ is recognised by the composition $h \circ g$. Consider now a unary polynomial operation $f : T\Sigma \rightarrow T\Sigma$. As we have remarked in the proof of Lemma 4.12, congruences are compatible with unary polynomial operation, which means that

$$h(w_1) = h(w_2) \quad \text{implies} \quad h(f(w_1)) = h(f(w_2)).$$

This, in turn, means that h also recognises the inverse image $h^{-1}(L)$.

- (2) Next we show that if \mathcal{L} is a class of recognisable languages closed under unions and intersections (which covers the case of language pseudo-varieties), then $\mathbf{A}\mathcal{L}$ is an algebra pseudo-variety¹⁴. Every language recognised by a sub-algebra of A is also recognised by A , and the same is true for quotients, and therefore $\mathbf{A}\mathcal{L}$ is closed under sub-algebras and quotients of A . Consider now products. Suppose that a language L is recognised by a homomorphism

$$h : T\Sigma \rightarrow A \times B \quad \text{with } A, B \in \mathbf{A}\mathcal{L}.$$

For every $a \in A$, the inverse image

$$L_a = h^{-1}(\{a\} \times B)$$

is recognised by the homomorphic

$$h_A : T\Sigma \rightarrow A,$$

which is the composition of h with the projection to A . Since the latter homomorphism has domain A , it follows that $L_a \in \mathcal{L}$. For similar reasons, if $b \in B$ then \mathcal{L} contains the language

$$L_b = h^{-1}(A \times \{b\}).$$

The intersection $L_a \cap L_b$ is the inverse image under h of the pair (a, b) . Every language recognised by h is a finite union of such languages; and therefore it belongs to \mathcal{L} by closure under unions and intersections.

¹⁴ The first two steps of this proof establish that the maps \mathbf{L} and \mathbf{A} form what is known as a Galois connection, between

- classes of finite algebras closed under products; and
- classes of recognisable languages closed under unions and intersections.

In the terminology of Galois connections, the pseudo-varieties are the closed sets, with respect to this Galois connection.

- (3) We now show that the maps \mathbf{A} and \mathbf{L} are mutual inverses. We first show that every algebra pseudo-variety \mathcal{A} satisfies

$$\mathcal{A} = \mathbf{A}\mathbf{L}\mathcal{A}.$$

This is the same as showing that $A \in \mathcal{A}$ if and only if

- (*) every language recognised by A is recognised by some algebra in \mathcal{A} .

Clearly every algebra $A \in \mathcal{A}$ satisfies (*). We now prove the converse implication. Suppose that an algebra A satisfies (*). The multiplication operation

$$\mu : \mathbb{T}A \rightarrow A$$

in the algebra A is a homomorphism from the free algebra $\mathbb{T}A$ to A . By the assumption that A satisfies (*), every language recognised by this homomorphism is recognised by some algebra from \mathcal{A} . In particular, for every $a \in A$ the language $\mu^{-1}(a)$ is recognised by some homomorphism

$$h_a : \mathbb{T}A \rightarrow B_a \in \mathcal{A}.$$

Consider the product homomorphism

$$h : \mathbb{T}A \rightarrow \prod_{a \in A} B_a \quad t \mapsto (h_a(t))_{a \in A}$$

Define B to be the image of h . This is a sub-algebra of the a product of algebras from \mathcal{A} , and therefore it belongs to \mathcal{A} . From now on we view h as surjective homomorphism onto its image B . This homomorphism recognises all languages $\mu^{-1}(a)$, and therefore μ factors through h , i.e. there is some function f which makes the following diagram commute:

$$\begin{array}{ccc} \mathbb{T}A & \xrightarrow{\mu} & A \\ & \searrow h & \uparrow f \\ & & B \end{array}$$

By Lemma 4.14, f is not just a function but also a homomorphism of algebras. This means that A is the image of B under a surjective homomorphism. In other words, A is a quotient of B , and therefore $A \in \mathcal{A}$.

- (4) In the final step, we show that every language pseudo-variety \mathcal{L} satisfies

$$\mathcal{L} = \mathbf{L}\mathbf{A}\mathcal{L}.$$

This is the same as showing that $L \in \mathcal{L}$ if and only if

- (*) L is recognised by an algebra that only recognises languages from \mathcal{L} .

Clearly (*) implies $L \in \mathcal{L}$, so we focus on the converse implication. Suppose that $L \in \mathcal{L}$. Since L is recognisable, it has a syntactic homomorphism

$$h : T\Sigma \rightarrow A$$

thanks to Theorem 4.11. To prove (*), we will show that all languages recognised by the syntactic algebra A belong to \mathcal{L} .

By the proof of Theorem 4.11, the kernel of the syntactic homomorphism h is the same as contextual equivalence for L . Therefore, by definition of contextual equivalence, we know that every $v, w \in T\Sigma$ satisfy

$$h(v) = h(w) \quad \text{iff} \quad \underbrace{p(w) \in L \Leftrightarrow p(v) \in L}_{\substack{\text{for every unary polynomial operation} \\ p : T\Sigma \rightarrow T\Sigma}}$$

Unary polynomial operations are compatible with congruences, which means that for every unary polynomial operation p there some function $p^A : A \rightarrow A$ which makes the following diagram commute

$$\begin{array}{ccc} T\Sigma & \xrightarrow{h} & A \\ p \downarrow & & \downarrow p^A \\ T\Sigma & \xrightarrow{h} & A \end{array}$$

Since p^A can be chosen in finitely many ways, and $p^{-1}(L)$ depends only on p^A , it follows that there is a finite set \mathcal{P} of unary polynomial operations in $T\Sigma$ such that

$$h(v) = h(w) \quad \text{iff} \quad \underbrace{p(v) \in L \Leftrightarrow p(w) \in L}_{\text{for every unary polynomial operation } p \in \mathcal{P}} \tag{4.13}$$

We now show that \mathcal{L} contains every language recognised by A . Let then

$$g : T\Gamma \rightarrow A$$

be some homomorphism. By surjectivity of the syntactic homomorphism and the universal property of the free algebra $T\Gamma$, we can choose some homomorphism f which makes the following diagram commute

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

We will show that for every $a \in A$, the language $g^{-1}(a)$ belongs to in \mathcal{L} .

Since \mathcal{L} is closed under finite unions, it will follow that every language recognised by g belongs to \mathcal{L} , thus completing the proof of (*) for L .

Fix some $a \in A$. Since the syntactic homomorphism is surjective, one can choose some $v \in T\Sigma$ such that $h(v) = a$. By (4.13), every $w \in T\Gamma$ satisfies

$$g(w) = h(f(w)) = a = h(v) \quad \text{iff} \quad \underbrace{p(f(w)) \in L \Leftrightarrow p(v) \in L}_{\text{for every unary polynomial operation } p \in \mathcal{P}}.$$

If v is fixed, then the right side of the above is a finite Boolean combination of constraints of the form

$$p(f(w)) \in L \quad \text{which is the same as} \quad w \in \underbrace{f^{-1}(p^{-1}(L))}_{\text{a language in } \mathcal{L}}.$$

Summing up, we can express the constraint $g(w) = a$ as a finite Boolean combination of constraints $w \in K$, where K is some language in \mathcal{L} . Therefore, $g^{-1}(a)$ belongs to \mathcal{L} .

□

Exercises

Exercise 137. Give an example of a monad which is not finitary, but where every language $L \subseteq T\Sigma$ with a finite alphabet Σ has a syntactic homomorphism.

Exercise 138. Consider the monad of finite words. Show that a class of languages \mathcal{L} is a pseudo-variety if and only if it is closed under Boolean combinations, inverse images under homomorphisms, and inverse images of unary polynomial operations of the form:

$$w \mapsto v_1 w v_2 \quad \text{for every choice of parameters } v_1, v_2 \in \Sigma^*.$$

Exercise 139. Consider the monad of finite words. Show that there are uncountably many algebra pseudo-varieties. In particular, for some algebra pseudo-varieties, the membership problem $A \stackrel{?}{\in} \mathcal{A}$ is undecidable.

Exercise 140. Consider the monad of \circ -words. Show that a class of languages \mathcal{L} is a pseudo-variety if and only if it is closed under Boolean combinations,

inverse images under homomorphisms, and inverse images of unary polynomial operations of the following forms:

$$\begin{aligned} w &\mapsto v_1 w v_2 && \text{for every choice of parameters } v_1, v_2 \in \Sigma^\circ \\ w &\mapsto w^\omega \\ w &\mapsto w^{\omega*} \end{aligned}$$

$$w \mapsto \text{shuffle of } \{w, v_1, \dots, v_n\} \quad \text{for every choice of parameters } v_1, \text{dots}, v_n \in \Sigma^\circ$$

Exercise 141. Let S be a finite set, and consider the category

$$\text{Set}^S$$

of S -sorted sets with sort-preserving functions. State and prove the Eilenberg Pseudo-Variety Theorem for monads over this category.

Exercise 142. Consider the monad from Example 23, which corresponds to weighted automata. We adapt to pseudo-varieties to the weighted setting as follows. Define an algebra pseudo-variety to be class of finite-dimensional algebras which is closed under sub-algebras, quotients and products. Define a language pseudo-variety to be a class \mathcal{L} of linear maps $\mathbb{T}\Sigma \rightarrow \mathbb{Q}$, recognised by finite-dimensional algebras, which is closed under inverse images of homomorphisms and polynomials, and which is closed under combinations in the following sense: if \mathcal{L} contains

$$\{\lambda_i : \mathbb{T}\Sigma \rightarrow U_i\}_{i \in \{1,2\}},$$

and $f : \mathbb{Q}^2 \rightarrow \mathbb{Q}$ is a linear map, then \mathcal{L} contains also

$$w \mapsto f(\lambda_1(w), \lambda_2(w)).$$

Show that the Eilenberg Pseudo-Variety Theorem holds for pseudo-varieties understood in this way.

Exercise 143. Consider the weighted pseudo-varieties from the previous example. What is the weighted analogue of star-free languages? Hint: consider the concatenation of two linear maps

$$\lambda_1, \lambda_2 : \mathbb{T}\Sigma \rightarrow U$$

to be the linear map which is defined as follows on Σ^*

$$(\lambda_1 \cdot \lambda_2)(a_1 \cdots a_n) = \sum_{i \in \{0, \dots, n\}} \lambda_1(a_1 \cdots a_i) \cdot \lambda_2(a_{i+1} \cdots a_n),$$

and which is extended to $\mathbb{T}\Sigma$ by linearity.

Bibliography

- [1] H Appelgate et al. *Seminar on triples and categorical homology theory*. Springer, 1969.
- [2] Mustapha Arfi. “Polynomial Operations on Rational Languages”. In: *Symposium on Theoretical Aspects of Computer Science, STACS, Passau, Germany*. 1987, pp. 198–206.
- [3] Stephen L. Bloom and Zoltán Ésik. “The equational theory of regular words”. In: *Information and Computation* 197.1 (2005), pp. 55–89.
- [4] Achim Blumensath. “Regular Tree Algebras”. In: *CoRR* abs/1808.03559 (2018).
- [5] Mikołaj Bojańczyk and Bartek Klin. “A non-regular language of infinite trees that is recognizable by a sort-wise finite algebra”. In: *Logical Methods in Computer Science* 15.4 (2019).
- [6] J. Richard Büchi. “On a decision method in restricted second order arithmetic”. In: *Logic, Methodology and Philosophy of Science (Proc. 1960 Internat. Congr .)* Stanford, Calif.: Stanford Univ. Press, 1962, pp. 1–11.
- [7] J. Richard Büchi. “Weak second-order arithmetic and finite automata”. In: *Z. Math. Logik und Grundl. Math.* 6 (1960), pp. 66–92.
- [8] Olivier Carton, Thomas Colcombet, and Gabriele Puppis. “An algebraic approach to MSO-definability on countable linear orders”. In: *The Journal of Symbolic Logic* 83.3 (2018), pp. 1147–1189.
- [9] Thomas Colcombet and A. V. Sreejith. “Limited Set Quantifiers over Countable Linear Orderings”. In: *International Colloquium on Automata, Languages and Programming, ICALP, Kyoto, Japan*. Ed. by Magnús M. Halldórsson et al. Vol. 9135. Lecture Notes in Computer Science. Springer, 2015, pp. 146–158.

- [10] Wojciech Czerwinski et al. “A Characterization for Decidable Separability by Piecewise Testable Languages”. In: *Discret. Math. Theor. Comput. Sci.* 19.4 (2017).
- [11] Samuel Eilenberg and Marcel-Paul Schützenberger. *On pseudovarieties*. IRIA. Laboratoire de Recherche en Informatique et Automatique, 1975.
- [12] Calvin C. Elgot. “Decision problems of finite automata design and related arithmetics”. In: *Trans. Amer. Math. Soc.* 98 (1961), pp. 21–51.
- [13] Ronald Fagin. “Generalized first-order spectra and polynomial-time recognizable sets”. In: *Complexity of computation (Proc. SIAM-AMS Sympos. Appl. Math., New York, 1973)*. Providence, R.I.: Amer. Math. Soc., 1974, 43–73. SIAM–AMS Proc., Vol. VII.
- [14] Gudmund Skovbjerg Frandsen, Peter Bro Miltersen, and Sven Skyum. “Dynamic Word Problems”. In: *J. ACM* 44.2 (Mar. 1997), pp. 257–271.
- [15] J. A. Green and D. Rees. “On semi-groups in which $x^n = x$ ”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 48.1 (1952), pp. 35–40.
- [16] Wilfrid Hodges. *Model Theory*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1993.
- [17] J.A. Kamp. “Tense Logic and the Theory of Linear Order”. PhD thesis. Univ. of California, Los Angeles, 1968.
- [18] Manfred Kufleitner. “The Height of Factorization Forests”. In: *Mathematical Foundations of Computer Science 2008, 33rd International Symposium, MFCS 2008, Torun, Poland, August 25-29, 2008, Proceedings*. Ed. by Edward Ochmanski and Jerzy Tyszkiewicz. Vol. 5162. Lecture Notes in Computer Science. Springer, 2008, pp. 443–454.
- [19] H Läuchli and J Leonard. “On the elementary theory of linear order”. In: *Fundamenta Mathematicae* 59.1 (1966), pp. 109–116.
- [20] Robert McNaughton. “Testing and generating infinite sequences by a finite automaton”. In: *Information and Control* 9 (1966), pp. 521–530.
- [21] Robert McNaughton and Seymour Papert. *Counter-free automata*. The M.I.T. Press, Cambridge, Mass.-London, 1971.
- [22] J.-E. Pin and P. Weil. “Polynomial closure and unambiguous product”. In: *Theory Comput. Syst.* 30.4 (1997), pp. 383–422.
- [23] Thomas Place and Marc Zeitoun. “Going Higher in First-Order Quantifier Alternation Hierarchies on Words”. In: *J. ACM* 66.2 (2019), 12:1–12:65.
- [24] Frank D. Ramsey. “On a problem of formal logic”. In: *Proc. of the London Math. Soc.* 30 (1929), pp. 338–384.
- [25] Jan Reiterman. “The Birkhoff theorem for finite algebras”. In: *Algebra Universalis* 14.1 (1982), pp. 1–10.

- [26] Chloé Rispal and Olivier Carton. “Complementation of Rational Sets on Countable Scattered Linear Orderings”. In: *International Journal of Foundations of Computer Science* 16.04 (Aug. 2005), pp. 767–786.
- [27] Hanamantagouda P Sankappanavar and Stanley Burris. “A course in universal algebra”. In: *Graduate Texts Math* 78 (1981).
- [28] Marcel-Paul Schützenberger. “On finite monoids having only trivial subgroups”. In: *Information and Control* 8 (1965), pp. 190–194.
- [29] Marcel-Paul Schützenberger. “Sur Le Produit De Concatenation Non Ambigu”. In: *Semigroup Forum* 13 (1976), pp. 47–75.
- [30] Saharon Shelah. “The Monadic Theory of Order”. In: *Annals of Mathematics* (1975), pp. 379–419.
- [31] Imre Simon. “Factorization Forests of Finite Height”. In: *Theoretical Computer Science* 72.1 (1990), pp. 65–94.
- [32] Imre Simon. “Piecewise testable events”. In: *Automata Theory and Formal Languages*. Ed. by H. Brakhage. Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 214–222. ISBN: 978-3-540-37923-2.
- [33] J. W. Thatcher and J. B. Wright. “Generalized Finite Automata Theory with an Application to a Decision Problem of Second-Order Logic”. In: *Mathematical systems theory* 2.1 (Mar. 1968), pp. 57–81.
- [34] Boris A. Trakhtenbrot. “The synthesis of logical nets whose operators are described in terms of one-place predicate calculus (Russian)”. In: *Dokl. Akad. Nauk SSSR* 118.4 (1958), pp. 646–649.
- [35] Thomas Wilke. “Classifying Discrete Temporal Properties”. In: *STACS 99*. Ed. by Christoph Meinel and Sophie Tison. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 32–46.

Author index

Subject index