Slightly Infinite Sets

Mikołaj Bojańczyk
April 25, 2025
The latest version can be downloaded from: mimuw.edu.pl/~bojan/paper/atom-book

April 25, 2025

Contents

Preface

This book is about algorithms that run on objects that are infinite, but finite up to certain symmetries. Under a suitably chosen notion of symmetry, such objects – called *orbit-finite sets* – can be represented, searched and processed just like finite sets. The goal of the book is to explain orbit-finiteness and demonstrate its usefulness. Most of the examples of orbit-finite sets are taken from automata theory, since this is where orbit-finite sets began.

Chapter 1

Polynomial orbit-finite sets

The general idea in this book is to discuss sets which are built from some basic infinite set $\mathbb{A}$, and which are simple enough to be represented finitely and manipulated algorithmically. These sets will be called *orbit-finite sets*. The fully general notion will be described in later chapters. We begin in this chapter with a special case, called *polynomial orbit-finite sets*, which is simpler to formalize, but general enough to describe most interesting examples.

In the basic infinite set $\mathbb{A}$, which will be used to build the other sets, the only structure is equality. The assertion "the only structure is equality" will be formalized later in the book, by using invariance under atom permutations. For the moment, this will be apparent in the examples, and our convention that elements of $\mathbb{A}$ – which will be called *atoms* – are names such as John or Eve. Everybody knows that names have no structure beyond equality.

Before formally defining polynomial orbit-finite sets, we begin with several examples. These examples are based on automata theory, which was the original motivation for these notions.

Example 1. Consider the language

$$\{ w \in \mathbb{A}^* \mid \text{the first and last letters of } w \text{ are the same} \}.$$

To recognise this language, we use a deterministic automaton that remembers the first letter seen in its state, plus one extra bit of information that tells us whether the last letter seen is the same as the first. This state space can be viewed as a disjoint union (denoted using +):

$$\{\text{initial}\} \quad + \quad \underbrace{\mathbb{A}}_{\text{equal}} \quad + \quad \underbrace{\mathbb{A}}_{\text{nonequal}}.$$

The transition function consists of the following transitions, where a and b range over $\mathbb{A}$:

$$\begin{aligned}
\text{initial}() &\xrightarrow{a} \text{equal}(a)\\
\text{equal}(a) &\xrightarrow{b} \begin{cases}\text{equal}(a) & \text{if } a = a\\ \text{nonequal}() & \text{if } a \neq a\end{cases}\\
\text{nonequal}(a) &\xrightarrow{b} \begin{cases}\text{equal}(a) & \text{if } a = a\\ \text{nonequal}() & \text{if } a \neq a.\end{cases}
\end{aligned}$$

The accepting states are those from the first copy of $\mathbb{A}$. □

The automaton in the above example was deterministic. Here is an example of a nondeterministic automaton.

Example 2. Consider the language

$$L = \{\, w \in \mathbb{A}^* \mid \text{ some letter appears twice } \}.$$

The recognizing automaton uses nondeterminism to guess the letter that appears twice. It then loads that letter into its state, and waits for a second appearance, upon which it enters an accepting sink state. The state space is

$$\{\text{initial, accept}\} \quad + \quad \mathbb{A}.$$

The first two states are the initial and accepting states, respectively. The transitions of this automaton are listed below, where a and b range over $\mathbb{A}$:

$$\begin{aligned}
\text{initial} &\xrightarrow{a} \text{initial}\\
\text{initial} &\xrightarrow{a} a\\
a &\xrightarrow{b} \begin{cases}\text{accept} & \text{if } a = b\\ a & \text{if } a \neq b\end{cases}\\
\text{accept} &\xrightarrow{a} \text{accept}.
\end{aligned}$$

We will later show that this language cannot be recognised by a deterministic automaton, but that will require a formal definition of the model. □

In the automata from the above examples, the state space could be infinite, but it had a very special form: each state would store some finite information (for example, is it accepting or reject), and some atoms. In the two examples above, each state would store zero or one atom, but one could imagine that more atoms are stored, e.g. we could have a state space of the form

$$\mathbb{A}^0 + \mathbb{A}^0 + \mathbb{A}^1 + \mathbb{A}^7.$$

As before, we write + for disjoint union of sets. In the disjoint union above, the components of the form $\mathbb{A}^0$ represent states where no atoms are stored, such as the initial states in the two examples. This leads us to the following definition.

Definition 1.1 (Pof set)**.** A *polynomial orbit-finite set*, pof set for short, is any set of the form

$$\mathbb{A}^{d_1} + \cdots + \mathbb{A}^{d_k}$$

for some $k, d_1, \ldots, d_k \in \{0, 1, \ldots\}$.

It is clear why we use the word "polynomial" in the name – syntactically a pof set is the same thing as a univariate polynomial with coefficients in the natural numbers. The meaning of the words "orbit-finite" will become apparent later in this book, when we discuss orbits under the action of atom permutations. In a pof set, we use the name *component* for summands in the disjoint union; each component in a pof set is a set of the form $\mathbb{A}^d$. The *atom dimension* of a component is the exponent d, the atom dimension of a pof set is the maximal atom dimension of its components.

We will be interested in computational models where instead of finite sets, we use pof sets. We already saw this in Examples 1 and 2; in these automata the state spaces and input alphabets where pof sets. This resulting theory will generalize the standard theory of finite objects, because a finite set can be seen as a pof set of atom dimension zero. For example, a set with three elements can be seen as a pof set

$$\mathbb{A}^0 + \mathbb{A}^0 + \mathbb{A}^0$$

that has three components of dimension zero.

In order to get a meaningful theory, we need to make some restrictions on the way that elements of pof sets are manipulated. Otherwise, we would be working with models that use countable sets instead of finite ones. The restriction that we make formalizes the idea that atoms have no structure beyond equality. The idea is that if atoms are renamed in a way that preserves equality, then all properties should be preserved. For example, if an automaton has a transition of the form

$$(\text{John}, \text{Eve}) \quad \overset{\text{Adam}}{\to} \quad (\text{Adam}, \text{John})$$

then the same automaton should also have a transition of the form

$$(\text{Tom}, \text{Adam}) \quad \overset{\text{John}}{\to} \quad (\text{John}, \text{Tom}),$$

because the equality patterns are the same in both transitions. This notion is formalized in the following definition, by using *atom permutations*, which are defined to be bijective functions $\mathbb{A} \to \mathbb{A}$.

Definition 1.2 (Equivariant subset)**.** A subset $X \subseteq \mathbb{A}^d$ is called *equivariant* if it is stable under applying atom permutations, i.e.

$$(a_1, \ldots, a_d) \in X \quad \Leftrightarrow \quad (\pi(a_1), \ldots, \pi(a_1)) \in X$$

holds for every atom permutation π. A subset of a pof set is called equivariant if its intersection with each component is equivariant.

Example 3. Consider the set $\mathbb{A}^3$. Up to atom permutations, this set contains five kinds of elements, namely a non-repeating triple

$$(\text{John}, \text{Eve}, \text{Tom}),$$

three kinds of triples that use two atoms

$$(\text{John}, \text{Eve}, \text{Eve}), \quad (\text{John}, \text{Eve}, \text{Tom}), \quad (\text{John}, \text{Tom}, \text{Eve}),$$

and a triple where all atoms are the same

$$(\text{John}, \text{John}, \text{John}).$$

Every other element of $\mathbb{A}^3$ can be mapped to one of the above five example using an atom permutation, and the five kinds are all different, i.e. none of them can be mapped to another by an atom permutation. If we want to choose an equivariant subset of $\mathbb{A}^3$, we need to decide for each of the five kinds whether we want to include it or not. The five decisions are independent, and therefore there are 2^5 possibilities of choosing an equivariant subset. □

The kinds of elements, as described in the above example, will be called *orbits*. This is because they are the special case of the general notion of orbits under a group action, in the case where the group is the group of atom permutations.

Definition 1.3 (Orbit)**.** The *orbit* of an element x in a pof set X is the set

$$\{ \pi(x) \mid \pi \text{ is an atom permutation } \}.$$

Example 4. In Example 3, we showed that the set $\mathbb{A}^3$ has five orbits. More generally, the number of orbits in $\mathbb{A}^d$ is the number of equivalence relations on the set $\{1, \ldots, d\}$. This number is called the Bell number, and it grows exponentially with d. For example, the 4-th Bell number is 15, and the 5-th Bell number is 52. □

In the above example, we have argued that for sets of the form $\mathbb{A}^d$, the number of equivariant subsets is finite, albeit exponential. This extends to pof sets, which are finite disjoint unions of such sets, and hence we get the following result, which explains the expression "orbit-finite" in the name "polynomially orbit-finite".

Lemma 1.4. *Every pof set has finitely many equivariant subsets.*

In Definition 1.2, we defined equivariant subsets of one pof set. This extends naturally to relations on pof sets, e.g. binary relations

$$R \subseteq X \times Y,$$

where X and Y are pof sets. This is because the product of two pof sets can itself be seen as a new pof set, by distributing products across disjoint unions:

$$\left(\sum_{i\in I} \mathbb{A}^{d_i}\right) \times \left(\sum_{j\in J} \mathbb{A}^{e_j}\right) \quad \equiv \quad \sum_{\substack{i\in I\\ j\in J}} \mathbb{A}^{d_i+e_j}.$$

Similarly, we can also talk about equivariant functions $f : X \to Y$. These are the same as binary relations that are both equivariant and functional, i.e. for every $x \in X$ there exactly one $y \in Y$ that belongs to the relation.

Example 5. To represent booleans, we can use the atomless set

$$2 \quad \overset{\text{def}}{=} \quad \underbrace{\mathbb{A}^0}_{\text{true}} + \underbrace{\mathbb{A}^0}_{\text{false}}$$

For a pof set X, and equivariant function of type $X \to 2$ is the same thing as an equivariant subset of X. □

Example 6. There is only one equivariant function of type $\mathbb{A} \to \mathbb{A}$, namely the identity. Clearly the identity is equivariant, since the corresponding set of pairs is the diagonal

$$\{ (a, a) \mid a \in \mathbb{A} \},$$

and this set is equivariant. Let us now prove that there is no other equivariant function of this type. Suppose then that an equivariant function would map an atom a to some atom $b \neq a$. From the pair (a, b) we can go to any pair (a, c) with $a \neq c$ by applying an atom permutation. This would yield a violation – in fact infinitely many violations – of the functionality condition, which says that each input has only one output. □

Example 7. Let us list all equivariant functions of type $f : \mathbb{A}^2 \to \mathbb{A}$. If the input to such a function is a repeating pair $(a, a) \in \mathbb{A}^2$, then the output has to be a, by the same argument as in the previous example. If the input is a non-repeating pair (a, b) with $a \neq b$, then the output could be either the first argument a or the second argument b. Furthermore, this is uniform: if for some non-repeating pair the output is the first coordinate, then this is true for all non-repeating pairs. This is because every non-repeating pair can be mapped to every other non-repeating pair by an atom permutation. Therefore, there are two possibilities for f: it is either the projection to the first coordinate, or the projection to the second coordinate. □

Example 8. An example of an equivariant function of type $\mathbb{A}^3 \to \mathbb{A}$ is

$$(a, b, c) \quad \mapsto \quad \begin{cases} c & \text{if } a \neq b \\ a & \text{if } a = b. \end{cases}$$

□

As shown in Lemma 1.4, a pof set will have finitely many equivariant subsets. Therefore, there will be finitely many equivariant relations $R \subset X \times Y$, and only some of these will be functions. Summing up, for every pof sets X and Y there will be finitely many equivariant functions of type $X \to Y$.

Exercises

Exercise 1. In the definition of an equivariant set from Definition 1.2, we have an equivalence $\Leftrightarrow$, and we quantify over atom permutations, which can be briefly written as

$$0. \quad \bar{a} \in X \quad \Leftrightarrow \quad \pi(\bar{a}) \in X \quad \text{for all permutations } \pi : \mathbb{A} \to \mathbb{A}$$

Instead of a two-way implication, we can have a one-way implication in either of the two directions, and we can quantify over functions that are not necessarily permutations, as in the following variants:

$$\begin{array}{lllll}
1. & \bar{a} \in X & \Rightarrow & \pi(\bar{a}) \in X & \text{for all permutations } \pi : \mathbb{A} \to \mathbb{A} \\
2. & \bar{a} \in X & \Leftarrow & \pi(\bar{a}) \in X & \text{for all permutations } \pi : \mathbb{A} \to \mathbb{A} \\
3. & \bar{a} \in X & \Leftrightarrow & \pi(\bar{a}) \in X & \text{for all functions } \pi : \mathbb{A} \to \mathbb{A} \\
4. & \bar{a} \in X & \Rightarrow & \pi(\bar{a}) \in X & \text{for all functions } \pi : \mathbb{A} \to \mathbb{A} \\
5. & \bar{a} \in X & \Leftarrow & \pi(\bar{a}) \in X & \text{for all functions } \pi : \mathbb{A} \to \mathbb{A}
\end{array}$$

Which ones are equivalent to the original definition, as in variant 0?

Exercise 2. Show that there is no equivariant function of type $\mathbb{A}^0 \to \mathbb{A}$.

Exercise 3. Show that the number of equivariant subsets of $\mathbb{A}^d$ is doubly exponential in d.

Exercise 4. Consider a pof set X and an equivariant binary relation $R \subseteq X \times X$. Show that the transitive closure of R is also equivariant.

1.1 Representation of equivariant subsets

The central idea of this book is that sets such as pof sets – and generalizations such as (not polynomial) orbit-finite sets that will be defined in later chapters – can be used as a new notion of finiteness, and the resulting computational problems can be studied. A typical example is pof automata, which are automata where the state space and input alphabet are pof sets, the initial and final subsets are equivariant, and the transition relation is also equivariant. The automata from Examples 1 and 2 are pof automata. As we will see in the next section, the emptiness problem is decidable for pof automata.

In order to meaningfully discuss the decision problems based on pof sets and equivariant subsets, we need to have some finite representation, so that they can be used as inputs to algorithms. For pof sets, there is little doubt: a pof set

$$\mathbb{A}^{d_1} + \cdots + \mathbb{A}^{d_k},$$

is represented by the list of natural numbers $d_1, \ldots, d_k$, which describe the atom dimensions of the various components. The relevant question is about representation of equivariant subsets. We think of an equivariant subset in a pof set as being a family of equivariant subsets, one for each component $\mathbb{A}^{d_i}$, and therefore we focus on representing equivariant subsets of individual components. There will be two representations

1.1.1 Generating sets

For a pof set X, define the set *generated* by a subset $Y \subseteq X$ to be all elements that can be obtained by applying atom permutations to elements of Y, i.e.

$$\{ \pi(y) \mid y \in Y \text{ and } \pi \text{ is an atom permutation } \}.$$

In other words, this is the union of orbits of the elements from Y.

Example 9. The full set $\mathbb{A}^2$ is generated by the two pairs

$$(\text{Eve}, \text{Eve}), (\text{John}, \text{Eve}).$$

As explained in Example 4, the set $\mathbb{A}^d$ is generated by a finite subset, whose size is the d-th Bell number. The identity function of type $\mathbb{A}^2 \to \mathbb{A}^2$, when seen as a subset of $\mathbb{A}^4$, is generated by

$$(\text{John}, \text{John}) \mapsto (\text{John}, \text{John}) \qquad (\text{John}, \text{Eve}) \mapsto (\text{John}, \text{Eve}).$$

In the above, we write $a \mapsto b$ instead of (a, b) when describing pairs in the graph of a function □

We can use finite generating subsets as a representation of equivariant subsets. This assumes that we can represent individual atoms; for the moment we simply assume that atoms are strings over some finite alphabet, but the issue of representations will be discussed in more detail in Section 3.5. The representation by generating subsets is general enough to cover all equivariant subsets, as shown in the following lemma.

Lemma 1.5. *Every equivariant subset of a pof set is generated by finitely many elements.*

Proof. There are finitely many orbits, and an equivariant subset is a union of some of these orbits. □

The above lemma shows that finite generating sets can be used as a way of representing equivariant subsets. The representation has several advantages, but conciseness is not one of them. (Non-conciseness can also be framed as an advantage, since making the inputs longer for an algorithm can give a better bound on its running time, as we will see in the next section.) For example, to represent the full subset of $\mathbb{A}^d$ we need a number of generators that is exponential in the dimension d. Also, this representation is not well suited to basic operations on sets. For example, the empty set has a very small representation, but its complement does not. Another example is taking pairs.

Example 10. Consider the subset $X \subseteq \mathbb{A}^d$ that contains only non-repeating pairs. This subset is generated by one element, e.g. if $d = 3$ then a generator is

$$(\text{John}, \text{Adam}, \text{Tom}).$$

However, if we want to take the square X^2, which is an equivariant subset of $\mathbb{A}^{2d}$, then we will need a number of generators that is exponential in d. This is because X^2 consists of tuples of length $2d$ where the first half is non-repeating and the second half is also non-repeating, but there is no further restriction on the equalities between the first half and the second half. In particular, X^2 will contain any tuple where the second half is a permutation of the first half, such as

$$((\text{John}, \text{Adam}, \text{Tom}), (\text{Tom}, \text{John}, \text{Adam})).$$

It will also contain tuples where some atoms are shared between the first and second half, and some atoms are not, such as

$$((\text{John}, \text{Adam}, \text{Tom}), (\text{Tom}, \text{John}, \text{Eve})).$$

□

1.1.2 Formulas

As an alternative to generating sets, we can use formulas. For example, the set of non-repeating tuples in $\mathbb{A}^3$ can be described by the formula

$$x_1 \neq x_2 \land x_1 \neq x_3 \land x_2 \neq x_3.$$

The formulas that we use have no quantifiers, they are only Boolean combinations of equalities on the coordinates (quantifiers will appear later in the book). This representation can be exponentially more concise than the generating set representation. For example, the full set $\mathbb{A}^d$ can be represented by the short formula "true", while the number of generators is exponential in d. Also, the representation efficiently supports such operations as complementation, which is implemented by adding $\neg$ to the formula. Similar comments apply to union and intersection, or to the product operation from Example 10. The following lemma shows that the formula representation is equivalent to the generating set representation, in the sense that both define the same subsets, namely the equivariant subsets.

Lemma 1.6. *A subset $X \subseteq \mathbb{A}^d$ is equivariant if and only if it can be defined by a formula $\varphi(x_1, \ldots, x_d)$ that is constructed using equality comparisons $x_i = x_j$ and Boolean operations $\land, \lor, \neg$.*

Proof. For the implication $\Leftarrow$, we observe that if we apply an atom permutation to a tuple in $\mathbb{A}^d$, then this will not change the pattern of equalities between coordinates, and therefore the truth value of a formula that uses only equality will be preserved.

For the implication $\Rightarrow$, consider an equivariant subset $X \subseteq \mathbb{A}^d$. This subset is generated by a finite set $Y \subseteq X$, thanks to Lemma 1.5. The orbit of generator $y \in Y$ is described by a formula, which asserts the pattern of equalities in this generator:

$$\underbrace{\Big(\bigwedge_{i,j} x_i = x_j\Big)}_{\substack{\text{conjunction ranges over}\\ i,j \in \{1,\ldots,d\}\\ \text{such that } y[i]=y[j]}} \quad \land \quad \underbrace{\Big(\bigwedge_{i,j} x_i \neq x_j\Big)}_{\substack{\text{conjunction ranges over}\\ i,j \in \{1,\ldots,d\}\\ \text{such that } y[i]\neq y[j]}}.$$

Since there are finitely many generators, to define X we can take the finite disjunction of these formulas, ranging over the generators. The size of the formula is the number of generators, times a factor that is polynomial in the dimension d. □

Exercises

Exercise 5. Consider the following problem: given two subsets of a pof set, represented using formulas, decide if they are equal. Show that this problem is: (a) in deterministic logarithmic space under generating set representation; and (b) complete for coNP under formula representation.

Exercise 6. To specify a subset of $X \subseteq \mathbb{A}^d$, we can also use a formula with quantifiers (which range over atoms). Show that for every such formula, there is an equivalent formula that is quantifier-free. For example, the formula

$$\varphi(x_1, x_2) = \exists y\ (x_1 \neq y) \land (x_2 \neq y),$$

is equivalent to "true".

Chapter 2

Automata for polynomial orbit-finite sets

In this section, we discuss in more detail the generalization of automata from finite sets to polynomial orbit-finite sets. We show that, despite being formally infinite, these automata can be treated algorithmically, and some of their decision problems can be decided. However, this comes at a certain cost – not all constructions are allowed, and the model is less robust than for finite sets. For example, determinization fails, because the powerset construction does not work.

2.1 Graph reachability

Before discussing automata and their emptiness, we begin with an even simpler computational problem, namely reachability in directed graphs.

Definition 2.1. A directed pof graph consists of a set of vertices V, which is a pof set, and an edge relation $E \subseteq V^2$ that is equivariant.

Example 11. A simple example is a clique on the atoms: the vertices are $\mathbb{A}$, and all edges are allowed, i.e.

$$E = \{ a \to b \mid a, b \in \mathbb{A} \}$$

Here is a second example, where the edge relation is no longer symmetric. The vertices are $\mathbb{A}^2$, and the edges are

$$E = \{ (a,b) \to (b,c) \mid a, b, c \in \mathbb{A} \}$$

Here is an example of a path in the second graph

$$(\text{John}, \text{John}) \to (\text{John}, \text{Eve}) \to (\text{Eve}, \text{Tom}).$$

Both graphs are strongly connected, i.e. for every vertices v and w there is a path from v to w. $\square$

A directed pof graph can be represented in a finite way, by giving the pof set for the vertices, and some representation (generating set or formula) for the edge relation. Therefore, it is meaningful to discuss decision problems for pof graphs, such as reachability.

Theorem 2.2. *The following problem is decidable:*

- ***Input:*** *A pof graph, and two equivariant subsets of vertices $S, T \subseteq V$.*
- ***Question:*** *Is there a path from some vertex in S to some vertex in T?*

The complexity depends on the representation of equivariant subsets:

- *PSPACE-complete under the formula representation;*
- *NL-complete under the generating set representation.*

Proof. The main idea is that the set of vertices reachable from S is equivariant. Therefore, we can search for a path from initial to final state in the automaton by just looking at orbits of vertices under atom permutations. The rest, including the complexity bounds, is mere bookkeeping.

Generating set representation. We begin with complexity of the problem under the generating set representation. In order to formally speak of this representation, we need to discuss how individual atoms are represented. We will assume that atoms are bit strings, i.e. $\mathbb{A} = 2^*$. (This is a bit inconsistent with our convention of representing atoms as names, but of course names can be encoded in bit strings.) We will show that, under this representation, the reachability problem is complete for the complexity class of nondeterministic logarithmic space (NL). When talking about logarithmic space, we need to use a two-tape model for Turing machines: a read-only input tape, and a read-write work tape of logarithmic size.

- **Lower bound.** A special case of our problem is reachability for finite graphs, since pof sets subsume finite sets. The problem reachability problem is hard for NL in the case of finite graphs, and therefore this lower bound carries over to the more general atom version of the problem.
- **Upper bound.** We reduce the problem to the special case of finite graphs. Reachability in the latter case can be solved in NL, using a naive algorithm that nondeterministically guesses a path, and stores the current vertex by using a pointer to the input tape (logarithmic space suffices for that). The reduction produces the following instance:
  - vertices are those that appear in generators of edges in the original instance, or in generators of the source and target sets S and T;
  - there is an edge $v \to w$ if the target state of the transition v is in the same orbit as the source state of the transition w;

– source vertices are generators of S;
– target vertices are generators of T.

The correctness of the reduction is given in the following claim.

Claim 2.3. *The original instance has a source-to-target path if and only if the same is true in the new instance.*

Proof. Using equivariance of the edge relation, one shows that for every vertex in the original instance, it is reachable from a source if and only if some vertex in the same orbit is reachable in the new instance. □

The reduction can be computed in logarithmic space, even deterministically, and the reachability problem is in NL.

Formula representation. We now discuss the reachability problem under the formula representation. Here, the complexity will be exponentially higher, namely polynomial space instead of logarithmic space.

- **Upper bound.** We use the same kind of nondeterministic guessing algorithm that was used in before. We are allowed to use nondeterminism, since PSpace is equal to NPSpace by Savich's Theorem. This time, we will store on the tape a reachable vertex. At each step, the algorithm guesses a new vertex, with atoms represented as strings, and it then checks if the formula for the edge relation allows a connection. The space used by this algorithm is polynomial in:

  1. the representation of the graph;
  2. the space used to represent atoms.

  We will now justify why the space used by atoms is small, in fact logarithmic in the graph. In every transition, there are at most

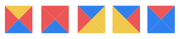
  $$d = \dim V + \dim V$$

  atoms that are used, where dim is the atom dimension. When we are guessing a new vertex, we might need to get some new atoms that were not seen in the previous vertex. We can always take the shortest unused atoms, and so we will always be using the first d atoms, which can be stored using $\log d$ bits.

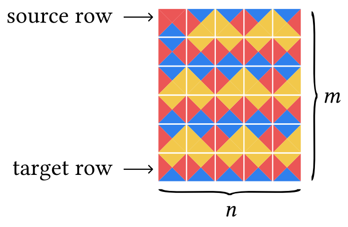
- **Lower bound.** Here, we reduce from the corridor tiling problem. In this problem, we have a finite set of square tiles, where each tile has a colour on each of its four sides. This is formalized as a subset

  $$\underbrace{\{N, S, E, W\} \to C}_{\text{a tile has colours on the four directions of the compass}}.$$

  Elements of this subset will be called tiles. Here is a picture of a set of tiles which uses three colours:



We are also given source and target rows s, t, which are sequences of tiles of the same length, say n. This length will be the width of the corridor. A solution to the corridor tiling problem is an $n \times m$ rectangle labelled by the tiles, such that the first row is s, the last row is t, and every two adjacent tiles have the same colour on their connecting side. Here is a picture of a solution:



The corridor tiling problem, i.e. deciding if there exists a solution, is PSpace-complete. We will show that the corridor tiling problem reduces to the graph reachability problem, under the formula representation, thus proving PSpace-hardness for the latter problem. A vertex of the graph will store the representation of a row in the solution. Assuming that there are $k = |C|$ colours, one row will be represented by $k + 4n$ atoms

$$(\underbrace{a_1, \ldots, a_k}_{\substack{\text{distinct atoms}\\ \text{that represent}\\ \text{the tile colours}}}, \underbrace{n_1, s_1, e_1, w_1, \ldots, n_n, s_n, e_n, w_n}_{\substack{\text{four atoms for each tile in the row,}\\ \text{corresponding to the colours on the sides}\\ \text{north, south, east, west}}}).$$

Not all tuples of length $k + 4n$ represent a row, but the tuples that do can be specified by a formula that has size polynomial in k and d, as follows:

$$\underbrace{\bigwedge_{i \neq j \in \{1,\ldots,n\}} a_i \neq a_j}_{\text{atoms for colours are distinct}}$$
$$\wedge \underbrace{\bigwedge_{i \in \{1,\ldots,n-1\}} e_i = w_{i+1}}_{\text{colours match horizontally}}$$
$$\wedge \underbrace{\bigwedge_{i \in \{1,\ldots,n\}} \bigvee_{t \in T} n_i = a_{t(\text{north})} \wedge s_i = a_{t(\text{south})} \wedge e_i = a_{t(\text{east})} \wedge w_i = a_{t(\text{west})}}_{\text{each position is occupied by a legitimate tile}} .$$

We can further refine the above formula to say that the row represents the source row, or the target row, by restricting the tile t from the last condition to be the one that should be used. This way, we get formulas for the initial and final vertices. Finally, we need to specify the formula for the edge relation. This formula has

$$\underbrace{k + 4n}_{\substack{\text{old}\\\text{row}}} + \underbrace{k + 4n}_{\substack{\text{new}\\\text{row}}},$$

variables. It says that both the old and new rows are valid, in the sense described above, and furthermore the south atoms of the old row match the north atoms of the new row. This, again, can be described by a formula polynomial in k and n. It is now easy to see that accepting runs of the automaton correspond to solutions of the corridor tiling problem, and therefore the nonemptiness problem is PSpace-hard.

□

Exercises

Exercise 7. Show that the reachability problem remains PSpace-complete when we restrict it to symmetric graphs, i.e. graphs where the edge relation is symmetric[1].

Exercise 8. Consider an undirected pof graph, i.e. a graph where the edge relation is symmetric. Does it necessarily have a spanning tree that is equivariant?

Exercise 9. Consider two undirected pof graphs, which are isomorphic. Is there necessarily an isomorphism that is equivariant?

Exercise 10. Show that given a directed pof graph, one can compute a number $k \in \{0, 1, \ldots\}$ such that for every vertices s and t, if there is a path from s to t, then there is a path of length at most k.

Exercise 11. Consider a directed pof graph. Show that there is an infinite path if and only if there is a cycle.

Exercise 12. Consider a directed pof graph. Show that if the graph is acyclic, then there is a finite upper bound k on the length of paths.

Exercise 13. Show that the following problem is decidable: given a directed pof graph, decide if it has finite outdegree, i.e. for every vertex v, there are finitely many vertices w with an edge $v \to w$.

Exercise 14. Assume the equality atoms. Show a graph which has an infinite path, but does not have any infinite finitely supported path.

2.2 Automata and their emptiness problem

In this section, we introduce pof variants of deterministic and nondeterministic automata, and use the graph reachability result from the previous section to show that the emptiness problem is decidable. We begin by formally defining the model.

[1]Note that in the case of finite graphs, the complexity drops from NL to L when restricting to symmetric graphs, as shown by Rheingold.

Definition 2.4 (Pof automaton)**.** A nondeterministic pof automaton consists of:

1. an input alphabet Σ, which is a pof set;
2. a state space Q, which is a pof set;
3. initial and accepting subsets $I, F \subseteq Q$, which are equivariant;
4. a transition relation $\Delta \subseteq Q \times \Sigma \times Q$, which is equivariant.

A deterministic pof automaton is the special case where there is exactly one initial state, and where the transition relation is a function.

As was the case for graphs, the above definition is simply the usual definition, except that finite sets are replaced by pof sets, and all subsets, relations and functions are required to be equivariant. Using the same principle, one can consider pof variants of other structures, such as graphs, pushdown automata, context-free grammars, Turing machines, etc. (This will be the content of Chapter 3.)

Using the result about graph reachability, we obtaine decidability of emptiness for pof automata, deterministic or not.

Theorem 2.5. *The emptiness problem is decidable for nondeterministic pof automata. The complexity is the same as for the reachability problem in pof graphs.*

Proof. The lower bounds for graph reachability transfer directly to the emptiness problem, by considering automata over a one-letter alphabet, which are the same as instances of graph reachability. For the upper bound under the formula representation, we can use the same straightforward nondeterministic algorithm as in graph reachability. For the upper bound under the generating set representation, we simply delete the input letters from the generators of the transitions, and then we solve the corresponding instance of graph reachability. □

A corollary of the above theorem is decidability of language equivalence for deterministic pof automata (as we will see in the following section, the problem is no longer decidable for nondeterministic automata).

Corollary 2.6. *The following problem is decidable:*

- ***Input:*** *Two deterministic pof automata.*
- ***Question:*** *Do they recognise the same language?*

The complexity is the same as for the emptiness problem.

Proof. We can use the product construction. A product $Q_1 \times Q_2$ of two pof sets is also a pof sets, and the corresponding operations on subsets and transitions preserve equivariance. We then check if in the product automaton, one can reach states that are accepting in one automaton, but rejecting in the other. □

In the equivalence algorithm from the above corollary, we use determinism. This is because we implicitly complement an automaton by complementing its accepting states, and this only works for deterministic automata. As we will see in the next section, the equivalence problem becomes undecidable for nondeterministic automata. Let us first show that we cannot solve this problem by determinizing. The natural idea would be to use the powerset construction. Unfortunately, this fails. For example, if we take the set $\mathbb{A}$, then its powerset $\mathsf{P}\mathbb{A}$ is not a pof set. In fact, not only this construction fails, but there is no successful construction at all, as shown in the following theorem.

Theorem 2.7. *Languages recognised by nondeterministic pof automata are not closed under complementation. Also, deterministic pof automata are strictly less expressive than nondeterministic ones.*

Proof. The first part of the theorem directly implies the second part, since deterministic automata are closed under complementation. To prove the first part, we use the language

$$L = \{ w \in \mathbb{A}^* \mid \text{ some letter appears twice } \}.$$

In Example 2, we showed that this language is recognised by a nondeterministic pof automaton. It remains to show that its complement is not recognised by any nondeterministic pof automaton.

The complement of L consists of words where all letters are pairwise different. Suppose, toward a contradiction, that this complement is recognised by a nondeterministic pof automaton. Let d be the atom dimension of the state space, i.e. the maximal number of atoms that can be stored in a state. Choose n so that it is strictly larger than the atom dimension of the state space, and consider a word w with $2n$ pairwise distinct atoms. This belongs to the complement of the language, and thus it must have an accepting run. Consider the state q in the middle of the accepting run, i.e. a state with

$$I \ni p \xrightarrow{w_1} q \xrightarrow{w_2} r \in F,$$

where w_1 and w_2 are the two halves of w that have length n. By assumption on n, there must be some atom a that appears in the first half w_1 but not in the state q. Similarly, there must be some atom b that appears in the second half w_2 but not in the state q. Let π be the atom permutation that swaps a and b. By equivariance, we have

$$\pi(q) \xrightarrow{\pi(w_2)} \pi(r) \in F.$$

Since neither a nor b appear in q, we have $\pi(q) = q$, and therefore we can stitch the two runs above to get an accepting run over the concatenation of w and $\pi(v)$, which is a word that has an atom repetition, and therefore should be rejected. □

Exercises

Exercise 15. Find a deterministic pof automaton for the following language:

$$\{ w \in \mathbb{A}^* \mid \text{ the first and last letters are different } \}.$$

Exercise 16. Find a deterministic pof automaton for the following language:

$$\{ w \in \mathbb{A}^* \mid \text{no two consecutive letters are the same} \}.$$

Exercise 17. Find a deterministic pof automaton for the language

$$\{ w \in \mathbb{A}^* \mid \text{there are at least three different letters} \}.$$

Exercise 18. Consider a deterministic pof automaton. Show that after reading an input string w, all atoms that appear in the state must also appear in w.

Exercise 19. Show that if the input alphabet is finite (i.e. a pof set of dimension zero), then deterministic pof automata recognise exactly the regular languages.

Exercise 20. Consider a nondeterministic pof automaton, and let k be the maximal number of atoms that can appear in a single transition. Let A be a finite set of k atoms. Show that if the automaton is nonempty, then it accepts some word that uses only atoms from A.

Exercise 21. Define a left derivative of a language $L \subseteq \Sigma^*$ to be a language of the form

$$v^{-1}w \stackrel{\text{def}}{=} \{ w \in \Sigma^* \mid \ vw \in L \ \}$$

for some word $v \in \Sigma^*$. Are languages recognised by deterministic pof automata closed under left derivatives?

Exercise 22. We say that two states p and q in a deterministic pof automaton are *behaviourally equivalent* if for every input string w, the states pw and qw are both accepting or both rejecting. Show that behavioural equivalence is equivariant.

Exercise 23. A deterministic pof automaton is called minimal if one cannot find reachable states $p \neq q$ that are behaviourally equivalent. Show a language that is recognised by some deterministic pof automaton but not by any minimal deterministic pof automaton.

Exercise 24. Show that the expressive power of nondeterministic pof automata does not change if we allow ε-transitions.

Exercise 25. Show that for every nondeterministic pof automaton, the set

$$\{ n \in \{0, 1, \ldots\} \mid \ \text{the automaton accepts some word of length } n \ \}$$

is ultimately periodic.

Exercise 26. Show that the question from Problem **??** has the same complexity if we restrict to undirected graphs[2]

Exercise 27. Show a family of deterministic pof automata, such that in the n-th automaton the input alphabet is $\mathbb{A}$, the state space is $\mathbb{A}^0 + \mathbb{A}^n$, but the shortest accepted word is exponential in n.

Exercise 28. Show that for every deterministic pof automaton one can compute some bound k such that if two states p and q are not behaviourally equivalent, then they can be distinguished

[2]This is in contrast to atomless graphs, where the complexity drops to deterministic logspace, as proved by Rheingold.

by some word of length at most k. Show that this k can be exponential in the dimension of the state space.

Exercise 29. Show that the following problem is decidable: given a deteriminstic pof automaton, decide if its language is commutative.

Exercise 30. A language is called *positively equivariant* if for every function $\pi : \mathbb{A} \to \mathbb{A}$ which is not necessarily a bijection, we have

$$w \in L \quad \Rightarrow \quad \pi(w) \in L.$$

Show that the following problem is decidable: given a determinstic pof automaton, decide if its language is positively.

Exercise 31. Let $L \subseteq \Sigma^*$ be a language recognised by a deterministic pof automaton. Show that there is some k with the following property: for every $w \in \Sigma^*$ there are atoms $a_1, \ldots, a_k \in \mathbb{A}$ such that for every atom permutation π that fixes all atoms from $a_1, \ldots, a_k$, and every $v \in \Sigma^*$ we have

$$wv \in L \quad \Leftrightarrow \quad \pi(w)v \in L.$$

Exercise 32. Suppose that $L \subseteq \Sigma^*$ is recognised by a nondeterministic pof automaton. Show that there is some k such that for every $w \in \Sigma^*$ longer than k one can find

1. a decomposition $w = xyz$ with y nonempty; and
2. an atom permutation π that moves finitely many atoms

such that for every $n \in \{0, 1, \ldots\}$ we have

$$xy\pi^1(y)\pi^2(y)\cdots\pi^n(y)\pi^n(z) \in L.$$

Exercise 33. Is there a deterministic pof automaton for the following language?

$$\{\ w \in \mathbb{A}^* \mid \text{ all letters are different }\ \}$$

Exercise 34. Which of the following closure properties are true for the class of languages recognised by deterministic pof automata?

1. complementation;
2. union;
3. intersection;
4. reverse;
5. concatentation;
6. Kleene star.

Exercise 35. Which of the closure properties from Problem 34 hold for nondeterministic pof automata?

Exercise 36. Show that the complement of the language

$$\{\ ww \mid \ w \in \mathbb{A}^* \text{ is non-repeating }\ \}$$

is recognised by a nondeterministic pof automaton.

Exercise 37. In this exercise, we consider a variant of regular expressions. Consider the least class of languages that:

1. contains every equivariant set of words that has bounded length;
2. is closed under union and concatenation;
3. is closed under Kleene star L^*.

Show that these languages are a strict subset of nondeterministic, pof automata.

Exercise 38. Show that the regular expressions from the previous exercise are not closed under intersection.

2.3 Undecidable universality

In Corollary 2.6, we showed that language equivalence is decidable for deterministic pof automata. We will now show that this problem becomes undecidable for nondeterministic automata. In fact, already a special case of the language equivalence problem will be undecidable: given one nondeterministic pof automaton, we want to decide if it accepts all words, i.e. it is equivalent to the automaton that accepts all words. For finite automata, this problem is solved by complementing, and then testing for emptiness. As we have seen above, this approach will fail for pof sets, since nondeterministic pof automata are not closed under complementation. The following theorem shows that no other approach will work, since the problem is undecidable.

Theorem 2.8. *The following problem is undecidable:*

- ***Input:*** *A nondeterministic pof automaton.*
- ***Question:*** *Does it reject some word?*

Proof. In the proof, we reduce from the halting problem for counter machines.

Let us begin by describing counter machines. This is a computational model that uses counters which store natural numbers, and which can updated via increments, decrements and zero tests. The syntax of the machine is given by a finite set of states Q, a finite set C of counters, and a finite set of transitions

$$\Delta \subseteq \underbrace{Q}_{\text{source state}} \times \underbrace{\{\text{inc}, \text{dec}, \text{zero}\}}_{\text{counter operations}} \times \underbrace{C}_{\text{which counter is affected}} \times \underbrace{Q}_{\text{target state}}.$$

The increment operation adds one to the affected counter and leaves the other counters unchanged, the decrement operation subtracts one, and the zero test transition does not change the counters but is only enabled if the corresponding counter is equal to zero. A decrement cannot be performed if the corresponding counter is zero, since we require counters to be natural numbers. The semantics of the machine is its *configuration graph*, which is a directed graph where the vertices are

$$\underbrace{Q}_{\text{state}} \times \underbrace{\mathbb{N}^C}_{\text{counter valuation}}$$

and where the edges are given by the transitions in the expected way. The following problem, which we call the *halting problem for counter machines*, is a classic undecidable problem:

- **Input:** a counter machine, and two states p and q.
- **Question:** is there a path from $(p, \bar{0})$ to $(q, \bar{0})$ in the configuration graph?

We will show that the above problem reduces to universality of nondeterministic pof automata, and therefore the latter problem is also undecidable. (The halting problem for counter machines is known to be undecidable even for two counters. However, we do not need this in our reduction, since it will also work with more than two counters.)

In the reduction, we will use words with atoms to represent accepting runs of the counter machine. The idea is to use atoms to represent the matching between increments and their corresponding decrements. Formally speaking, define L to be the set of words over the alphabet

$$\mathbb{A} \times \Delta$$

which satisfy the following two conditions:

1. The first transition has the initial state p, the last transition has the target state q, and consecutive transitions agree on states.
2. An atom can appear at most twice. Also:
   (a) if an atom appears once, then its only appearance labels a zero test;
   (b) if an atom appears twice, then the first appearance labels an increment, the second appearance labels a decrement on the same counter, and there are no zero tests on this counter between them.

It is not hard to see that L is nonempty if and only if there is a path as in the halting problem. Therefore, if we could decide emptiness of L, then we could decide the halting problem. Emptiness of L is the same as universality for its complement. The following lemma shows that the complement is recognised by a pof automaton, thus completing the reduction.

Lemma 2.9. *The complement of L is recognised by a nondeterministic pof automaton.*

Proof. A word belongs to the complement of L if and only if it violates one of the two conditions 1 or 2 in the definition of L. Since nondeterministic pof automata are closed under unions, it is enough to give separate automata for the two conditions. Condition 1 does not refer to atoms, and therefore its complement can be recognised by a finite automaton, because regular languages on finite (i.e. atom-less) alphabets are closed under complementation. The interesting part is violations of condition 2. This condition is violated if and only if at least one of the following holds:

(a) some atom appears at least three times; or

(b) some atom appears at least twice, but in a way that violates condition 2(ii), i.e. either the two appearances are not matching increment/decrement pairs, or there is a zero test between them; or

(c) some atom appears exactly once, but is label is not a zero test.

It is enough to give a separate automaton for each of the three kinds of violations. Violations of kind (a) can be recognised by an automaton which stores the repeated atom in its state, using state space

$$\underbrace{\mathbb{A}^0}_{\text{initial}} + \underbrace{\mathbb{A}^1}_{\text{after first}} + \underbrace{\mathbb{A}^1}_{\text{after second}} + \underbrace{\mathbb{A}^0}_{\text{accepting}}.$$

Violations of kind (b) can be recognised by several copies of the following automaton, with one copy for each counter:

$$\underbrace{\mathbb{A}^0}_{\text{initial}} + \underbrace{\mathbb{A}^1}_{\text{after first}} + \underbrace{\mathbb{A}^0}_{\text{accepting}}.$$

The most interesting automaton is for violations of kind (c). The challenge is that the automaton needs to check that the atom appears exactly once. For this reason, it must guess the atom immediately, before having seen it, to check that it does not appear earlier in the word. This is done by an automaton with a state space

$$\underbrace{\mathbb{A}^1}_{\text{initial}} + \underbrace{\mathbb{A}^1}_{\text{after first}} + \underbrace{\mathbb{A}^0}_{\text{accepting}},$$

where the initial states stores an atom that is nondeterministically guessed. □

Observe that the automaton in the proof of the lemma above had atom dimension one, and therefore the universality problem is undecidable already for such automata. □

Exercises

Exercise 39. To express properties of words in $\mathbb{A}^*$, we can use first-order logic, where the quantifiers range over positions, and there are predicates for the order on positions, and equality of data values. For example, the following formula says that the first position has the same atom as some later position:

$$\forall x \quad \underbrace{(\forall y\, y \geq x)}_{x \text{ is the first position}} \quad \Rightarrow \quad \underbrace{(\exists y\, y > x \land y \sim x)}_{\substack{x \text{ has the same atom} \\ \text{as some later position}}}.$$

Show that satisfiability is undecidable for this logic, i.e. one cannot decide if a given formula is true in some word from $\mathbb{A}^*$.

2.4 A decidable case of universality

In this section, we show that under extra assumptions, we can recover decidability of universality for nondeterministic pof automata. There is not much space here, since the undecidability argument used automata with atom dimension one, i.e. a state would store at most one register. Of course atom dimension zero would be sufficient for decidability, since such automata determinize (even if the input alphabet is infinite), but we are looking for something more exciting. It turns out that the crucial distinction is nondeterministic guessing of atoms that was used in the reduction in the previous section. We formalize this in the following definition.

Definition 2.10. A nondeterministic pof automaton is called *guessing* if there is some initial state that contains an atom, or some transition

$$p \xrightarrow{a} q,$$

where q contains an atom that appears neither in p nor a.

The automaton for condition (c) in the proof of Lemma 2.9 used guessing, since its initial states contained atoms. We will show that if the automaton is non-guessing, and its state space has dimension at most one, then the universality problem is decidable. The bound is tight – if we lift allow guessing then we can use the undecidability proof from Theorem 2.8, and if we allow dimension two, then we also get undecidability, which is left as an exercise for the reader.

Theorem 2.11. *The following problem is decidable:*

- ***Input:*** *A nondeterministic pof automaton, which is non-guessing and has a state space of dimension at most one.*
- ***Question:*** *Does it accept all words?*

We do not give any complexity bounds, since the algorithm that we provide is highly inefficient, and is based on a brute-force procedure that searches through all possible witnesses of some kind, with no explicit bound on the size of these witnesses.

The rest of Section 2.4 is devoted to proving the above theorem. As mentioned in Section **??**, the powerset construction does not work for pof sets. However, as we will see in this proof, the two extra assumptions – dimension at most one and non-guessing – will enable us to exhaustively search the state space in the powerset automaton, despite this automaton not being a pof automaton.

For the rest of this proof, fix a nondeterministic pof automaton

$$\mathcal{A} = (Q, \Sigma, \Delta, I, F)$$

that we want to check for universality. We will discuss the usual powerset automaton, which we denote by $\mathsf{P}\mathcal{A}$, despite this automaton not being a pof automaton. Let us recall the construction of the powerset automaton. The input alphabet is the same. States in the powerset automaton are sets of states in the original automaton, with the initial state being I, and the final states being those that intersect F. The point of the powerset automaton is that it is deterministic: when it is in a state $P \subseteq Q$, and it reads an input letter a, then it deterministically goes to the state

$$\{ q \in Q \mid \ p \xrightarrow{a} q \text{ for some } p \in P \ \}.$$

Although the powerset automaton is not a pof automaton, we will be able to represent its reachable states in a finite way, under the extra assumptions from the theorem.

The non-guessing assumption will ensure that only finite sets of states appear in the powerset automaton.

Lemma 2.12. *If the automaton $\mathcal{A}$ is non-guessing, then every reachable state in the powerset automaton is a finite subset of Q.*

Proof. By the non-guessing assumption, we know that if the powerset automaton reads an input word w, then its state will be a subset of

$$\{ q \in Q \mid \text{ all atoms from } q \text{ appear in } w \ \}.$$

The above set is finite, since a pof set can only have finitely many elements that use a given finite set of atoms. □

Thanks to the above lemma, from now on, we will be working in the *finite powerset automaton*, which is obtained from the powerset automaton by restricting its state space to finite sets. The general idea behind our universality algorithm is that it will exhaustively search for two kinds of witnesses:

- a finite witness that the automaton rejects some word; or
- a finite witness that the automaton accepts all words.

The witnesses of the first kind are straightforward: they are rejected words. The witnesses of the second kind are described in the following lemma (we will later explain why these witnesses can be viewed as finite).

Lemma 2.13. *The automaton $\mathcal{A}$ accepts all words if and only if there is a family*

$$\mathcal{R} \subseteq \mathsf{P}_{fin} Q$$

of states in the finite powerset automaton with the following properties:

1. *$\mathcal{R} \ni I$, i.e. $\mathcal{R}$ contains the initial state of the finite powerset automaton;*
2. *$\mathcal{R}$ contains only sets that intersect F, i.e. it only contains accepting states of the finite powerset automaton;*
3. *$\mathcal{R}$ is closed under applying transitions of the finite powerset automaton;*
4. *$\mathcal{R}$ is equivariant, i.e. closed under applying atom permutations;*
5. *$\mathcal{R}$ is upward closed with respect to inclusion, when restricted to finite sets: if a finite set $P \subseteq Q$ contains some set from $\mathcal{R}$, then also $P \in \mathcal{R}$.*

Proof. The implication $\Leftarrow$ is immediate; in fact it holds already if we only keep the first three conditions 1–3. Let us now prove the implication $\Rightarrow$. Define $\mathcal{R}$ to be the upward closure of the reachable states of the powerset automaton, i.e. $\mathcal{R}$ is the finite sets that contain some reachable state of the powerset automaton. Conditions 1, 2 and 5 follow from the definition of $\mathcal{R}$. By equivariance of the original automaton, the set of reachable states in the powerset automaton is also equivariant, and therefore so is its upward closure, thus proving condition 4. Finally, condition 3 follows from monotonicity of the transition function of the powerset automaton: if we increase the source state, then we also increase the target state. □

As mentioned in the proof, the equivalence in the above lemma would continue to hold without the last two conditions 4 and 5 about equivariance and upward closure. The purpose of These conditions, and of the assumption that Q has dimension at most one, is to ensure that $\mathcal{R}$ can be represented in a finite way. This representation will be based on the following order on finite subsets of Q:

$$R_1 \sqsubseteq R_2 \qquad \text{iff} \qquad \pi(R_1) \subseteq R_2 \text{ for some atom permutation } \pi. \tag{2.1}$$

It is not hard to see that two subsets are equivalent under the above order, i.e. they are compared in both directions, if and only if they are in the same orbit under atom permutations. It is also not hard to see that a subset $\mathcal{R}$ is upward closed with respect to this order if and only if it satisfies conditions 4 and 5. The following lemma shows that every upward closed set – and therefore also the set $\mathcal{R}$ from Lemma 2.13 – is the upward closure of a finite set, and thus it can be represented in a finite way. The lemma crucially uses the assumption on dimension one, and it would fail for atom dimension two or more.

Lemma 2.14. *Let Q be a pof set of dimension one. If $\mathcal{R} \subseteq \mathsf{P}_{fin} Q$ is upward closed with respect to the order $\sqsubseteq$, then there is a finite set $\mathcal{R}_0 \subseteq \mathcal{R}$ such that*

$$R \in \mathcal{R} \qquad \textit{iff} \qquad R_0 \sqsubseteq R \textit{ for some } R_0 \in \mathcal{R}_0.$$

Proof. To prove the lemma, we will use a similar observation about vectors of natural numbers, equipped with the coordinatewise ordering

$$(x_1, \ldots, x_d) \le (y_1, \ldots, y_d) \quad \stackrel{\text{def}}{=} \quad x_1 \le y_1 \land \cdots \land x_d \le y_d.$$

This observation is called Dickson's Lemma, and is stated below.

Dickson's Lemma *Let $X \subseteq \mathbb{N}^d$ be a set that is upward closed with respect to the coordinatewise ordering. Then there is some finite set $X_0 \subseteq X$ such that*

$$x \in X \qquad \textit{iff} \qquad x_0 \le x \textit{ for some } x_0 \in X_0.$$

We will reduce the present lemma to Dickson's Lemma, using the assumption that Q has atom dimension at most one. Let us decompose Q into components of dimension zero and one:

$$Q \quad = \quad \underbrace{\mathbb{A}^0 + \cdots + \mathbb{A}^0}_{\substack{k \text{ components of} \\ \text{dimension zero}}} \quad + \quad \underbrace{\mathbb{A}^1 + \cdots + \mathbb{A}^1}_{\substack{\ell \text{ components of} \\ \text{dimension zero}}}$$

For a finite set $R \subseteq Q$, define its *profile* to be the following information:

i. which elements from components of dimension zero belong to R;

ii. for each nonempty subset $I \subseteq \{1, \ldots, \ell\}$, how many atoms a satisfy

$$i \in I \qquad \text{iff} \qquad \text{the } i\text{-th copy of } a \text{ belongs to } P.$$

The profile of a set identifies the set up to atom permutations, i.e. two finite subsets of Q have the same profile if and only if they are in the same orbit. Together with equivariance of $\mathcal{R}$, this implies that membership in $\mathcal{R}$ can be seen as a question about profiles, i.e. we have

$$R \in \mathcal{R} \qquad \Leftrightarrow \qquad \text{profile}(R) \in \mathcal{P}, \tag{2.2}$$

where $\mathcal{P}$ is defined to be the image of $\mathcal{R}$ under the profile map.

The point of profiles is that they are essentially vectors of natural numbers. More precisely, we can view the profile as a function

$$\mathsf{P}_{\text{fin}} Q \to \underbrace{\{0,1\}^{\{1,\ldots,k\}}}_{\substack{\text{answers to}\\ \text{question (i)}}} \times \underbrace{\mathbb{N}^{\text{nonempty subsets of } \{1,\ldots,\ell\}}}_{\substack{\text{answers to}\\ \text{question (ii)}}}$$

This allows us to view finite sets of states as vectors of natural numbers of fixed dimension, which will enable us to use Dickson's Lemma. (Observe that this technique would not longer work for dimension two, since we would need to describe how pairs of atoms interact).

The profile map has the following monotonicity property, where profiles are ordered coordinatewise:

$$R_1 \sqsubseteq R_2 \qquad \Leftarrow \qquad \text{profile}(R_1) \leq \text{profile}(R_2). \tag{2.3}$$

This is because increasing the profile corresponds to adding states to the set. Thanks (2.3) and upward closure of $\mathcal{R}$, the set of profiles $\mathcal{P}$ is upward closed under $\leq$. Therefore, we can apply Dickson's Lemma to conclude that $\mathcal{P}$ is generated by some finite set of profiles $\mathcal{P}_0$. Together with (2.2), this gives us

$$R \in \mathcal{R} \quad \Leftrightarrow \quad P_0 \leq \text{profile}(P) \text{ for some } P_0 \in \mathcal{P}. \tag{2.4}$$

Choose $\mathcal{R}_0$ so that its profiles are exactly those from $\mathcal{P}_0$. The conclusion of the lemma is proved in the following diagram:

$$\begin{tikzcd}
 & R \in \mathcal{R} & \\
\begin{array}{c} P_0 \leq \text{profile}(R) \\ \text{for some } P_0 \in \mathcal{P}_0 \end{array} \arrow[ur, Leftrightarrow, "(2.4)"] \arrow[rr, Rightarrow, "(2.3)"'] & & \begin{array}{c} R_0 \sqsubseteq R \\ \text{for some } R_0 \in \mathcal{R} \end{array} \arrow[ul, Rightarrow, "\text{upward closure of } \mathcal{R}"']
\end{tikzcd}$$

□

We are now ready to complete the proof of the theorem. Define a witness for universality to be a set $\mathcal{R}$ as in Lemma 2.13, which is represented by a finite set $\mathcal{R}_0$ as in Lemma 2.14. Define a witness for non-universality to be a rejected word. The algorithm exhaustively searches for witnesses of both kinds, and it is guaranteed to find one in finite time. (As mentioned before, we have no explicit bounds on the running time of the algorithm, beyond saying that it runs in finite time.) It remains to show that one can verify a witness for universality, i.e. given a finite set $\mathcal{R}_0$, one can check if its upward closure $\mathcal{R}$ satisfies the conditions of Lemma 2.13. The only interesting condition to check is condition 3, which says that $\mathcal{R}$ is closed under applying transitions. By monotonicity of the transition function in the powerset automaton, this reduces to checking if for every R in the finite set $\mathcal{R}_0$ and every input letter $a \in \Sigma$, the resulting state is in $\mathcal{R}$. Although there are infinitely many possible letters a, we only need to check this for finitely many choices, since we only need to use at most d fresh atoms, where d is the atom dimension of the input alphabet and fresh atoms are those that do not appear in R. This completes the proof of Theorem 2.11.

Exercises

Exercise 40. Show that languages recognised by one way non-guessing alternating automata are not closed under reversals.

Exercise 41. Show that the order defined in (2.1) is no longer a well-quasi ordering if we use infinite subsets of $\mathbb{A}$.

Exercise 42. Show that the order defined in (2.1) is no longer a well-quasi ordering if we use finite subsets of $\mathbb{A}^2$ instead of $\mathbb{A}$. For example, we have

$$\{(\text{John}, \text{Eve}), (\text{John}, \text{John})\} \quad \leq \quad \{\underbrace{(\text{Eve}, \text{Tom})}_{\substack{\text{this pair} \\ \text{is deleted}}}, \underbrace{(\text{Mark}, \text{John}), (\text{Mark}, \text{Mark})}_{\substack{\text{to the remaining elements, we} \\ \text{apply an atom permutation with} \\ \text{Mark} \mapsto \text{John and John} \mapsto \text{Eve}}}.\}$$

Exercise 43. Consider the following ordering on $\mathbb{A}^*$. We say that $w \leq v$ if one can obtain w from v as follows: (a) first delete some letters from v; then (b) apply some atom permutation. Is this a well-quasi-ordering?

Exercise 44. We say that a language $L \subseteq \Sigma^*$ is *upward closed* if it is closed under inserting letters. In other words,

$$wv \in L \quad \Rightarrow \quad wav \in L \quad \text{for every } w, v \in \Sigma^* \text{ and } a \in \Sigma.$$

Is it true that every language that is both equivariant and upward closed is necessarily recognised by a nondeterministic pof automaton?

Chapter 3

More computational models with atoms

In the previous sections, we discussed pof variants of deterministic and nondeterministic automata. In this chapter, we give a sample of other models of computation, namely alternating automata, two-way automata, circuits, context-free grammars, and Turing machines. This material below is nothing but a collection of exercises, each one preceded by a brief description of the relevant model of computation.

3.1 Alternating automata

Earlier in this chapter, we showed that in the pof setting, nondeterministic automata are no longer equivalent to deterministic. There are two more examples of this phenomenon, namely two other variants of automata that are equivalent to the usual automata in the atom-free case, but are no longer equivalent in the presence of atoms. The first of these is alternating automata.

An *alternating pof automaton* is a generalization of nondeterministic automata, which is self-dual in the sense that it does not priviledge existential choice over universal choice. Let us describe this model. The automaton has the same syntax as a pof nondeterministic automaton, except that there is an additional equivariant partition of the states into two parts, called the *existential* and *universal* states. The semantics are changed as follows. We assume that the automaton has one initial state, and the language of the automaton is defined to be the words accepted from this initial state. The set of words w accepted from a state q is defined by induction on the length of the word as follows[1]. The empty word is accepted from q if and only if q is a final state. Consider now a nonempty word, say of the form aw where a is a letter and w is some shorter word. The word aw is accepted from a state q if:

[1]This model is often considered with ϵ-transitions, but we avoid them for simplicity. However, we do use ϵ-transitions in Exercise 59, with the semantics being left to the reader.

- the state q is existential, and w is accepted from p for some transition

$$q \xrightarrow{a} p;$$

- the state q is universal, and w is accepted from q for every transition

$$q \xrightarrow{a} p.$$

Exercises

Exercise 45. Give an alternating pof automaton that recognises the language

$$\{ w \in \mathbb{A}^* \mid \text{ all letters in } w \text{ are different } \}.$$

Exercise 46. Show that in the atom-less case, i.e. when the states and input alphabet have atom dimension zero, this model recognises exactly the regular languages.

Exercise 47. Show that languages recognised by alternating pof automata are closed under complement.

Exercise 48. Show that emptiness is undecidable for alternating pof automata.

Exercise 49. Show that emptiness continues to be undecidable for alternating pof automata even if we require the state space to have atom dimension 1.

Exercise 50. Show a language that witnesses point 3 in Figure **??**.

Exercise 51. Show a language that witnesses point 4 in Figure **??**, possibly assuming conjectures about complexity classes being distinct.

Exercise 52. Show that the non-guessing alternating pof automata are strictly weaker than the general model.

Exercise 53. Show that emptiness becomes decidable for alternating pof automata if we require the state space to have atom dimension 1, and the automaton must be non-guessing.

3.2 Two-way automata

We now describe a second extension of finite automata, which is equivalent to the usual automata in the atom-free case, but not in the presence of atoms. This is a two-way automaton, where the head can move both left and right. This model is the same as Turing machines that have a read-only input tape and no work tape. We will consider the pof variant of this model, in the deterministic case. A *deterministic two-way pof automaton* is defined like a deterministic pof automaton, except that the transition function is of type

$$Q \times \underbrace{(\Sigma + \{\vdash, \dashv\})}_{\text{input letters or endmarkers}} \to \{\text{accept}, \text{reject}\} + (Q \times \underbrace{\{\text{left}, \text{stay}, \text{right}\}}_{\text{head movement}}).$$

The automaton can also reject by entering an infinite loop.

Exercises

Exercise 54. Show that in the atom-less case, i.e. when the states and input alphabet have atom dimension zero, this model recognises exactly the regular languages.

Exercise 55. Suppose that atoms are names, which can be written using the latin alphabet. The *atomless representation* of an element in a pof set is the string over the finite alphabet, which is obtained from the latin alphabet by adding letters for brackets and commas, that is obtained by writing out each atom as a string. For example the triple

$$(\text{John}, \text{Eve}, \text{John}) \in \mathbb{A}^3$$

has an atomless representation of 15 letters, where the first letter is an opening bracket and the second letter is J. The atomless representation extends to words over a pof alphabet. Show that for every two-way pof automaton, the set

$$\{ \text{ atomless representation of } w \mid \text{ the automaton accepts } w \ \}$$

is in the complexity class L, i.e. deterministic logarithmic space.

Exercise 56. Consider the nondeterministic variant of the previous exercise. Show that the language of atomless representations is in NL, i.e. nondeterministic logarithmic space, and it can be complete for that class[2].

Exercise 57. Find a deterministic two-way register automaton which recognises the language

$$\{a_1 \cdots a_n : a_1, \ldots, a_n \text{ are distinct and } n \text{ is a prime number}\}.$$

Exercise 58. Consider nondeterministic two-way register automaton $\mathcal{A}$ with one register and labels Σ. Show that the following language is regular (in the usual sense, without data values):

$$\begin{aligned}\{b_1 \cdots b_n \in \Sigma^* \quad : \quad & \mathcal{A} \text{ accepts } (b_1, a_1) \cdots (b_n, a_n) \\ & \text{for some distinct atoms } a_1, \ldots, a_n \in \mathbb{A}\}.\end{aligned}$$

Exercise 59. Show that for every nondeterministic two-way pof automaton, there is an equivalent (one-way) alternating pof automaton with ε-transitions.

3.3 Circuits

In this group of problems, we consider the pof version of circuits. A pof circuit consists of:

1. a pof set X of variables;
2. a pof directed acyclic graph whose vertices are called *gates*;
3. a distinguished output gate;
4. an equivariant labelling from gates to $X + \{\vee, \wedge\}$.

[2] A corollary of Exercises 55 and 56 is that

$$\text{pof two-way automata determinize} \quad \Rightarrow \quad \text{NL} = \text{L}.$$

It is likely that the assumption is false, but no proof is known as of this time.

Given a valuation of the variables $X \to \{\text{true}, \text{false}\}$, the circuit computes a value in the natural way, which is the value of the output gate.

Exercises

Exercise 60. Show that satisfiability is undecidable for pof circuits.

Exercise 61. A circuit is called a *formula* if the directed acyclic graph is a tree. Show that every pof circuit can be transformed into an equivalent formula.

Exercise 62. A pof circuit is said to be in DNF form if the root gate is a disjunction, its children are conjunctions, and their children are variables or their negations. Show that satisfiability is decidable for circuits in DNF form.

Exercise 63. CNF normal form is defined dually to DNF normal form. Show that not every pof DNF circuit can be transformed into an equivalent pof CNF circuit.

3.4 Pushdown automata and context-free grammars

In this section, we discuss pof variants of pushdown automata[3] and context-free grammars. We show that basic results, such as equivalence of pushdown automata and context-free grammars, or decidability of emptiness, transfer easily to the pof setting. We also motivate the models by giving examples of automata and grammars that use atoms.

Definition 3.1. A *pof pushdown automaton* consists of

$$\underbrace{Q}_{\text{states}} \quad \underbrace{\Sigma}_{\substack{\text{input}\\\text{alphabet}}} \quad \underbrace{\Gamma}_{\substack{\text{stack}\\\text{alphabet}}} \quad \underbrace{q_0 \in Q}_{\text{initial state}} \quad \underbrace{\gamma_0 \in \Gamma}_{\substack{\text{initial stack}\\\text{symbol}}},$$

such that the initial state and initial stack symbol are equivariant, together with an equivariant transition relation

$$\delta \quad \subseteq \quad Q \times \overbrace{\Gamma^*}^{\text{popped}} \times \overbrace{(\Sigma \cup \epsilon)}^{\text{input}} \times Q \times \overbrace{\Gamma^*}^{\text{pushed}}$$

such that the popped and pushed strings have bounded length.

The language recognised by such an automaton is defined in the usual way. We assume that the automaton accepts via empty stack, i.e. a run is accepting if the last configuration (state, stack contents) has an empty stack.

Similarly, we can define a pof pushdown grammar.

[3]Context-free languages for infinite alphabets were originally introduced by Cheng and Kaminski (1998), who proved equivalence for register extensions of context-free grammars and pushdown automata. The generalisation to orbit-finite pushdown automata and context-free grammars is from Bojańczyk et al. (2014). See also Murawski et al. (2014); Clemente and Lasota (2015a,b).

Definition 3.2. A *pof context-free grammar* consists of

$$\underbrace{N}_{\text{nonterminals}} \qquad \underbrace{\Sigma}_{\substack{\text{input}\\ \text{alphabet}}} \qquad \underbrace{R \subseteq N \times (N+\Sigma)^*}_{\text{rules}} \qquad \underbrace{S \in N}_{\substack{\text{initial}\\ \text{nonterminal}}}$$

where the nonterminals and input alphabet are pof sets, the set of rules is equivariant and has bounded length, and the initial nonterminal is equivariant.

The language generated by a grammar is defined in the usual way.

Example 12. [Pushdown automaton for palindromes.] For a pof alphabet Σ, consider the language of palindromes, i.e. words which are equal to their reverse. This language is recognised by a pof pushdown automaton which works exactly the same way as the usual automaton for palindromes, with the only difference being that the stack alphabet Γ is now a pof set, namely Σ. For instance, in the case when $\Sigma = \mathbb{A}$, the automaton keeps a stack of atoms during its computation. The automaton has two control states: one for the first half of the input word, and one for the second half of the input word. As in the standard automaton for palindromes, this automaton uses nondeterminism to guess the middle of the word. □

Example 13. [Pushdown automaton for modified palindromes.] The automaton in Example 12 had two control states, which did not store any atoms. In some cases, it might be useful to have a set Q of control states that uses atoms. Consider the set of odd-length palindromes where the middle letter is equal to the first letter. A natural automaton recognising this language would be similar to the automaton for palindromes, except that it would store the first letter a_1 in its control state.

Another solution would be an automaton which keeps the first letter in every token on the stack. This automaton has a stack alphabet of $\Gamma = \Sigma \times \Sigma$, and after reading letters $a_1 \cdots a_n$ its stack is

$$(a_1, a_1), (a_1, a_2), \ldots, (a_1, a_n).$$

This automaton needs only two control states. Actually, using the standard construction, one can show that every orbit-finite pushdown automaton can be converted into one that has one control state, but a larger stack alphabet. □

The following example gives some motivation for studying orbit-finite pushdown automata.

Example 14. [Modelling recursive programs] Pushdown automata without atoms are sometimes used to model the behaviour of recursive programs with Boolean variables. By adding atoms, we can also model programs that have variables ranging over atoms. Consider a recursive function such as the following one (this program does not do anything smart):

```
function f(a: atom)
begin
    b:=read() // read an atom from the input
    if b != a then
```

```
        f(b)
        if b != read() then
            fail() // terminate the computation
end
```

The behaviour of this program can be modelled by a pof pushdown automaton. The input tape corresponds to the `read()` functions. The stack corresponds to the call stack of the recursive functions; the stack stores atoms since the functions take atoms as parameters. Since the only variables are atoms, the set of possible call frames (i.e. the stack alphabet) is a pof set. Pof pushdown automata could also be used to model more sophisticated behaviour, including mutually recursive functions and boolean variables. □

Theorem 3.3. *Pushdown automata recognize the same languages as context-free grammars. Furthermore, emptiness is decidable.*

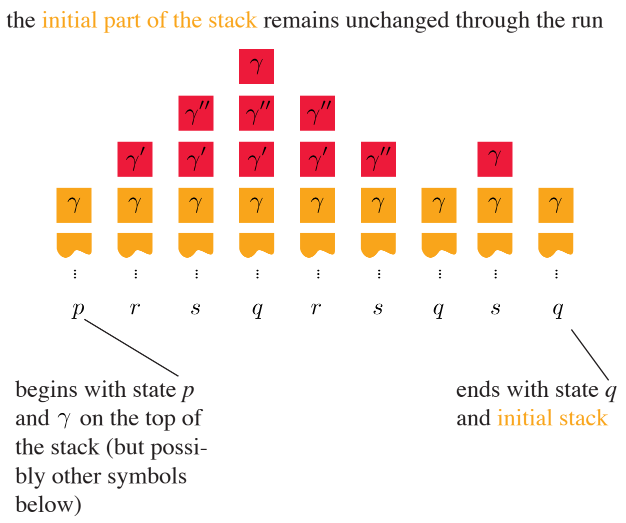
Proof. We just redo the classical constructions, which are so natural that they easily go through in the pof extension.

- *From a pushdown automaton to a context-free grammar.* Without loss of generality, we assume that each transition either: pops nothing and pushes one symbol; or pops one symbol and pushes nothing. We also assume that in every accepting run, the stack is nonempty until the last configuration. Every pushdown automaton can be transformed into one of this form, without changing the recognised language, by using additional states and ε-transitions. The transformation can be done in polynomial time, assuming that equivariant subsets are represented using formulas.

  Assuming that the pushdown automaton has the form discussed above, the corresponding grammar is defined as follows. The nonterminals are

  $$N = \underbrace{\{S\}}_{\text{an initial nonterminal}} + Q \times \Gamma \times Q.$$

  The language generated by a nonterminal (p, γ, q) is going to be the set of words which label runs of the following form:


the initial part of the stack remains unchanged through the run
p
r
s
q
r
s
q
s
q
begins with state p and γ on the top of the stack (but possibly other symbols below)
ends with state q and initial stack

To describe these runs, we use the following grammar rules. All the sets below are equivariant and have bounded length:

1. *Transitive closure.* For every $p, q, r \in Q$ and $\gamma \in \Gamma$, there is a rule

$$(p, \gamma, q) \to (p, \gamma, r)(r, \gamma, q).$$

2. *Push-pop.* For every transitions

$$\underbrace{(p, \epsilon, a, p', \gamma')}_{\text{push}} \qquad \underbrace{(q', \gamma', b, q, \epsilon)}_{\text{pop}}$$

there is a rule

$$(p, \gamma, q) \to a(p', \gamma', q')b.$$

3. *Starting.* For every transition that pops the initial stack symbol γ_0

$$\underbrace{(p, \gamma_0, a, q, \epsilon)}_{\text{pop}}$$

there is a rule

$$S \to (q_0, \gamma_0, p)a.$$

- *From a context-free grammar to a pushdown automaton.* The automaton keeps a stack of nonterminals. It begins with just the starting nonterminal, and accepts when all nonterminals have been used up. In a single transition, it replaces the nonterminal on top of the stack by the result of applying a rule. This automaton has one state (if we disregard the restriction that all transitions have to be either push or pop).

- *Emptiness is decidable.* We now show that emptiness is decidable. We use the context-free grammars, and the usual algorithm. This algorithm stores an equivariant subset of the nonterminals that are known to be nonempty (also known as productive nonterminals). Initially, the subset is empty. In each step, we add a nonterminal X to the subset if there is some rule in the grammar, where the left hand side has X, and the right hand side has only terminals and nonterminals that are already in the subsets. Because the set of rules is equivariant, in each step the subset is equivariant. Therefore, the subset can grow only in finitely many steps before stabilizing. The number of steps is at most the number of orbits in the set of nonterminals, which is at most exponential in the representation of the grammar.

□

Exercises

Exercise 64. Consider the following extension[4] of pof pushdown automata, where a new kind of transition is allowed:

$$q \xrightarrow{\text{fresh}(a)} p \qquad \text{for states } p, q \text{ and an input letter } a \in \mathbb{A}.$$

When executing this transition, the automaton reads letter a and changes state from q to p, but only under the condition that all atoms from the input letter a are fresh (i.e. do not appear in) with respect to every letter on the stack and the current state q. Show that emptiness is decidable.

Exercise 65. Consider the following higher-order variant of orbit-finite pushdown automata[5]. The automaton has a stack of stacks (one could also consider stacks of stacks of stacks, etc., but this exercise is about stacks of stacks). There are operations as in a usual pushdown automaton, which apply to the topmost stack. There is also an operation "duplicate the topmost stack" and an operation "delete the topmost stack". Show that emptiness is undecidable.

Exercise 66. Show a language that is generated by a pof context-free grammar, but not by any pof context-free grammar with a finite (not just pof) set of nonterminals.

Exercise 67. Show that emptiness for pof context-free grammars is ExpTime-complete.

Exercise 68. Show that if the set of terminals (i.e. the input alphabet), is finite (i.e. pof of dimension zero), then pof context-free grammars are the same as usual context-free grammars.

Exercise 69. Show that pof context-free grammars can be converted into Chomsky normal form, where all rules are of the form $X \to YZ$ with X, Y, Z nonterminals, or $X \to a$ with X a nonterminal and a a terminal.

3.5 Turing machines

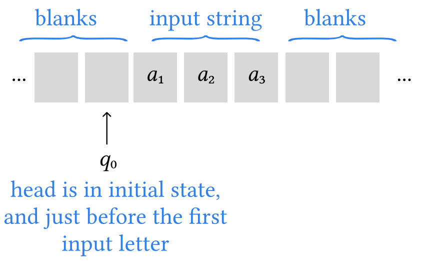
In this section, we discuss the pof version of Turing machines. A pof Turing machine is defined like a Turing machine, except that the set of states, and the alphabets are pof sets, and the transition function is equivariant.

[4] This extension is based on Murawski et al. (2014).
[5] This exercise is based on (Murawski et al., 2014, Section 6).

We assume that the reader is familiar with Turing machines, but we give a more detailed description of our modal to fix notation. The input alphabet Σ, the work alphabet Γ, and the set of states Q are all pof sets. We assume that the work alphabet contains the input alphabet, and there is some designated blank symbol

$$\text{blank} \in \Gamma \setminus \Sigma$$

that is equivariant. One could have a two tape model, but since we will not be interested in machines with sublinear space (e.g. logarithmic space), we use the one tape model for simplicity. In this model, there is one tape that is read-write, which initially contains the input string, and which is also used for storing intermediate computations. The tape is infinite in both directions. A configuration of the Turing machine consists of the tape contents (i.e. each cell has some letter from the work alphabet), a head position (which points to some cell), and a state from Q. The initial configuration looks like this:


blanks
input string
blanks
...
a_1
a_2
a_3
...
q_0
head is in initial state, and just before the first input letter

The behaviour of the Turing machine is specified by its transition function, which is an equivariant function of type

$$\underbrace{Q \times \Gamma}_{\text{current state and letter under the head}} \quad \to \quad \{\text{accept, reject}\} + (Q \times \underbrace{\{\text{left, stay, right}\}}_{\text{head movement}} \times \underbrace{\Gamma}_{\text{what is written on the tape}}).$$

Using the transition function, the machine computes a new configuration in the expected way, or it accepts/rejects. This leads to a computation (a sequence of configurations), which is either finite – when an accept/reject instruction is executed – or infinite. In a nondeterministic machine, instead of a function we have a binary relation, and an input string might have more than one computation. The language *recognized by* a (possibly nondeterministic) Turing machine is the set of words that have at least one accepting computation.

Example 15. [A Turing machine checking that all letters are different] Consider the equality atoms. Assume that the input alphabet is $\mathbb{A}$. We show a deterministic Turing machine which accepts words where all letters are distinct. The idea is that the machine iterates the following procedure until the tape contains only blank symbols: if the first non-blank letter on the tape is a, replace it by a blank and load a into the state, scan the word to check that a does not appear again (if it does appear again, then reject immediately), and after reading the entire word go back to the beginning

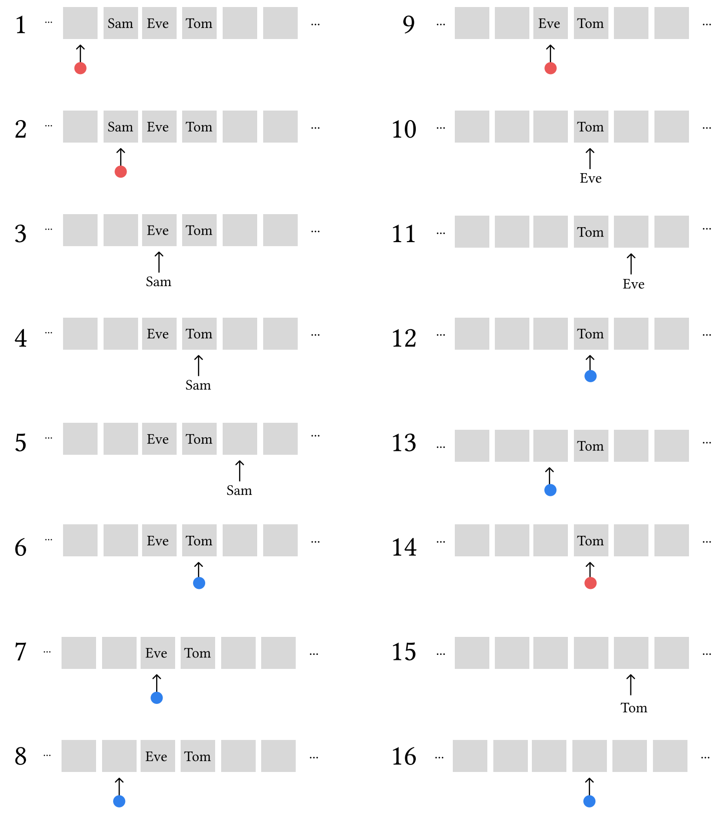


Figure 3.1: An accepting run of the Turing machine from Example 15.

of the tape. If the tape is entirely erased, then accept. The sets of states is $\mathbb{A}$, plus two extra states for the scanning, which are depicted using red and blue in Figure 3.1. $\square$

Having defined Turing machines, we get the usual notions of semi-decidability (the language of some Turing machine) and decidability (the language of some Turing machine that always halts). The Church-Turing Thesis states that there is only one notion of decidable language, which is captured by Turing machines. Does introducing atoms give a violation of this thesis? What does that even mean? One way of answering this question is to relate computation with atoms to the classical notion of computation without atoms. A word with atoms can be represented by a word without atoms, by writing down the atoms, such as "John" or "Mary" using a finite alphabet. Under such a representation, we get a usual word over a finite alphabet, which can be used as an input for the classical atom-free models of computation. We will show later in this section that Turing machines with atoms can be simulated by machines without atoms, and vice versa, and thus the two models of computation are essentially equivalent. Using this equivalence, we can carry over to the atom world classical results, such as equivalence of deterministic and nondeterministic machines in the presence of unbounded computation time. However, in Chapter 8 we will discover a twist in the story – if we use a more general notion of pof sets, namely (not necessarily polynomial) orbit-finite sets, then some of the equivalences break down, for example nondeterministic machines are not equivalent to deterministic ones. Before we get to the twist, let us tell the un-twisted story, which involves pof sets.

We begin by formalizing what it means to "write down" an atom.

Definition 3.4. A *representation* of the atoms is any function

$$r : 2^* \to \mathbb{A}$$

which is surjective (every atom has at least one representation) and such that one can decide if two strings represent the same atom.

An alternative choice of definition would require the function to be bijective, which would also give a simpler algorithm for deciding if two strings represent the same atom. We choose to use the above definition because it will more naturally extend to atoms with more structure.

Suppose that we have a representation of the atoms. We can extend it to represent elements of a pof set: an element of such a set is described by indicating which component $\mathbb{A}^{d_i}$ is used, followed by a representation of the d_i atoms in the tuple. We can also extend the representation to describes words over a pof set, by using separator symbols between the letters. Summing up, once we know how to represent atoms with atom-less strings, we can do the same for words over a pof alphabet. In the following theorem, we show that the atom version of Turing machines correspond to the usual Turing machines without atoms, via the representation. Furthermore, the choice of representation is not important.

Theorem 3.5. *The following conditions are equivalent for every language $L \subseteq \Sigma^*$ over a pof alphabet:*

1. *L is recognised by a deterministic pof Turing machine;*

2. *L is recognised by a nondeterministic pof Turing machine;*

3. *L is equivariant and for every representation r,*

$$\{ w \mid w \text{ represents, under } r\text{, some word in } L \}$$

   is recognised by a nondeterministic Turing machine;

4. *as in the previous item, but the machine is deterministic;*

5. *as in the previous item, but the representation r is quantified existentially.*

Proof. The implications 1 ⇒ 2 and 3 ⇒ 5 are trivial. For the implication 2 ⇒ 3, we use a straightforward simulation, where the simulating machine stores representations of the simulated Turing machine. Implication 3 ⇒ 4 is the classical fact that, without atoms, deterministic and nondeterministic Turing machines compute the same languages. The interesting implication is 5 ⇒ 1, which is proved below.

Let r be a representation as in the assumption 5, and let us write $s : 2^* \to \Sigma^*$ for the extension of this representation to words over the alphabet Σ. The main idea is that this representation can be inverted, up to atom permutations, by a deterministic pof Turing machine. This is proved in the following lemma, which we call the deatomisation lemma, because it transforms a word with atoms into a representation without atoms. (We use the standard notion of Turing machines for computing a function – there is an output tape, the machine always halts, and the contents of the output tape is the output of the function.)

Lemma 3.6 (Deatomisation)**.** *There is a function $f : \Sigma^* \to 2^*$, computed by a deterministic pof Turing machine, such that every word $w \in \Sigma^*$ is in the same orbit as $s(f(w))$.*

Before proving the above lemma, we use it to prove the implication 5 ⇒ 1. Using the atom-less Turing machine from the assumption, we know that there is a Turing machine that in puts $w \in \Sigma^*$, and checks if $s(f(w))$ belongs to the language. By the assumption that the language is equivariant, this is the same as checking if w belongs to the language. It remains to prove the Deatomisation Lemma.

Proof. Consider some computable enumeration of representations of the atoms, i.e. an infinite list of strings in 2^* which is computed by a Turing machine, and such that every atom is represented by exactly one string on the list. Such an enumeration can be found for every representation.

Using this enumeration, we define the deatomisation function f from the statement of the lemma. Consider an input string $w \in \Sigma^*$. The string w contains some atoms, and these atoms can be listed in the order of their first appearance in the string. For each of these atoms, we choose a string representation according to the enumeration in the previous paragraph, i.e. the atom with the leftmost appearance gets the first representation, the atom with the second leftmost appearance gets the second representation, and so on. We then apply this choice consistently to the entire string. All of this can be implemented by a deterministic pof Turing machine. □

This completes the proof of the implication 5 ⇒ 1, and therefore also of the theorem. We would like to remark that the proof of the Deatomisation Lemma given above will fail for more general input alphabets which will be considered later in the book. The issue is that the proof above refers to the order of appearance of atoms in the input string, and this will no longer be meaningful for some input alphabets, such as unordered pairs of atoms, which will be legitimate alphabets in the more general settings. □

Exercises

Exercise 70. Give a deterministic pof Turing machine for the language

$$\{ w\#v \mid \ w, v \in \mathbb{A}^* \text{ are in the same orbit } \}.$$

Exercise 71. Consider a two-tape model, which has a work tape with a separate head. Show that for every pof Turing machine, deterministic or not, there is an equivalent one (in the two-tape model) where the state space has atom dimension zero. (Attention: this will no longer be true for orbit-finite sets, which are not polynomial orbit-finite.)

Exercise 72. Show that the answer to the previous problem is negative in the one-tape model.

Chapter 4

Orbit-finite sets

In this chapter, we define two of the main notions discussed in this book, namely finite supports and orbit-finiteness. The motivating example is minimization of deterministic automata.

Minimization of automata. Suppose that we want to minimize a deterministic pof automaton. The classical construction restricts the state space to the subset of reachable states, and then quotients this subset under the Myhill-Nerode equivalence relation

$$q \sim p \qquad \overset{\text{def}}{=} \qquad qw \in F \Leftrightarrow pw \in F \text{ for every word } w \in \Sigma^*.$$

For pof automata, we cannot apply this construction, since pof sets are not closed under taking subsets, or quotients. This is illustrated in the following example.

Example 16. Consider the language

$$L = \{ w \in \mathbb{A}^* \mid w \text{ uses at most two different atoms } \}.$$

This language is recognised by a pof automaton, which has a state space

$$\underbrace{\mathbb{A}^0 + \mathbb{A}^1 + \mathbb{A}^2}_{\text{atoms seen so far}} \quad + \quad \underbrace{\mathbb{A}^0}_{\text{reject}}.$$

This automaton is not minimal, because the states from $\mathbb{A}^2$ store the order in a pair (a, b), which is not needed for the language. To make this automaton minimal, we should use a state space of the form

$$\mathbb{A}^0 + \mathbb{A}^1 + \underbrace{\binom{\mathbb{A}}{2}}_{\substack{\text{sets of exactly}\\ \text{two atoms}}} + \mathbb{A}^0$$

This kind of feature is not available in pof automata. More formally, we will show that for every deterministic pof automaton for this language, the states after reading the input words *ab* and *ba* must be different (while they should be equal in a minimal automaton). Indeed, if the same state q would be reached after reading both *ab* and *ba*, then this state would satisfy

$$\pi(q) = q \qquad \text{where } \pi \text{ is the atom permutation that swaps } a \text{ and } b.$$

If an element in a pof set satisfies the above condition, then it cannot use the atoms a and b. This cannot happen in an automaton that recognises the language. □

If we would like our automata to be closed under minimization, then they should support taking subsets and quotients. A quick and dirty solution is to simply add these two features. We will describe this solution in Section 4.1. Later in this chapter, we will show that the solution is not so dirty after all.

Exercises

Exercise 73. Define the *orbit count* of a deterministic pof automaton to be the number of orbits of reachable states. For a language, we can consider the set of deterministic pof automata that recognise it, and have minimal orbit count for this property. Show that this set can contain automata that are non-isomorphic. Here, an isomorphism between two automata is an equivariant bijection

$$\text{reachable states of } \mathcal{A} \xrightarrow{f} \text{reachable states of } \mathcal{B}$$

such that for every input word, applying f to the state of $\mathcal{A}$ after reading a word gives the state of $\mathcal{B}$ after reading the same word.

4.1 Quotiented pof sets

To formalize subsets and quotients, we use partial equivalence relations. A *partial equivalence relation* on a set X is defined to be a binary relation that satisfies

$$\underbrace{x \sim y \quad \Rightarrow \quad x = y}_{\text{symmetric}} \quad \text{and} \quad \underbrace{x \sim y \text{ and } y \sim z \quad \Rightarrow \quad x \sim z}_{\text{transitive}}.$$

This is like an equivalence relation, but the missing axiom is reflexivity $x \sim x$. Defining a partial equivalence relation is the same as indicating some subset (the elements that are equivalent to themselves), and then defining a (total) equivalence relation on the subset. Therefore, partial equivalence relations subsume both subsets and quotients. We write $X_{/\sim}$ for the set of equivalence classes of $\sim$, and we call this a *quotient* of the original set[1].

Definition 4.1. A *quotiented pof set*, qpof for short, is any quotient $X_{/\sim}$ of a pof set X by a partial equivalence relation $\sim$ that is equivariant.

[1]A more complete name would be *quotiented subset*, but we use the shorter name, despite the fact that it subsumes both quotients and subsets.

Example 17. Quotiented pof sets subsume both subsets and quotients. Let us begin by illustrating subsets. Every equivariant subset $Y \subseteq X$, can be seen as a quotiented pof set, which corresponds to the partial equivalence relation

$$x \sim y \quad \text{if} \quad x = y \text{ and } x \in Y.$$

One example of such a set is the set

$$\mathbb{A}^{(d)} \stackrel{\text{def}}{=} \{\ (a_1, \ldots, a_d) \in \mathbb{A}^d \mid\ a_1, \ldots, a_d \text{ are pairwise different }\ \},$$

which we call the *set of non-repeating tuples*. We will use these sets a lot. □

Example 18. In Example 16, we discussed the set

$$\binom{\mathbb{A}}{2}.$$

This is an example of a quotiented pof set. It is the quotient of $\mathbb{A}^2$ under the partial equivalence relation defined by

$$(a, b) \sim (a', b') \quad \text{if} \quad \{a, b\} = \{a', b'\} \wedge a \neq b.$$

□

Similarly to the case of pof sets, all structure on quotiented pof sets will be required to be equivariant. Equivariance is defined in the same way as for subsets of pof sets: membership in the set must be stable under applying atom permutations. Let us define this more formally.

Definition 4.2. [Equivariant subset of a qpof set] A subset

$$Y \subseteq X_{/\sim}$$

of a quotiented pof set is called *equivariant* if for every element of $X_{/\sim}$, which is an equivalence class $[x]_\sim$ of some element $x \in X$, we have

$$[x]_\sim \in Y \quad \Leftrightarrow \quad \underbrace{\pi([x]_\sim)}_{\substack{\text{image of } \pi\text{, when applied to the set} \\ \text{of elements that are in the equivalence class } [x]}} \in Y \quad \text{for every atom permutation } \pi.$$

In the above definition, when we apply an atom permutation to an element of an equivalence class, we take the image of all elements in the equivalence class. An alternative approach would be to apply the atom permutation to a representative of the equivalence class, and then take the equivalence class of the image. This would give the same effect, since applying atom permutations commutes with taking equivalence classes, as long as $\sim$ is equivariant:

$$\underbrace{\pi([x]_\sim)}_{\substack{\text{first take the equivalence class,} \\ \text{then apply the atom permutation}}} \quad = \quad \underbrace{[\pi(x)]_\sim}_{\substack{\text{first apply the atom permutation,} \\ \text{then take the equivalence class}}}.$$

Example 19. Consider the quotiented pof set

$$\binom{\mathbb{A}}{2}$$

from Example 18, which is obtained by a quotient of $X = \mathbb{A}^2$ with respect to a certain equivalence relation $\sim$. An element of the quotient is an equivalence class of pairs, as in the following example:

$$\underbrace{(\text{John}, \text{Mary})}_{\text{element } x \in X} \qquad \underbrace{\{(\text{John}, \text{Mary}), (\text{Mary}, \text{John})\}}_{\text{its equivalence class in } X_{/\sim}}. \tag{4.1}$$

If we apply to the above equivalence class the atom permutation that swaps John and Mary, then we get the same equivalence class. □

In Definition 4.2, we defined equivariance for subsets of quotiented pof sets. Since quotiented pof sets are closed under taking products $X \times Y$, we can also talk about equivariant relations on quotiented pof sets, by considering equivariant subsets of the product $X \times Y$. As a special case of relations, we can discuss equivariant functions between quotiented pof sets.

Example 20. We will show that there is no equivariant function

$$f : \binom{\mathbb{A}}{2} \to \mathbb{A}.$$

In this proof, we treat elements of $\binom{\mathbb{A}}{2}$ as sets $\{a, b\}$ of size two, although formally they are defined to be equivalence classes of ordered pairs. Consider some hypothetical function f, and some input-output pair

$$f(\{a, b\}) = c.$$

We first rule out the case that $c \notin \{a, b\}$. If this were the case, then we could apply an atom permutation to the input-output pair that moves c without moving a and b, and get a violation of functionality. Let us now rule out the case $c \in \{a, b\}$. If this were the case, then we could apply an atom permutation that swaps a and b; this atom permutation would not change the input, but it would change the output, and so it would also be a violation of functionality. □

Quotiented pof sets are a solution to the problem of minimization of deterministic pof automata. Indeed, if we have some pof automaton, then we can define a partial equivalence relation on its states by

$$p \sim q \quad \overset{\text{def}}{=} \quad p \text{ and } q \text{ are both reachable, and accept the same words.}$$

By equivariance of the automaton, this is an equivariant partial equivalence relation. Therefore, the quotient $Q_{/\sim}$ is a quotiented pof set. As usual, one can define a quotient automaton $\mathcal{A}_{/\sim}$ which recognises the same language as the original automaton $\mathcal{A}$. This automaton is well-defined (i.e. its structure is equivariant), and it is minimal in the appropriate sense. More details will be provided later in this chapter, when we prove Myhill-Nerode Theorem.

Exercises

Exercise 74. How many equivariant functions are there of type

$$\binom{\mathbb{A}}{d} \to \binom{\mathbb{A}}{e}$$

for $d, e \in \{0, 1, \ldots\}$?

Exercise 75. Find a deterministic qpof automaton for the language

$$\{\ w \in \binom{\mathbb{A}}{3}^* \mid \text{ some atom } a \text{ appears in all letters }\ \}.$$

Exercise 76. Find a deterministic qpof automaton for the language

$$\{\ w \in \binom{\mathbb{A}}{3}^* \mid \text{ there are at most distinct 5 atoms used in the word }\ \}.$$

Exercise 77. Find a deterministic qpof automaton for the language

$$\{\ w \in \binom{\mathbb{A}}{3}^* \mid \begin{array}{l}\text{some atom from the set in the first letter appears}\\ \text{an even number of times in the remaining letters}\end{array}\ \}.$$

Exercise 78. Find a deterministic qpof automaton for the language:

$$\{\ w \in \binom{\mathbb{A}}{2}^* \mid \text{ there exist } a, b \in \mathbb{A} \text{ such that every letter in } w \text{ intersects } \{a, b\}\ \}.$$

Consider the first two letters in the input string that are not equal to each other, which are sets x and y of size two. If:

1. If the sets are disjoint, then the only candidates for a, b are from $x \cup y$. Then, we can use the same kind of solution as in Exercise 77.
2. Otherwise, the only candidates for a and b are:
   (a) the two atoms that are in the symmetric difference $(x \setminus y) \cup (y \setminus x)$; or
   (b) the atom in the intersection, and some other atom.

4.2 Orbit-finiteness

One could worry that quotiented pof sets are a hack that fixes the minimization issue for automata, but does not have proper theoretical justification. In this section, we ease such worries, by giving a more semantic concept, namely orbit-finite sets, and showing that they are exactly the same as quotiented pof sets.

Group actions. The semantic characterization will use two of the central notions in this book: finite supports, and orbit-finiteness. These notions are defined in terms of group actions, so begin by defining those.

Definition 4.3. An action of a *group* G on a set X is defined to be a function

$$G \times X \to X,$$

which we denote by

$$(\pi, x) \mapsto \pi(x),$$

that satisfies the following two axioms:

$$\underbrace{(\pi \circ \sigma)(x)}_{\substack{\text{compose in the group}\\ \text{and then apply the action}}} = \underbrace{\pi(\sigma(x))}_{\substack{\text{apply the action}\\ \text{two times}}} \qquad \text{and} \qquad \underbrace{1(x) = x}_{\substack{\text{the group identity}\\ \text{does not move anything}}}.$$

In this book, the group G will always be permutations of the atoms, although in later chapters will restrict the permutations to those that respect some extra structure on the atoms, such as a linear order.

Example 21. Sets constructed using atoms are naturally equipped with an action of the group of atom permutations. For example, the set $\mathbb{A}^d$ is equipped with the action defined by

$$\pi(a_1, \ldots, a_d) = (\pi(a_1), \ldots, \pi(a_d)).$$

We have been using this group action implicitly in the previous chapters. The action also extends to other sets, such as $\mathbb{A}^*$ or the powerset $\mathsf{P}\mathbb{A}$. In the case of the powerset, we use the image, i.e.

$$\pi(X) = \{ \pi(a) \mid a \in X \}.$$

$\square$

Finite supports. We now introduce the fundamental notion of supports. The general idea is that the support of an element x in a set X consists of the atoms that are needed to describe it. Since the notion is defined in abstract terms of group actions, it will be useful to keep the following examples in mind while reading the formal definition.

Set X	Element $x \in X$	Support of x
$\mathbb{A}^2$	(John, Mary)	John, Mary
$\mathsf{P}\mathbb{A}$	$\{ a \in \mathbb{A} \mid a \neq \text{John} \}$	John
$\mathsf{P}\mathbb{A}$	$\mathbb{A}$	$\emptyset$

Definition 4.4 (Supports)**.** Consider a set X that is equipped with an action of atom permutations. An element $x \in X$ is said to be *supported* by a list of atoms $a_1, \ldots, a_n$ if

$$\underbrace{\pi(a_1) = a_1 \wedge \cdots \wedge \pi(a_n) = a_n}_{\text{we write this as } \pi(\bar{a}) = \bar{a}, \text{ where } \bar{a} \text{ is the list } a_1, \ldots, a_n} \quad \Rightarrow \quad \pi(x) = x \tag{4.2}$$

holds for every atom permutation π. We say that x is *finitely supported* if it is supported by some finite list of atoms.

Before continuing, let us remark on the notation. In the above definition, we use finite lists for supports. The order of atoms in this list, or their repetitions, are not relevant for the notion of support, since they do not affect the assumption of the implication (4.2). Therefore, the only relevant information is the set of atoms that appears on this list, which is why many authors present the support as a finite set of atoms, and not a list. If one uses sets $\{a_1, \ldots, a_n\}$ for supports, then one should remember that the assumption in implication (4.2) is not

$$\pi(\{a_1, \ldots, a_n\}) = \{a_1, \ldots, a_n\},$$

which is a weaker assumption, because it allows π to swap atoms inside the set.

Example 22. Let us discuss which elements of the powerset $\mathsf{P}\mathbb{A}$ are finitely supported. If $x \in \mathsf{P}\mathbb{A}$ is finite, then it is finitely supported, namely by any list that contains all atoms in this set. A similar result holds for co-finite sets, i.e. sets obtained by removing finitely many atoms. For example, if we take

$$x = \mathbb{A} \setminus \{\text{John}, \text{Mary}\},$$

then x is supported by the atoms John, Mary, because any atom permutation that fixes both John and Mary will map the set x to itself, even if it permutes the other atoms. Therefore, all finite and co-finite elements in $\mathsf{P}\mathbb{A}$ are finitely supported.

The remaining elements of the powerset are not finitely supported. Let us prove this formally. Suppse that $x \in \mathsf{P}\mathbb{A}$ is neither finite nor cofinite, and take some hypothetical finite support $\bar{a}$. There must be some atoms b, c that are not in this finite support, and such that $b \in x$ and $c \notin x$. Take the atom permutation that swaps b and c, and leaves all other atoms fixed. This permutation fixes the support, but moves x, which contradicts the definition of finite support. □

Orbits and orbit-finiteness. We now introduce the second fundamental notion of this book, which is orbit-finiteness. This idea was already discussed informally before, but we now give a formal definition in terms of group actions. Consider a set X equipped with an action of atom permutations. An *orbit* of X is defined to be any subset of the form

$$\{\ \pi(x) \mid \ \pi \text{ is an atom permutation } \},$$

for some $x \in X$. We will be interested in sets that have finitely many orbits, and where all elements are finitely supported. (In the exercises, we explain why these two conditions are necessary for the theory to work.)

Definition 4.5 (Orbit-finite set)**.** An *orbit-finite set* is defined to be a set X, together with an action of the group of atom permutations, such that every element $x \in X$ has finite support, and there are finitely many orbits.

Example 23. Recall the set $\mathbb{A}^{(d)}$ that consists of non-repeating d-tuples of atoms. This set has one orbit. The set $\mathbb{A}^3$ is orbit-finite. Like any orbit-finite set, this set decomposes into a disjoint union of one-orbit sets, which in this case uses five orbits:

$$\underbrace{\mathbb{A}^1}_{\substack{\text{all atoms}\\\text{are equal}}} \quad + \quad \underbrace{\mathbb{A}^{(2)} + \mathbb{A}^{(2)} + \mathbb{A}^{(2)}}_{\substack{\text{two atoms are equal, and}\\\text{one is diffferent, which}\\\text{can happen in three ways}}} \quad + \quad \underbrace{\mathbb{A}^{(3)}}_{\substack{\text{all atoms}\\\text{are different}}}.$$

□

Example 24. The powerset $\mathsf{P}\mathbb{A}$ is not orbit-finite, because sets of different finite size are in different orbits. □

Example 25. Consider the set $\mathbb{A}^{(3)}$ of non-repeating triples. On this set, consider the equivalence relation that identifies triples modulo cyclic shift:

$$(a, b, c) \sim (b, c, a) \sim (c, a, b) \quad \text{for all } a, b, c \in \mathbb{A}.$$

Consider the quotient of under this equivalence relation. This is a one-orbit set. It is also an example of a quotiented pof set, since we can view $\sim$ as a partial equivalence relation on (possibly repeating) triples that removes the repeating triples. □

The following theorem shows that orbit-finite sets are exactly the same as the quotiented pof sets.

Theorem 4.6. *Let X be a set equipped with an action of atom permutations. Then X is orbit-finite if and only if it admits an equivariant bijection with a quotiented pof set.*

Proof. The bottom-up implication is easy, so we only prove the top-down implication. If a set is finitely many orbits, then it is a disjoint union of one-orbit sets. Since quotiented pof sets are closed under disjoint union, it is enough to prove the top-down implication for a one-orbit set X. Choose some $x \in X$. By assumption on finite supports, this element is supported by some atoms $a_1, \ldots, a_d$. Define a partial function from $\mathbb{A}^d$ to X as follows: it consists of input-output pairs

$$\pi(a_1, \ldots, a_d) \mapsto \pi(x),$$

where π ranges over atom permutations. By definition of supports, this is a partial function, i.e. if the inputs are equal then the outputs are equal. This function is surjective, since its range is the orbit of x, and we have assumed that X is a one-orbit set. The function defines a bijection between X and the inputs, quotiented by the equivalence relation "same output". □

A Myhill-Nerode Theorem. We now use the theory developped above to prove an orbit-finite version of the Myhill-Nerode Theorem. In the classical, finite version, the theorem says that a language is regular if and only if its syntactic congruence has finite index (i.e. finitely many equivalence classes), with the syntactic congruence defined as follows.

Definition 4.7. The *syntactic congruence* of a language $L \subseteq \Sigma^*$ is the equivalence relation on Σ^* that identifies two words w and w' if they cannot be distinguished by any future, i.e.

$$wv \in L \Leftrightarrow w'v \in L \qquad \text{for every } v \in \Sigma^*.$$

The syntactic congruence can be applied also for infinite alphabets, such as quotiented pof sets. We will show that in this case, orbit-finite index will correspond to being recognised by a deterministic quotiented pof automaton.

We begin by explaining what it means for the syntactic congruence to have orbit-finite index. The first observation is that the syntactic congruence is equivariant, as long as the language itself is equivariant.

Lemma 4.8. *Let Σ be a quotiented pof set. If a language $L \subseteq \Sigma^*$ is equivariant, then the same is true for its syntactic congruence, i.e.*

$$w \sim w' \quad \Leftrightarrow \quad \pi(w) \sim \pi(w')$$

for every words $w, w' \in \Sigma^*$ and atom permutations π.

Proof. If the words w and w' can be distinguished by some future v, then the words $\pi(w)$ and $\pi(w')$ can be distinguished by the future $\pi(v)$, thanks to equivariance of concatenation and of the language L. □

We now explain why the notion of orbits is applicable to the quotient of Σ^* under the syntactic congruence. Consider some set X with an action of atom permutations (we care about $X = \Sigma^*$ in this example). Let $\sim$ be an equivalence relation on this set that is equivariant (we care about the syntactic congruence). The quotient $X_{/\sim}$ is also equipped with an action of atom permutations, with the action defined by

$$\text{equivalence class of } x \quad \overset{\pi}{\mapsto} \quad \text{equivalence class of } \pi(x). \tag{4.3}$$

By equivariance of $\sim$, it is easy to check that this action is well-defined, i.e. it does not depend on the choice of representative x in the equivalence class. Thanks to the above observations, if Σ is a quotiented pof set, and $L \subseteq \Sigma^*$ is an equivariant language, then we can equip the quotient

$$\Sigma^*_{/\text{syntactic congruence of } L}$$

with an action of atom permutations, and therefore we can ask if this quotient is orbit-finite. As the following theorem shows, orbit-finiteness is equivalent to recognizability by a deterministic quotiented pof automaton.

Theorem 4.9. *The following conditions are equivalent for an equivariant language $L \subseteq \Sigma^*$ over a quotiented pof alphabet Σ:*

1. *L is recognised by a deterministic quotiented pof automaton;*
2. *the quotient of Σ^* under the syntactic congruence of L is orbit-finite.*

Proof. We use essentially the same proof as in the classical Myhill-Nerode Theorem, except that we use "orbit-finite" instead of "finite".

1 ⇒ 2 Let us first show that if L is recognised by a deterministic quotiented pof automaton, then the quotient from item 2 is orbit-finite. Let Q be the reachable states of the automaton. If two words give the same state of the automaton, then they must be equivalent under the syntactic congruence. This gives us a function from Q to the quotient. This function is surjective, since every word gives some state, and it is easily seen to be equivariant. Therefore, we can deduce orbit-finiteness of the quotient by applying the following straightforward lemma.

Lemma 4.10. *Let $f : X \to Y$ be a surjective equivariant function between two sets equipped with actions of atom permutations. If X is orbit-finite, then so is Y.*

Proof. Every orbit of X is mapped to an orbit of Y. □

2 ⇒ 1 We use the standard syntactic automaton whose state space is the quotient, and whose transition function is given by

$$\text{equivalence class of } w \quad \xrightarrow{a} \quad \text{equivalence class of } wa.$$

We will justify that this is indeed a quotiented pof automaton. Directly from the definition of the action on the quotient described in (4.3), we deduce that quotienting preserves finite supports: if a tuple of atoms supports a word $w \in \Sigma^*$, then the same tuple supports its equivalence class under syntactic congruence. Therefore, every element in the quotient has a finite support. By Theorem 4.6, the quotient is isomorphic to a quotiented pof set.

□

In the theorem above, we use quotiented pof sets. What about (non-quotiented) pof sets, as discussed at the beginning of this book? If the input alphabet is non-trivally quotiented, then we will also need quotients for the state space of the automaton, as explained in the following example.

Example 26. Consider the input alphabet

$$\Sigma = \binom{\mathbb{A}}{2},$$

and a deterministic (non-quotiented) pof automaton. We claim that in this automaton, all reachable states will have atom dimension zero, i.e. they will come from atom-free components $\mathbb{A}^0$. To see why this is true, we use the following observation.

Lemma 4.11. *Let $d > 0$. There is no equivariant function*

$$f : \binom{\mathbb{A}}{2} \to \mathbb{A}^d.$$

Thanks to the observation in the above lemma, if the current state is of atom dimension zero, then the next state also has this property. Therefore, the reachable states of the automaton have atom dimension zero. This will preclude recognizing any language that depends on atoms in any way. □

The above example shows that we may need quotients if the input alphabet has quotients. But what if the alphabet does not have quotients, i.e. it is a pof set? As we will see later in this chapter, the Myhill-Nerode Theorem does hold in this case, because we have an implication

$$\begin{array}{c}\text{recognised by a quotiented pof automaton and input alphabet is a pof set}\\ \Downarrow \\ \text{recognised by a pof automaton}\cdot\end{array}$$

However, proving this implication will require developping some extra theory, namely least supports.

Exercises

Exercise 79. Show that a tuple $\bar{a}$ supports x if and only if

$$\pi(\bar{a}) = \sigma(\bar{a}) \quad \text{implies} \quad \pi(x) = \sigma(x) \qquad \text{for every atom automorphisms } \pi, \sigma.$$

Exercise 80. Find all equivariant binary relations on $\mathbb{A}$.

Exercise 81. Show that a function $f : X \to Y$ is equivariant if and only if the following diagram commutes for atom permutation π:

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "\pi"'] & Y \arrow[d, "\pi"] \\ X \arrow[r, "f"'] & Y \end{tikzcd}$$

Exercise 82. Consider an enumeration $a_1, a_2, \ldots$ of some countably infinite set A. Define the distance between two permutations of A to be $1/n$ where a_n is the first argument where the permutations disagree. Let X be a countably infinite set equipped with an action of permutations of the equality atoms. Show that all elements of X are finitely supported if and only if

$$\underbrace{\pi}_{\text{permutation of } \mathbb{A}} \qquad \mapsto \qquad \underbrace{(x \mapsto \pi(x))}_{\text{permutation of } X}$$

is a continuous mapping, and that this continuity does not depend on the choice of enumerations of $\mathbb{A}$ or X.

Exercise 83. Show a counterexample, in the equality atoms, to the converse implication from Exercise 112. In other words, show a set which is not orbit-finite, but where every tuple of atoms supports finitely many elements.

Exercise 84. Assume the equality atoms. Let $R \subseteq \mathbb{A}^{n+k}$ be a finitely supported relation which is total in the following sense: for every $\bar{a} \in \mathbb{A}^n$ there is some $\bar{b} \in \mathbb{A}^k$ such that $R(\bar{a}\bar{b})$. Show that there is a finitely supported function $f : \mathbb{A}^n \to \mathbb{A}^k$ whose graph is contained in R.

Exercise 85. Show that in the equality atoms (actually, under any oligomorphic atoms), every orbit-finite is Dedekind finite[2], i.e. does not admit a finitely supported bijection with a proper subset of itself.

Exercise 86. Show that in the equality atoms, there is a set that is not orbit-finite, but Dedekind finite in the sense from Exercise 85.

Exercise 87. Call a family of sets *directed* if every two sets from the family are included in some set from the family. Consider the equality atoms. Show that a set with atoms X is finite (in the usual sense) if and only if it satisfies: for every set with atoms $\mathcal{X} \subseteq \mathsf{P}X$ which is directed, there is a maximal element in $\mathcal{X}$.

Exercise 88. Call a family $\mathcal{X}$ of sets *uniformly supported*[3] if there is some tuple of atoms which supports all elements of $\mathcal{X}$. Assume that the atoms are oligomorphic. Show that a set X is orbit-finite if and only if: (*) there is a maximal element in every set of atoms $\mathcal{X} \subseteq \mathsf{P}X$ which is directed and uniformly supported.

Exercise 89. Show the following variant of König's lemma. If a tree has orbit-finite branching and arbitrarily long branches, then it has an infinite branch.

4.3 Least supports

An element $x \in X$ might have different supports. For example, we can add atoms to a support, and it will still be a support. In this section, we show that adding useless atoms to the support is the only phenomenon that can arise, because there is a least support[4].

Theorem 4.12 (Least Support Theorem)**.** *Let X be a set with an action of atom permutation. If $x \in X$ has some finite support, then one can find atoms $a_1, \ldots, a_d$ that support x, and such that every finite support of x contains all atoms $a_1, \ldots, a_d$.*

Another way of stating the above theorem is that finite supports are closed under intersection, if they are viewed as sets (and not lists). It is important that we consider finite supports. For example, any atom a is supported by the infinite set $\mathbb{A} - \{a\}$, since fixing this set is the same as fixing a. The intersection of the two supports $\{a\}$ and $\mathbb{A} - \{a\}$ is empty, but a does not have empty support.

Proof of the Least Support Theorem. It is enough to prove the theorem in the case when X has one orbit; this is because every other set is a disjoint union of (possibly infinitely

[2]This exercise is inspired by Blass (2013).

[3]This exercise is inspired by (Pitts, 2013, Section 5.5).

[4]The Least Support Theorem was first proved in (Gabbay and Pitts, 2002, Proposition 3.4). A generalisation of this theorem, for other kinds of atoms, can be found in (Bojańczyk et al., 2014, Section 10).

many) one-orbit sets. Recall the set $\mathbb{A}^{(d)}$ of non-repeating tuples that was described in Example 17. This is an equivariant single-orbit set. The key observation is the following lemma.

Lemma 4.13. *Assume that X has one orbit. There is some $d \in \{0, 1, \ldots\}$ an equivariant surjective function*

$$f : \mathbb{A}^{(d)} \to X$$

such that tuples with the same value under f are equal as sets:

$$f(a_1, \ldots, a_d) = f(b_1, \ldots, b_d) \qquad \textit{implies} \qquad \{a_1, \ldots, a_d\} = \{b_1, \ldots, b_d\}.$$

Proof. By Lemma **??** there is an equivariant surjective function

$$f : Y \to X \qquad \text{for some equivariant } Y \subseteq \mathbb{A}^n.$$

Take some equivariant orbit of f, with f viewed as a subset of $Y \times X$. This orbit is still an equivariant function whose image is also X. In other words, we can assume without loss of generality that Y is a single equivariant orbit in $\mathbb{A}^d$. Such an orbit is an equality type. By projecting away the duplicated coordinates in the equality type, we can assume that Y contains only nonrepeating tuples. Summing up, we know that there is a surjective equivariant function

$$f : \mathbb{A}^{(d)} \to X.$$

We show below that the function either satisfies the condition in the statement of the lemma, or the dimension d can be made smaller. If the condition in the statement of the lemma is not satisfied, then

$$f(a_1, \ldots, a_d) = f(b_1, \ldots, b_d) \tag{4.4}$$

holds for some tuples $\bar{a}, \bar{b}$ which are not equal as sets. Without loss of generality, we assume that the last atom a_d in $\bar{a}$ does not appear in the tuple $\bar{b}$. Choose some atom permutation π which fixes the first $d-1$ atoms in $\bar{a}$ and all atoms in $\bar{a}$, but does not fix a_n. We have

$$f(\bar{a}) \overset{(4.4)}{=} f(\bar{b}) \overset{\pi \text{ fixes } \bar{b}}{=} f(\pi(\bar{b})) \overset{\text{equivariance}}{=} \pi(f(\bar{b})) \overset{(4.4)}{=} \pi(f(\bar{a})) \overset{\text{equivariance}}{=} f(\pi(\bar{a})),$$

which proves that

$$f(a_1, \ldots, a_{d-1}, a_d) = f(a_1, \ldots, a_{d-1}, a) \qquad \text{for some distinct } a, a_1, \ldots, a_d.$$

The set of tuples $a, a_1, \ldots, a_d$ which satisfies the condition above is an equivariant subset of $\mathbb{A}^{(d+1)}$, by equivariance of f. Therefore, if some tuple satisfies the condition, then all tuples in $\mathbb{A}^{(d+1)}$ satisfy it as well, i.e. we could also write "for all distinct" in the above condition. In other words, the value of f depends only on the first $d-1$ coordinates. Therefore, we can use the induction assumption. □

Using the above lemma, we complete the proof of the Least Support Theorem. Apply Lemma 4.13 yielding some equivariant function

$$f : \mathbb{A}^{(n)} \to X.$$

Let $x \in X$, and choose some tuple $(a_1, \ldots, a_n)$ which is mapped by f to x. To prove the Least Support Theorem, we will show that the atoms $a_1, \ldots, a_n$ appear in every support of x. Let then $\bar{b}$ be some atom tuple which supports x. Toward a contradiction, suppose that $\bar{b}$ is not a permutation of $a_1, \ldots, a_d$, and therefore one can choose atom permutation π such that

$$\begin{aligned} &\pi(b_i) = b_i \quad \text{for every } i \in \{1, \ldots, d\} \\ &\pi(a_i) \notin \{a_1, \ldots, a_d\} \quad \text{for some } i \in \{1, \ldots, d\}. \end{aligned}$$

We have

$$\begin{aligned} x &= && (\pi \text{ fixes the support of } x) \\ \pi(x) &= && (\text{choice of } a_1, \ldots, a_d) \\ \pi(f(a_1, \ldots, a_d)) &= && (\text{equivariance of } f) \\ f(\pi(a_1, \ldots, a_d)). \end{aligned}$$

Since the tuple $\pi(a_1, \ldots, a_n)$ is not equal to $(a_1, \ldots, a_n)$ as a set, it must have a different value than x, by assumption on the function f. □

A representation theorem

Apart from the Least Support Theorem, another application of Lemma 4.13 is the following representation theorem for equivariant orbit-finite sets. Let X be a one-orbit set. Apply Lemma 4.13, yielding an equivariant function

$$f : \mathbb{A}^{(d)} \to X.$$

Because f is equivariant and permutations of coordinates commute with atom automorphisms, the following conditions are equivalent for every permutation g of the coordinates $\{1, \ldots, d\}$:

$$f(a_1, \ldots, a_d) = f(a_{g(1)}, \ldots, a_{g(d)}) \qquad \text{for some } (a_1, \ldots, a_d) \in \mathbb{A}^{(d)} \tag{4.5}$$

$$f(a_1, \ldots, a_d) = f(a_{g(1)}, \ldots, a_{g(d)}) \qquad \text{for every } (a_1, \ldots, a_d) \in \mathbb{A}^{(d)}. \tag{4.6}$$

Permutations g which satisfy condition (4.6) form a group, call it G. This is a subgroup of the group of permutations of the coordinates $\{1, \ldots, d\}$. We claim:

$$\begin{aligned} f(a_1, \ldots, a_d) &= f(b_1, \ldots, b_d) \\ &\text{iff} \\ \exists g \in G \; (a_1, \ldots, a_d) &= (b_{g(1)}, \ldots, b_{g(d)}). \end{aligned}$$

The bottom-up implication is by definition. For the top-down implication, recall that Lemma 4.13 asserted that tuples with the image under f must contain the same atoms, and therefore some $g \in G$ must take one tuple to the other. Let us write

$$\mathbb{A}^{(d)}/_G$$

to be $\mathbb{A}^{(d)}$ for the set of non-repeating atom tuples modulo coordinate permutations from the group G. Since quotienting by G is exactly the kernel of the function f, we have just proved the following theorem[5]:

Theorem 4.14. *Let X be an orbit-finite set that has one orbit. Then X admits an equivariant bijection to a set of the form*

$$\mathbb{A}^{(n)}/_G$$

for some $d \in \mathbb{N}$ and some subgroup G of the group of permutations of the set $\{1, \ldots, d\}$.

Example 27. Let $d \in \{1, 2, \ldots\}$ and let G be the group of all permutations of $\{1, \ldots, d\}$. In this case, $\mathbb{A}^{(d)}/_G$ is the same as

$$\binom{\mathbb{A}}{d},$$

i.e. unordered sets of atoms with exactly d elements. □

Myhill-Nerode for pof sets

As we have mentioned after the proof of the Myhill-Nerode characterization in Theorem 4.9, in the case of pof sets that are not quotiented, deterministic pof automata are equivalent to deterministic quotiented pof automata. This is proved below.

Theorem 4.15. *Assume that the input alphabet is a (non-quotiented) pof set Σ. Then the two equivalent conditions in Theorem 4.9 are also equivalent to*

(3) L is recognised by a deterministic pof automaton.

Proof. Since pof sets are a special case of quotiented pof sets, it is enough to show that if the input alphabet is a (non-quotiented) pof set, then for every deterministic quotiented pof automaton, there is an equivalent (non-quotiented) pof automaton. Consider a deterministic quotiented pof automaton. Let Q be its state space. By Theorem 4.14, we can assume that the state space is

$$Q = \sum_{i \in I} \mathbb{A}^{(d_i)}/G_i$$

Consider the pof set

$$P = \sum_{i \in I} \mathbb{A}^d.$$

[5]This result is from (Bojańczyk et al., 2014, Theorem 10.17), although a similar construction can already be found in (Ferrari et al., 2002, Definition 2).

There is a natural projection from P to Q, which is a partial function

$$\Pi : P \to Q.$$

This projection is defined on elements with non-repeating tuples of atoms, and it returns the corresponding equivalence class. An important property of this projection is:

(*) every output element arises from finitely many input elements.

Let us pull back the transition function of the original automaton to a transition relation on states P. In other words, define

$$\Delta \subseteq P \times \Sigma \times P$$

to be the inverse image, under the projection Π, of the transition function of the original automaton. We view Δ as a binary relation between two pof sets, namely $P \times \Sigma$ and P. Because the original automaton was deterministic, and thanks to the finiteness condition (*), we know that for every input in $P \times \Sigma$, the transition relation Δ has finitely many outputs in P. Therefore, we can apply the following lemma to extract an equivariant function contained in Δ.

Lemma 4.16. *Let X and Y be pof sets, and let $\Delta \subseteq X \times Y$ be an equivariant binary relation such that for every $x \in X$ the set*

$$\{ y \in Y \mid (x, y) \in \Delta \}$$

is nonempty and finite. Then there is an equivariant function $\delta : X \to Y$ whose graph is contained in Δ.

Proof. Homework. □

□

Exercises

Exercise 90. Consider a qpof group, i.e. the underlying set is a qpof, and the group operation is equivariant. Show that such a group must be finite.

Exercise 91. Let X be a qpof. Show that if $f : X \to X$ is an equivariant surjective function, then f is a bijection.

Exercise 92. Consider a chain

$$X_0 \overset{f_1}{\twoheadrightarrow} X_1 \overset{f_2}{\twoheadrightarrow} X_2 \overset{f_3}{\twoheadrightarrow} \ldots \overset{f_n}{\twoheadrightarrow} X_n$$

of equivariant surjective function between qpof sets. Show that the length of this chain is bounded by a polynomial of the following two parameters of the first set X_0: the orbit count, and the atom dimension.

Chapter 5

Atoms beyond equality

So far, we have worked with atoms that have equality only. It turns out that the theory developed in this book is also meaningful when the atoms have extra structure, like an order. We take a logical approach, where the notion of atoms is specified by a relational structure, i.e. a set with some relations on it. Here are some examples:

$$\underbrace{(\mathbb{N})}_{\substack{\text{the natural numbers}\\\text{with no relations}}} \quad \underbrace{(\mathbb{N}, <)}_{\substack{\text{the natural numbers}\\\text{with order}}} \quad \underbrace{(\mathbb{Z}, <)}_{\substack{\text{the integers}\\\text{with order}}} \quad \underbrace{(\mathbb{Q}, +)}_{\substack{\text{the rational nunmbers}\\\text{with a ternarny relation}\\\text{for adition } x+y=z}}.$$

All of these structures will be candidates for atoms, however only the first and last one will turn out to be appropriate. This chapter explains when a structure is appropriate, and how the theory works when that happens.

5.1 Oligomorphic structures

The choice of atoms will be formalized by a relational structure, as in model theory.

Definition 5.1 (Relational structure)**.** A *relational structure* consists of:

1. an underlying set A, called the *universe* of the structure;
2. a family of relations on this set, each one of the form $R \subseteq A^d$ for some d.

We use letters like $\mathbb{A}$ or $\mathbb{B}$ to describe the atoms. A candidate for the atoms is a relational structure. Not all candidates are appropriate, however. Here is an example of an inappropriate one.

Example 28. [Presburger Arithmetic] Suppose that for the atoms we would like to use the natural numbers with successor, i.e. the relational structure

$$\mathbb{A} = (\{0, 1, 2, \ldots\}, \underbrace{x + y = z}_{\text{a ternary relation for the successor}}).$$

This structure is also known as Presburger Arithmetic. We could consider polynomial orbit-finite sets for this structure, and subsets of them that are definable using formulas. This time, the formulas could use the relations given in the structure, namely the successor relation and a zero test. In this setting, many of the problems that were decidable previously, will become undecidable for the new choice of atoms. An example is graph reachability – one can easily encode the halting problem for counter machines. To implement a zero test on variable x, we check if $x + x = x$. □

As we see from the above example, some structures, such as Presburger Arithmetic, will not be a good choice for the atoms. This is despite Presburger Arithmetic being a very tame structure, in particular it has a decidable first-order theory, as formalized in the following definition. (We assume that the reader is familiar with the basics of first-order logic, such as what it means for a formula to be true in a structure, or what free variables are. For a more detailed introduction, see Hodges (1993).)

Definition 5.2 (Decidable first-order theory)**.** The *first-order theory* of a structure is the set of first-order sentences that are true in it. Here, a first-order sentence is a formula that is built using the following constructors

$$\underbrace{\forall x \quad \exists x}_{\text{quantifiers}} \qquad \underbrace{\vee \quad \wedge \quad \neg}_{\text{Boolean combinations}} \qquad \underbrace{x = y}_{\text{equality}} \qquad \underbrace{R(x_1, \ldots, x_d)}_{\text{relations from the structure}},$$

and which has no free variables. A structure has a decidable first-order theory if there is an algorithm that decides if first-order sentence belongs to the theory.

We will typically be interested in structures with a decidable first-order theory, such as Presburger Arithmetic. In fact the latter structure is named Mojżesz Presburger, who proved that its first-order theory is decidable. As we saw in Example 28, having a decidable first-theory will not – on its own – be sufficient for our theory. It will be equally important that the notion of equivariant subset is well-behaved.

So far, equivariance, which was defined in terms of atom permutations. When the atoms are a structure with relations beyond equality, the role of permutations is played by automorphisms, as described in the following definition.

Definition 5.3 (Automorphisms)**.** An *automorphism* of a relational structure $\mathbb{A}$ is a bijection π of its universe with itself, which preserves all relations, i.e.

$$\bar{a} \in R \qquad \Leftrightarrow \qquad \pi(\bar{a}) \in R$$

holds for every relation R of arity d in the structure and every tuple $\bar{a} \in \mathbb{A}^d$.

Example 29. [Presburger arithmetic is rigid] Suppose that we define the atoms $\mathbb{A}$ to be Presburger Arithmetic. The problem with this choice is that there are no non-trivial automorphisms (such structures are called rigid). Indeed, an automorphism must map 0 to 0, and then it must map 1 to 1, and so on. Therefore, the only automorphism is the identity. This means that every subset of $\mathbb{A}$, or more generally $\mathbb{A}^d$, is going to be equivariant. In particular, this precludes any finite representation or algorithms that would deal with equivariant subsets. □

Example 30. [Equality atoms] Suppose that we define the atoms $\mathbb{A}$ to be a structure where the universe is some countably infinite set, and which has no relations. (Equality is assumed to be a given for first-order logic.) An automorphism in this case is the same as a bijection of the universe with itself, i.e. a permutation of the universe, as was the case for atoms with equality only. For this reason, we call this structure the *equality atoms*. □

Example 31. [Integers with order] Suppose that we define the atoms $\mathbb{A}$ to be $(\mathbb{Z}, <)$, i.e. the integers with order. Automorphisms of this structure must preserve the order. Therefore, they must also preserve the successor relation. Indeed, if two consecutive elements x and $x+1$ would be mapped to non-consecutive elements, then the resulting gap would be a violation of bijectivity. Therefore, automorphisms of this structure are translations, i.e. functions of the form $x \mapsto x+c$ for some $c \in \mathbb{Z}$. In this structure, $\mathbb{A}$ has one orbit, because one can go from every integer to every other integer by applying some translation. However, $\mathbb{A}^2$ has infinitely many orbits, because the difference $x_1 - x_2$ between the two coordinates is preserved by translations. Therefore, there are uncountably many equivariant subsets of $\mathbb{A}^2$. □

As illustrated in the above examples, we want the structure to have finitely many orbits under its automorphisms. This should not only hold for the structure $\mathbb{A}$ itself, but also for its powers $\mathbb{A}^d$, since such powers will arise in our constructions (such as pof sets). Hence the following definition.

Definition 5.4. A structure $\mathbb{A}$ is called *oligomorphic*[1] if for every $d \in \{0, 1, \ldots\}$, the structure $\mathbb{A}^d$ has finitely many elements up to automorphisms of $\mathbb{A}$. More precisely, for every d, the equivalence relation on $\mathbb{A}^d$ defined by

$$\bar{a} \sim \bar{b} \qquad \text{if } \pi(\bar{a}) = \bar{b} \text{ for some automorphism } \pi \text{ of } \mathbb{A}$$

has finitely many equivalence classes.

Example 32. The equality atoms from Example 30, i.e. an infinite set without any structure except equality, are oligomorphic. These are the atoms that we have studied so far: automorphisms are permutations, and the number of orbits in $\mathbb{A}^d$ is the d-th Bell number. The other structures

$$(\mathbb{N}, +) \qquad (\mathbb{Z}, <)$$

discussed in Examples 28 and 31 are not oligomorphic. For the second one, we need to got to the second power to get infinitely many orbits. □

Example 33. Every structure with a finite universe is oligomorphic. □

[1]The notion of oligomorphic structures comes from Ryll-Nardzewski (1959), Engeler (1959) and Svenonius (1959), who proved that countable oligomorphic structures are exactly those which are ω-categorical, i.e. are the unique countable models of their first-order theory. This connection with first-order logic will be important in Chapter **??**, which discusses how orbit-finite sets can be represented using formulas of first-order logic.

Example 34. [Ordered atoms] Consider the structure $(\mathbb{Q}, <)$ of ordered rational numbers. We will call this structure the *ordered atoms.* This is because it will turn out that this structure is the canonical way of modelling a total order in our theory, as we will describe in Chapter 6. We will show that this structure is oligomorphic. An automorphism of this structure is any order-preserving permutation. For example, the affine function

$$x \mapsto \frac{x}{3} - 2$$

is an automorphism. On the other hand, x^3 is not an automorphism, despite preserving the order. The reason is that cubing is not invertible on the rationals. To show that this structure is oligomorphic, we will prove that two tuples

$$(a_1, \ldots, a_d), (b_1, \ldots, b_d) \in \mathbb{Q}^d$$

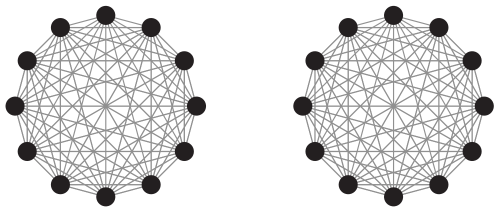
are in the same orbit, with respect to atom automorphisms, if and only if they have the same order type, i.e.

$$a_i \le a_j \quad \Leftrightarrow \quad b_i \le b_j \qquad \text{for all } i, j \in \{1, \ldots, d\}.$$

This will imply oligomorphism, since there are finitely many different order types for each dimension d. Clearly if the tuples are in the same orbit, then they must have the same order type, by definition of automorphisms. The converse implication is also not hard to see, for example it is sufficient to consider piecewise affine maps. □

Example 35. The real numbers with order $(\mathbb{R}, <)$ are also oligomorphic. The same argument as for the rationals works. However, we will not study this structure since we care about countable ones only. □

Example 36. Consider an undirected graph with two countably infinite cliques (without self-loops). Here is a picture, with only 12 vertices shown for each of the two cliques:



The graph, like any graph, can be viewed as a logical structure, where the universe is the vertices, and there is one binary relation for edges, which is symmetric and irreflexive. The automorphisms of this structure (which are the same as graph automorphisms in the usual sense) are generated by: permutations of the first clique, permutations of the second clique, and swapping the two cliques. In particular, the tuples

$$(a_1, \ldots, a_d) \quad \text{and} \quad (b_1, \ldots, b_d)$$

are equal up to atom automorphisms if and only if they have the same equality types and the same equivalence types with respect to the equivalence relation “in the same clique”. Since there are finitely many possibilities for every choice of n, it follows that these atoms are oligomorphic. □

Polynomial orbit-finite sets. Many of the notions that we have described so far make sense for other atoms, and not just the equality atoms. The only difference is that instead of atom permutations, we use the more general notion of atom automorphism. In the special case of the equality atoms, this will be the same as atom permutations.

We begin with the generalization of pof sets and their equivariant subsets.

Definition 5.5 (Polynomial orbit-finite sets for general atoms)**.** Let $\mathbb{A}$ be an oligomorphic structure. A *pof set* over this structure is any set of the form

$$\mathbb{A}^{d_1} + \cdots + \mathbb{A}^{d_k}.$$

A subset X of a pof set is called *equivariant* if membership in the subset is invariant under atom automorphisms, i.e.

$$x \in X \quad \Leftrightarrow \quad \pi(x) \in X \qquad \text{for every automorphism } \pi \text{ of } \mathbb{A},$$

with the expected action of automorphisms on elements of the pof set.

The first part of the above definition, i.e. applying a polynomial to some structure, makes sense for structures that are not necessarily oligomorphic. However, we intend to study pof sets equipped with equivariant subsets, and equivariant subsets are well-behaved only for oligomorphic structures.

As was the case for the equality atoms, we can consider pof automata, now for a general structure.

Example 37. Consider the ordered atoms $\mathbb{A} = (\mathbb{Q}, <)$ from Example 34, and the language

$$\{\, w \in \mathbb{A}^* \mid \text{ the letters in } w \text{ are strictly growing } \}.$$

This language is recognised by a deterministic pof automaton. The automaton stores the most recent letter, and enters a rejecting sink state if it sees a decrease. The state space is

$$\underbrace{\mathbb{A}^0}_{\text{initial}} + \underbrace{\mathbb{A}^1}_{\substack{\text{last}\\\text{atom}}} + \underbrace{\mathbb{A}^0}_{\text{error}}$$

and the transition function is defined as expected. □

Exercises

Exercise 93. Show that the structure $(\mathbb{Z}, <)$ is not oligomorphic.

Exercise 94. For the atoms $(\mathbb{Q}, <)$, find all equivariant binary relations on $\mathbb{A}$.

Exercise 95. Consider a structure $\mathbb{A}$ that is oligomorphic. Let $\mathbb{B}$ be a new structure, whose universe is a pof set over $\mathbb{A}$, and whose relations are equivariant (under automorphism of $\mathbb{A}$). Show that $\mathbb{B}$ is also an oligomorphic structure.

5.2 Representation of equivariant subsets

The purpose of the theory that is developed in this book is to have a generalization of finiteness that is amenable to algorithms. In particular, equivariant sets should allow for finite representations, and should have other good properties of finite sets. From the very definition of oligomorphism we see that an equivariant subset can be chosen in finitely many ways, as explained in the following lemma.

Lemma 5.6. *Let $\mathbb{A}$ be a relational structure that is oligomorphic. Then every pof set has finitely many equivariant subsets.*

Proof. It is enough to show that pof sets of the form $\mathbb{A}^d$ have finitely many equivariant subsets, and the result will transfer to general pof sets, which are disjoint unions of such sets. By definition of oligomorphism, there are finitely many orbits in $\mathbb{A}^d$, and each equivariant subset is a union of such orbits. Therefore, there are finitely many choices. □

A corollary of the above lemma is that certain fixpoint algorithms will be guaranteed to terminate in finite time. A typical example is graph reachability, as illustrated below.

Example 38. In Theorem 2.2, we showed that graph reachability is decidable pof sets under the equality atoms. Suppose that we want to generalise this to any oligomorphic atoms. A natural idea is to consider the chain

$$V_0 \subseteq V_1 \subseteq V_2 \subseteq \cdots \tag{5.1}$$

where V_n is the set of vertices that can be reached from some source vertex via a path of length at most n. Assuming that set of source vertices is equivariant, and the edge relation is also equivariant, the set V_n will also be equivariant for every n. It follows from Lemma 5.6 that the chain (5.1) will stabilize after finitely many steps, and therefore the set of reachable vertices can be obtained in finitely many steps. □

In the above example, we showed an "algorithm" that decides graph reachability by computing a finite increasing chain of equivariant subsets of vertices. However, for this to be an algorithm, we need to be to represent subsets V_n from the chain in a finite way; compute the new subsets based on the previous ones, and test equality between such subsets. This leads us to the question:

How can we represent equivariant subsets of a pof set?

Of course, we want the representation to support certain basic operations, such as checking if two subsets are the same (because the same subset might have several representations), or Boolean combinations on subsets. Since a pof set is a finite union of sets of the form $\mathbb{A}^d$, the question reduces to

How can we represent equivariant subsets of $\mathbb{A}^d$?

In the case of the equality atoms, in Section 1.1 we proposed two representations, namely generating subsets, and formulas. As it turns out, these two representations carry over to general oligomorphic structures.

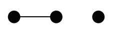
By definition of oligomorphic atoms, see Lemma 5.6, a pof set will have finitely many orbits, and therefore every equivariant subset can be represented by giving one element for each orbit. Therefore, we can use finite sets of generators to describe equivariant subsets, at least as long as we can write down individual elements of the structure. There are, however, some unresolved questions about this representation for subsets. For example: how do we test equality of two subsets given by generators? Or: how do we compute the complement? We will return to these questions in Section 7.2; for the moment we will stick to the formula representation. As we will see below, the formula representation is very well suited to the oligomorphic case, since oligomorphic structures are exactly those where equivariant subsets can be defined by first-order formulas.

Under the equality atoms, we represented an equivariant subset of $\mathbb{A}^d$ by a formula

$$\varphi(x_1, \ldots, x_d)$$

that used Boolean combinations and equality. In the general oligomorphic case, we will also use such formulas, but we will allow quantifiers, and other relations – beyond equality – that are present in the structure. Subset of $\mathbb{A}^d$ that can be defined this way are called *first-order definable.* For some structures, such as the equality atoms, we can avoid quantifiers, but for others the quantifiers will be necessary, as illustrated in the following example.

Example 39. Consider the following three-vertex graph:



As mentioned in Example 36, this graph can be seen as a relational structure with one binary relation. Like any finite structure, this structure is oligomorphic. There is no quantifier-free formula that distinguished the isolated vertex from a non-isolated vertex, despite the two vertices being in different orbits. □

The following theorem shows that for oligomorphic structures which are countable (i.e. have a countable universe), equivariant subsets are exactly the first-order definable ones.

Theorem 5.7. *Let $\mathbb{A}$ be a countable oligomorphic structure. A subset $X \subseteq \mathbb{A}^d$ is equivariant if and only if it is first-order definable.*

Proof. In this proof, we use the name *atom* for elements of the universe. Consider the following game (known as the Ehrenfeucht-Fraïssé game), which is parametrised by two tuples $\bar{a}, \bar{b} \in \mathbb{A}^d$ and a number of rounds $k \in \{0, 1, 2, \ldots, \omega\}$. The game is played by two players, called Spoiler and Duplicator. In each round:

- Spoiler chooses one of the tuples and extends it with one atom.
- Duplicator responds by extending the other tuple with one atom.

Spoiler wins the game if, for some finite $i \leq k$, the (extended) tuples after playing i rounds can be distinguished by some quantifier-free formula (using the relations from the structure), otherwise Duplicator wins. The theorem follows immediately from the equivalence of items 1 and 4 in the following lemma.

Lemma 5.8. *The following conditions are equivalent for every tuples $\bar{a}, \bar{b} \in \mathbb{A}^d$:*

1. *the tuples belong to the same first-order definable subsets;*
2. *Duplicator has a winning strategy in the k-round game for every $k < \omega$;*
3. *Duplicator has a winning strategy in the ω-round game;*
4. *the tuples are in the same orbit.*

Proof.

- *1 implies 2.* This is (half of) the classical Ehrenfeucht-Fraïssé theorem[2], which says that if two tuples satisfy the same formulas of quantifier rank at most k, then Duplicator has a winning strategy in the k-round game. Recall that the quantifier rank of a formula is the maximal nesting of quantifiers.

- *2 implies 3.* In this step, we use oligomorphism. The key observation is in the following claim, which shows that Duplicator has a strategy that ensures staying in positions that satisfy 2.

  Claim 5.9. *Consider one round of the Ehrenfeucht-Fraïssé game, which begins in a position (i.e. a pair of atom tuples of same finite length) that satisfies condition 2. For every move of player Spoiler, there is a response of player Duplicator which ensures that the resulting position also satisfies condition 2.*

  Proof. Suppose that the round begins in a position $(\bar{a}, \bar{b})$. By symmetry, we only consider the case when Spoiler extends the tuple $\bar{a}$ with some atom $a \in \mathbb{A}$. By condition 2, we know that for every k there is some response $b_k \in \mathbb{A}$ of player Duplicator, which guarantees that

  $$\text{Duplicator can win the } k\text{-round game from position } (\bar{a}a, \bar{b}b_k). \tag{5.2}$$

  Observe that the above condition is, which describes a property of tuples of some fixed length, is equivariant. This is because the dynamics of the game would not be affected if we applied an atom automorphism to all choices. By oligomorphism, we know that there are finitely many orbits of tuples

  $$(\bar{a}a, \bar{b}b_k)$$

  that can be realized. Therefore, some orbit is hit by infinitely many choices of b_k. By equivariance of (5.2), we can pick some b_k that witnesses an orbit that is hit infinitely often, and this b_k will guarantee winning the k-round game for infinitely many k, and therefore for all k. □

[2]See (Hodges, 1993, Section 3.2)

Thanks to the above claim that if we play the ω-round game and we start in a position that satisfies 2, then Duplicator can play in a way that guarantees always staying in positions that satisfy 2. In particular, Duplicator can win ω-rounds, thus witnessing 3.

- *3 implies 4.* In this step, we use countability. We need to show that if Duplicator has a winning strategy in the ω-round game for tuples $\bar{a}$ and $\bar{b}$, then there is an automorphism that maps one tuple to the other. This is proved using a back-and-forth argument. Fix some enumeration of the model $\mathbb{A}$, which exists by assumption on countability. Consider a play in the ω-round game, where Spoiler uses the following strategy:

  – in even-numbered rounds, extend the $\bar{a}$ tuple with the least (according to the enumeration) atom that does not appear in it;

  – in odd-numbered rounds, do the same for the $\bar{b}$ tuple.

  Suppose that Duplicator responds to the above strategy with a winning strategy. In the resulting play, we get two infinite sequences

  $$a_1, a_2, \ldots \qquad b_1, b_2, \ldots$$

  of atoms that extend the tuples $\bar{a}$ and $\bar{b}$, respectively. By the choice of Spoiler's strategy, every atom appears in the first infinite sequence, and also every atom appears in the second infinite sequence. Therefore, the function $a_i \mapsto b_i$ is permutation of the atoms. Furthermore, this permutation is an automorphism, since at every step in the game, the same same quantifier-free formulas must be satisfied on both sides.

- *4 implies 1.* By induction on the quantifier rank k, one shows that tuples in the same equivariant orbit must satisfy the same first-order formulas of quantifier rank k.

This completes the proof of the lemma, and therefore also of the theorem. □

□

Graph reachability

In the previous chapters, we showed that some decision problems – such as graph reachability or emptiness for nondeterministic automata – can be decided. We now show that these results carry over to other structures, under the suitable assumptions. The first of these assumptions is that the structure is countable and oligomorphic, and so we can use Theorem 5.7 to conclude that equivariant subsets can be represented in a finite way, namely by first-order formulas. This assumption makes the decision problems well-posed, because the inputs (such as graphs or automata) can be represented in a finite way. However, we also need to be able to operate on equivariant subsets. For example, the same equivariant subset might have several representations, and we need to be able to test equality between them. This boils down to the question: given two first-order formulas

$$\varphi(x_1, \ldots, x_d) \qquad \text{and} \qquad \psi(x_1, \ldots, x_d),$$

decide if they define the same subset of $\mathbb{A}^d$. Already in the special case when $d = 0$, i.e. when the formulas are sentences, this problem is the same as checking which sentences are true in the structure. Therefore, in order to manipulate equivariant subsets represented by formulas, we will want the first-order theory to be decidable; this will be our second assumption. These two assumptions will be enough for many algorithms. An example is graph reachability – the following theorem shows that the decidability result from Section 2.1 transfers over from the equality atoms to general oligomorphic structures.

Theorem 5.10. *Assume that the atoms $\mathbb{A}$ are a countable oligomorphic structure with a decidable first-order theory. Then reachability for pof graphs is decidable.*

Proof. Although we have essentially described the algorithm in Example **??**, we spell out the details about the representation in this proof, to explain how exactly we manipulate equivariant subsets represented by formulas. The input to the problem consists of a pof set

$$V = \sum_{i \in I} \mathbb{A}^{d_i},$$

together with three equivariant relations:

$$\underbrace{E \subseteq V^2}_{\text{edges}} \qquad \underbrace{S, T \subseteq V}_{\substack{\text{source and target}\\ \text{vertices}}}$$

An equivariant subset of V – such as the source and target sets – is represented by a family of first-order formulas, with one formula for each component $i \in I$ of the disjoint union in the set V. The formula for component i has d_i free variables, and tells us when a tuple of atoms belongs to the i-th component. A similar representation is used for the binary relation E – for each pair of components $i, j \in I$, there is a formula with $d_i + d_j$ free variables, which tells us when a tuple of atoms from the i-th component is related to a tuple of atoms from the j-th component. We will use these representations to implement a reachability algorithm.

We intend to compute the chain

$$V_0 \subseteq V_1 \subseteq V_2 \subseteq \cdots$$

of sets, such that V_n is the vertices that can be reached by a path of length at most n. Each set V_n will be represented by a family of formulas, call these formulas $\{\varphi_i^n\}_{i \in I}$. For $n = 0$, we use the formulas for the source set. Let us now show how to compute the formulas for V_{n+1} based on the formulas for V_n:

$$\overbrace{\varphi_i^{n+1}(\bar{x})}^{\substack{\text{the formula for}\\ \text{component } i \text{ in } V_{n+1}}} \quad = \quad \overbrace{\varphi_i^{n}(\bar{x})}^{\substack{\text{the formula for}\\ \text{component } i \text{ in } V_{n}}} \;\vee\; \underbrace{\bigvee_{j\in I} \exists \bar{y}}_{\substack{\text{choosing a component}\\ j \in I \text{ and a tuple } \bar{y}\\ \text{of } d_j \text{ atoms is the same as}\\ \text{choosing an element of } V}} \; (\overbrace{\varphi_j^{n}(\bar{y})}^{\substack{\text{the formula for}\\ \text{component } j \text{ in } V_{n}}} \;\wedge\; \underbrace{\varphi^E_{ji}(\bar{y},\bar{x})}_{\substack{\text{the formula for}\\ \text{components } i \text{ and } j \text{ in}\\ \text{the edge relation } E}}).$$

As explained in Example **??**, this chain cannot grow infinitely often, because the set of vertices has finitely many orbits, and each set in the chain is a union of these orbits. Also, a new set in the chain is defined in terms of the previous one, and therefore once we have $V_{n+1} = V_n$ for some n, then the chain stabilizes forever. We can check when the chain stabilizes by asking if the following first-order formula – which says that no new elements have been added – is true in the atoms:

$$\bigwedge_{i\in I} \forall \bar{x}\, \varphi_i^n(\bar{x}) \;\Rightarrow\; \varphi^{n+1}(\bar{x}).$$

We can get an answer to this question, by the assumption that the atoms have a decidable first-order theory. □

In the proof above, we did not give a more precise estimate on the computational complexity of the problem, beyond saying that it is decidable. Later on in this book, we will see that the algorithm is in PSpace for most choices of atoms that we consider, including the equality atoms (this was already shown in Section 2.1), and the ordered atoms.

Exercises

Exercise 96. Consider a structure with a countable vocabulary. Show that if it is not oligomorphic, then there is some subset of $\mathbb{A}^d$ that is equivariant, but not first-order definable.

Exercise 97. Consider an oligomorphic structure with a decidable first-order theory. Show that the following conditions are equivalent:

1. the following function, called the *Ryll-Nardzewski function*, is computable:
$$d \in \{0,1,\ldots\} \qquad \mapsto \qquad \text{number of orbits in } \mathbb{A}^d;$$
2. there is an algorithm that inputs $d \in \{0,1,\ldots\}$ and returns a formula with $2d$ free variables that defines the "same orbit" relation on $\mathbb{A}^d$.

Exercise 98. Consider a countable oligomorphic structure. Show that the following conditions are equivalent for a sequence of orbits

$$X_1 \subseteq \mathbb{A}^1, X_2 \subseteq \mathbb{A}^2, \ldots :$$

1. there is some enumeration of the atoms such that X_n is the orbit of the first n atoms in the enumeration;

2. X_n is obtained from X_{n+1} by deleting the last coordinate, and every orbit in $\mathbb{A}^*$ can be obtained from some X_n by deleting some coordinates.

Exercise 99. Consider the two conditions in Definition **??**. Show that in the presence of the first condition **??**, the second condition **??** is equivalent to any of the following conditions:

1. the Ryll-Nardzewski function is computable, as in item 1 from Exercise 97;
2. there is an algorithm that inputs $d \in \{0, 1, \ldots\}$ and returns a list of tuples that generate $\mathbb{A}^d$, with one tuple for each orbit (i.e. no orbit is represented twice).

Exercise 100. Show that the following are equivalent for a countable oligomorphic structure:

1. has a representation;
2. has a decidable first-order theory and a computable Ryll-Nardzewski function.

Exercise 101. Consider the following two conditions for an orbit-finite graph.

1. there is an directed path;
2. there is a cycle.

Find an atom structure where the two conditions are equivalent, and also an atom structure where only the implication 1 $\Leftarrow$ 2 is true.

Exercise 102. Show that under the assumptions of Theorem **??**, there is an algorithm that checks if condition 1 of Exercise 101 is satisfied, assuming that the graph, source and targets are all hereditarily orbit-finite. Likewise for condition 2.

Exercise 103. An instance of *alternating reachability* is defined in the same way as an instance of graph reachability, i.e. there is a directed graph with distinguished source and target vertices. The difference is in the semantics: we play a game between players Odd and Even, with Odd choosing the next edge in odd rounds, and Even choosing the next edge in even rounds. We want to decide if player Odd has a strategy that guarantees seeing a target vertex in a finite number of rounds, regardless of the choice of initial vertex in the source set[3]. Show that this problem is decidable under the assumptions of Theorem 5.10.

Exercise 104. Assume the equality atoms. A *Büchi game* has the same syntax as alternating reachability from Exercise 103. The game is played similarly, except that the objective of player 0 is to see vertices from T infinitely often. Give an algorithm that decides the winner in a Büchi game represented by a set builder expression. Hint: use memoryless determinacy of Büchi games without atoms, see (Thomas, 1990, Theorem 6.4).

Exercise 105. Consider the graph which is obtained by taking a disjoint union of all cliques, one for each size $n \in \{1, 2, \ldots\}$. This structure is not oligomorphic, but we can still consider pof sets with first-order definable subsets. Show that graph reachability is decidable.

5.3 Orbit-finite sets

In Section 4.2, we gave a more semantic notion of finiteness for the equality atoms, called orbit-finiteness. This notion, which is the central one for this book, extends to other structures by using automorphisms instead of permutations.

[3]This type of game is called a *reachability game*. More general games, namely parity games, are studied in (Klin and Łełyk, 2017, Section 5.2)

Definition 5.11 (Finite supports and orbit-finiteness)**.** Let $\mathbb{A}$ be a relational structure, and consider a set X that is equipped with an action of atom automorphisms, i.e. automorphisms of the structure $\mathbb{A}$.

- *Supports.* An element $x \in X$ is *supported* by a list of atoms $a_1, \ldots, a_n$ if

$$\pi(\bar{a}) = \bar{a} \quad \Rightarrow \quad \pi(x) = x$$

holds for every automorphism π of the structure $\mathbb{A}$. We say that x is *finitely supported* if it is supported by some finite list of atoms.

- *Orbit-finite set.* The set X is called *orbit-finite* if every element $x \in X$ has finite support, and there are finitely many orbits under the group action.

We will only be interested in orbit-finite sets for atoms that are oligomorphic. The oligomorphic assumption will guarantee that basic operations, such as product $X \times Y$, can be implemented on orbit-finite sets.

Example 40. [Finitely supported subsets in the ordered atoms] Consider the ordered atoms $\mathbb{A} = (\mathbb{Q}, <)$. In this case, the automorphisms are order-preserving bijections. Consider a subset $X \subseteq \mathbb{A}$ which is supported by a tuple of atoms $\bar{a}$. We claim that X is a union of intervals (open, closed, open-closed or closed-open) whose endpoints are either $-\infty, \infty$, or appear in $\bar{a}$. Indeed, consider atoms b, c that are not in $\bar{a}$ and are not separated by an atom in $\bar{a}$ in terms of the order. There is an automorphism that fixes $\bar{a}$, and which maps b to c. Since the set X is supported by $\bar{a}$, it follows that $b \in X$ if and only if $c \in X$. $\square$

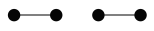
In Chapter 4, we defined quotiented pof sets under the equality atoms, and we showed that they were the same as orbit-finite sets, up to equivariant bijections. The notion of quotiented pof set extends to oligomorphic structures (a pof set quotiented by an equivariant partial equivalence relation). Also, the characterization carries over, as stated in the following theorem.

Theorem 5.12. *Let $\mathbb{A}$ be an oligomorphic structure, and let X be a set that is equipped with an action of atom automorphisms. Then X is orbit-finite if and only if it admits an equivariant bijection with a quotiented pof set.*

Proof. Same proof as the special case for equality atoms from Theorem 4.6. Oligomorphism is used in the easer right-to-left implication: every quotiented pof set is orbit-finite. This is because (non-quotiented) pof sets are orbit-finite by definition of oligomorphism, and quotienting does not increase the number of orbits. $\square$

A corollary of the above theorem is that orbit-finite sets enjoy the same closure properties as quotiented pof sets. For example, they are closed under Cartesian products $X \times Y$, since quotiented pof sets have this property. Not all results carry over to oligomorphic structures. For example, least supports can fail, as explained below.

Example 41. Consider an atom structure $\mathbb{A}$ which is the following graph:



Consider the quotiented pof set $\mathbb{A}/\sim$, where $\sim$ is the equivalence relation "in the same connected component". An element of this set, i.e. a connected component, is supported by any of the two atoms in it, but none of these supports is a least support. A similar phenomenon can be observed in the structure of two cliques from Example 36. In this case, each of the two cliques – when seen as an element of the finitely supported powerset – is supported by any atom that appears in it. □

Exercises

Exercise 106. Assume that the atoms are oligomorphic. Show that for every orbit-finite set X, there is some $d \in \{0, 1, \ldots\}$ and a surjective equivariant function $f : \mathbb{A}^d \to X$.

Exercise 107. Show that the atoms $(\mathbb{Q}, <)$ also have least supports.

Exercise 108. Show an example of oligomorphic atoms without least supports.

Exercise 109. Assume that the atoms are oligomorphic. Let X be a set with an action of group automorphisms, which is not known to be orbit-finite. Let $R \subseteq X \times X$ be an equivariant binary relation which is orbit-finite. Show that the transitive closure of R is also orbit-finite.

Exercise 110. Assume that the atoms are oligomorphic, and there are infinitely many atoms. Show that orbit-finite sets are not closed under taking finitely supported function spaces:

$$X \xrightarrow{\text{fs}} Y \quad \overset{\text{def}}{=} \quad \{ f : X \to Y \mid f \text{ is finitely supported} \}.$$

Exercise 111. Assume oligomorphic atoms. Let X, Y be orbit-finite sets and let F be an equivariant subset of the finitely supported function space from the previous exercise. Show that F is orbit-finite if and only if there is some $n \in \{0, 1, 2, \ldots\}$ such that every function $f \in F$ has a support of size at most n.

Exercise 112. Assume oligomorphic atoms. Show that in an orbit-finite set, for every atom tuple $\bar{a}$ there are finitely many elements supported by $\bar{a}$.

Exercise 113. Show that Exercise 84 fails in $(\mathbb{Q}, <)$.

Exercise 114. Show that Exercise 84 fails in some atoms, even for a relation R such that for every first argument, there are finitely many second arguments related by the relation.

Exercise 115. Assume that the atoms are oligomorphic. Let X be an orbit-finite set and let $\bar{a}$ be a tuple of atoms. Consider the family of equivalence relations on X which are supported by $\bar{a}$ and where every equivalence class is finite. Show that this family has a greatest element with respect to inclusion (i.e. a coarsest equivalence relation).

z

Exercise 116. Show that the following statement is true in the equality atoms but not in $(\mathbb{Q}, <)$. Let X be a set equipped with an action of atom automorphisms, where every element is finitely supported. Then X is orbit-finite if and only if: (***) for every equivariant family of finitely supported subsets of X which is totally ordered by inclusion, there is a maximal element.

Chapter 6

Homogeneous atoms

To define orbit-finiteness, we need atoms that are oligomorphic. How does one get oligomorphic structures?

This chapter is devoted to a method of producing oligomorphic structures, which is called the Fraïssé limit[1]. The idea behind the Fraïssé limit is that it inputs a class of finite structures, sufficiently well-behaved, and outputs a single countably infinite structure which contains all the finite structures, and does so in a certain homogeneous way. The Fraïssé limit can be applied to classes of finite structures such as all finite total orders, all finite directed graphs, all equivalence relations on finite sets, etc.

6.1 Homogeneous structures

Before defining homogeneous structures, we begin by recalling some terminology from logic. In the previous chapter, there was one structure, and we used logic to define relations on this structure. In this chapter, there will be many structures, but we will still want to compare them using a single formula. To do this, we use the notion of a *vocabulary*: this is a set of names for relations, each one with an associated arity in $\{0, 1, \ldots\}$. Here are some examples of vocabularies:

$$\underbrace{x \leq y}_{\substack{\text{the vocabulary for}\\ \text{ordered structures}\\ \text{has one binary relation}}} \qquad \underbrace{\text{edge}(x, y)}_{\substack{\text{the vocabulary for graphs}\\ \text{has one binary relation}}} \qquad \underbrace{x + y = z \quad x \times y = z}_{\substack{\text{the vocabulary for rings}\\ \text{has two ternarny relations}}}.$$

Note that the first two vocabularies are essentially the same, since they both have one relation of arity two. However, is useful to give different names to convey different intentions. In the third vocabulary, we use ternary relations instead of binary functions – this is because we want to stick to relational vocabularies for simplicity. A structure over a given vocabulary is a structure in the sense of Definition 5.1, together with a function (called the *interpretation* of the vocabulary) that assigns each relation

[1]This is a basic notion in model theory. For further information, see e.g. (Hodges, 1993, Section 7).

name from the vocabulary to relation in the structure of the same arity. Thanks to the interpretation, we can evaluate a formula over the vocabulary in any structure over this vocabulary.

Consider two structures $\mathbb{A}, \mathbb{B}$ over the same vocabulary. An *embedding* $f : \mathbb{A} \to \mathbb{B}$ is any injective function from the universe of $\mathbb{A}$ to the universe of $\mathbb{B}$ which preserves and reflects the relations in the following sense

$$\underbrace{R(a_1, \ldots, a_n)}_{\text{in } \mathbb{A}} \quad \Leftrightarrow \quad \underbrace{R(f(a_1), \ldots, f(a_n))}_{\text{in } \mathbb{B}}.$$

An *isomorphism* is the special case of an embedding where the function is a bijection. An *embedded substructure* of a structure is defined to be any structure that embeds into it. A *substructure* is the special case where the embedding is simply an inclusion map. We will be mainly interested in (embedded or not) substructures that are finite, i.e. have finite universes. Here is the fundamental definition for this chapter.

Definition 6.1 (Homogeneous structure)**.** A structure is called *homogeneous* if every isomorphism between finite substructures extends to a full automorphism of the entire structure.

Here is a diagram that describes the above definition.

$$\forall\exists \qquad \begin{tikzcd} \mathbb{B} \arrow[r, "\text{isomorphism}"] \arrow[d, "\text{subset}"'] & \mathbb{C} \arrow[d, "\text{subset}"] \\ \mathbb{A} \arrow[r, "\text{automorphism}"'] & \mathbb{A} \end{tikzcd}$$

Example 42. The equality atoms and the ordered atoms $(\mathbb{Q}, <)$ are homogeneous. Let us do the proof for the ordered atoms. A finite substructure is the same as a choice of d rational numbers $x_1 < \cdots < x_d$. Any two such choices will be isomorphic, assuming the same dimension d. If we take two such choices, they will be in the same orbit, i.e. the isomorphism will extend to an automorphism. □

Example 43. Consider the structure which consists of finite subsets of natural numbers, equipped with a binary relation for subset inclusion:

$$\mathbb{A} = (\mathsf{P}_{\text{fin}}(\mathbb{N}), \subseteq).$$

We will show that this structure is not homogeneous. Consider a finite substructure $\mathbb{B}$ that has only one element, namely the empty set, and another finite substructure $\mathbb{C}$ that also has only one element, namely the singleton set $\{1\}$. As finite structures, they are isomorphic – the subset relation connects the unique element with itself in both of them. However, there is no automorphism of $\mathbb{A}$ that maps the empty set to a nonempty set. This structure is also not oligomorphic, because it has infinitely many orbits already in $\mathbb{A}^1$, namely sets of different finite sizes will be in different orbits. □

In principle, oligomorphism and homogeneity are incomparable notions, as explained in the following two examples.

Example 44. [Oligomorphic $\not\Rightarrow$ homogeneous] Every finite structure is oligomorphic, but not every finite structure is a homogeneous. For example, consider the three element path

Let the vertices be $1, 2, 3$. The substructures $\{1\}$ and $\{2\}$ are isomorphic, but there is no automorphism that maps 1 to 2. □

Example 45. [Homogeneous $\not\Rightarrow$ oligomorphic] Consider an infinite structure which has one unary relation for every possible singleton. This structure has no automorphism, and therefore it has infinitely many orbits. However, it is homogeneous for the trivial reason that the only isomorphism between finite substructures are between identical subsets. □

However, the differences exhibited in the above examples are rather superficial. Every oligomorphic structure can be made homogeneous one by adding (infinitely many) relations: we can simply add a d-ary relation for every orbit in $\mathbb{A}^d$. For the converse implication, the following theorem shows that most reasonable homogeneous structures are in fact oligomorphic.

Theorem 6.2. *If a structure is homogeneous, then it is oligomorphic if and only if*

(*) *for every $d \in \{0, 1, \ldots\}$, it has finitely many substructures of size d, up to isomorphism.*

Proof. If we take tuples $(a_1, \ldots, a_d)$ and $(b_1, \ldots, b_d)$ that are in the same orbit, then the function $a_i \mapsto b_i$ is an isomorphism between the substructures generated by the tuples. Therefore, if there are finitely many orbits in $\mathbb{A}^d$, then there are finitely many kinds of substructures of size d, up to isomorphism, which proves the left-to-right implication. For the converse implication, we use homogeneity: the orbit of a tuple is uniquely determined by the isomorphism type of the induces substructure, and the order and repetitions of the elements in the tuple, which can be chosen in finitely many ways. □

A corollary of the above theorem is that for finite vocabularies, homogeneity implies oligomorphism. This is because condition (*) will automatically hold in the presence of a finite vocabulary.

One of the fundamental properties of oligomorphic structures was that equivariant relations were exactly those that could be defined in first-order logic, see Theorem 5.7. For homogeneous structures, quantifier-free formulas are enough.

Theorem 6.3. *Consider a homogeneous structure $\mathbb{A}$.*

1. *Two tuples in $\mathbb{A}^d$ are in the same orbit if and only if they satisfy the same quantifier-free formulas.*

2. *If $\mathbb{A}$ additionally satisfies condition (*) from Theorem 6.2, then the equivariant subsets of $\mathbb{A}^d$ are exactly those that are definable by quantifier-free formulas.*

Proof. We begin with the equivalence in the first item. The right-to-left implication is immediate: the truth-value of quantifier-free formula does not change when an automorphism is applied. Conversely, if two tuples of atoms satisfy the same quantifier-free formulas, then one can build an isomorphism between the substructures generated by them, which will extend to an automorphism of the entire structure by homogeneity.

The second item follows from the first item, and the observation that under assumption (*), there are finitely many possible quantifier-free formulas with d variables, up to logical equivalence. □

Exercises

Exercise 117. Which finite graphs are homogeneous?

6.2 The Fraïssé limit

In this section, we describe the Fraïssé limit, which is a way – in fact the only way – of constructing countable homogeneous structures. Before defining the Fraïssé limit, consider the following problem: for a class $\mathscr{A}$ of finite structures, find some (possibly infinite) structure $\mathbb{A}$ such that

$$\mathscr{A} = \underbrace{\{ \mathbb{B} \mid \mathbb{B} \text{ is a finite structure that embeds into } \mathbb{A} \}}_{\text{this is called the } \textit{age} \text{ of the structure } \mathbb{A}}.$$

For example, if $\mathscr{A}$ is the class of all finite structures over an empty vocabulary, then it is the age of any infinite structure over the empty vocabulary. If $\mathscr{A}$ is the class of finite total orders, then it is the age of any infinite total order, such as

$$(\mathbb{N}, <) \quad (\mathbb{Z}, <) \quad (\mathbb{Q}, <) \quad (\mathbb{R}, <).$$

However, if we want the total order to be countable and homogeneous, then the only choice is the rational numbers. Finally, not every class arises as the age of some structure. A necessary condition is that every two structures from $\mathscr{A}$ can be embedded into a single structure from $\mathscr{A}$, since this is a property that will hold for the age of a single structure $\mathbb{A}$. For this reason, the class

$$\mathscr{A} = \{ G \mid G \text{ is a graph with at most 10 edges } \}$$

is not the age of any structure. The purpose of this chapter is to identify conditions which guarantee that $\mathscr{A}$ can be obtained as the age of some structure, and furthermore we want this structure to be homogeneous. These conditions are described in terms of amalgamations, so we begin by defining amalgamation.

Definition 6.4 (Amalgamation)**.** An *instance of amalgamation* is two embeddings with a common source:

$$\begin{tikzcd} & \mathbb{A} \arrow[dl, "f_1"'] \arrow[dr, "f_2"] & \\ \mathbb{B}_1 & & \mathbb{B}_2 \end{tikzcd} \tag{6.1}$$

A *solution* of the instance is a structure $\mathbb{C}$ and two embeddings g_1, g_2 such that the following diagram commutes:

$$\begin{tikzcd} & \mathbb{A} \arrow[dl, "f_1"'] \arrow[dr, "f_2"] & \\ \mathbb{B}_1 \arrow[dr, "g_1"'] & & \mathbb{B}_2 \arrow[dl, "g_2"] \\ & \mathbb{C} & \end{tikzcd} \tag{6.2}$$

Definition 6.5 (Fraïssé class)**.** A *Fraïssé class* is a class of finite structures over a common vocabulary which is closed under isomorphism, substructures, and also:

- it is *closed under amalgamation*, which means that for every instance of amalgamation which uses structures from the class, there is a solution which also uses a structure from the class.

A Fraïssé class is called *countable* if it has countably many structures, up to isomorphism. We are now ready to state the Fraïssé theorem, which says that Fraïssé classes are in one-to-one correspondence with countable homogeneous structures.

Theorem 6.6 (Fraïssé Theorem)**.** *The map*

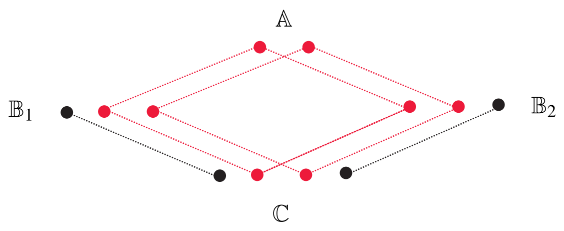
$$\mathbb{A} \quad \mapsto \quad \textit{age of } \mathbb{A}$$

is a bijection between countable homogeneous structures (modulo isomorphism) and countable Fraïssé classes. In other words:

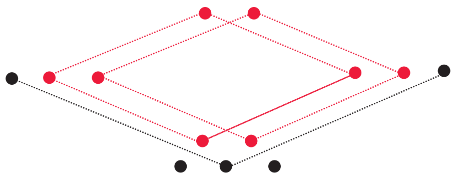
1. *the age of every countable homogeneous structure is a countable Fraïssé class; and*
2. *every countable Fraïssé class is obtained this way; and*
3. *if two countable homogeneous structures have the same age, then they are isomorphic.*

The inverse of the age operation, i.e. the map which inputs a Fraïssé class and outputs the corresponding countable homogeneous structure (which is unique up to isomorphism thanks to the above theorem), is called the *Fraïssé limit*. Before proving Theorem 6.6, we give some examples and non-examples of Fraïssé classes. In all these examples, closure under substructures and isomorphism is immediate, and only amalgamation need be discussed.

Example 46. Consider the class of all finite structures over an empty vocabulary (in which case formulas can talk only about equality). This class is closed under amalgamation, by taking the disjoint union of two sets with a common subset. Here is an example of an instance of amalgamation and its solution:

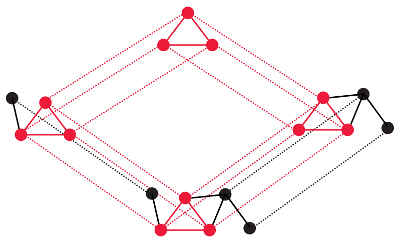


When drawing amalgamation diagrams, we use the red colour for elements of $\mathbb{A}$. In general, the same instance might have several solutions. Here is an example of a different solution to the instance above:



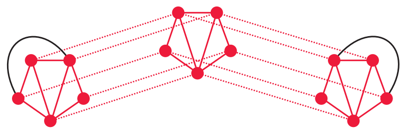
In fact, the above instance has infinitely many solutions (because the solution can be arbitrarily large). Note how the second solution uses the same black element as the target of both black nodes in the second row. □

Example 47. Consider the class of finite undirected graphs. In other words, this is the class of all finite structures over a vocabulary which has one binary relation that is required to be symmetric and irreflexive. This class is closed under amalgamation (the same argument works for directed graphs), by taking the disjoint union of two directed graphs with a common induced subgraph. Here is an example:



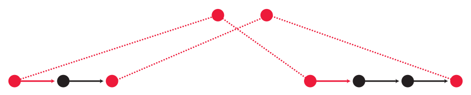
As in Example 46, there are also other solutions to the above instance. More generally, for every relational vocabulary, the class of all finite structures over this vocabulary is closed under amalgamation. In particular, by Theorem 6.6, each of these classes has a Fraïssé limit. The limit for undirected graphs will be discussed in more detail in Section 6.3.1. □

Example 48. Consider the class of finite planar graphs. To simplify this example, we assume that a graph is modelled (unlike in Example 47) as a structure where the universe is vertices and edges, and there is a binary incidence relation between edges and vertices. (This way of modelling a graph means that an embedding can add edges without adding vertices.) The class is not closed under amalgamation. Here is an instance without a solution:



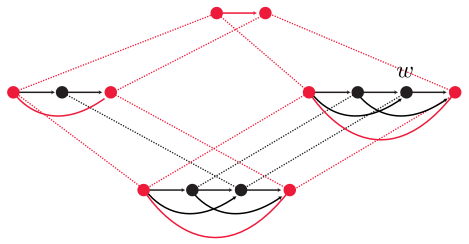
Any hypothetical solution to the above instance would have the 5-clique as a minor, and would therefore not be planar. A similar but more elaborate example would show failure of amalgamation for planar graphs under the modelling of graphs used by Example 47, where the universe of the structure is the vertices and there is a binary relation for the edges. □

Example 49. Consider directed graphs where the edge relation is a partial successor, i.e. vertices have out-degree and in-degree at most one, and no loops. The class is not closed under amalgamation, here is an instance without a solution:



□

Example 50. Consider the class of finite total orders. This class is closed under amalgamation. Here is an example of an instance of amalgamation and its solution:



□

We now begin the proof of the Fraïssé Theorem. We first show item 1, which says that the age of a countable homogeneous structure is a Fraïssé class. We prove a slightly stronger result, which does not assume countability.

Lemma 6.7. *For every homogeneous structure, not necessarily countable, its age is a Fraïssé class.*

Proof. The only nontrivial part is amalgamation. Let $\mathbb{H}$ be a homogeneous structure. Consider an instance of amalgamation which uses structures that embed into $\mathbb{H}$, as in the following diagram (all arrows are embeddings):

$$\begin{tikzcd}
 & \mathbb{A} \arrow[dl, "f_1"'] \arrow[dr, "f_2"] & \\
\mathbb{B}_1 \arrow[d, "h_1"] & & \mathbb{B}_2 \arrow[d, "h_2"] \\
\mathbb{H} & & \mathbb{H}
\end{tikzcd}$$

The diagram distinguishes the targets of h_1 and h_2 because the embeddings $h_1 \circ f_1$ and $h_2 \circ f_2$ need not be the same embedding of $\mathbb{A}$ in $\mathbb{H}$. However, the images of both of these embeddings are isomorphic finite substructures of $\mathbb{H}$. Therefore, by homogeneity there is an automorphism π which extends this partial automorphism. In other words, the following diagram commutes:

$$\begin{tikzcd} & \mathbb{A} \arrow[dl, "f_1"'] \arrow[dr, "f_2"] & \\ \mathbb{B}_1 \arrow[d, "h_1"] & & \mathbb{B}_2 \arrow[d, "h_2"] \\ \mathbb{H} \arrow[rr, "\pi"] & & \mathbb{H} \end{tikzcd}$$

If we restrict the right copy of $\mathbb{H}$ to the union of the images of the maps h_2 and $\pi \circ h_1$, then we get a solution of amalgamation. □

If a homogeneous structure is countable, then it has countably many embedded finite substructures. Therefore, by the above lemma, the age of a countable homogeneous structures is a countable Fraïssé classes. We now establish item 3 in the theorem, which says that the age uniquely identifies a countable homogeneous structure.

Lemma 6.8. *A countable structure $\mathbb{H}$ is homogeneous if and only if:*

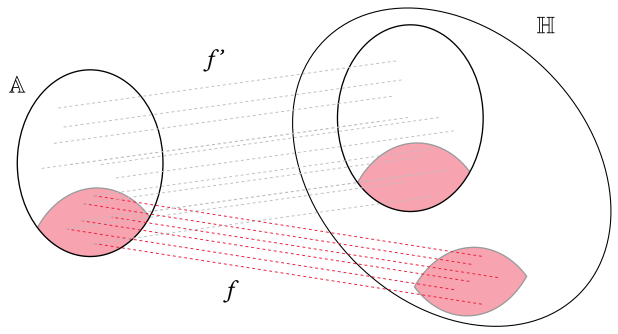
(*) If $\mathbb{A}, \mathbb{B}$ are finitely generated substructures of $\mathbb{H}$ then

$$\forall\exists \quad \begin{tikzcd} \mathbb{B} \arrow[r, "g"] \arrow[dr, "f"'] & \mathbb{A} \arrow[d, "h"] \\ & \mathbb{H} \end{tikzcd}$$

Furthermore, countable homogeneous structures with the same age are isomorphic.

Proof.

- *Homogeneous structures satisfy (*).* Let g, f be as in (*). We assume without loss of generality that g is an inclusion. Let f' be an embedding of $\mathbb{A}$ into $\mathbb{H}$, which exists by the assumption that $\mathbb{A}$ is a substructure. Here is a picture:

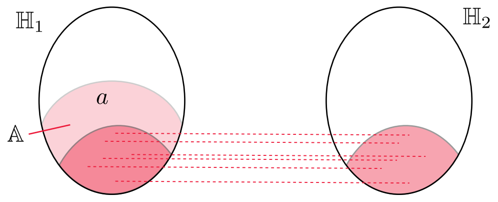


By following the inverse of f and then f', we get a partial automorphism between two finitely generated substructures of $\mathbb{H}$, namely the two red parts on the right. By homogeneity, this partial automorphism extends to a full automorphism. The function f' composed with the inverse of that automorphism is the desired embedding.

- *Structures satisfying (*) are homogeneous.* Here we use countability. The following claim, in the special case of $\mathbb{H} = \mathbb{H}_1 = \mathbb{H}_2$, shows that $\mathbb{H}$ is homogeneous.

  Claim 6.9. *Let $\mathbb{H}_1, \mathbb{H}_2$ be countable structures with the same age. If both satisfy (*), then every partial isomorphism between finite substructures of $\mathbb{H}_1$ and $\mathbb{H}_2$ extends to a full isomorphism.*

  Proof. Let f be an isomorphism between structures in the ages of $\mathbb{H}_1$ and $\mathbb{H}_2$, respectively, and let a be an element of $\mathbb{H}_1$. Let $\mathbb{A}$ be the substructure of $\mathbb{H}_1$ whose universe is a plus the domain of f. Here is a picture:



  The structure $\mathbb{A}$ is in the age of $\mathbb{H}_1$, and therefore by the assumption of the claim it embeds into $\mathbb{H}_2$. By (*), f extends to an embedding of $\mathbb{A}$ into $\mathbb{H}_2$. This argument, and a symmetric one where a is in $\mathbb{H}_2$, establishes that:

  (**) For every isomorphism between structures in the ages of $\mathbb{H}_1$ and $\mathbb{H}_2$, respectively, and every element a of either $\mathbb{H}_1$ or $\mathbb{H}_2$, the partial isomorphism can be extended to be defined also on a.

  The conclusion of the claim follows from (**) using a back-and-forth construction. Define inductively a sequence of partial isomorphisms between finitely generated substructures of $\mathbb{H}_1$ and $\mathbb{H}_2$, such that the next one extends the previous one, and every element of both structures appears eventually in the source or target of a partial isomorphism from the sequence. The full isomorphism is then the limit of these partial isomorphisms. □

- *Homogeneous structures are uniquely determined by their finitely generated substructures.* By Claim 6.9 applied to the empty partial isomorphism between $\mathbb{H}_1$ and $\mathbb{H}_2$, we see that countable homogeneous structures are uniquely determined – up to isomorphism – by their age.

□

To finish the proof of Fraïssé Theorem, we need to show item 2.

Lemma 6.10. *Every countable Fraïssé class $\mathscr{A}$ arises as the age of some countable homogeneous structure.*

Proof. Choose some enumeration

$$\mathbb{A}_1, \mathbb{A}_2, \ldots \tag{6.3}$$

of the structures in $\mathscr{A}$, which represents every structure up to isomorphism. We define a sequence

$$\mathbb{H}_0 \subseteq \mathbb{H}_1 \subseteq \cdots \tag{6.4}$$

of structures in $\mathscr{A}$ as follows. Choose the first structure $\mathbb{H}_0$ arbitrarily, say the empty structure. A new structure is obtained by applying the following claim.

Claim 6.11. *Suppose that $\mathbb{H}_n$ is already defined. There is a structure $\mathbb{H}_{n+1} \supseteq \mathbb{H}_n$ in $\mathscr{A}$ such that for every instance of amalgamation*

$$\begin{tikzcd} & \mathbb{A} \arrow[dl, hook, "f"'] \arrow[dr, hook, "g"] & \\ \mathbb{B} & & \mathbb{H}_n \end{tikzcd}$$

where both $\mathbb{A}, \mathbb{B}$ are among the first n structures in the enumeration of $\mathscr{A}$, there is a solution of the form

$$\begin{tikzcd} \mathbb{B} \arrow[dr, hook, "f'"'] & & \mathbb{H}_n \arrow[dl, hook', "inclusion"] \\ & \mathbb{H}_{n+1} & \end{tikzcd}$$

Proof. There are finitely many possible instances of amalgamation as in the claim, because $\mathbb{A}$ and $\mathbb{B}$ can be chosen in finitely many ways, and there are finitely many possible embeddings between two finite structures. Let m be the number of instances. By induction on $i \in \{1, \ldots, m\}$, we create a structure $\mathbb{H}^i_{n+1}$ that solves the first i instances; once we have done this we can use $\mathbb{H}^m_{n+1}$ as the solution for all instances. The induction step is proved by applying amalgamation to the previous solution. □

Define $\mathbb{H}$ to be the limit (i.e. union) of the sequence $\mathbb{H}_1, \mathbb{H}_2, \ldots$. By construction, $\mathbb{H}$ satisfies condition (*) from Lemma 6.8, and is therefore homogeneous.

To complete the proof, we justify that the age of $\mathbb{H}$ is exactly $\mathscr{A}$. Every finite structure that embeds into the limit $\mathbb{H}$ must embed into some $\mathbb{H}_n$, and is therefore in $\mathscr{A}$, because $\mathbb{H}_n \in \mathscr{A}$ and the class is closed under substructures. Therefore, the age of $\mathbb{H}$ is contained in $\mathscr{A}$. Let us prove the converse inclusion. Suppose that $\mathbb{A} \in \mathscr{A}$. At some point n in the enumeration, we have seen both $\mathbb{A}$ and the empty structure. Therefore, $\mathbb{H}_{n+1}$ will contain a solution to an instance of amalgamation where the empty structure is embedded into both $\mathbb{A}$ and $\mathbb{H}_{n+1}$. This means that $\mathbb{H}_{n+1}$ contains an isomorphic copy of $\mathbb{A}$. □

This completes the proof of Fraïssé Theorem.

Computability. The Fraïssé limit not only exists, but under mild assumptions on the Fraïssé class, it can be computed. What does it mean to compute an infinite structure? This is formalized in the following theorem, which shows that one perform basic computational operations, such as counting orbits, deciding the first-order theory, and representing elements.

Theorem 6.12. *Let $\mathscr{A}$ be a Fraïssé class such that:*

(*) there is an algorithm that inputs d and returns a finite list of all structures that represents all structures of size d in $\mathscr{A}$, up to isomorphism.

Then its Fraïssé limit, call it $\mathbb{A}$, has the following properties:

1. *it is oligomorphic, and given d one can compute the number of orbits in $\mathbb{A}^d$;*
2. *it has effective quantifier elimination, i.e. for every first-order formula, one can compute an equivalent one that is quantifier-free;*
3. *there is a function $\rho : 2^* \to \mathbb{A}$, called a* representation, *which has the following properties (when atoms are used in algorithms, they are represented as strings using the representation):*
   (a) *every atom is represented by at least one string;*
   (b) *given a first-order formula $\varphi(x_1, \ldots, x_d)$ and $a_1, \ldots, a_d \in \mathbb{A}$, one can decide if*
   $$\mathbb{A} \models \varphi(a_1, \ldots, a_d);$$
   (c) *given two tuples in $\mathbb{A}^d$, decide if they are in the same orbit.*

Proof. We begin with item 1. By assumption (*), there are finitely many substructures of size d in $\mathscr{A}$, up to isomorphism. Therefore, $\mathbb{A}$ is oligomorphic by Theorem 6.2. An orbit in $\mathbb{A}^d$ is the same thing as a substructure with at most d elements, together with a list of length d that covers all of its elements, possibly with repetitions. We can count such objects, up to isomorphism, using assumption (*).

We now show item 2, about quantifier elimination. We assume that "true" and "false" are quantifier-free formulas; these will be the only possible formulas when we apply the quantifier elimination to a sentence, i.e. a formula without free variables. The proof is by induction on the size of the formula. The only non-trivial case is eliminating a single quantifier, say an existential one (because eliminating a universal quantifier reduces to this case by De Morgan's laws):

$$\exists x \underbrace{\varphi(x_1, \ldots, x_n, x)}_{\text{quantifier free}}.$$

The inner formula φ can be seen as describing structures with $n + 1$ distinguished elements; with the distinguished elements not being necessarily pairwise distinct. Let us write $\mathscr{A}_\varphi$ for the corresponding structures, i.e. this is the class

$$\mathscr{A}_\varphi = \{(\mathbb{A}, \overbrace{a_1, \ldots, a_n}^{\bar{a}}, a) : \mathbb{A} \in \mathscr{A} \text{ and } a_1, \ldots, a_n, a \text{ are elements that satisfy } \varphi\}.$$

Up to isomorphism, the above class is finite and can be computed thanks to assumption (*). Because the Fraïssé limit is homogeneous, a tuple $\bar{a}a$ in the Fraïssé limit satisfies φ if and only if $\mathscr{A}_\varphi$ contains the substructure generated by $\bar{a}$ (together with the distinguished $\bar{a}$). Define $\mathscr{A}_{\exists x\varphi}$ to be the following projection of $\mathscr{A}_\varphi$: for each $(\mathbb{A}, \bar{a}a) \in \mathscr{A}_\varphi$, remove the last element a from the list of distinguished elements. A tuple $\bar{a}$ in the Fraïssé limit satisfies the quantified formula $\exists x\varphi$ if and only if $\mathscr{A}_{\exists x\varphi}$ contains the substructure generated by $\bar{a}$ (together with the distinguished $\bar{a}$). This property can be expressed using a quantifier-free formula.

We now prove the last item 3, about the representation. Here, we revisit the construction of the Fraïssé limit in the proof of Lemma 6.10. In that proof, we started off with an enumeration, see (6.3), which represents all structures in $\mathscr{A}$ up to isomorphism. Thanks to assumption (*), we can assume that this enumeration is effective in the following sense: there is an algorithm that inputs n and returns the n-th structure $\mathbb{A}_n$ in the enumeration. The construction in Claim 6.11 preserves this notion of effectiveness, and therefore also the sequence $\mathbb{H}_n$ is effective. Since the Fraïssé limit is defined to be the union of the latter enumeration, it follows that the Fraïssé limit is effective in the sense that one can define a surjective representation $\rho : 2^* \to \mathbb{H}$ that allows us to test if a given tuple of elements satisfies a given relation from the vocabulary. (The string representing an element from $\mathbb{H}$ stores the following information: at which stage n did the element appear in $\mathbb{H}_n$, and which element of $\mathbb{H}_n$ it is.) If we can decide the relations from the vocabulary, then we can decide quantifier-free formulas, and so we can also decide first-order formulas, as required by item 3b, thanks to the previous item about quantifier elimination. The last part of the theorem, in item 3c, is about deciding if two tuples are in the same orbit. By Theorem 6.3, we know that two tuples are in the same orbit if and only if they satisfy the same quantifier-free formulas. Although the vocabulary is in principle infinite, we can use assumption (*) to show that for every d, there is some finite part of the vocabulary such that quantifier-free formulas using only that part are enough to distinguish different orbits in $\mathbb{A}^d$. In combination with the previous observations about deciding quantifier-free formulas, we can get an effective criterion for checking if two tuples from $\mathbb{A}^d$ are in the same orbit. □

All Fraïssé classes discussed in this chapter satisfy the assumptions of the above theorem. In particular, the corresponding Fraïssé limit will have a decidable first-order theory, thanks to the special case of item 3b for formulas without free variables. Therefore, we can apply Theorem 5.10 to decide graph reachability. In the Chapter 7, we will see many other examples of algorithms, beyond graph reachability, which can be used for atoms that arise a Fraïssé limit. Before we do that, however, we present several interesting examples of Fraïssé limits, which will illustrated the scope of applicability for the algorithms that will be presented in Chapter 7.

Exercises

Exercise 118. Consider the class of all finite partial orders, i.e. binary relations that are reflexive and transitive. Show that this class is closed under amalgamation.

Exercise 119. Are series parallel graphs closed under amalgamation?

Exercise 120. Show a Fraïssé class where solutions to amalgamation necessarily violate the following condition:

(*) the intersection of the images of g_1 and g_2, as per diagram (6.2), is exactly the image of $\mathbb{A}$.

Exercise 121. Assume a finite relational vocabulary. Suppose that $\mathscr{A}$ is a class of structures that satisfies the assumptions of Theorem 6.6, and let $\mathbb{A}$ be its Fraïssé limit. Show that if membership in $\mathscr{A}$ is decidable, $\mathbb{A}$ is an effective structure.

Exercise 122. Let $\mathscr{A}$ be a class of structures over a finite vocabulary, possibly including functions, which:

1. has decidable membership;
2. is closed under substructures, isomorphism and amalgamation;
3. given $k \in \mathbb{N}$ one can compute some $n \in \mathbb{N}$ such that structures in $\mathscr{A}$ with k generators have size at most n.

Show that the Fraïssé limit of $\mathscr{A}$ has a decidable first-order theory with constants and a computable Ryll-Nardzewski function.

Exercise 123. Define *monadic second-order logic* (MSO) to be the extension of first-order logic where one can also quantify over sets of vertices. A famous result on MSO is Rabin's Theorem[2], which says that the structure $\{0,1\}^*$ equipped with functions $x \mapsto x0$ and $x \mapsto x1$ has decidable MSO theory, i.e. one can decide if a sentence of MSO is true in it. Show that $(\mathbb{Q}, <)$ has decidable MSO theory.

Exercise 124. If Σ is a finite alphabet. We model a word $w \in \Sigma^*$ as a structure, where the universe is positions in w, there is a binary predicate $<$ for the order relation, and for every label $a \in \Sigma$ there is a unary predicate $a(x)$. We denote the vocabulary used for this structure by $\Sigma_<$. Show that for every regular language $L \subseteq \Sigma^*$ there is a homogeneous structure $\mathbb{A}$ over a vocabulary containing $\Sigma_<$ such that the age of $\mathbb{A}$ after restricting to $\Sigma_<$ is exactly the structures corresponding to L.

6.3 Examples of homogeneous atoms

We end this chapter with three extended examples of homogeneous structures.

6.3.1 The random graph

We begin with the Fraïssé limit of all finite undirected graphs. As shown in Example 47, this is a Fraïssé class, and therefore it has a Fraïssé limit. Call this limit the *random graph*. The name is justified by the following observation.

Theorem 6.13. *Consider a countably infinite undirected graph, where each the presence/absence of an edge is chosen independently with equal probability one half*[3]*. Almost surely (i.e. with probability one) this graph is isomorphic to the random graph.*

[2]For an introduction to MSO and Rabin's Theorem, see (Thomas, 1990, Theorem 6.8).

[3]The conclusion of the theorem would not change if we used a different distribution, e.g. there would be an edge with probability 0.99.

Proof. Let us write $\mathbb{H}$ for the graph that is chosen randomly. For a finite graph G, and a function h from vertices of an induced subgraph $F \subseteq G$ to vertices of $\mathbb{H}$, consider the event: "either h is not an embedding, or it can be extended to an embedding of G". This event happens almost surely because failing the event would require infinitely many independent random events that go wrong. Since there are countably many choices of $F \subseteq G$ and functions h, up to isomorphism, it follows that almost surely the graph $\mathbb{H}$ satisfies condition (*) of Lemma 6.8, and therefore it is isomorphic to the random graph. □

Since the class of finite undirected graphs is clearly countable, its Fraïssé limit is oligomorphic and has all of the computability properties in the conclusion of Theorem 6.12. It follows that problems such as graph reachability or automaton emptiness are decidable, assuming that the inputs are pof automata, or quotiented pof automata.

Exercises

Exercise 125. Assume that the atoms are the random graph. Is the language

$$\{a_1 \cdots a_n \in \mathbb{A} : \text{the subgraph induced by } a_1, \ldots, a_n \text{ is connected}\}$$

recognised by a nondeterministic orbit-finite automaton?

Exercise 126. Assume that the atoms are the random graph. Give examples and non-examples of graph properties X such that the following language is recognised by a nondeterministic orbit-finite automaton:

$$L_X = \{a_1 \cdots a_n : \text{ the subgraph induced by } a_1, \ldots, a_n \text{ satisfies } X\}.$$

To recognise L_X, the automaton should be prepared for an arbitrary enumeration of the vertices of the graph, possibly with repetitions.

Exercise 127. Assume that the atoms are the random graph. Show that there is no finitely supported total order on the random graph.

Exercise 128. Show that there is no orbit-finite automaton, even nondeterministic, which recognises the language of width k path decompositions.

Exercise 129. Assume that the atoms are the random graph. Show that for every MSO formula $\varphi(x_1, \ldots, x_n)$ with free variables that represent vertices (not sets of vertices) there is formula of first-order logic which is equivalent on the random graph. Nevertheless, there is no algorithm which computes such equivalent formulas.

Exercise 130. Assume that the atoms are the random graph. Show that solving equations, as discussed in Section 7.4, is undecidable.

6.3.2 Bit vectors

This section is about the Fraïssé limit of finite vector spaces over the two element field. These atoms will also be discussed in Chapter 8, where we will show that, over these atoms, deterministic polynomial time orbit-finite Turing machines are weaker than the nondeterministic ones.

For the rest of this section, we only study vector spaces over the two element field, so we say vector space with the implicit assumption that the underlying field is the two element field. Every vector space of finite dimension (which is equivalent to having finitely many vectors) is isomorphic to

$$(\{0,1\}^d, +) \qquad \text{for some } d \in \{1,2,\ldots\}$$

where addition is modulo two. We model a – possibly infinite – vector space V as a structure over the following infinite vocabulary: for every $d \in \{0,1,\ldots\}$ there is a relation which selects d-tuples of vectors that are linearly independent. Here, a d-tuple $\bar{v} \in V^d$ is called linearly independent if it does not satisfy any non-trivial dependency

$$\alpha_1 v_1 + \cdots + \alpha_d v_d = 0,$$

where non-trivial means that at least one of the coefficients α_i is nonzero. In particular, if the tuple contains a repetition, then it is linearly dependent. If the vector space has finite dimension, then the relation will not select any tuple, once d exceeds the dimension.

It is not hard to see that finite vector spaces are a Fraïssé class. Embeddings are the same thing as injective linear maps. To amalgamate two vector spaces, of dimensions say d_1 and d_2, one needs a vector space of dimension $\max(d_1, d_2)$. Therefore, there is a Fraïssé limit of the finite vector spaces; and thanks to Theorem 6.12 this limit is a countably oligomorphic structure.

One can also construct the Fraïssé limit explicitly. The Fraïssé limit must be a vector space, since any violation of the vector space axioms would need to happen already in a finitely generated substructure. Since the Fraïssé limit is countable, its dimension must be countable, and since the Fraïssé limit embeds all finite vector spaces, its dimension must be infinite. Therefore, the Fraïssé limit is a vector space of countably infinite dimension. Up to isomorphism, there is a unique vector space like this. One way of representing this unique vector space is as follows. The elements are *bit vectors*, which are defined to be ω-sequences of zeroes and ones which have finitely many ones (if we allowed infinitely many ones, the resulting vector spaces would have uncountable dimension). By ignoring trailing zeroes, a bit vector can be represented as a finite sequence, such as 00101001. Define the *bit vector atoms* to be the bit vectors equipped with a function for coordinatewise addition modulo two:

$$01011 + 11001 = 1001.$$

An example basis consists of bit vectors which have a 1 on the n-th coordinate:

$$1, 01, 001, 0001, \ldots.$$

Another example of a basis is

$$1, 11, 111, 1111, \ldots.$$

Least supports. We prove below that for the bit vector atoms, a version of the Least Support Theorem is true. For bit vectors, least supports are not unique as sets, but as spanned subspaces. For example, the pair of atoms $(01, 10)$ is supported by itself, but it is also supported by $(11, 01)$. More generally, the following lemma shows that supporting and spanning are the same concepts, when talking about tuples of atoms.

Lemma 6.14. *Assume the bit vector atoms. An atom tuple $\bar{a}$ supports an atom tuple $\bar{b}$ if and only if all atoms in $\bar{b}$ are spanned by $\bar{a}$.*

Proof. The right-to-left implication is immediate. For the converse implication, suppose that some atom in $\bar{b}$ is not spanned by $\bar{a}$. By the Steinitz exchange lemma, this atom can be mapped to some other atom by a $\bar{a}$-automorphism. □

We are now ready to state the Least Support Theorem for bit vector atoms.

Theorem 6.15 (Least Support Theorem)**.** *Assume the bit vector atoms. Let X be a set equipped with an action of atom automorphisms. If $x \in X$ has finite support, then there exists a tuple $\bar{a}$ of atoms which supports x, and which is least in the sense that if $\bar{b}$ supports x, then $\bar{a}$ supports $\bar{b}$.*

Proof. Without loss of generality, we assume that X has one orbit. The proof follows the same lines as the proof for the equality atoms, except that vector independence plays the role of equality. Let us write $\mathbb{A}^{\langle d \rangle}$ for the set of d-tuples of atoms which are linearly independent. This is a one orbit set.

Lemma 6.16. *There is an equivariant function*

$$f : \mathbb{A}^{\langle d \rangle} \to X$$

which satisfies the following condition[4] *for every $\bar{a}, \bar{b} \in \mathbb{A}^{\langle d \rangle}$:*

$$f(\bar{a}) = f(\bar{b}) \quad \Rightarrow \quad \text{every atom in } \bar{a} \text{ is spanned by } \bar{b} \text{ and vice versa.}$$

Proof. We start with some function $f : \mathbb{A}^{\langle d \rangle} \to X$ that is equivariant, but which does not necessarily satisfy the condition in the lemma. Such a function can be found, by taking some tuple $\bar{a}$ of independent atoms that supports some element $x \in X$, and extending it to an equivariant function. We will now show that either f satisfies the condition, or the dimension d can be made smaller. By iterating this argument at most d times, we get the conclusion of the lemma.

Suppose that f violates the condition in the lemma, as witnessed by tuples $\bar{a}$ and $\bar{b}$, which have the same image under f but do not span each other. Some coordinates in $\bar{a}$ are spanned by $\bar{b}$, but at least one coordinate is not. Without loss of generality, we assume that the first i coordinates in the tuple $\bar{a}$ are not spanned by $\bar{b}$, and the remaining coordinates are spanned by $\bar{b}$. In other words, the tuple

$$(\underbrace{a_1, \ldots, a_i}_{\substack{\text{first } i \text{ atoms} \\ \text{in the tuple } \bar{a}}}, \underbrace{b_1, \ldots, b_d}_{\substack{\text{all atoms in} \\ \text{the tuple } \bar{b}}})$$

is linearly independent. Since the vector space $\mathbb{A}$ has infinite dimension, one can choose $a'_1, \ldots, a'_i \in \mathbb{A}$ which are linearly independent, and which are not spanned by $\bar{a}\bar{b}$. It follows that

$$(a_1, \ldots, a_i) \quad \overset{\pi}{\mapsto} \quad (a'_1, \ldots, a'_i)$$

[4]Condition (??), as well as the related condition from Lemma 4.13, is equivalent to saying that $\bar{a}$ and $\bar{b}$ have the same algebraic closure, in the model theory sense, see (Hodges, 1993, Chapter 4).

holds for some atom automorphism π that fixes $\bar{b}$. Because $\bar{b}$ supports $f(\bar{b})$, which is the same as $f(\bar{b})$, we have

$$f(\bar{a}) = f(\pi(\bar{a})).$$

Since we have assumed that the last $d - i$ coordinates of $\bar{a}$ are supported by $\bar{b}$, it follows that the last $d - i$ coordinates in $\pi(\bar{a})$ are the same as in $\bar{a}$. Summing up, we have found two inputs for the function f, namely $\bar{a}$ and $\pi(\bar{a})$, which agree on the last $d - i$ coordinates, but which have independent atoms on the first i coordinates. By equivariance of f, this means that the first i coordinates in a tuple from $\mathbb{A}^{\langle d \rangle}$ can be replaced by fresh independent atoms without affecting the value of f. It follows that f does not depend on the first i coordinates, and hence we can lower the dimension d. □

Take the function f from the above lemma. This function is surjective, since an input orbit is mapped to an output orbit, and X is assumed to be a one-orbit set. We will show that if $\bar{a}$ is a least support, in the sense of the theorem, for $f(\bar{a})$. Indeed, suppose that $f(\bar{a})$ would be supported by some tuple $\bar{b}$ which does not span $\bar{a}$. Then there would be an atom automorphism π that would fix $\bar{b}$ – and therefore also the output of the function – but would map $\bar{a}$ to some tuple not spanned by $\bar{a}$. In this case, the inputs $\bar{a}$ and $\pi(\bar{a})$ would be a violation of the above lemma. □

Exercises

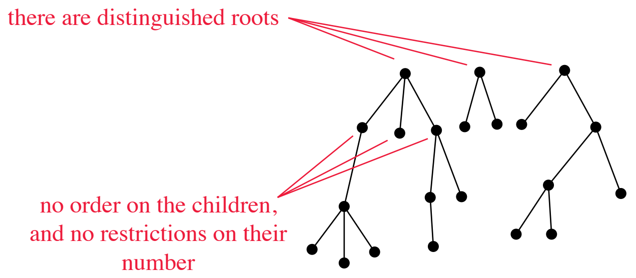
Exercise 131. Let $\mathbb{B}$ be the structure where the universe is the same as in the bit vector atoms, but we only have the independence predicate for dimension 3, i.e. there is a ternary predicate "the atoms a, b, c are linearly independent". Show that $\mathbb{B}$ has the same automorphisms as the bit vector atoms.

Exercise 132. Show that the structure $\mathbb{B}$ from Exercise 131 is not homogeneous.

Exercise 133. Consider vector spaces over the three element field, with the independence relations. Is the class of finite-dimensional vector spaces a Fraïssé class?

6.3.3 Trees and forests

In this section, we study the Fraïssé limit of trees and forests[5]. The trees and forests we study are rooted, unlabelled, and unordered, as explained in the following picture:

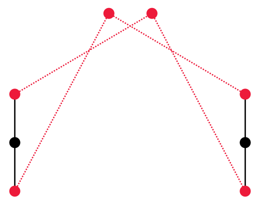

there are distinguished roots
no order on the children, and no restrictions on their number

[5]This section is based on Bojańczyk et al. (2013b).

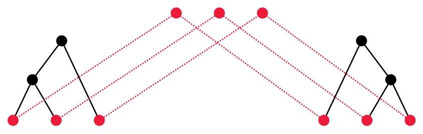
A tree is the special case of a forest when there is exactly one root.

The purpose of this section is to show that care is needed when choosing predicates and functions to model a combinatorial object, like a tree or forest, if we want to have a Fraïssé limit. The following list shows three ways of modelling trees as logical structures; only the third way will admit a Fraïssé limit. In all cases, the universe of the structure is the nodes of the tree.

1. There is a binary predicate for the parent relation. A finite forest is characterised by the requirement that each node has at most one parent. This way of modelling forests leads to a class that is not closed under amalgamation. Here is an instance of amalgamation that has no solution:



2. There is a binary predicate for the ancestor relation. A finite forest is characterised by the requirement that for every node, its ancestors are totally ordered. This way of modelling forests also leads to a class that is not closed under amalgamation. Here is an instance of amalgamation that has no solution:



3. We have a ternary relation

$$z = \text{closest common ancestor of } x \text{ and } y.$$

The class of trees modelled this way is closed under amalgamation, as illustrated in Figure 6.1. Therefore, it has a Fraïssé limit, which we call the *universal forest*. (This forest is connected, because by amalgamation we can connect any two forests.)

Exercises

Exercise 134. Assume the universal forest atoms. Find a finitely supported equivalence relation on the atoms which has infinitely many infinite equivalence classes.

Exercise 135. Assume the universal forest atoms. Show that one cannot find an infinite equivariant set X and an equivariant relation on it which is a total dense order. Equivariance is important here, since if we only want a finitely supported one then this is easily accomplished

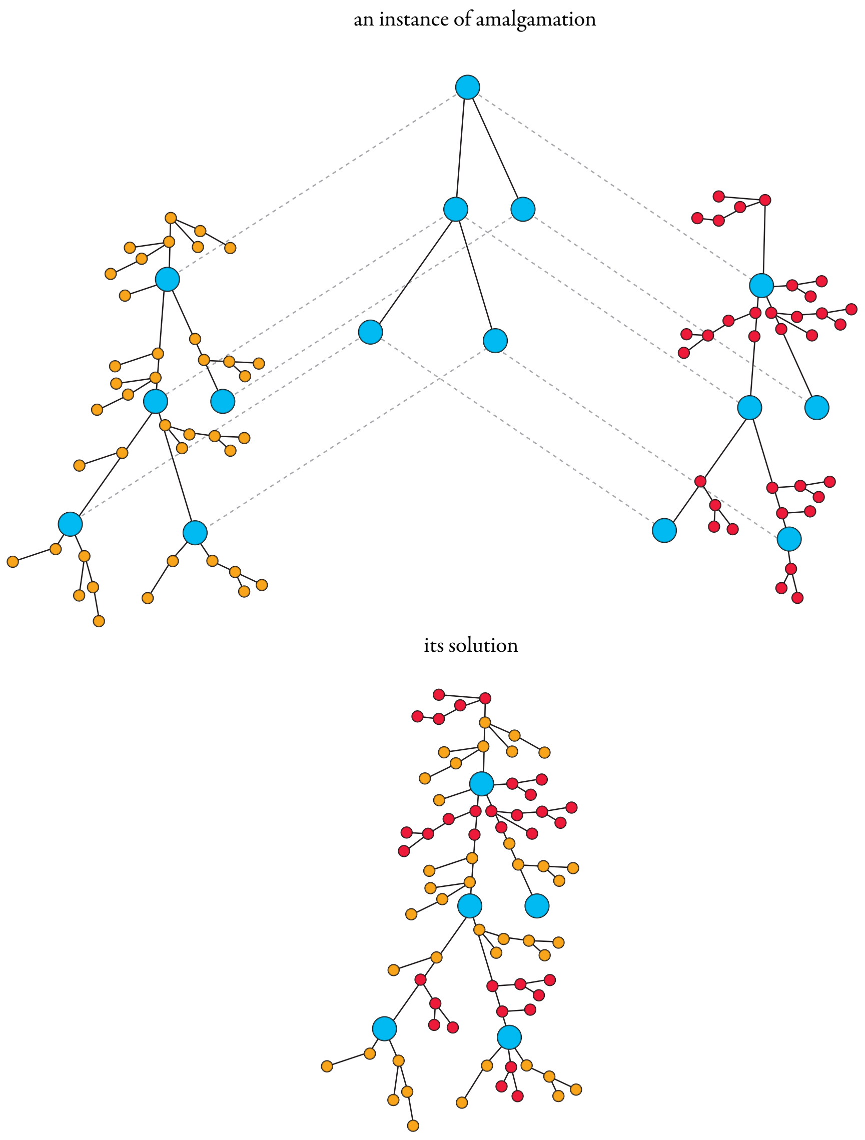


Figure 6.1: Amalgamation for forests with a closest common ancestor relation.

by taking the path connecting some two atoms $a < b$, and using the order inherited from the atoms.

Exercise 136. Show that the universal forest has decidable MSO theory.

Chapter 7

Algorithms on orbit-finite sets

In this chapter, we give examples that illustrate how algorithms can be generalized from finite sets to orbit-finite sets. For all algorithms in this chapter, we make the following assumptions about the atom structure.

Definition 7.1. A structure $\mathbb{A}$ is called *effectively oligomorphic* if:

1. it is oligomorphic and countable;
2. it has a decidable first-order theory;
3. given $d \in \{0, 1, \ldots\}$, the number of orbits in $\mathbb{A}^d$ can be computed.

These assumptions are satisfied by all atom structures that have been discussed so far, such as the equality atoms, the order atoms, the graph atoms, or the bit vector atoms.

7.1 Representing orbit-finite sets

To discuss algorithms, we need a finite representation of orbit-finite sets and their equivariant subsets. In the special case of polynomial orbit-finite sets, we already discussed such a representation in Section 5.2, when deciding graph reachability. This representation was defined for polynomial orbit-finite sets, but it extends naturally to (not necessarily polynomial) orbit-finite sets, as described below.

- *How do we represent an orbit-finite set?* By Theorem 5.12, every orbit-finite set admits an equivariant bijection with a quotiented pof set, and so we will use quotiented pof sets as our representation of orbit-finite sets. A quotiented pof set is of the form

$$\mathbb{A}^{d_1}_{/\sim_1} + \cdots + \mathbb{A}^{d_k}_{/\sim_k},$$

  where each $\sim_i$ is an equivariant partial equivalence relation on $\mathbb{A}^{d_i}$. By Theorem 5.7, each of the equivalence relations $\sim_i$ is necessarily first-order definable, and therefore it can be represented by a first-order formula. This formula has $2d_i$ variables, because it is a binary relation on $\mathbb{A}^{d_i}$. Summing up, an orbit-finite set is represented by a list of dimensions $d_1, \ldots, d_k \in \{0, 1, \ldots\}$, one for each component, together with a list of first-order formulas $\varphi_1, \ldots, \varphi_k$ that describe partial equivalence relations on these components. We would like the syntax to be decidable, which in this case means checking if the first-order formulas do indeed define partial equivalence relations. This can be formalized by writing a first-order sentence:

$$\bigwedge_{i \in \{1,\ldots,k\}} \forall \bar{x}, \bar{y}, \bar{z} \in \mathbb{A}^{d_i} \quad \underbrace{\varphi_i(\bar{x}, \bar{y}) \Leftrightarrow \varphi_i(\bar{y}, \bar{x})}_{\text{symmetry}} \quad \land \quad \underbrace{\varphi_i(\bar{x}, \bar{y}) \land \varphi_i(\bar{y}, \bar{z}) \Rightarrow \varphi_i(\bar{x}, \bar{z})}_{\text{transitivity}}.$$

Since the first-order theory is decidable for an effectively oligomorphic structure, we can check if the above formula is true, and so the syntax is decidable.

- *How do we represent an equivariant subsets of an orbit-finite set?* Apart from orbit-finite sets, we also need to represent their equivariant subsets (which themselves can be seen as new orbit-finite sets.) Suppose that we have an orbit-finite set as in the previous item. To describe an equivariant subset, we need to specify for each component $\mathbb{A}^{d_i}$ an equivariant subset, which is required to be stable under the equivalence relation $\sim_i$. This subset can be described by a first-order formula ψ_i, and stability can be formalized by a first-order sentence

$$\bigwedge_{i \in \{1,\ldots,k\}} \forall \bar{x} \in \mathbb{A}^{d_i} \quad \psi_i(\bar{x}) \quad \Rightarrow \quad \underbrace{\varphi_i(\bar{x}, \bar{x})}_{\substack{\text{each element} \\ \text{is in some} \\ \text{equivalence class}}} \quad \land \quad \underbrace{\forall \bar{y} \in \mathbb{A}^{d_i}\ \varphi_i(\bar{x}, \bar{y}) \Rightarrow \psi_i(\bar{y})}_{\substack{\text{subset is closed under replacing} \\ \text{elements with equivalent ones}}}.$$

In the rest of this chapter, we will present algorithms that operate on orbit-finite sets and their equivariant subsets, using the representation described above. For the moment, we need to care about the representation, and we will need to justify how various operations, such as Boolean operations or certain kinds of loops, can be implemented on this representation. Later on in this book, we will present a more principled approach, namely a programming language that takes care of all of these operations.

7.2 Representing elements of orbit-finite sets

In the previous section, we explained how we can represent orbit-finite sets and their subsets. It would also be desirable to represent individual atoms; for example this would be needed to use a representation of sets in terms of generators, as we did in Section 1.1.1 for the equality atoms. Before moving on to the algorithms, we discuss how individual elements can be represented. Of course such a representation should support certain basic operations, such as testing equality. This is formalized in the following definition, which uses the same three conditions as in item 3 of Theorem 6.12.

Definition 7.2 (Atom representation)**.** An *atom representation* for a structure $\mathbb{A}$ is a function $r : 2^* \to \mathbb{A}$ which has the following properties (when atoms are used in algorithms, they are represented as strings using the representation):

(a) every atom is represented by at least one string;

(b) given a first-order formula $\varphi(x_1, \ldots, x_d)$ and $a_1, \ldots, a_d \in \mathbb{A}$, one can decide if

$$\mathbb{A} \models \varphi(a_1, \ldots, a_d);$$

(c) given two tuples in $\mathbb{A}^d$, decide if they are in the same orbit.

In Exercise 137 we show that if an atom representation exists, then it is essentially unique, since there are computable translations between any two atom representations. The following theorem shows that atom representations exist, under our usual effectivity assumption.

Theorem 7.3. *Every effectively oligomorphic structure has an atom representation.*

Proof. We will use Theorem 6.12, whose conclusion says that there is an atom representation. An oligomorphic structure $\mathbb{A}$ can be seen as a homogeneous structure $\mathbb{H}$, which has the same elements, but its vocabulary is extended so that it has one relation for every first order formula. (The vocabulary is infinite.) Let $\mathscr{H}$ be the age of $\mathbb{H}$. By the assumption that $\mathbb{A}$ is effectively oligomorphic, there is an algorithm that inputs d, and returns all structures in $\mathscr{H}$ up to isomorphism[1] In other words, $\mathscr{A}$ satisfies assumption (*) of Theorem 6.12. Therefore, we can apply that theorem, which yields an atom representation for the Fraïssé limit $\mathbb{H}$, because item 3 of Theorem 6.12 is the same as the definition of an atom representation. This in turn yields a representation for $\mathbb{A}$. □

The atom representation which arises from the above theorem will be very inefficient. In cases of interest, such as the equality atoms or the order atoms, it will be better to manually prepare more efficient atom representations.

One application of atom representations will be discussed in Chapter 8, where we will study what it means for a language $L \subseteq \mathbb{A}^*$ to be decidable. In the presence of atom representations, we can simply require that this language is decidable in the usual sense, assuming that atoms are given as strings that represent them. This requirement will be a baseline, to which we will compare other notions, such as orbit-finite Turing machines.

Another application is to represent equivariant subsets of $\mathbb{A}^d$ by finite generating sets. This is the same method as in Section 1.1.1: a *generating set* for a subset $X \subseteq \mathbb{A}^d$ is any set that contains at least one tuple per orbit. The assumptions on an atom representation will enable us to perform basic operations on subsets represented this way, such as Boolean combinations. For union, we can simply combine the two sets of generators (which might result in some orbits being represented by more than one generator). For intersection, we can use the "same orbit" test to check which orbits are represented in both sets. The most interesting operation is complementation, where we need to be able to describe orbits that are *not* represented. For this, we use the following lemma.

[1]It is worth explaining how one "returns" a structure over an infinite vocabulary. This means that we return a list of its elements, together with an algorithm which inputs a relation name, and tells us which tuples are selected by the relation.

Lemma 7.4. *For an atom representation of an effectively oligomorphic structure, there is an algorithm which does the following:*

(d) given d, compute a list of tuples that represents every orbit in $\mathbb{A}^d$.

Proof. By assumption on being effectively oligomorphic, we know the number of orbits. We can then start enumerating all of $\mathbb{A}^d$, until we have represented all orbits, which can be checked using the "same orbit" test from item (c). □

Of course, in cases of interest we will want to use more efficient algorithms than the one which is given in the proof of the above lemma.

Exercises

Exercise 137. Consider an effectively oligomorphic structure, and two atom representations r_1, r_2. Show that there is some computable function $f : 2^* \to 2^*$ which inputs a representation of some atom under r_1, and returns a representation of the same atom under r_2.

7.3 Orbit-finite graphs and automata

Having discussed representations of orbit-finite sets and their elements, we now start to present algorithms that operate on them. The first group of results, presented in this section, is about orbit-finite automata. These results are mainly based on the graph reachability result from Theorem 5.10, which showed that graph reachability is decidable for effectively oligomorphic atoms (in fact, the proof did not use the full power of the assumption, since it did not require that we can compute the number of orbits in $\mathbb{A}^d$). Quotients do not affect the algorithm, and so it extends to orbit-finite graphs, as stated in the following theorem.

Theorem 7.5. *Assume that the atoms are effectively oligomorphic. Then the reachability problem for orbit-finite graphs (represented as in Section 7.2) is decidable.*

Proof. Same as for Theorem 5.10. □

The emptiness problem for automata is the same as the reachability problem for graphs, and therefore we can use the above theorem to decide emptiness for orbit-finite automata. Let us begin by formally defining the model. Similarly to graphs, the definition of an orbit-finite automaton is the same as in the finite case, except that the word "finite" is replaced by "orbit-finite", and all subsets must be equivariant.

Definition 7.6 (Nondeterministic orbit-finite automaton)**.** Let $\mathbb{A}$ be an oligomorphic structure. A *nondeterministic orbit-finite automaton* over $\mathbb{A}$ is a tuple

$$\mathcal{A} = (\underbrace{Q,}_{\text{states}} \quad \underbrace{\Sigma,}_{\text{input alphabet}} \quad \underbrace{I \subseteq Q,}_{\text{initial states}} \quad \underbrace{F \subseteq Q,}_{\text{accepting states}} \quad \underbrace{\delta \subseteq Q \times \Sigma \times Q}_{\text{transitions}}),$$

where Q and Σ are orbit-finite sets, and the subsets I, F, δ are equivariant.

The semantics of the automaton are defined as for nondeterministic finite automata. The language recognised by such an automaton is equivariant, since the set of accepting runs is equivariant. An automaton is called *deterministic* if it has one initial state, and δ is a function from $Q \times \Sigma$ to Q.

Theorem 7.7. *Assume that the atoms are effectively oligomorphic. Then the emptiness problem for nondeterministic orbit-finite automata (represented as in Section 7.2) is decidable.*

Proof. An immediate corollary of Theorem 7.5. □

Other positive results that generalise easily to orbit-finite automata include:

- ϵ-transitions do not add to the power of nondeterministic automata (Exercise 138);
- one can minimize deterministic automata (Theorem 7.9);
- one can decide if a nondeterministic automaton is unambiguous (Exercise 139).

Negative results for the equality atoms generalise to other oligomorphic atoms, when the atoms are infinite:

- nondeterministic automata are not closed under complement;
- the universality problem is undecidable;
- deterministic automata are strictly weaker than nondeterministic ones.

To illustrate the scope of the above results, we give several examples of deterministic or nondeterministic orbit-finite automata, in various oligomorphic atoms.

Example 51. Assume that the atoms are the random graph from Section 6.3.1. This structure is effectively oligomorphic, because it is the Fraïssé limit of a Fraïssé class that can be enumerated as in the assumptions of Theorem 6.12. The set of paths in the random graph can be viewed as a language

$$\{a_1 \cdots a_n \in \mathbb{A} : \text{for every } i < n \text{ there is an edge from } a_i \text{ to } a_{i+1}\} \subseteq \mathbb{A}^*.$$

This language is recognised by a deterministic orbit-finite automaton, which uses its state to stores the last seen vertex. The set of cycles is also recognised by a deterministic automaton, this automaton also needs to remember the first vertex to check if it is connected to the last one. □

Example 52. Assume that the atoms are the random graph, and consider the language

$$\{\ w \in \mathbb{A}^* \mid \text{ the subgraph of } \mathbb{A} \text{ induced by atoms from } w \text{ is a clique }\ \}$$

This language cannot not be recognized by an orbit-finite automaton, even nondeterministic, as we show in the Exercise 140. □

Example 53. Assume that the atoms are the random graph. The graph atoms are a natural setting to talk about path and tree decompositions of graphs, as used in the graph minor project of Robertson and Seymour. To make notation lighter, we only discuss path decompositions. A *width k path decomposition* for a finite subset $V \subseteq \mathbb{A}$ is defined to be a list of (not necessarily disjoint) subsets $V_1, \ldots, V_n \subseteq V$ such that: (a) every vertex from V appears in at least one bag; and (b) if two vertices from V are connected by an edge, then they appear together in at least one bag; and (c) if a vertex appears in some two bags, then it also appears in all other bags between them.

If k is fixed, then such a path decomposition can be seen as a word over an orbit-finite alphabet, namely the sets of at most k atoms. (The number of orbits in this alphabet is the number of isomorphism types of graphs with at most k vertices.) Therefore, a path decomposition can be used as the input to an orbit-finite automaton. We now show that interesting properties of the underlying graph can be recognised by such automata.

Claim 7.8. *There is a deterministic orbit-finite automaton $\mathcal{A}$ such that*

$$\mathcal{A} \text{ accepts } V_1 \cdots V_n \quad \text{iff} \quad \text{the graph } V_1 \cup \cdots \cup V_n \text{ is connected}$$

holds for input which is a width k path decomposition[2].

Proof. After reading a path decomposition $V_1, \ldots, V_n$ the automaton stores in its state the last bag V_n together with the equivalence relation $\sim_n$ on it which identifies vertices from the last bag if they are in the same connected component of the underlying graph $V_1 \cup \cdots \cup V_n$. The states of the automaton are pairs (set of at most k atoms, an equivalence relation on this set); this state space is orbit-finite. The initial state is the empty set equipped with an empty equivalence relation, and the accepting states are those where the equivalence relation has one equivalence class. The definition of the transition function is left to the reader. □

Similar constructions as in the above claim can be done for any property of graphs of bounded pathwidth that is recognisable in the sense of Courcelle, which covers all graph properties that can be defined in monadic second-order logic[3]. Using tree automata instead of word automata, one can also cover tree decompositions. □

Example 54. Consider the bit-vector atoms and the language

$$\{ w \in \mathbb{A}^* \mid w \text{ is linearly dependent } \}.$$

The linear dependence in the above language is the same as the one discussed when defining the bit-vector atoms, i.e. w is viewed as a list and not as a set. This means that any repetition in the list will immediately be a dependence. This language is recognised by a nondeterministic orbit-finite automaton. The state space is $\mathbb{A}$, the

[2]The automaton does not check if the input is a path decomposition, in fact this cannot be done, see Exercise 128.

[3]For more on recognisability, pathwidth, and monadic second-order logic, see (Courcelle and Engelfriet, 2012, Chapter 5.3).

initial subset is the singleton of the zero vector {0}, and the accepting subset is the set of non-zero vectors. The transition relation is

$$\{ \ p \xrightarrow{a} q \mid \ p + a = q \text{ or } p = q \ \ \}.$$

One can show that the nondeterminism in the above automaton is unavoidable – the language is not recognised by a deterministic orbit-finite automaton. In fact, we will show an even stronger result later in this book, namely that the language is not recognised by any deterministic Turing machine running in polynomial time. □

Minimization of deterministic automata. In Chapter 5, one of our motivations for introducing orbit-finite sets as a generalization of pof sets was to minimize deterministic automata. In Theorem 4.9, we showed a version of the Myhill-Nerode Theorem for the equality atoms, which gave a machine independent characterization of deterministic orbit-finite automata, in terms of an orbit-finite syntactic congruence. The same result carries over to general oligomorphic atoms.

Theorem 7.9. *Assume that the atoms are oligomorphic. The following conditions are equivalent for an equivariant language $L \subseteq \Sigma^*$ over an orbit-finite alphabet Σ:*

1. *L is recognised by a deterministic orbit-finite automaton;*
2. *the quotient of Σ^* under the syntactic congruence of L is orbit-finite.*

Proof. The same proof as for Theorem 4.9, and in fact, for the original Myhill-Nerode Theorem for finite sets. We simply construct a deterministic automaton on the equivalence classes of syntactic congruence. The assumption that the language is equivariant guarantees that the structure of the automaton – the transition function and the accepting states – is also equivariant. □

If the atoms are not only oligomorphic, but they are effectively oligomorphic, then the syntactic automaton (i.e. the automaton that arises from the above theorem, also known as the minimal automaton) can be computed based on any other deterministic automaton.

Theorem 7.10. *If the atoms are effectively oligomorphic, then the syntactic automaton can be computed based on any deterministic orbit-finite automaton.*

Proof. We use what is called the *Moore algorithm*, i.e. a fixpoint procedure that computes equivalence on states[4]. Suppose that we are given a deterministic orbit-finite automaton, whose states are Q. We first use the graph reachability algorithm from Theorem 7.5 to restrict the state space to reachable ones. Next, we quotient the state space with respect to syntactic equivalence, i.e. recognizing the same language, as described below.

For $n \in \{0, 1, \ldots\}$, define $\sim_n$ to be the equivalence relation on states, which identifies two states if they accept the same words of length at most n. It is easy to see

[4]The computational complexity of automata minimisation is studied in Murawski et al. (2015), using the equality atoms and a more concrete model with registers and control states.

that this equivalence relation is equivariant, and each $\sim_n$ can be computed using the formula representation of equivariant subsets. The chain

$$\sim_1 \quad \sim_2 \quad \sim_3 \quad \cdots$$

is a decreasing sequence of equivariant subsets of $Q \times Q$, and therefore it must stabilize after finitely many steps. The stable value of this sequence is the syntactic equivalence relation, and the minimal automaton is obtained by quotienting its state space under this relation. □

Exercises

Exercise 138. Show that adding ϵ-transitions does not change the expressive power of nondeterministic orbit-finite automata.

Exercise 139. Show that, under the assumptions of Theorem 7.5, one can check if a nondeterministic orbit-finite automaton is deterministic. Likewise for unambiguous (each input admits at most one accepting run).

Exercise 140. Consider the graph atoms. Show that the language of cliques, i.e. words in $\mathbb{A}^*$ where every two letters are connected by an edge, is not recognised by a nondeterministic orbit-finite automaton.

Exercise 141. Consider the equality atoms. For a language $L \subseteq \Sigma^*$, consider the two-sided Myhill-Nerode equivalence relation which identifies words $w, w' \in \Sigma^*$ if

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

The quotient of Σ^* under this equivalence relation is called the *syntactic monoid* of L. Show that if the syntactic monoid is orbit-finite, then the syntactic automaton is orbit-finite, but the converse implication fails.

Exercise 142. Let $L \subseteq \Sigma^*$ be a language, and let Q be the states of its syntactic automaton. Show that the syntactic monoid defined in the previous exercise is isomorphic to the submonoid of functions $Q \to Q$ which is generated by the state transition functions $\{q \mapsto qa\}_{a \in \Sigma}$ of the syntactic automaton.

Exercise 143. Let $L \subseteq \Sigma^*$ and let $h : \Sigma^* \to M$ be its syntactic homomorphism, i.e. the function which maps a word to its equivalence class under two-sided Myhill-Nerode equivalence. Show that M is orbit-finite if and only if the syntactic automaton of L is orbit-finite and there is some $k \in \{0, 1, \ldots\}$ such that all elements of M have support of size at most k.

Exercise 144. We say that a monoid M is aperiodic if for every $m \in M$ there is some $k \in \{0, 1, \ldots\}$ such that $m^k = m^{k+1}$. Let L be a language with an orbit-finite syntactic automaton. Show that the syntactic monoid of L is aperiodic if and only if for every state q of the syntactic automaton and every $w \in \Sigma^*$ there is some $k \in \{0, 1, \ldots\}$ such that $qw^k = qw^{k+1}$.

Exercise 145. Suppose that M is an orbit-finite monoid. Can one find an infinite sequence

$$M \supsetneq M_1 \supsetneq M_2 \supsetneq M_3 \supsetneq \cdots$$

such that each M_i is a submonoid?

Exercise 146. Consider an orbit-finite monoid M. We define the prefix relation on this monoid as follows: a is an infix of b if $b = ax$ for some $x \in M$. Show that under the equality atoms, the prefix relation is well-founded, but this is no longer true under the order atoms.

7.4 Systems of equations

In the previous section, we discussed automata problems, which were based on graph reachability. Using a similar approach, the results on context-free grammars from Section 3.4 can be extened from the equality atoms to effectively oligomorphic atoms. Let us now give a new algorithm, which is based on a different approach[5]. In this algorithm, we use only two kinds of atoms, namely the equality atoms and the ordered atoms, but curiously enough, the ordered atoms are needed to analyze the equality atoms.

Consider a system of equations in the two element field $\mathbb{Z}_2$, like this one:

$$\begin{aligned} x + y &= 1 \\ x + z &= 1 \\ y + z &= 1 \end{aligned}$$

The system above does not have a solution, because some two variables need to get the same value, violating the equations. The system has finitely many equations. In this section, we consider systems where the set of equations is orbit-finite, but each individual equation is finite.

Example 55. Consider the equality atoms. The variables are pairs of distinct atoms, and the set of equations is

$$\underbrace{(a, b)}_{\text{one variable}} + \underbrace{(b, a)}_{\text{one variable}} \quad = 1 \qquad \text{for all } a \neq b \in \mathbb{A}.$$

A solution in $\mathbb{Z}_2$ to this system amounts to a choice function, which chooses for every two atoms $a \neq b \in \mathbb{A}$ exactly one of the pairs (a, b) or (b, a). It follows that the above system has a solution, but no equivariant supported solution. □

The above example shows that, under the equality atoms, an equivariant system of equations might have a solution, but it might not have an equivariant solution. If we use the ordered atoms, then the problem goes away, as shown in the following theorem.

Theorem 7.11. *Assume the atoms $(\mathbb{Q}, <)$. Let $\mathcal{E}$ be an equivariant orbit-finite set of equations. If $\mathcal{E}$ has any solution in $\mathbb{Z}_2$, then it has a solution in $\mathbb{Z}_2$ that is equivariant.*

Proof.

1. In the first step, we show that without loss of generality we can assume that the variables are tuples of atoms. Let X be the orbit-finite set of variables that appear in the equations $\mathcal{E}$. By the representation result from Theorem 5.12, see also Exercise 106, there is some $k \in \{0, 1, 2 \ldots\}$ and an equivariant surjective function

$$f : \mathbb{A}^k \to X.$$

[5] This section is based on Klin et al. (2015)

Define $\mathcal{F}$ to be the following set of equations over variables $\mathbb{A}^k$:

$$\underbrace{x = y}_{\text{when } f(x) = f(y)} \qquad \underbrace{y_1 + \cdots + y_n = i.}_{\substack{\text{when } \mathcal{E} \text{ contains an equation} \\ x_1 + \cdots + x_n = i \\ \text{where } f(y_1) = x_1, \ldots, f(y_n) = x_n}}$$

It is easy to see that if $\mathcal{E}$ has a solution if and only if $\mathcal{F}$ has a solution. Likewise for equivariant solutions.

2. Let $\mathcal{F}$ be the system of equations produced in the previous item. To prove the theorem, it remains to show that if $\mathcal{F}$ has a solution

$$s : \mathbb{A}^k \to \mathbb{Z}_2$$

then it also has an equivariant one. We prove this using the Ramsey Theorem. By the Ramsey Theorem, there is an infinite set $A \subseteq \mathbb{A}$ such that

$$s(a_1, \ldots, a_n) = s(b_1, \ldots, b_n)$$

holds for all $\bar{a}$ and $\bar{b}$ which are strictly growing tuples from A. Again by the Ramsey Theorem, there is an infinite set $B \subseteq A$ such that

$$s(a_1, \ldots, a_n) = s(b_1, \ldots, b_n)$$

holds for all $\bar{a}$ and $\bar{b}$ which are strictly decreasing tuples from B. Repeating this argument for all finitely many order types, i.e. for all orbits in $\mathbb{A}^k$, we get an infinite set $Z \subseteq \mathbb{A}$ such that

$$s(a_1, \ldots, a_n) = s(b_1, \ldots, b_n)$$

holds whenever $\bar{a}$ and $\bar{b}$ are tuples from Z^k with the same order type (in other words, in the same equivariant orbit of $\mathbb{A}^k$). Define

$$s' : \mathbb{A}^k \to \mathbb{Z}_2$$

to be the function that maps $\bar{a}$ to $s(\bar{b})$ where $\bar{b}$ is some tuple from Z^k in the same equivariant orbit as $\bar{a}$. Such a tuple $\bar{b}$ exists, and furthermore $s(\bar{b})$ does not depend on the choice of $\bar{b}$ by construction. Because $s'(\bar{a})$ depends only on the equivariant orbit of $\bar{a}$, the function s' is equivariant. It is also a solution to $\mathcal{F}$. This is because every equation from $\mathcal{F}$ can be mapped to some equation in $\mathcal{F}$ which uses only variables from Z, and s' satisfies those equations.

□

Corollary 7.12. *Assume that the atoms are* $(\mathbb{Q}, <)$. *Given an equivariant orbit-finite system of equations, one can decide if the system has a solution in* $\mathbb{Z}_2$. *Likewise for the equality atoms.*

Proof. Assume the atoms are $(\mathbb{Q}, <)$. By Theorem 7.11, it is enough to check if the system has an equivariant solution. By Lemma **??**, we can compute all equivariant orbits of the variables, and therefore we can check all equivariant functions from the variables to $\mathbb{Z}_2$, to see if there is any solution.

Consider now the equality atoms. We reduce to $(\mathbb{Q}, <)$. Every equivariant orbit-finite set over the equality atoms can be viewed as an equivariant orbit-finite set over $(\mathbb{Q}, <)$, by using the same set builder expressions. This transformation does not affect the existence of solutions, and for systems of equations over atoms $(\mathbb{Q}, <)$ we already know how to answer the question. □

Exercises

Exercise 147. Assume that the atoms are Presburger arithmetic $(\mathbb{N}, +)$. Consider sets of equations over the field $\mathbb{Z}_2$, where both the variables and the set of equations are represented by set builder expressions. Show that having a solution is undecidable.

Exercise 148. What is the effect on the decidability of the problem in Exercise 147 if we assume that the set of variables is $\mathbb{A}$, i.e. the natural numbers? What if the variables are atoms and every equation has at most two variables?

Exercise 149. Consider the following atoms[6]. The universe is the set of bit strings $\{0,1\}^\omega$ which have finitely many 1's. The structure on the atoms is given by the following relation of arity four:

$$a + b = c + d,$$

where addition is coordinatewise. This structure is oligomorphic. Show two sets that are equivariant and orbit-finite, such that there is a finitely supported bijection between them, but there is no equivariant bijection.

[6]Suggested by Szymon Toruńczyk.

Chapter 8

Turing machines

Without atoms, the notion of "computability" is formalised using Turing machines, and there is a thesis – called the Church-Turing Thesis – which says that there is any other hypothetical model of computation would need to be captured by Turing machines.

What happens to this thesis in the presence of atoms? We begin a discussion in this chapter, by introducing orbit-finite Turing machines. This is an interesting and natural model of computation. However, as we will discover already in this chapter and then later on in this book, using Turing machines as the definition of computability is problematic. The main issue is that the basic data structure in a Turing machine is a list, and arranging objects in lists requires a form of choice, which typically impossible in the presence of atoms. We will overcome these limitations in later chapters, by using models of computation that are based on sets instead of lists.

Nevertheless, there are interesting things to say about Turing machines with atoms, and we say some of them in this chapter. The highlights are that, for Turing machines with atoms, one can prove nontrivial separations for complexity classes:

- In Section 8.2, we prove that $\textsc{p} \neq \textsc{np}$ holds in the bit vector atoms. It is worth underlying that this result has no bearing on the usual version of the question, without atoms. This is because we prove a much stronger result, which is clearly false without atoms, namely that deterministic orbit-finite Turing machines running in polynomial time are incapable of recognizing even some languages that are recognized by nondeterministic orbit-finite automata. The separating language is

  $$\{a_1 \cdots a_n \in \mathbb{A} : a_1, \ldots, a_n \text{ are not linearly independent}\}.$$

  This language is recognised in by a nondeterministic automaton (guess the linear combination that gives zero) or deterministic exponential time Turing machine (try all combinations), but it is not reocognized in deterministic polynomial time.

- In Section 8.3, we prove a similar separation for the equality atoms. We show that there is a language which is recognised by a nondeterministic Turing machine (even in polynomial time), but not recognised by any deterministic Turing machine (even without restrictions on running time). This separation is harder than in Section 8.2, and uses the Cai-Fürer-Immerman construction from finite model theory. As in the previous item, this separation is unlikely to be useful in separating complexity classes without atoms. The accompanying proof is based on the limited access that Turing machines have to their input and on the symmetries that result from applying atom automorphisms.

8.1 Orbit-finite Turing machines

Before presenting the separation results about Turing machines, we begin by defining the model and discussing some examples.

Definition 8.1 (Orbit-finite Turing machine)**.** An *orbit-finite Turing machine* is defined in the same way as in Section 3.5, except that instead of polynomial orbit-finite sets, we use (not necessarily polynomial) orbit-finite sets, and the atoms need not be the equality atoms, but they can be any oligomorphic structure.

At first glance, the difference seems to be minor. By Theorems 4.6 and 5.12, an orbit-finite set is the same as a polynomial orbit-finite set quotiented by some partial equivalence relation, and therefore the only difference between pof sets and orbit-finite sets is the presence of quotients. As we will see, the quotients have profound impact, and break results such as the characterization from Theorem 3.5, even under the equality atoms.

To illustrate the roles of quotients, consider the following example. The example is a positive one – i.e. we can still solve the problem using a deterministic orbit-finite Turing machine – but later on we will see negative examples.

Example 56. [An input alphabet with quotients] Consider the equality atoms. Let the input alphabet Σ be sets of atoms of size at most ten. This is the quotient of $\mathbb{A}^{10}$ under the partial equivalence relation that identifies two tuples if they use the same set of atoms. (Formally speaking, this quotient does not cover the empty set. To work around this minor issue, we can either use $\mathbb{A}^{10} + \mathbb{A}^{0}$, or we can stay with $\mathbb{A}^{10}$ and modify the equivalence relation, so that the empty set is represented by some redundant orbit. This is because sets of non-maximal size, say size 1, are represented by many orbits, and one of these can be spared to represent the empty set.)

Here is an example of an input word over this alphabet Σ that has six letters:

$$\{\text{John}, \text{Eve}, \text{Mary}\}\{\text{John}, \text{Eve}, \text{Tom}\}\{\text{John}, \text{Eve}\}\{\text{John}, \text{Eve}, \text{Mary}\}\{\text{John}, \text{Eve}\}\{\text{Tom}\}.$$

The letters are sets, which means that there is no order on the atoms that appear in a given letter. This will prevent a machine from doing operations like "load the first atom from the letter under the head", witnessing the difficulties of input alphabets with quotients. However, as we show in this example, the diffiulties can be overcome for this particular input alphabet.

Consider the language: "some atom appears in an odd number of letters". This language can easily be recognized by a nondeterministic Turing machine, in fact even a nondeterministic orbit-finite automaton. The challenge is doing this with a deterministic machine, because there is no apparent mechanism for pointing to the atom that appears in an odd number of letters. To see this, consider the example input word from the previous paragraph. Both atoms John and Eve appear in an odd number of letters, but an equivariant deterministic Turing machine cannot see any difference between these two atoms, because every set contains either both or none of the atoms John and Eve.

Here is a solution to the problem, i.e. a deterministic orbit-finite Turing machine that recognizes the language. Suppose that the input word is $A_1 \cdots A_n$, where each A_i is a letter from the alphabet, i.e. a set of at most ten atoms. The Turing machine executes the following program:

1. Generate a copy of the input word. After this step the tape has the form

$$\underbrace{A_1 \cdots A_n}_{\text{first copy}} | \underbrace{A_1 \cdots A_n}_{\text{second copy}}.$$

2. As long as possible, iterate these steps on the second copy of the tape:

   (a) if some set appears multiple times, remove all the duplicates;

   (b) if A, B are two distinct intersecting sets that appear in the second copy, then remove them and add the three sets $A \setminus B, B \setminus A, A \cap B$.

   This stage ends when all sets in the second copy are pairwise disjoint.

3. At this point, the second contains contains equivalence classes of the relation "appears in the same sets from $A_1, \ldots, A_n$". Check if some equivalence class (i.e. some set from the second copy) is contained in an even number of sets from the first copy.

All the above steps can be performed by a deterministic orbit-finite Turing machine. To process sets, the machine can use state space which stores a set of at most 10 atoms. □

In the above example, the symmetries in the input alphabet required some non-trivial programming tricks. As we will see later in this chapter, when the symmetries get more complicated, the tricks will run out.

Before showing these negative results, we begin with a positive result, which involves nondeterministic machines. In Theorem 3.5, we showed that for the equality atoms and polynomial orbit-finite sets, Turing machines with atoms are computationally complete, in the sense that they recognize the same languages as the usual atom-less Turing machines, assuming that atoms are represented in an atom-less way. The proof of that theorem does not extend to (not necessarily polynomial) orbit-finite sets. The issue is with the deatomisation construction in Lemma 3.6, which assumes that one can order the atoms from the input string in their order of appearance, and speak of the "leftmost atom", or the "second leftmost atom". Such an order exists when the alphabet is a pof set, because letters are ordered tuples of atoms, but it does not necessarily exist for more general alphabets, e.g. when letters are sets as in Example 56. Therefore, we need to revisit the questions for the more general setting of orbit-finite sets.

Later in this chapter, we will show that not only the proof of Theorem 3.5 fails to work for orbit-finite sets that are not polynomial, but the result itself is false, because deterministic and nondeterministic orbit-finite machines have different computational power. However, we can at least show that the nondeterministic machines are computationally complete, under a mild assumption on the atoms, which is true for all atom structures discussed in this book.

The additional assumption is that the vocabulary is finite, and every first-order formula is equivalent to an existential formula, which is a formula of the shape

$$\varphi(x_1, \ldots, x_n) \quad = \quad \underbrace{\exists y_1, \ldots, y_m}_{\substack{\text{prefix of existential}\\ \text{quantifiers}}} \underbrace{\varphi(x_1, \ldots, x_n, y_1, \ldots, y_m)}_{\text{a quantifier-free formula}}.$$

Most atom structures that we have used so far – such as the equality atoms, the ordered atoms, or the graph atoms – have an even stronger property, namely every formula is equivalent to a quantifier-free formula. This is because they are homogeneous structures over a finite vocabulary, and for homogeneous structures, every first-order formula is equivalent to a quantifier-free formula, see Theorem 6.3. The only exception is the bit-vector atoms, which are homogeneous, but over an infinite vocabulary. However, also these atoms can be made compliant, if we change the vocabulary, as explained in the following example.

Example 57. Consider the bit-vector atoms. As we have defined them in Section 6.3.2, this is a homogeneous structure over an infinite vocabulary, which has a dependence relation for every dimension d. We can, however, use a different vocabulary, namely one ternary relation

$$x + y = z$$

for addition of vectors. This relation is equivariant, since automorphisms of the atoms are linear maps, and therefore it can be defined using a quantifier-free formula by Theorem 6.3, see Exercise 150 for an explicit formula. Conversely, each of the dependence relations can be expressed using only the addition relation. Therefore, we can think of the bit-vector atoms as having only on relation, namely addition. (For this new vocabulary, the structure is no longer homogeneous, see Exercise 151.) In the new vocabulary, every first-order formula is equivalent to an existential one, since: (a) every first-order formula is equivalent to a quantifier-free formula using dependence only, and (b) both dependence and independence can be defined using existential formulas that use addition only. □

In Theorem 8.2 below, we will show that nondeterministic orbit-finite Turing machines are computationally complete, which means that they are equivalent to Turing machines that use atom-less strings as representations of atoms. This notion of representation was formalised in Definition 7.2, as a function $r : 2^* \to \mathbb{A}$, subject to certain assumptions. Once we know how to represent individual atoms as atom-less strings, we can easily extend the representation to represent elements of pof sets, or elements of orbit-finite sets, or words over an orbit-finite alphabets. These extensions are used in the following theorem, which shows that nondeterministic orbit-finite Turing machines are the "right" model for languages over orbit-finite alphabets.

Theorem 8.2. *Assume that the atoms are effectively oligomorphic, have a finite vocabulary, and every first-order formula is equivalent to an existential one. Then the following conditions are equivalent for every $L \subseteq \Sigma^*$ where Σ is an orbit-finite alphabet:*

1. *L is recognised by a nondeterministic orbit-finite Turing machine;*
2. *L is equivariant and for every atom representation, see Definition 7.2, the language*
$$\{ w \in 2^* \mid w \text{ represents some word in } L \}$$
is recognised by a nondeterministic Turing machine (without atoms);
3. *as in the previous item, but the machine is deterministic;*
4. *as in the previous item, but the representation is quantified existentially.*

Proof. The implications $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4$ are proved in the same way as in Theorem 3.5. Let us very briefly recall those arguments. For the implication $1 \Rightarrow 2$, we extend the representation from atoms to configurations of Turing machines, and show that the one-step successor relation is decidable, which can be used to recognize the language in item 2. The implication $2 \Rightarrow 3$ follows from the fact that deterministic and nondeterministic Turing machines without atoms recognize the same languages. Finally, the implication $3 \Rightarrow 4$ is trivial ("every" implies "some", as long as an atom representation exists, and it does exist by by Theorem 7.3).

We are left with the implication $4 \Rightarrow 1$. We begin by reducing to the case where the input alphabet is $\mathbb{A}$.

Lemma 8.3. *If the implication $4 \Rightarrow 1$ holds in the special case when the input alphabet Σ is $\mathbb{A}$, then it holds for every orbit-finite alphabet.*

Proof. The reduction is based on the following observation.

Claim 8.4. *For every orbit-finite set Σ there exists $d \in \{0, 1, \ldots\}$ and a surjective equivariant function $f : \mathbb{A}^d \to \Sigma$.*

Proof. By Theorem 5.12, we know that Σ admits an equivariant bijection with a quotiented pof set. By removing the quotients, it follows that there is a surjective equivariant function from some (un-quotiented) pof to Σ. Therefore, it enough to prove the claim for (un-quotiented) pof sets. To get such a surjective function

$$f : \mathbb{A}^d \to \underbrace{\mathbb{A}^{d_1} + \cdots + \mathbb{A}^{d_k}}_{\text{an (un-quotiented pof set)}},$$

we choose d to be the maximal dimension among $d_1, \ldots, d_k$ plus some constant e, which is chosen to be so that $\mathbb{A}^e$ has at least k orbits. We use the orbit of the last e atoms to choose the component in $i \in \{1, \ldots, k\}$, and the remaining atoms to get the content of tuple in $\mathbb{A}^{d_i}$. □

We use the above claim to prove the lemma. Let $L \subseteq \Sigma^*$ be a language over some orbit-finite alphabet, which satisfies condition 4. Apply the above claim to get a surjective function f, and extend it to lists, giving a surjective equivariant function

$$f^* : (\mathbb{A}^d)^* \to \Sigma^*.$$

We can pull back the language L along f^*, yielding a new language

$$K = \{\, v \in (\mathbb{A}^d)^* \mid \; f^*(v) \in L \;\}.$$

We can think of the new language as being a language over alphabet $\mathbb{A}$, which contains only words with length divisible by k. Therefore, we can apply the implication 4 ⇒ 1 to the new language. To complete the proof of the lemma, we will show that condition 4 transfers from the original language L to the new language K, and that condition 1 transfers in the opposite direction.

Let us first explain how condition 4 transfers from the original language to the new language. Condition 4 speaks about machines which work on atom-less representations. Given a representation of a word for the new language, an orbit-finite Turing machine can compute a representation for its image under f^*, and then use the Turing machine for the original language. This shows the transfer of condition 4.

Let us now explain how condition 1 transfers from the new language to the original language. Given an input word for the original language, the orbit-finite machine uses nondeterminism to guess some input for the new language which maps to it along f^*, and then calls on the Turing machine for the new language. □

The proof of the above lemma crucially uses nondeterminism, in the last step where an inverse image under f^* is guessed. As we will see later in this chapter, this is unavoidable, since nondeterministic orbit-finite Turing machines are strictly more expressive than deterministic ones.

Having proved the reduction in Lemma 8.3, it remains to prove implication 4 ⇒ 1 for languages over the alphabet $\mathbb{A}$. Let then $L \subseteq \mathbb{A}^*$ be a language that satisfies condition 4. When we speak of representations of atoms or lists of atoms below, we refer to the representation this condition.

We begin by showing that if we bound the computation time, then the accepted words can be defined using first-order formulas.

Lemma 8.5. *Given an input length d and bounds t and s on the time and space of the computation, one can compute a first-order formula which defines the language*

$$\{\, w \in \mathbb{A}^d \mid \begin{array}{l}\textit{some word in the orbit of } w \textit{ has a representation which}\\ \textit{is accepted by } M \textit{ in time at most } t \textit{ and space at most } s\end{array} \}$$

Proof. Let F be the finite set of words (without atoms) that are accepted by M in at most n computation steps, and which represent some word in $\mathbb{A}^d$. This set can be computed given the parameters d, s, t. To prove the lemma, we show that we can compute formulas that describe the orbits of the words represented by F. The main observation is in the following claim, which shows that the partition of $\mathbb{A}^d$ into orbits, with each orbit described by a first-order formula, can be computed.

Claim 8.6. *Given d, one can compute first-order formulas that describe all orbits in $\mathbb{A}^d$.*

Proof. By the assumption that the structure is effectively oligomorphic, we can compute the number of orbits in $\mathbb{A}^d$, say it is n. Next, we can start enumerating n-tuples of first-order formulas in d variables, until we find an n-tuple where all formulas describe nonempty subsets which are pairwise disjoint. These subsets must be the orbits of $\mathbb{A}^d$. The stopping criterion is decidable, since the structure has a decidable first-order theory, and the procedure must stop, since every orbit is first-order definable by Theorem 5.7. □

Using the above claim, we can check which of the orbit formulas are satisfied by the tuples (represented by strings) in F. This check is decidable, by definition of atom representations. The disjunction of the resulting formulas defines the language in the statement of the lemma. □

We are now ready to prove that the language L is recognised by a nondeterministic orbit-finite Turing machine. Thanks to the above lemma, we can compute a first-order formula which tells us if the input word belongs to the language as witnessed by a run of given length. The Turing machine recognising L will evaluate the formula, for ever larger resource bounds, and it will accept once it finds some resource bounds for which the formula is true. If the word is not in the language, then the procedure will not terminate, since no resource bounds be found. The important thing, however, is that the procedure will terminate with acceptance when the word is in the language. To evaluate the formulas, we use the following lemma, which completes the proof that L is recognised by a nondeterministic orbit-finite Turing machine, and also the proof of the theorem.

Lemma 8.7. *There is a nondeterministic orbit-finite Turing machine that inputs*

$$\underbrace{a_1 \cdots a_d \in \mathbb{A}^*}_{\textit{a list of atoms}} \qquad \textit{and} \qquad \underbrace{\varphi(x_1, \ldots, x_d)}_{\textit{a first-order formula}}$$

and answers if the formula is true for the given atoms.

Proof. The input alphabet is the disjoint union of $\mathbb{A}$, which is used for the list of atoms, and some finite alphabet which is used to represent the formula. The proof of this lemma is the only place where we use the assumption that every formula is equivalent to an existential one.

We begin by proving the lemma in the special case when the formula φ is quantifier-free. A quantifier-free formula is a Boolean combination of atomic formulas

$$R(x_{i_1}, \ldots, x_{i_k})$$

where R is a relation from the vocabulary, and the indices $i_1, \ldots, i_k$ indicate the variables in question. For each such atomic formula, the Turing machine moves it head to the positions $i_1, \ldots, i_k$ on the input list of atoms, and collects their values into a single tuple from $\mathbb{A}^k$. Since the vocabulary is finite, the dimension k has a fixed upper bound that does not depend on the input, and therefore this tuple can be stored in the state.

The relations from the vocabulary can be hard-coded into the transition function of the Turing machine, since they are equivariant, and there are finitely many possibilities by the assumption that the vocabulary is finite. Therefore, the Turing machine can use a single transition to check if the relation R is satisfied by a tuple that is stored in its state.

We now consider the general case, where the input formula is not necessarily quantifier-free. By the assumption on the atom structure, we know that the input formula is equivalent to an existential formula

$$\exists y_1 \cdots \exists y_n\ \psi(x_1, \ldots, x_d, y_1, \ldots, y_n).$$

Not only does this existential formula exist, but we can also compute it. This is done by enumerating all candidates for the existential formula and using decidability of the first-order theory to check if the candidate is equivalent to the input formula. We know that an equivalent candidate will eventually be found. Once we have found the appropriate candidate, it can be evaluated for the input list of atoms: use nondeterminism to guess the values of the existential variables, write them on the tape, and then check if the quantifier-free part ψ is true using the procedure from the previous paragraph. □

□

In the above theorem, we use nondeterministic Turing machines and the assumption that first-order logic collapses to its existential fragment. This assumption covers all atom structures used in this book, and we are not aware of any example that is not covered. However, it is natural to ask about what happens without this assumption. We will show that without this assumption, a variant of the above theorem can still be recovered, at the cost of using a slightly more general model, namely alternating machines.

Let us begin by describing the model. The syntax of an alternating Turing machine is the same as for a nondeterministic one, except that the states are additionally partitioned into two parts: *existential* and *universal*. (The idea is that a nondeterministic machine is the special case when all states are existential.) The semantics of is defined in terms of a game, which is played by two players, called the *existential player* and the *universal player*. A position in the game is a configuration of the machine. In a given configuration, the player who owns the control state chooses a transition, which results in a new configuration, or in acceptance/rejection. The language recognized by the machine is defined to be the words for which the existential player has a strategy in the game that ensures acceptance, assuming that one begins with the initial configuration for the input string. This means that for every strategy of the universal player, a finite number of rounds is played and then the machine accepts.

Theorem 8.8. *Assume that the atoms are effectively oligomorphic and have a finite vocabulary. Then for every language $L \subseteq \Sigma^*$ over orbit-finite alphabet, conditions 2–4 from Theorem 8.2 are equivalent each other, and also to:*

1 L is recognised by an alternating orbit-finite Turing machine.*

Proof. The implications 1* ⇒ 2 ⇒ 3 ⇒ 4 are proved in the same way as in Theorem 3.5. The only difference is that for the first implication, we need the observation that without atoms, alternating and nondeterministic Turing machines are equivalent. This is a classical result, which is proved by showing that a nondeterministic Turing machine can enumerate through all possible strategies for the existential player.

For the implication 4 ⇒ 1*, we can use the same proof structure as in the proof of Theorem 8.2. As we remarked in that proof, Lemma 8.7 is the only part of the proof which used the assumption that every formula is equivalent to an existential one. Therefore, it remains to prove that lemma, without this assumption, but using alternating machines instead of nondeterministic ones in the conclusion. In this version, the lemma becomes essentially trivial, since the alternation can be used to simulate both kinds of quantifiers, universal and existential. □

Exercises

Exercise 150. Consider the bit-vector atoms. Write a quantifier-free formula for $x + y = z$, which uses the dependence relations.

Exercise 151. Consider the bit-vector atoms with the ternary addition relation from Example 57 and no other relations. Show that this structure is not homogeneous.

Exercise 152. Assume oligomorphic atoms. Let M be a nondeterministic orbit-finite Turing machine with input alphabet $\mathbb{A}$. Show that there is a finite family $\mathcal{R}$ of equivariant relations on the atoms such that for every $n, t, s \in \{0, 1, \ldots\}$, the property

$$\{ w \in \mathbb{A}^n \mid M \text{ accepts } w \text{ in time at most } t \text{ and space at most } s \}$$

can be defined by an existential formula that uses only relations from $\mathcal{R}$.

Exercise 153. Assume that the atoms are effectively oligomorphic. Prove a converse of Theorem 8.2: if the conditions in the theorem are equivalent, then every first-order formula is equivalent to an existential one.

Exercise 154. Assume that the atoms are oligomorphic. Let Σ be an orbit-finite input alphabet. Show that a language $L \subseteq \Sigma^*$ is recognised by a deterministic orbit-finite Turing machine if and only if:

(*) There is an orbit-finite set $A \supseteq \Sigma$, a finite set $\mathcal{F}$ of functions (each one being an equivariant function $A^k \to A$ for some k) and an equivariant subset $F \subseteq A$ such that given $n \in \mathbb{N}$, one can compute a term using the functions $\mathcal{F}$ and has n variables such that

$$a_1 \cdots a_n \in L \quad \text{iff} \quad t(a_1, \ldots, a_n) \in F \qquad \text{for every } a_1, \ldots, a_n \in \Sigma.$$

Exercise 155. Assume that the atoms are oligomorphic and admit least supports. Show that a language $L \subseteq \mathbb{A}^*$ is recognised by a deterministic orbit-finite Turing machine if and only if:

(**) There exists a finite family $\mathcal{F}$ of functions (each one being an equivariant function $\mathbb{A}^k \to \mathbb{A}$ for some k) and relations (each one being a subset of $\mathbb{A}^k$ for some k) such that given $n \in \mathbb{N}$, one can compute a quantifier-free formula with functions $\mathcal{F}$ that has n free variables and defines $L \cap \mathbb{A}^n$.

Exercise 156. Assume that the atoms admit least supports, and are homogeneous over a relational vocabulary. Show that nondeterministic and deterministic orbit-finite Turing machines recognise the same languages over input alphabet $\mathbb{A}$.

8.2 For bit vector atoms, P ≠ NP

Recall the bit vector atoms that were introduced in Section 6.3.2. This is the vector space over the two-element field of countably infinite dimension, i.e. these are vectors in $\{0,1\}^\omega$ that have finitely many nonzero entries. Equivalently, this is the Fraïssé limit of finite-dimensional vector spaces over this field. As explained in Example 57, we can view this as a structure with one ternary relation $x + y = z$, or alternatively with infinitely many relations for linear dependence.

In this section, we show that P ≠ NP holds for these atoms. Actually, we prove that deterministic polynomial time orbit-finite Turing machines are not even capable of simulating nondeterministic orbit-finite automata. The separating language consists of lists of vectors that have some nontrivial linear dependency.

Theorem 8.9. *Assume the bit vector atoms. The language*

$$\{ a_1 \cdots a_n \in \mathbb{A}^* \mid \text{ for some nonempty subset } I \subseteq \{1,\ldots,n\} \text{ we have } 0 = \textstyle\sum_{i\in I} a_i \ \}$$

is recognised by a nondeterministic orbit-finite automaton (and therefore also by a nondeterministic polynomial time orbit-finite Turing machine), but it is not recognised by any deterministic polynomial time orbit-finite Turing machine.

Proof. The upper bound in the theorem – about recognisability by an orbit-finite automaton – was shown in Example 54. One can also have an alternative upper bound: the language can be recognised by a deterministic orbit-finite Turing machine in exponential time, by enumerating through all possible coefficients in the linear combination.

The rest of this section is devoted to proving the lower bound for deterministic Turing machines that run in polynomial time. Fix some deterministic orbit-finite Turing machine. We will show that if the machine runs in polynomial time and rejects some linearly independent tuple, then it will also reject some linearly dependent tuple.

We begin by introducing some notation. Let Γ be the work alphabet and let Q be the state space of the fixed Turing machine. A computation of the machine that uses t time steps and s units of space can be seen as a rectangular grid

$$\rho : \underbrace{\{1,\ldots,t\} \times \{1,\ldots,s\}}_{\substack{\text{pairs in this set}\\ \text{will be called tiles}}} \to \underbrace{\Gamma + \Gamma \times Q}_{\text{labels of tiles}},$$

where labels from $\Gamma \times Q$ are used for tiles containing the head, and labels from Γ are used for the other tiles. Not every function ρ of the above type is a computation, because ρ must also respect the transition function of the machine. The following straightforward lemma says that respecting the transition function is a property that depends on at most three tiles at a time.

Lemma 8.10. *Suppose that*

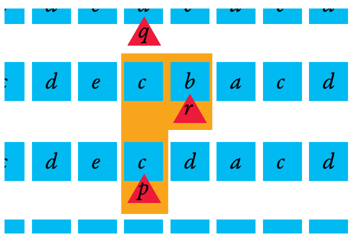
$$\rho, \sigma : \{1,\ldots,t\} \times \{1,\ldots,s\} \to \Gamma + \Gamma \times Q$$

are similar in the sense that for every three tiles x, y, z, the triples

$$(\rho(x), \rho(y), \rho(z)) \qquad (\sigma(x), \sigma(y), \sigma(z))$$

are in the same orbit. Then ρ is a computation if and only if σ is a computation, and ρ is rejecting if and only if σ is rejecting.

Proof. The semantics of a Turing machine involves comparing at most three tiles at the same time, as in the following picture:



The assumption of the lemma could even be weakened to triples of tiles that are adjacent in the grid as in the above picture, but we will not need this stronger variant of the lemma. □

We use the above lemma to show that a rejecting computation of the Turing machine that has polynomial size can be converted into another rejecting computation, whose input word is linearly dependent, and therefore should be accepted. The resulting contradiction will show that the language cannot be recognised by a deterministic polynomial time Turing machine.

By Claim 8.4, there is a surjective equivariant function

$$f : \mathbb{A}^d \to \Gamma + \Gamma \times Q.$$

Consider an input string $w \in \mathbb{A}^*$ that is linearly independent, and let

$$\rho : \{1, \ldots, t\} \times \{1, \ldots, s\} \to \Gamma + \Gamma \times Q$$

be the corresponding computation of the Turing machine. This is a rejecting computation, because the input string is linearly independent. Define a *support list* of this computation to be any

$$\bar{\rho} : \{1, \ldots, t\} \times \{1, \ldots, s\} \to \mathbb{A}^d$$

which is yields ρ after extending with f. The support list can be viewed as a list of atoms of length std. This length of this list is polynomial in the input length, since the time and space of the Turing machine is polynomial, and the dimension d is fixed. We claim that if the input length n is large enough, then there is some atom $a \in \mathbb{A}$ which

1. is spanned by the n independent atoms in the input string w;
2. is not spanned by any subset of $3d$ atoms in the support list $\bar{\rho}$.

This is because the number of atoms spanned by $\bar{a}$ is exponential in n, while the number of subsets from the second item is polynomial.

Choose a linear map $\varphi : \mathbb{A} \to \mathbb{A}$ whose kernel is $\{0, a\}$. (This is done by choosing a basis of $\mathbb{A}$ which contains a, and then sending the basis vector a to zero, and the remaining basis vectors to themselves). This linear map is not an atom automorphism, since it is not invertible. Nevertheless, we can still apply it to atoms and lists of atoms (however, it will not be guaranteed to preserve orbits). Apply it to the support list $\bar{\rho}$, yielding a new support list

$$\bar{\sigma} : \{1, \ldots, t\} \times \{1, \ldots, s\} \to \mathbb{A}^d.$$

Finally, let

$$\sigma : \{1, \ldots, t\} \times \{1, \ldots, s\} \to \mathbb{A}^d$$

be the result of extending $\bar{\sigma}$ with f. We will now show that σ is in fact a computation of the Turing machine, and that it is also rejecting. This will yield a contradiction, since the input string becomes linearly dependent after applying φ, and therefore we get a rejecting computation on a string that should be accepted.

Lemma 8.11. *The function σ is a rejecting computation of the Turing machine.*

Proof. The essential idea is that while the linear map φ is not an atom automorphism, it still preserves the orbit of short lists of atoms, and this will be enough to preserve computations of the Turing machine.

Thanks to Lemma 8.10, it suffices to show that for every three tiles x, y, z, the triples orbits of these three tiles are the same in ρ and σ. To prove this, we use the following straightforward characterisation of the same orbit relation.

Claim 8.12. *Two tuples $\bar{a}, \bar{b} \in \mathbb{A}^k$ are in the same orbit if and only if*

$$\sum_{i \in I} a_i = 0 \quad \textit{iff} \quad \sum_{i \in I} f(a_i) = 0 \qquad \textit{for every } I \subseteq \{1, \ldots, k\}.$$

Proof. The left-to-right implication is immediate. For the right-to-left implication, we observe that if the same subsets of coordinates have zero sum, then the two tuples satisfy the same quantifier-free formulas under the vocabulary that uses the dependence relations. Under this vocabulary, the structure $\mathbb{A}$ is homogeneous, see the discussion in Example 57. Therefore, if the two tuples satisfy the same quantifier-free formulas, then they are in the same orbit, by Theorem 6.3. □

Using the claim, we complete the proof of the lemma. By definition, the support list $\bar{\sigma}$ was obtained from $\bar{\rho}$ by applying the linear map φ, which preserved non-zeroness of linear combinations of size at most $3d$. Since a tile uses at most d atoms, it follows from the above claim that for every three tiles, the corresponding triples of labels in $\bar{\sigma}$ and $\bar{\rho}$ are in the same orbit. This relation continues to hold after applying the equivariant map f, and therefore every triple of tiles in ρ has a triple of labels that is in the same orbit as the corresponding triple of labels in σ. This shows that σ is also a rejecting computation of the Turing machine. □

The above lemma shows that the Turing machine must also reject some linearly dependent tuple, which is a contradiction. Therefore, the language cannot be recognised by a deterministic polynomial time Turing machine. □

Exercises

Exercise 157. Assume the bit vector atoms. Show that if the input alphabet is $\mathbb{A}$, then nondeterministic orbit-finite Turing machines have the same expressive power as deterministic orbit-finite Turing machines (although with possibly exponential slowdown).

8.3 For equality atoms, Turing machines cannot be determinised

This section describes another thing that deterministic Turing machines with atoms cannot do. This time, the atoms are the equality atoms.

Theorem 8.13. *Assume the equality atoms. There is a language which:*

1. *is recognised by a nondeterministic orbit-finite Turing machine;*
2. *is not recognised by any deterministic orbit-finite Turing machine.*

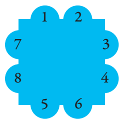
In other words, deterministic orbit-finite Turing machines are not computationally complete[1], which witnesses the tightness of Theorem 8.2.

The rest of Section 8.3 is devoted to proving Theorem 8.13.

Recall from Theorem 3.5 that, when the input alphabet is a pof set, then deterministic orbit-finite Turing machines are computationally complete. Therefore, the language in the theorem needs to use an input alphabet that is not a pof set.

The language in Theorem 8.13 will be recognised by a polynomial time nondeterministic machine. Therefore, the theorem gives another example of NP≠P. Again, as mentioned at the beginning of this chapter, the theorem is unlikely to shed new light on the NP≠P question without atoms, since the proof is based on the limited way that a Turing machine can access the atoms in its tape.

The separating language. Define a *tile* to be a tuple of 8 distinct atoms, i.e. an element of $\mathbb{A}^{(8)}$. We draw tiles like this:

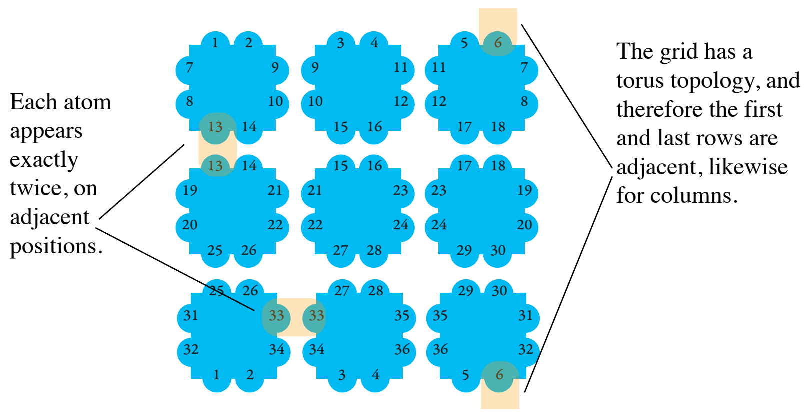
$(1, 2, 3, 4, 5, 6, 7, 8)$ is drawn as 

[1]The results in this section are based on Bojańczyk et al. (2013a).

We will arrange tiles on a square grid with torus topology. For $n \in \{1, 2, \ldots\}$, define an $n \times n$ tiling to be a function

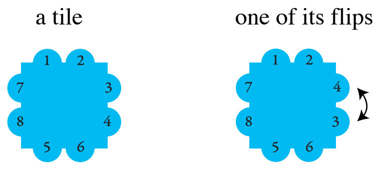
$$\mathcal{T} : n \times n \to \mathbb{A}^{(8)} \qquad \text{where } n \times n \overset{\text{def}}{=} \{0, 1, \ldots, n-1\} \times \{0, 1, \ldots, n-1\}.$$

A tiling is called *consistent* if it satisfies the following constraints:

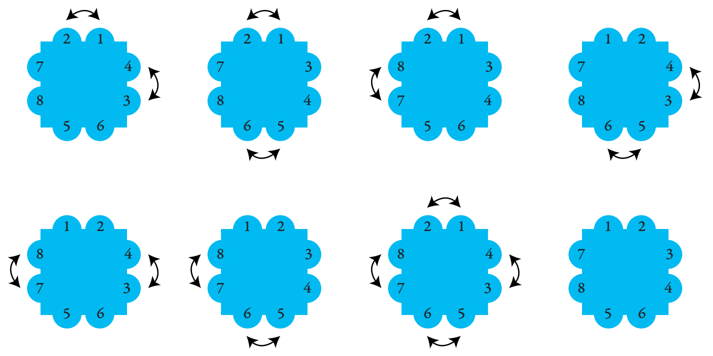

Each atom appears exactly twice, on adjacent positions.
The grid has a torus topology, and therefore the first and last rows are adjacent, likewise for columns.

We begin with an informal description of the language that is difficult for deterministic Turing machines. One is given partial information about a tiling, namely each tile is known up to an even number of flips (see below). The question is: can the partial information be instantiated to a tiling that is consistent? This question will turn out to be doable using a nondeterministic machine – by guessing the instantiation – but will be impossible for a deterministic machine.

We now describe the partial information in more detail. A *flip* on a tile is defined to be a transposition of atoms that appear on one side, as shown in the following picture:



Define $\approx$ to be the equivalence relation on tiles, which identifies two tiles if one can be obtained from the other by doing an even number of flips. Each equivalence class of $\approx$ has eight tiles, as shown in the following picture:



Define $\mathbb{A}^{(8)}_{/\approx}$ to be the set of equivalence classes of tiles. This is an orbit-finite set. We are now ready to define the separating language.

Definition 8.14 (CFI property)**.** Define an $n \times n$ $\approx$-tiling to be a function

$$\mathcal{T} : n \times n \to \mathbb{A}^{(8)}_{/\approx}.$$

We say that $\mathcal{T}$ satisfies the CFI property[2] if there exists a consistent tiling

$$\mathcal{S} : n \times n \to \mathbb{A}^{(8)}$$

which projects to $\mathcal{T}$ when tiles are replaced by their equivalence classes.

Formally speaking, the separating language required for Theorem 8.13 should be a set of words, and not $\approx$-tilings, because Turing machines input words. Therefore, we assume some convention on linearly ordering the tiles in an $\approx$-tiling, e.g. the tiles are ordered first by columns then by rows. Under such a convention, an $n \times n$ $\approx$-tiling can be encoded uniquely as a word of length n^2 over the alphabet $\mathbb{A}^{(8)}_{/\approx}$.

To prove Theorem 8.13, we will show that a nondeterministic orbit-finite Turing machine can check if an $\approx$-tiling satisfies the CFI property, but a deterministic one cannot.

The positive part about nondeterministic machines is immediate. The work alphabet of the machine is $\mathbb{A}^{(8)}_{/\approx} \cup \mathbb{A}^{(8)}$ plus additional symbols that are used as markers. Given an input word representing some $\approx$-tiling $\mathcal{T}$, the machine uses nondeterminism to guess the consistent tiling $\mathcal{S}$ which witnesses the CFI property. Then, it deterministically checks if the adjacency constraints of a consistent tiling are satisfied by $\mathcal{S}$. This computation can be done in a polynomial number of steps.

The interesting part is that deterministic machines cannot check the CFI property.

The CFI property is not recognised by any deterministic Turing machine. We begin by discussing a doubt the reader might have at this point. Given an input representing a $\approx$-tiling $\mathcal{T}$, there are only finitely many (if exponentially many) possibilities

[2]The name stands for Cai, Fürer and Immerman, who first studied this property in Cai et al. (1992).

for choosing the witness $\mathcal{S}$ as in Definition 8.14. Why not use a deterministic algorithm that exhaustively enumerates all the possibilities? The problem is that such an algorithm cannot be implemented as a deterministic Turing machine. The intuitive reason is that even if a $\approx$-equivalence class has only 8 tiles, one cannot choose deterministically any single one among them (i.e. there is no notion of the "first" or "second" element of the equivalence class) to write it down on the tape.

We now proceed to give a formal proof of why the CFI property is not recognised by any deterministic Turing machine. This will be a consequence of Lemma 8.16 below, which says that a deterministic Turing machine, unlike the CFI property, is insensitive to certain well chosen flips in an $\approx$-tiling.

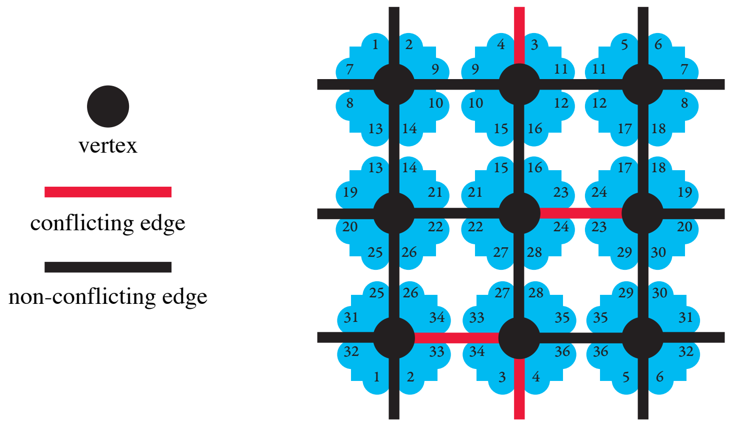
We lift the notion of flips from tiles to their $\approx$-equivalence classes as follows. If τ is a tile, then the *flip* of its $\approx$-equivalence class is defined to the $\approx$-equivalence class which contains some (equivalently, any) flip of τ. It is easy to see that this notion does not depend on the choice of τ in its $\approx$-equivalence class, nor does it depend on the choice of which side was flipped. Flipping is an involution on $\approx$-equivalence classes, i.e. doing a flip twice leads back to the same $\approx$-equivalence class.

The following lemma shows that flips violate the CFI property.

Lemma 8.15. *Let $\mathcal{T}$ be an $n \times n$ $\approx$-tiling which satisfies the* CFI *property. Then for every $x \in n \times n$, the following $\approx$-tiling violates the* CFI *property:*

$$\mathcal{T}_x(y) \stackrel{\text{def}}{=} \begin{cases} \textit{flip of } \mathcal{T}(y) & \textit{if } y = x; \\ \mathcal{T}(y) & \textit{otherwise.} \end{cases}$$

Proof. A parity argument. We view an $n \times n$ grid as a graph, where vertices are grid positions, and grid positions are connected by an edge if they are adjacent in the (torus) grid topology. For $\mathcal{S} : n \times n \to \mathbb{A}^{(8)}$ define the *conflict set* to be the set of edges e in the graph corresponding to $n \times n$ such that the colours of the two sides adjoining on e are different. Here is a picture:



Using this terminology, an $\approx$-tiling $\mathcal{T}$ satisfies the CFI property only if there exists some $\mathcal{S}$ which has an empty conflict set and such that $\mathcal{T}$ is the $\approx$-equivalence class of $\mathcal{S}$. The key observation is that $\mathcal{S} \approx \mathcal{S}'$ implies that the conflict sets have the same parity (i.e. size modulo two); and furthermore making one flip makes this parity change. □

We are now ready to prove the main lemma which witnesses that the CFI property is not recognised by any deterministic orbit-finite Turing machine. Fix a deterministic orbit-finite Turing machine. We use the formalisation of computations from Section 8.2, i.e. a computation is a function $\rho : \mathbb{N}^2 \to \Delta$, where the Δ is the work alphabet plus pairs (letter of the work alphabet, state of the machine). If $\mathcal{T}$ is an $\approx$-tiling, we write $\rho_{\mathcal{T}}$ for the unique computation of the fixed Turing machine on the word representing $\mathcal{T}$.

Lemma 8.16. *There exists $k \in \{0, 1, \ldots\}$ with the following property. Let $n \in \{0, 1, \ldots\}$ be sufficiently large, and let $\mathcal{T}$ be an $n \times n$ $\approx$-tiling which satisfies the CFI property. Assuming the notation $\mathcal{T}_x$ defined in Lemma 8.15, the following holds for every $i, j \in \mathbb{N}$:*

$$\rho_{\mathcal{T}}(i, j) = \rho_{\mathcal{T}_x}(i, j) \qquad \textit{for all } x \in n \times n \textit{ with at most } k^2 \textit{ exceptions.} \tag{*}$$

Before proving the lemma, we use it to finish the proof of Theorem 8.13. Take k as in the lemma, and let n be sufficiently large. Let $\mathcal{T}$ be some $n \times n$ $\approx$-tiling which satisfies the CFI property. Consider the computation $\rho_{\mathcal{T}}$, and let (i, j) be the place in the computation which contains the head at the moment when it accepts. If $n > k^2$, then (*) in the lemma implies that there is some $x \in n \times n$ such that $\rho_{\mathcal{T}_x}$ has the same contents. In particular, the machine also accepts $\mathcal{T}_x$. This contradicts Lemma 8.15.

Proof of Lemma 8.16. Choose k so that

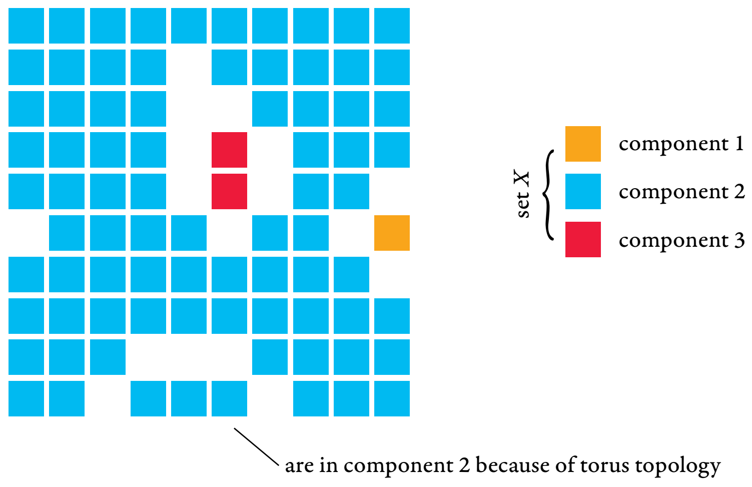
$$k/2 > \text{support size for the Turing machine} + \underbrace{\text{support size for cell contents}}_{\substack{\text{smallest } l \text{ such that } \rho_{\mathcal{T}}(i,j) \text{ has a} \\ \text{support of size } l \text{ for every } i, j \in \mathbb{N}}}.$$

We prove (*) by induction on i, i.e. the number of computation steps of the Turing machine. For the induction base of $i = 0$, we observe that the contents of a cell in time $i = 0$ depend only on the value of the input in at most one grid position, and hence (*) holds with at most one exception.

For the induction step, suppose that (*) is true for $i - 1$ and consider the case of i. In the computation of a Turing machine, the contents of a cell in time i are uniquely determined by the contents of at most two cells in time $i - 1$: the cell in the same column (offset from the beginning of the tape), plus possibly the contents of the unique cell in time $i - 1$ which contains the head of the machine. Hence, using the induction assumption we can conclude the following weaker version of (*), which uses $2k^2$ exceptions instead of k^2:

$$\rho_{\mathcal{T}}(i, j) = \rho_{\mathcal{T}_x}(i, j) \qquad \text{for all } x \in n \times n \text{ with at most } 2k^2 \text{ exceptions.} \tag{**}$$

In the rest of this proof, we bring back the number of exceptions down to k^2. To do this, we talk about connected components in $\mathcal{T}$ after removing some grid positions from the input $\mathcal{T}$. For a subset $X \subseteq n \times n$ of grid positions, define its *connected components* to be the connected components in the subgraph of the graph of $n \times n$ (as defined in the proof of Lemma 8.15) induced by X. Here is a picture of a set X together with its partition into connected components:



We now resume the proof of the implication from (**) to (*). Choose a tuple of atoms $\bar{a}$ which supports both the Turing machine and the cell contents $\rho_{\mathcal{T}}(i, j)$. By choice of k, we can assume that $\bar{a}$ less than $k/2$ atoms. Define

$$Z \subseteq n \times n$$

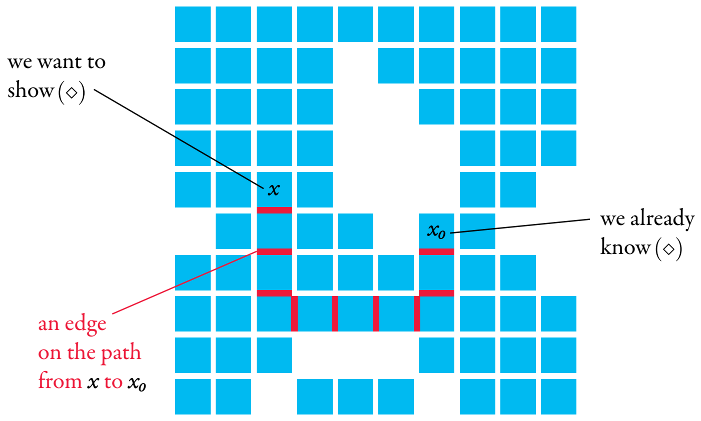
to be the grid positions where $\mathcal{T}$ uses at least one atom from $\bar{a}$. The set Z has less than k grid positions, since every atom appears in at most two grid positions and k is more than twice the size of $\bar{a}$. By a straightforward analysis of connectivity in an $n \times n$ grid, one can conclude that if n is big enough, then the graph corresponding to $n \times n - Z$ has a connected component, call it X, which contains all grid positions from $n \times n$ with at most k^2 exceptions. If n is big enough, then

$$\underbrace{2k^2}_{\text{number of exceptions in (**)}} < \underbrace{n^2 - k^2}_{\text{size of } X},$$

and therefore there is some $x_0 \in X$ which satisfies

$$\rho_{\mathcal{T}}(i, j) = \rho_{\mathcal{T}_x}(i, j). \tag{$\diamond$}$$

Using this x_0, we will show that all $x \in X$ also satisfy ($\diamond$), thus proving (*). Let $x \in X$. Since X is connected and disjoint from Z, in the graph corresponding to $n \times n$ there is a path which goes from x to x_0 and avoids grid positions from Z. Here is a picture:



Every edge e of the grid $n \times n$ corresponds to two distinct atoms. Define π to be the atom automorphism which swaps, for every e on the path from x to x_0, the two atoms that correspond to the edge e. This atom automorphism fixes all atoms from $\bar{a}$. For each tile except those corresponding to x and x_0, the automorphism flips an even number of sides, and hence we have:

$$\mathcal{T}_x = \pi(\mathcal{T}_{x_0}). \tag{8.1}$$

The path from x to x_0 was chosen so that it avoids atoms in the support of the Turing machine, and therefore

$$\pi(\rho_{\mathcal{T}}) = \rho_{\pi(\mathcal{T})} \qquad \text{for every input } \mathcal{T} \text{ to the machine.} \tag{8.2}$$

We are now ready to prove that x satisfies (◇):

$$\begin{array}{rcl}
\rho_{\mathcal{T}_x}(i,j) & = & \text{(by (8.1))} \\
\rho_{\pi(\mathcal{T}_{x_0})}(i,j) & = & \text{(by (8.2))} \\
\pi((\rho_{\mathcal{T}_{x_0}})(i,j)) & = & \text{(because } x_0 \text{ satisfies (◇))} \\
\pi((\rho_{\mathcal{T}})(i,j)) & = & \text{(because } \pi \text{ fixes the support of } \rho_{\mathcal{T}}(i,j)) \\
\rho_{\mathcal{T}}(i,j). & &
\end{array}$$

This completes the proof of the lemma, and therefore also of Theorem 8.13. □

Exercise 158. Assume the equality atoms. Show that if $k \le 3$ and the input alphabet Σ is k-tuples of atoms modulo some equivariant equivalence relation, then every nondeterministic Turing machine over input alphabet Σ can be determinised.

Exercise 159. In the proof of Theorem 8.13, we used an input alphabet which consisted of 8-tuples of atoms modulo some equivalence relation. Improve the proof to use 6-tuples modulo some equivalence relation[3].

Exercise 160. Assume the equality atoms and consider the alphabet

$$\{\{\{a,b,c\},\{d,e,f\}\} : a,b,c,d,e,f \text{ are distinct atoms}\}.$$

[3]This exercise is based on Klin et al. (2014); in particular Section 5.1 of that paper shows that 5 is the smallest dimension where Theorem 8.13 holds.

Show that Turing machines over this input alphabet cannot be determinised.

Bibliography

Blass, Andreas. 2013. *Power-Dedekind Finiteness*. http://www.math.lsa.umich.edu/ablass/pd-finite.pdf. [Online; accessed September 6, 2019].

Bojańczyk, Mikołaj, Klin, Bartek, Lasota, Sławomir, and Toruńczyk, Szymon. 2013a. Turing Machines with Atoms. Pages 183–192 of: *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*.

Bojańczyk, Mikołaj, Segoufin, Luc, and Toruńczyk, Szymon. 2013b. Verification of database-driven systems via amalgamation. Pages 63–74 of: *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*.

Bojańczyk, Mikołaj, Klin, Bartek, and Lasota, Sławomir. 2014. Automata theory in nominal sets. *Logical Methods in Computer Science*, **10**(3).

Cai, Jinyi, Fürer, Martin, and Immerman, Neil. 1992. An optimal lower bound on the number of variables for graph identifications. *Combinatorica*, **12**(4), 389–410.

Cheng, Edward Y. C., and Kaminski, Michael. 1998. Context-Free Languages over Infinite Alphabets. *Acta Inf.*, **35**(3), 245–267.

Clemente, Lorenzo, and Lasota, Sławomir. 2015a. Reachability Analysis of First-order Definable Pushdown Systems. Pages 244–259 of: *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*.

Clemente, Lorenzo, and Lasota, Sławomir. 2015b. Timed Pushdown Automata Revisited. Pages 738–749 of: *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6–10, 2015*.

Courcelle, Bruno, and Engelfriet, Joost. 2012. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*. Encyclopedia of Mathematics and Its Applications, vol. 138. Cambridge University Press.

Engeler, ERWIN. 1959. A characterization of theories with isomorphic denumerable models. *Notices Amer. Math. Soc*, **6**, 161.

Ferrari, Gian Luigi, Montanari, Ugo, and Pistore, Marco. 2002. Minimizing Transition Systems for Name Passing Calculi: A Co-algebraic Formulation. Pages 129–158 of: *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8-12, 2002, Proceedings.*

Gabbay, Murdoch, and Pitts, Andrew M. 2002. A New Approach to Abstract Syntax with Variable Binding. *Formal Asp. Comput.*, **13**(3-5), 341–363.

Hodges, Wilfrid. 1993. *Model Theory.* Encyclopedia of Mathematics and its Applications. Cambridge University Press.

Klin, Bartek, and Łełyk, Mateusz. 2017. Modal Mu-Calculus with Atoms. 21 pages.

Klin, Bartek, Lasota, Sławomir, Ochremiak, Joanna, and Toruńczyk, Szymon. 2014. Turing machines with atoms, constraint satisfaction problems, and descriptive complexity. Pages 58:1–58:10 of: *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14–18, 2014.*

Klin, Bartek, Kopczyński, Eryk, Ochremiak, Joanna, and Toruńczyk, Szymon. 2015. Locally finite constraint satisfaction problems. Pages 475–486 of: *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS).* IEEE Computer Society.

Murawski, Andrzej, Ramsay, Steven, and Tzevelekos, Nikos. 2014. Reachability in Pushdown Register Automata. Pages 464–473 of: *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part I.*

Murawski, Andrzej, Ramsay, Steven, and Tzevelekos, Nikos. 2015. Bisimilarity in Fresh-Register Automata. Pages 156–167 of: *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6–10, 2015.*

Pitts, Andrew M. 2013. *Nominal Sets: Names and Symmetry in Computer Science.* Cambridge Tracts in Theoretical Computer Science, vol. 57. Cambridge University Press.

Ryll-Nardzewski, Czesław. 1959. On the categoricity in power $\leq \aleph_0$. *Bull. Acad. Polon. Sci. Sér. Sci. Math. Astr. Phys*, **7**, 545–548.

Svenonius, Lars. 1959. No-categoricity in first-order predicate calculus 1. *Theoria*, **25**(2), 82–94.

Thomas, Wolfgang. 1990. Automata on Infinite Objects. Pages 133–192 of: *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B).*

Author index

Subject index