Logic and Automata over Infinite Trees

Der Fakultät für Mathematik, Informatik und Naturwissenschaften
der Rheinisch-Westfälischen Technischen Hochschule Aachen
vorgelegte Habilitationsschrift zur Erlangung der Venia Legendi.

von

Dr. rer. nat. Christof Löding
aus Neumünster, Schleswig-Holstein

Contents

Introduction

In computer science, infinite trees are used as a mathematical model for capturing the possible behaviors of state based systems of various kinds such as, e.g., protocols or circuits. A discrete system can usually be described by a transition graph modelling the states of the system and the possible transitions between these states. Unwinding such a graph into an infinite tree gives a representation of the possible executions of the system. In this thesis we deal with two formalisms for reasoning about infinite trees: automata and logic.

Logic provides a natural framework for describing properties of infinite trees. This approach is used in the field of verification, where the central problem is to check whether a given system satisfies certain properties. These properties are usually expressed in a suitable logical formalism. For solving decision problems in logic, automata have proven to be a powerful tool in many situations. The translation of logical specifications into equivalent automata allows to reduce problems in mathematical logic to automata theoretic problems, which are often of graph theoretic nature. This connection has been exploited in various settings to solve central decision problems for logical theories or in the model-checking approach in verification.

In the past decades, many results in the field of automata and logic have been obtained, and a rich theory has been developed, as documented in various surveys and textbooks [118, 128, 62, 48, 122]. In this thesis we pursue this line of research. The results we present can roughly be divided into two categories:

1. We study purely automata theoretic questions to obtain a deeper understanding of the automaton model (Chapters 4 and 5).

2. We apply automata theoretic methods to solve questions on decidability and definability in logic (Chapters 3 and 6).

Of course, there is an interplay between these two types of problems. Learning more about the automaton model certainly enables us to develop new methods for applying automata. Vice versa, solving logical questions can also help to gain new insights on the automaton model.

In the following, we briefly sketch the development of automata theoretic methods and their applications to logic, and then give an overview of the material contained in this thesis.

The automata and logic connection

The connection between finite automata and logic has first been used to solve decision problems for certain restricted forms of second-order arithmetic. First results showed the decidability of the weak monadic second-order theory of one successor [16, 43, 119]. From a different perspective, this logical formalism can be seen as an extension of boolean logic with a time component: It can be used to describe finite sequences of boolean vectors by using boolean logic and additionally allowing to quantify over points in time and over finite sets of such points. This is a natural extension of boolean logic because such sequences of boolean vectors naturally arise, e.g., during an execution of a sequential circuit.

Since finite sequences of boolean vectors can be interpreted as finite words over a suitable alphabet, finite automata also provide a natural framework for describing such sequences. The fundamental theorem of Büchi, Elgot, and Trakhtenbrot states that the languages (sets of sequences or words) that can be defined in weak monadic second-order logic are the same as those that can be described by finite automata. The proof of this theorem makes use of the strong closure properties of finite automata on finite words.

A generalization of finite word automata is obtained by passing to finite trees as input structures. The theory of tree automata was established in the late sixties by Thatcher and Wright [117] and Doner [41]. They showed that the theory of standard automata, including the connection to logic, can be transferred from the domain of finite words to the domain of finite trees. Quoting from the introduction of [117], where the theory of tree automata is called generalized finite automata theory: "...the results presented here are easily summarized by saying that conventional finite automata theory goes through for the generalization – and it goes through quite neatly!"

In this thesis we are interested in automata on infinite objects, particularly in automata on infinite trees. The theory of automata on infinite words was founded by Büchi in [17], where he showed that the monadic second order theory of $(\mathbb{N}, \mathit{succ})$, i.e., of the natural numbers with successor function is decidable (this theory is also called S1S). In analogy to finite words and trees, the proof uses a translation of formulas into automata, which in this setting run on infinite input words.

The principle of finite automata on infinite words is the same as for finite automata on finite words: a transition relation or function describes the possible moves of the automaton. On an infinite word an automaton produces an infinite sequence of states. This implies that we cannot simply lift the acceptance condition of finite automata, which requires the automaton

to reach a final state at the end of the input, to this new setting. Instead, Büchi proposed to consider the states that occur infinitely often during a run, and to declare a run accepting if at least one of these states is from a designated set of accepting states. This model of automaton is nowadays referred to as Büchi automaton.

Besides their use for decidability proofs in logic, Büchi automata have become a major tool in verification of finite state systems by model checking. The corresponding theory goes back to the work of Vardi and Wolper [123] who showed that formulas of the temporal specification logic LTL introduced by Pnueli [98] can be translated into equivalent Büchi automata of singly exponential size. This contrasts the non-elementary blow-up when translating monadic second-order formulas as considered by Büchi into automata [115] (see also [105]). Since the work of Vardi and Wolper, many optimization techniques have been developed to improve the translation from LTL to automata and also the algorithmic analysis of Büchi automata, leading to tools like SPIN [54], which are able to solve large instances of verification problems.

While in verification the problem is to check if a given system satisfies a given specification, the problem of synthesis is to automatically construct a system satisfying a given specification. In [27] the problem of circuit synthesis for specifications written in monadic second-order logic was raised, sometimes referred to as Church's synthesis problem. In this setting the circuit receives a sequence of input signals and produces a sequence of output signals. The specification relates these two types of signals and describes the desired behavior of the circuit. The task is to automatically construct a circuit from the formula such that the specification is satisfied for all possible inputs. This problem was solved by Büchi and Landweber in [19] using a game-theoretic formulation. Their solution requires the use of deterministic automata. Above we have mentioned the model of Büchi automaton, which is a nondeterministic automaton model. A more complex acceptance condition was proposed by Muller in [87], claiming that deterministic automata equipped with this acceptance condition (called Muller automata) are as expressive as nondeterministic Büchi automata. This question has been settled by McNaughton in [80]. The techniques developed by Büchi/Landweber and McNaughton for solving the described synthesis problem can be found in the monograph [120].

Another path that generalizes the results of Büchi on the decidability of S1S and also allows to solve the synthesis problem was taken by Rabin who introduced the model of automata on infinite trees [102], called Rabin tree automata. In the same way as Büchi automata are used to show the decidability of S1S, Rabin tree automata can be used to show the decidability of S2S, the monadic second-order theory of two successors (i.e., of the infinite binary tree). In [104] Rabin shows how to use tree automata for solving Church's synthesis problem. Similar to the game-theoretic approach by

Büchi and Landweber, a main ingredient is the determinization theorem for automata on infinite words. In fact, there is a very close relationship between automata on infinite trees and games of infinite duration, as used by Büchi and Landweber in [19]. These connections are well documented in the survey articles [118] and [128]. From that perspective, the two approaches for the solution of Church's synthesis problem are very similar.

As for the complexity, the solution of the synthesis problem for monadic second-order specifications is non-elementary because it is a generalization of the decision problem for S1S. We have mentioned above the temporal logic LTL that is nowadays used in verification because it admits a more efficient translation to Büchi automata. Pnueli and Rosner studied the synthesis problem for LTL specifications and showed that it can be solved in doubly exponential time [99]. The additional exponential compared to the model checking problem for LTL originates from the required determinization of the Büchi automaton constructed from the LTL formula. Although the construction for the determinization of Büchi automata has been improved in the last decades [46, 107, 97, 59], the structure of the resulting automata is rather complex, and the lack of methods for efficient simplification of deterministic automata has prevented usable implementations of the synthesis algorithms. There are more recent approaches trying to avoid determinization constructions [69], or using simpler specification formalisms than LTL [14].

Besides their applications in synthesis problems, tree automata can also be used for the verification of branching time specifications. The modal μ-calculus is an extension of basic modal logic by least and greatest fixpoint operators [6]. The model checking problem for μ-calculus specifications on finite transition structures is polynomial time equivalent to the emptiness problem for tree automata with a parity acceptance condition or the problem of finding a winning strategy in a so called parity game [114]. The parity condition has been introduced in [84] for automata on infinite trees. Its importance for infinite games has been realized later in [45] and [85]. Today it plays a central role in the theory of infinite games and automata on infinite trees and words. Sketched briefly, a parity condition gives a partial order on the state set, grouping states in levels according to their importance, and additionally assigns to each level the status of being "good" or "bad". An infinite sequence of states is accepting if the most important level that is visited infinitely often is good. Technically, this is usually done by assigning to each state a natural number. All states with the same number belong to the same level, the order of the levels is inherited from the order on the numbers, and the levels with an even number are declared as good and those with an odd number as bad (this convention using even and odd numbers explains the name parity condition).

Several algorithms for solving parity games have been developed in the last years (see [57, 124, 58, 108] for some of them), but the question whether

these games can be solved in polynomial time remains open. By the connection mentioned above, a polynomial time algorithm for parity games would also mean that the model checking problem for the μ-calculus would be solvable in polynomial time.

The above sketch of the development of the interplay between automata and logic should give an impression of how useful automata are for dealing with problems arising in mathematical logic, and that some of the developed techniques have reached the state of industrial applicability. This justifies a further study of these models to obtain a deeper understanding of their properties and to find new areas of application for them.

Overview of this thesis

In this thesis we present some recent results in the theory of automata on infinite trees as well as some applications of these automata in the area of logic. For this purpose, we have to work with several models of automata, with games of infinite duration, and with different logical formalisms. The first two chapters are devoted to the presentation of this basic material. We give all required definitions and state the main results that we refer to throughout this thesis. Some of these results are even reproved because we have to refer to the details of the proofs in later chapters. The presentation of the basic material is split into two parts. Chapter 1 introduces automata on infinite words, games of infinite duration, and automata on infinite trees. A large part is devoted to the connection between games and tree automata, and we give a detailed proof of the complementation theorem for tree automata because this construction is used in Section 4.3. In Chapter 2 we introduce the basic notions from logic as required for this thesis, and we present some results on the decidability of logical theories, in particular ways to represent or transform structures in such a way that the obtained structure has a decidable theory (for a logic depending on the specific representation or transformation).

After these introductory chapters the main part of the thesis is contained in Chapters 3 to 6. We give short descriptions of the background and results of the respective chapters and how they are connected.

Choice functions and well-orderings. Chapter 3 deals with definability questions for monadic second-order (MSO) logic over the infinite binary tree. The main question considered in the chapter originates in the *uniformization problem*, i.e., the question whether a relation that is definable in some logic can be transformed into a function that is compatible with this relation and can also be defined in this logic. The synthesis problem that we touched upon above can be seen as a variation of uniformization: A logical specification for the behavior of a sequential circuit defines a relation between the inputs and the outputs of this circuit, which can be viewed as a

relation between infinite words. The goal is to construct a circuit that satisfies this specification. Since a circuit computes a function from inputs to outputs, this corresponds to the construction of a (specific) function that is compatible with the given relation. This is a variation of the uniformization problem because it asks for a very specific function, namely one that can be computed sequentially, i.e., without anticipating the input.

We are interested in relations that are defined in monadic second-order logic over the infinite line (the natural numbers with successor) or the infinite binary tree. In this setting, given a formula $\phi(\bar{X}, \bar{Y})$ with vectors $\bar{Y}, \bar{X}$ of free variables, such that $\forall \bar{X} \exists \bar{Y} \phi(\bar{X}, \bar{Y})$ is satisfiable, uniformization asks for a formula $\phi^*(\bar{X}, \bar{Y})$ such that

1. ϕ^* implies ϕ (each interpretation of $\bar{Y}, \bar{X}$ making ϕ^* true also makes ϕ true),

2. and ϕ^* defines a function in the sense that for each interpretation of $\bar{X}$ there is exactly one interpretation of $\bar{Y}$ making ϕ^* true.

MSO on the infinite line is known to have the uniformization property [112]. In the same paper, Siefkes conjectures that MSO on the infinite binary tree does not have the uniformization property. In [49] Gurevich and Shelah published the result that there is no MSO-definable choice function on the infinite binary tree. An MSO-definable choice function is given by a formula $\phi(X, y)$ with one free set variable X and one free element variable y, such that for each non-empty set X there is exactly one y such that ϕ is satisfied. That is, the formula ϕ chooses one element from each non-empty set. Such a choice function corresponds to a uniformization of the element relation, hence the result of Gurevich and Shelah shows that uniformization is not possible on the infinite binary tree in general.

However, the proof given in [49] is very complex and uses the set-theoretic technique of forcing, while it seems that automata offer a simpler and more direct approach to prove the same result. In Section 3.1 we present such a proof of the result of Gurevich and Shelah, and use the result in Section 3.2 to infer an undefinability result for strategies in infinite games.

The last section of Chapter 3 deals with well-orderings of the nodes of the infinite binary tree. In general, a well-ordering is an order relation without infinite decreasing chains. Such orderings are particularly useful because they allow to pick from each set a unique element, namely the minimal one. Hence, as a consequence of the undefinability of choice functions we obtain that it is not possible to define a well-ordering relation in MSO on the infinite binary tree. Nevertheless, it might be possible that there exists some well-ordering relation that we can add to the infinite binary tree such that the resulting structure still has a decidable MSO-theory. In Section 3.3 we show that this is not the case and illustrate with some examples how this result can be applied.

The results in this chapter have been obtained in collaboration with Arnaud Carayol and are published in [21].

Unambiguous and guidable automata. In Chapter 4 we study automata on infinite trees with specific properties. The underlying motivation is to find normal forms of automata or at least classes of automata with good properties that allow to represent all automaton definable languages. Consider, for example, regular languages of finite words. When reasoning about such languages one can always pick some canonical object representing them, e.g., the minimal deterministic automaton or the syntactic monoid. Similarly, for regular languages of finite trees, there are unique minimal deterministic bottom-up automata. For regular languages of infinite words there is no normal form for automata but one can work with the syntactic ω-semigroup or at least pick a deterministic automaton for the language, which has a unique run on each input word.

For automata on infinite trees no canonical representation is known, and it is rather easy to see that there is no useful notion of deterministic automaton on infinite trees that would be expressive enough to capture the whole class of regular languages of infinite trees (see Section 4.1). For this reason, we study two other properties of automata: unambiguity and guidability. Unambiguous automata have been introduced in [113] on finite words because they allow more efficient algorithms for equivalence and inclusion testing. These are nondeterministic automata that have exactly one accepting run for each input that is accepted. In Section 4.2 we use the results from Chapter 3 to obtain negative results concerning unambiguous automata on infinite trees, following a proof strategy from unpublished work of Niwiński and Walukiewicz [92].

In Section 4.3 we define the concept of guidable automaton. Intuitively, such an automaton can simulate every other automaton for the same language in the following sense: It can build up an accepting run (on a tree that is in the language) in a deterministic way if it is given an accepting run of the other automaton for the same language. We call these automata "guidable" because any other automaton for the language can serve as a guide for constructing accepting runs. The main result is that each regular language of infinite trees can be accepted by a guidable automaton. Although this is not a canonical representation of a language because guidable automata are not unique, we have found in a certain sense a most general way of representing a language of infinite trees by an automaton.

The results from this chapter have been obtained in collaboration with Arnaud Carayol [21] and Thomas Colcombet [32].

The parity index problem. The parity index problem refers to the complexity of the acceptance condition of infinite tree automata. While for non-

deterministic automata on infinite words it is enough to consider the Büchi acceptance condition, which is specified by a set of accepting states that has to be visited infinitely often, more involved acceptance conditions are necessary for automata on infinite trees to obtain the full expressive power of MSO. Many different acceptance conditions have been proposed but the parity condition [84] has turned out to have the best properties of those conditions that are strong enough to capture the whole class of MSO-definable languages.

We have already described the parity condition above. Technically it is usually given by a mapping from the states to natural numbers. The parity index problem is to determine for a given regular language of infinite trees the minimal range of this mapping required in the acceptance condition of an automaton for this language. This parameter gives a measure of the complexity of the language, and it also plays an important role for the complexity of several algorithms related to parity automata and games. More details on the parity index problem and known results on it can be found in Section 5.1.

Our contribution is a reduction of the parity index problem to a completely different problem for an automaton model called distance parity automata. This model is an extension of the nested distance desert automata used by Kirsten to solve the star-height problem for regular languages of finite words [64]. In the reduction we use the concept of guidable automata from Chapter 4. For more details on the proof scheme and the model of distance parity automata we refer the reader to the introduction of Chapter 5.

The material in Chapter 5 results from a joint work with Thomas Colcombet published in [32].

Finite set interpretations on trees. As already described earlier in this introduction, automata-theoretic methods are successfully used in the verification of finite state systems. A more recent branch of research considers verification of infinite structures because in many settings accurate models of given systems need to be infinite. Starting from early decidability results for prefix rewriting systems and pushdown graphs [18, 89], efficient algorithms for the verification of recursive programs over finite data types have been developed [47]. Besides the task of developing practically usable algorithms for specific applications, this also raises the question of what kind of systems admit automatic decision procedures.

This is the subject of computational model theory (cf. [12]), where the focus is on the problem of model checking infinite structures against specifications written in some logic, i.e., deciding for a given structure and logical formula if the formula holds in this structure. This problem setting has been studied for various specification logics and different formalisms for representing the infinite structures. The most prominent logics in this

context are first-order (FO) logic and monadic second-order (MSO) logic, and they have led to two tracks of research trying to identify classes of structures for which the respective logic is decidable.

One way of defining such classes uses, e.g., words or trees for representing the elements of the structure and simple transformations based on transducers or rewriting to define the relations of the structure. In this way, one obtains, e.g., the classes of automatic [53, 61, 11] and tree-automatic [39, 11] structures, for which the FO-theory is decidable, and the classes of pushdown-graphs [89] and prefix recognizable graphs [25] with decidable MSO-theory.

In the case of automatic structures, the decidability results are based on the strong closure properties of finite automata, which are used to define the relations. Other techniques, e.g., in [70] for rewriting in trace monoids, are based on Gaifman's locality theorem. The decidability results for MSO logic on pushdown and prefix recognizable graphs are derived from the results of Büchi and Rabin establishing the equivalence of MSO with corresponding types of finite automata, as explained above.

A different and more systematic approach for defining and studying classes of infinite structures is to use operations for transforming structures. An important operation of this kind is the model-theoretic interpretation. Such an interpretation defines a new structure 'inside' a given one by means of logical formulas describing the domain and the new relations. Depending on whether these defining formulas are FO or MSO one speaks of FO- and MSO-interpretations. An important property of these interpretations is that decidability results easily transfer from the given structure to the resulting structure, i.e., applying an FO-interpretation to a structure with a decidable FO-theory results in a structure with decidable FO-theory, and similarly for MSO.

As mentioned in [12], this suggests a new way of defining interesting classes of infinite structures: fix an underlying structure (with good algorithmic properties) and consider all structures that can be obtained by applying interpretations of a certain kind. In this way, one obtains the automatic structures by FO-interpretations from, e.g., a suitable extension of Presburger arithmetic [9], and the prefix recognizable structures by MSO-interpretations from the infinite binary tree [10]. This idea has been pursued further in [26], where MSO-interpretations and unravelling of graphs are iterated, leading to an infinite hierarchy of graphs (or structures) with a decidable MSO-theory.

All the methods and results described so far can be separated into those concerned with FO logic (sometimes extended by a reachability relation [39, 28]) and those dealing with MSO logic (sometimes with only restricted kind of set quantification as in [78]). Our goal is to study a method for relating these two areas. We consider a new kind of interpretation, named finite set interpretation, allowing to define classes of structures with decid-

able FO-theory from structures with decidable MSO-theory. To be more precise, we are considering weak MSO (WMSO) logic, i.e., MSO logic where quantification is restricted to finite sets. The idea for these interpretations is rather simple: the domain of the new structure does not consist of elements of the old structure but of finite sets of elements of the old structure. The relations are specified by WMSO-formulas with free set variables (the number of which corresponds to the arity of the relation). In this way, FO-formulas over the new structure can directly be translated into WMSO-formulas over the old structure.

Using the equivalence of WMSO logic and finite automata (over finite words and trees) it is not difficult to see that the classes of automatic and tree automatic structures can be obtained by finite set interpretations from $(\mathbb{N}, succ)$ and from the infinite binary tree, respectively (see [29]). The connection between automatic structures and finite set interpretations applied to $(\mathbb{N}, succ)$ has already appeared before in the literature. In [44] the authors show that the infinite binary tree together with the equal level relation can be generated from $(\mathbb{N}, succ)$ by a finite set interpretation. Today this extension of the infinite binary tree is known as a generator of automatic structures in the sense that every automatic structure can be obtained from it by an FO-interpretation. A similar result is given in [106] but for another generator of the automatic structures.

This raises the question of what happens when we apply finite set interpretations to other structures with decidable WMSO-theory, e.g., the structures from the hierarchy defined in [26]. Though this hierarchy is strict, it is not a priori clear whether this is also true for the hierarchy obtained after applying finite set interpretations. To answer questions of this kind one has to study the expressiveness of finite set interpretations and to provide tools for showing that a structure cannot be obtained by such an interpretation applied to a given structure. In particular, such tools can then be used to answer questions on automatic structures because these can be obtained by finite set interpretations (as mentioned above). More precisely, we concentrate ourselves on understanding what are the structures which are finite set interpretable in trees. All the examples mentioned so far fall in this category.

We contribute two results to this study. The first one establishes that the quotient of a structure that is finite set interpretable in a deterministic tree is itself finite set interpretable in that tree. This result was known for automatic structures, and was open for tree-automatic structures. Our result is a generalization of those two cases.

The second and main result allows to reduce questions on definability by finite set interpretations to questions on WMSO-interpretability. Those questions of WMSO-interpretability in trees are well understood since they can be reformulated in terms of clique-width. The clique-width is a measure of the complexity of graphs which has been first introduced for finite graphs

[34], and then extended to infinite ones [36]. We also show how our method can be used to obtain non-definability results for specific structures and a strictness result for a hierarchy of structures.

Chapter 6 is based on joint work with Thomas Colcombet published in [30].

Acknowledgements

Most of my research in the last years was carried out in collaborations with my colleagues and friends Arnaud Carayol, Thomas Colcombet, and Olivier Serre. I would like to thank them for having worked with me and for the good times we spent during the mutual visits in Paris and in Aachen. And of course I hope that these collaborations will continue in the future.

All these collaborations, which are central to my scientific work, would have been impossible without the continuous support of Wolfgang Thomas. I am grateful to him for having guided me towards this habilitation thesis and for helping me finding my place in the scientific community.

Certainly, there are many other people that I have to thank. But since their contributions are not directly connected to this thesis I do not mention all their names here.

Chapter 1

Automata and games

The historical background for automata and games has already been presented in the introduction. In this chapter we introduce the necessary technical background. We start with some basic terminology and general notations, and then present automata on infinite words, infinite games, and automata on infinite trees, where we focus on the definitions and results that are used throughout this thesis. For further results and references we refer the reader to the surveys [118, 128, 122], the textbooks [62, 95], and the seminar volume [48].

General notations

The set of natural numbers (non-negative integers) is denoted by $\mathbb{N}$. A finite interval $\{i, \ldots, j\}$ of natural numbers is written as $[i, j]$.

An *alphabet* is a finite set of symbols, called *letters*. Usually, alphabets are denoted by Σ. The set of finite words over Σ is written as Σ^* and the set of infinite words as Σ^ω. For a word w (finite or infinite) we denote by $w(i)$ the ith letter of w, starting with 0, i.e., $w = w(0)w(1)w(2)\cdots$. The infix of w from position i to position j is denoted by $w[i, j]$.

The length of a finite word $w \in \Sigma^*$ is denoted by $|w|$ and ε is the empty word. For an infinite word $\alpha \in \Sigma^\omega$ we write $\mathit{Inf}(\alpha)$ for the set of symbols occurring infinitely often in α.

For all words $w_1, w_2 \in \Sigma^*$, w_1 is a *prefix* of w_2 (written $w_1 \sqsubseteq w_2$) if there exists $w \in \Sigma^*$ such that $w_2 = w_1 w$. If $w \in \Sigma^+$ then w_1 is a *strict prefix* of w_2 (written $w_1 \sqsubset w_2$). The *greatest common prefix* of two words w_1 and w_2 (written $w_1 \wedge w_2$) is the longest word which is a prefix of w_1 and w_2.

1.1 Automata on infinite words

The concept of automata on infinite words is derived from standard automata on finite words. We use a finite set of states and a transition relation

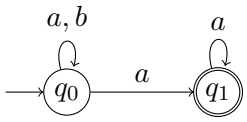


Figure 1.1: A Büchi automaton checking for finitely many b

or function (depending on whether we consider nondeterministic or deterministic automata) to describe possible computation steps of the automata. As usual, a run of the automaton on an input is a sequence of states that is consistent with the transition function. Since the inputs under consideration are infinite words, the runs are now infinite sequences of states. It remains to clarify when a run is accepting. There are various acceptance conditions, the most prominent one being the Büchi condition: A run is accepting if a designated set of accepting states is visited infinitely often during the run. This leads to the following automaton model.

A *nondeterministic Büchi word automaton* $\mathcal{A}$ is given as a tuple $\mathcal{A} = (Q, \Sigma, q^{\mathrm{i}}, \Delta, F)$ with a finite set Q of states, input alphabet Σ, initial state $q^{\mathrm{i}} \in Q$, transition relation $\Delta \subseteq Q \times \Sigma \times Q$, and accepting states $F \subseteq Q$.

A *run* of $\mathcal{A}$ on an infinite word $\alpha \in \Sigma^\omega$ is an infinite sequence $\rho = q_0 q_1 q_2 \cdots$ of states such that $q_0 = q^{\mathrm{i}}$, and $(q_i, \alpha(i), q_{i+1}) \in \Delta$ for all $i \geq 0$.

A run ρ is *accepting* if it contains infinitely many accepting states, i.e., if $\mathit{Inf}(\rho) \cap F \neq \emptyset$. A word α is accepted by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on α. The language accepted by $\mathcal{A}$ is the set of all accepted words

$$L(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } \alpha\}.$$

Büchi automata define a robust class of ω-languages that is closed under the boolean operations of union, intersection, and complement [17]. However, the proof of closure under complement is more involved than for automata on finite words. For finite words one can rely on determinization by the subset construction. Büchi automata require more involved constructions. This is witnessed, for example, by the fact that deterministic Büchi automata are weaker in expressive power than nondeterministic ones. The language of all words over the alphabet $\{a, b\}$ that contain finitely many b can be accepted by the Büchi automaton depicted in Figure 1.1. As usual, a small incoming arrow designates the initial state and a double circle the accepting states. The automaton nondeterministically chooses a point in the input after which no more b occur and moves from q_0 to q_1. If this guess is wrong the run cannot be continued with the next b. Hence, this automaton accepts exactly those words containing finitely many b.

It is not difficult to show that this language cannot be accepted by a deterministic Büchi automaton. To obtain a deterministic automaton model that is as expressive as Büchi automata, we have to use a stronger acceptance

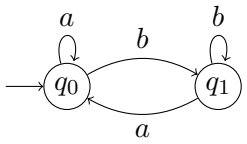


Figure 1.2: The transition graph of a deterministic automaton checking for finitely many b.

condition. The first model proposed for this purpose is the model of Muller automata [87]. In a Muller automaton one explicitly specifies all infinity sets of runs that lead to acceptance. Figure 1.2 shows the transition graph of such a deterministic automaton. Using a Muller condition expressing that only the infinity set $\{q_0\}$ is accepting, we obtain a deterministic Muller automaton that is equivalent to the Büchi automaton from above.

A theorem of McNaughton states that deterministic Muller automata are as expressive as nondeterministic Büchi automata.

THEOREM 1.1 ([80]) *For each nondeterministic Büchi automaton one can construct a deterministic Muller automaton accepting the same language.*

A construction with improved complexity is given in [107], which turns out to be asymptotically optimal (see [81], [118], [75], and [127]).

However, it is rather difficult to work with Muller automata because explicitly listing all accepting infinity sets might result in a rather complex acceptance condition. We prefer here to work with parity automata, a specialization of Muller automata, for which an analogue of Theorem 1.1 exists.

A *parity word automaton* is a tuple $\mathcal{A} = (Q, \Sigma, q^{\mathrm{i}}, \Delta, c)$ where the first four components are the same as in a Büchi automaton, and $c : Q \to \mathbb{N}$ is a *priority function* that assigns a natural number to each Q. A run is defined as for Büchi automata, and for a run ρ we refer by $c(\rho)$ to the corresponding infinite sequence of priorities obtained by applying c to each element of the sequence ρ.

The name "parity automaton" is justified by the way we interpret the priority function for the acceptance condition: A run is accepting if the maximal priority in $Inf(c(\rho))$ is even. For example, if we take the transition structure from Figure 1.2 and let $c(q_0) = 0$ and $c(q_1) = 1$, then the only accepting runs are those with infinity set $\{q_0\}$.

The parity condition has been introduced by Mostowski in [84] for automata on infinite trees, and has been used in the context of games independently by Emerson and Jutla [45], and Mostowski [85]. An explicit construction for deterministic parity automata on words is given in [23] (where they are called chain automata). The following result can for example be

obtained by combining the determinization construction from [107] with the construction from [23].

THEOREM 1.2 *For each nondeterministic Büchi automaton one can construct a deterministic parity automaton accepting the same language.*

There are also some more recent constructions that can be used transform nondeterministic Büchi automata into deterministic parity automata: One construction is due to Piterman [97], a refined version of Safra's construction [107], and another recent construction by Kähler and Wilke [59] builds on work of Muller and Schupp [90].

1.2 Games

There is a tight connection between automata on infinite trees and two-player games of infinite duration. This connection if often used to develop constructions or decision procedures for tree automata such as complementation or an emptiness test. Before we turn to automata on infinite trees we introduce in this section the basic definitions and results concerning games.

The games we are interested in are played on game graphs, also called arenas. An *arena* is a tuple $G = (V_\circ, V_\Box, E, col)$, where

- $V_\circ$ is the set of vertices of player *Eva* (graphically represented by circles),
- $V_\Box$ is the set of vertices of player *Adam* (disjoint from $V_\circ$, graphically represented by boxes),
- $E \subseteq V \times V$ for $V = V_\circ \cup V_\Box$ is the set of edges or moves, and
- $col : V \to Cols$ is a mapping that assigns to each vertex a color from a finite set *Cols* of colors.

A *play* in G is an infinite sequence of vertices $\gamma \in V^\omega$. One can imagine such a play being built up by the two players moving a token along the edges. Whenever the token is on a vertex of Eva, then she can choose the edge, otherwise Adam chooses.

Sometimes we are only interested in games starting in a specific initial vertex. In these cases we specify this vertex additionally to the other components of an arena.

To turn an arena into a game we have to add a winning condition. This is what the coloring of the vertices is used for. Given a play γ, we look at the corresponding sequence $col(\gamma) \in Cols^\omega$ of colors. The winning condition specifies whether this sequence is winning for Eva or for Adam. Hence, formally a *winning condition* is given by the set $Win \subseteq Cols^\omega$ of plays that

are winning for Eva. A *game* is a pair $\mathcal{G} = (G, \mathit{Win})$ of an arena and a winning condition (for Eva) as above.

The central notion in the theory of infinite games is the one of winning strategy. Given a game, we are interested in the question if one of the players can guarantee to win, no matter how the opponent plays. This is captured by the notion of winning strategy. Formally, a *strategy* $\sigma_\circ$ for Eva is a function that takes a finite prefix of a play ending a vertex of Eva, and outputs her next move:

$$\sigma_\circ : V^* V_\circ \to V$$

with the property that $\sigma_\circ(wv) = v'$ implies that $(v, v') \in E$. Similarly, a strategy for Adam is a function $\sigma_\Box : V^* V_\Box \to V$ such that $\sigma_\Box(wv) = v'$ implies that $(v, v') \in E$.

A play γ is said to be *played according to a strategy* $\sigma_\circ$ for Eva if for each i with $\gamma(i) \in V_\circ$ the next vertex is obtained by applying the strategy: $\sigma_\circ(\gamma[0, i]) = \gamma(i + 1)$. For a strategy of Adam the definition is analogous.

When we fix a strategy $\sigma_\circ$ for Eva, a strategy $\sigma_\Box$ for Adam, and an initial vertex v_0, then there is a unique play starting in v_0 that is played according to the two strategies. We call this the outcome of the two strategies from v_0 and denote it by $\mathit{out}(v_0, \sigma_\circ, \sigma_\Box)$.

Given a game $\mathcal{G} = (G, \mathit{Win})$ and an initial vertex v_0, a strategy $\sigma_\circ$ for Eva is a *winning strategy* from v_0 if all possible outcomes are winning for Eva: $\mathit{out}(v_0, \sigma_\circ, \sigma_\Box) \in \mathit{Win}$ for all strategies $\sigma_\Box$ of Adam. Similarly, a strategy $\sigma_\Box$ is a winning strategy for Adam from v_0 if $\mathit{out}(v_0, \sigma_\circ, \sigma_\Box) \notin \mathit{Win}$ for all strategies $\sigma_\circ$ of Eva.

The set of vertices from which Eva (Adam) has a winning strategy is called the *winning region* of Eva (of Adam). A game is called *determined* if from each of the vertices one of the players has a winning strategy. A winning condition *Win* is called determined if every game with this winning condition is determined.

Determinacy is interesting because we can conclude from the nonexistence of a winning strategy for one player that the other player has a winning strategy. This corresponds to an exchange of a universal and an existential quantifier: Eva does not have a winning strategy from v_0 is formalized has

$$\forall \sigma_\circ \exists \sigma_\Box \ : \ \mathit{out}(\sigma_\circ, \sigma_\Box, v_0) \notin \mathit{Win} \ .$$

Determinacy tells us that in this case Adam has a winning strategy:

$$\exists \sigma_\Box \forall \sigma_\circ \ : \ \mathit{out}(\sigma_\circ, \sigma_\Box, v_0) \notin \mathit{Win} \ .$$

This exchange of quantifiers can in some situations be used to decrease the complexity of problems. One example is the complementation of automata on infinite trees, presented in Section 1.4.

We are mainly interested in *parity games*. The parity condition is defined as for automata in Section 1.1. Each vertex is assigned a natural number

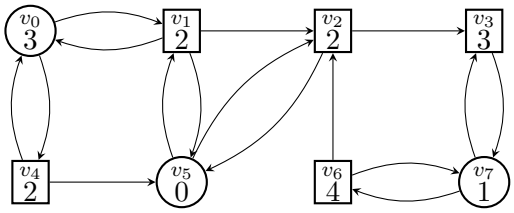


Figure 1.3: A parity game

from a finite range. As for automata we denote this special kind of coloring function by c instead of *col*. To determine the winner of a play γ we look at the maximal number that appears infinitely often in $c(\gamma)$. If this color is even, then Eva wins, otherwise Adam wins.

From this definition it is clear that we do not have to explicitly specify the winning condition *Win* for parity games because it is already specified by the coloring function c. Hence, parity games are completely specified by the game arena $(V_\circ, V_\Box, E, c)$.

An example for a parity game is shown in Figure 1.3. The numbers in the vertices determine the priority function, e.g., $c(v_0) = 3$.

Parity games are important because they are not only determined but the determinacy result already holds when restricting to a very simple type of strategies. If a player has a winning strategy from some node, then there is also a winning strategy that does not consider the history of the play for choosing the next move, but only the current vertex. Such strategies are called positional or memoryless.

Formally, a *positional strategy* for Eva is a function $\sigma_\circ : V_\circ \to V$ such that $(v, \sigma_\circ(v)) \in E$ for all $v \in V$. The same definition applies for Adam with $V_\Box$ instead of $V_\circ$.

For the parity game shown in Figure 1.3, the winning region of Eva is $\{v_2, v_3, v_5, v_6, v_7\}$, using the positional winning strategy $v_5 \to v_2$ and $v_7 \to v_6$. The winning region of Adam is $\{v_0, v_1, v_4\}$ with the positional winning strategy $v_1 \to v_0$ and $v_4 \to v_0$.

We call a game *positionally determined* if from each vertex one of the players has a positional winning strategy. The reason why parity games play a central role in the theory of automata and logics on infinite trees is the following theorem.

THEOREM 1.3 ([45, 85]) *Parity games are positionally determined.*

There are various proofs of this result. The proofs in the survey articles [128, 118] are based on an induction on the number of priorities. In [126] Walukiewicz gives a proof based on the concept of signatures or progress measures. On finite graphs this concept is used to derive an algorithm for

computing the winning regions in [57].

There are many other algorithms for solving finite parity games (see Chapter 7 of [48] for an overview, and the more recent works [58] and [108]). The most obvious algorithm can directly be derived from the positional determinacy: On a finite graph one can systematically check all possible strategies. The test whether a positional strategy is winning from some initial vertex boils down to a rather simple graph theoretic problem of checking for loops that are winning for the other player. This can be done in polynomial time. Hence, we obtain an NP algorithm for the problem of determining whether a player has an positional winning strategy in a parity game with a given initial vertex. Since parity games are symmetric in both players, the problem is in NP ∩ co-NP. We refer to the above decision problem as the problem of solving parity games.

THEOREM 1.4 *The problem of solving finite parity games is in NP∩ co-NP.*

The question whether parity games can be solved in polynomial time is one of the main open questions in this area.

Concerning the positional determinacy, a more general result is the positional determinacy for one player in Rabin games [65]. A *Rabin condition* over a finite set *Cols* of colors consists of a set $\Omega = \{(E_1, F_1), \ldots, (E_m, F_m)\}$ of pairs (E_i, F_i), where $E_i, F_i \subseteq \mathit{Cols}$ are sets of colors. An infinite sequence $\alpha \in \mathit{Cols}^\omega$ satisfies the Rabin condition Ω if there exists an index $i \in \{1, \ldots, m\}$ such that $\mathit{Inf}(\alpha) \cap E_i = \emptyset$ and $\mathit{Inf}(\alpha) \cap F_i \neq \emptyset$. In other words, there has to be a pair (E_i, F_i) such that α visits E_i only finitely often and F_i infinitely often. This type of condition originates from [102], where it is used as acceptance condition for automata on infinite trees.

A *Rabin game* $\mathcal{G} = (G, \Omega)$ is given by an arena $(V_\circ, V_\Box, E, \mathit{col})$ and a Rabin condition Ω. Eva wins a play γ if $\mathit{col}(\gamma)$ satisfies Ω.

THEOREM 1.5 ([65]) *Rabin games are determined and Eva has a positional winning strategy on her winning region.*

The dual of a Rabin condition is a Streett condition [116]. Syntactically, a *Streett condition* Ω is given in the same way as a Rabin condition as a set $\{(E_1, F_1), \ldots, (E_m, F_m)\}$ of pairs of color sets. An infinite sequence $\alpha \in \mathit{Cols}^\omega$ satisfies a Streett condition if for all $i \in \{1, \ldots, m\}$ either $\mathit{Inf}(\alpha) \cap E_i \neq \emptyset$ or $\mathit{Inf}(\alpha) \cap F_i = \emptyset$. Clearly, a Streett condition for Eva is a Rabin condition for Adam. Hence we directly obtain the following result.

COROLLARY 1.6 *Streett games are determined and Adam has a positional winning strategy on his winning region.*

We mentioned above that Theorem 1.5 is more general than Theorem 1.3. The reason is that a parity condition can easily be written as a Rabin con-

dition, and also as a Streett condition. Hence, Theorem 1.3 is a direct consequence of Theorem 1.5 and Corollary 1.6.

We briefly explain how to express a parity condition as a Rabin or Streett condition. W.l.o.g. we assume that the range of the priority function is $Cols = \{1, \ldots, 2k\}$. The Rabin condition $\{(E_1, F_1), \ldots, (E_k, F_k)\}$ is defined as follows: for $j \in \{1, \ldots, k\}$ the set E_j contains all $i > 2j$, and the set F_j only contains $2j$. The Rabin pair (E_j, F_j) expresses that the priorities bigger than $2j$ are visited only finitely often, and priority $2j$ is visited infinitely often, i.e., $2j$ is the maximal priority occurring infinitely often. Hence, the Rabin condition is satisfied iff the maximal priority occurring infinitely often is even.

As already explained, Streett conditions are complementary to Rabin conditions. To express a parity condition as a Streett condition we can simply express the fact that the maximal priority occurring infinitely often is odd by a Rabin condition in the same way as above, and then complement this condition to obtain a Streett condition.

We use the fact that Adam can win positionally in Streett games later for the special case of conjunctions of two parity conditions. A *conjunction of two parity conditions* is defined on a color set of the form $Cols = \{1, \ldots, k\} \times \{1, \ldots, \ell\}$. A sequence $\alpha \in Cols$ satisfies the conjunction of the two parity conditions if the projection to the first components yields a sequence satisfying the parity condition over $\{1, \ldots, k\}$, and the projection to the second components satisfies the parity condition over $\{1, \ldots, \ell\}$.

It is easy to see that Streett conditions are closed under conjunctions (because the nature of the condition is conjunctive). Hence the following remark is not surprising.

Remark 1.7 The conjunction of two parity conditions can be expressed by a Streett condition.

1.3 Automata on infinite trees

In this section we introduce the basic theory of automata on infinite trees. A very good survey of this subject can be found in [118]. To be self-contained we present here the definitions and constructions that are used in the other chapters.

The theory of automata on infinite trees as it originates from [102] uses the full infinite binary tree as underlying structure. In Chapter 6 we also consider trees that can have finite branches. The corresponding notations for this extended setting are introduced in Chapter 6.

We view the set $\{0, 1\}^*$ of finite words over $\{0, 1\}$ as the *domain* of an infinite binary tree. The *root* is the empty word ε, and for a node $u \in \{0, 1\}^*$ we call $u0$ its left successor (or 0-successor), and $u1$ its right successor (or

1-successor). A *branch* π is an infinite sequence $u_0, u_1, u_2 \ldots$ of successive nodes starting in the root, i.e., $u_0 = \varepsilon$ and $u_{i+1} = u_i 0$ or $u_{i+1} = u_i 1$ for all $i \geq 0$.

An (infinite binary) *tree* labeled by a finite alphabet Σ is a mapping $t : \{0,1\}^* \to \Sigma$. We denote by $\mathcal{T}_\Sigma^\omega$ the set of all trees labeled by Σ. In a tree t, a branch π induces an infinite sequence of labels in Σ^ω. We sometimes identify a branch with this infinite word. It should always be clear from the context to which meaning of a branch we are referring to.

In connection with logic, the labels of the infinite tree are used to represent interpretations of set variables. This motivates the following definitions. For a set $U \subseteq \{0,1\}^*$, we write $t[U] \in \mathcal{T}_{\{0,1\}}^\omega$ for the *characteristic tree* of U, i.e., the tree which labels all nodes in U with 1 and all the other nodes with 0. This notation is extended to the case of several sets. The characteristic tree of $U_1, \ldots, U_n \subseteq \{0,1\}^*$ is the tree labeled by $\{0,1\}^n$ written $t[U_1, \ldots, U_n]$ and defined for all $u \in \{0,1\}^*$ by $t[U_1, \ldots, U_n](u) := (b_1, \ldots, b_n)$ where for all $i \in [1,n]$, $b_i = 1$ if $u \in U_i$ and $b_i = 0$ otherwise. For a singleton set $U = \{u\}$ we simplify notation and write $t[u]$ instead of $t[\{u\}]$.

We now turn to the definition of automata for infinite trees. A *parity tree automaton* (PTA) on Σ-labeled trees is a tuple $\mathcal{A} = (Q, \Sigma, q^{\mathrm{i}}, \Delta, c)$ with a finite set Q of states, initial state $q^{\mathrm{i}} \in Q$, transition relation $\Delta \subseteq Q \times \Sigma \times Q \times Q$, and a priority function $c : Q \to \mathbb{N}$. A *run* of $\mathcal{A}$ on a tree $t \in \mathcal{T}_\Sigma^\omega$ from a state $q \in Q$ is a tree $\rho \in \mathcal{T}_Q^\omega$ such that $\rho(\varepsilon) = q$, and for each $u \in \{0,1\}^*$ we have $(\rho(u), t(u), \rho(u0), \rho(u1)) \in \Delta$. We say that ρ is *accepting* if the parity condition is satisfied on each branch of the run, i.e., on each branch the maximal priority appearing infinitely often is even. If we only speak of a run of $\mathcal{A}$ without specifying the state at the root, we implicitly refer to a run from q^{i}.

It is also possible to allow automata with a set of initial states. In this case we just require a run to start with some state from this set at the root. One can easily transform an automaton with a set of initial states into one with only a single initial state.

A tree t is accepted by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on t. The *language* accepted by $\mathcal{A}$ is the set of all accepted trees:

$$T(\mathcal{A}) = \{t \in \mathcal{T}_\Sigma^\omega \mid t \text{ accepted by } \mathcal{A}\}.$$

A tree language is called *regular* if it is accepted by some PTA.

For two trees t, t' we say that they are $\mathcal{A}$-*equivalent*, written as $t \equiv_{\mathcal{A}} t'$, if for each state q of $\mathcal{A}$ there is an accepting run from q on t iff there is an accepting run from q on t'. Intuitively, this means that $\mathcal{A}$ cannot distinguish the two trees.

As already mentioned, there is a tight connection between automata on infinite trees and games. For example, we can express the question whether the language $T(\mathcal{A})$ of a PTA is empty in terms of a game. We present this construction here because it is used in Section 4.3.

The language $T(\mathcal{A})$ is non-empty if the following holds:

$\exists t \in \mathcal{T}_\Sigma^\omega \exists$ run ρ on t $\forall$ branches π : $\rho(\pi)$ satisfies parity condition.

This quantifier alternation can be captured by a game in which one player chooses t and ρ, and the other player chooses π. Played like this, the game consists of only two moves (one for each player), each of which provides an uncountable number of choices. To avoid this, we let the players build up the objects incrementally, instead of choosing them in one shot. The idea is as follows:

- Eva chooses the symbol that should be put at the root of the tree and also chooses the first transition of ρ (compatible with the initial state and the chosen label). To complete t and ρ, she has to complete the left subtree and the right subtree. In case the language $T(\mathcal{A})$ is empty, she will not be able to complete one of the two subtrees. But once she has made her choice for the first transition, the direction in which she will not be able to complete the subtree is fixed. Hence Adam can make his first choice for the direction on π.
- Once Eva and Adam have made their choices as just explained, the game continues in the same way from the new state determined by the transition chosen by Eva and the direction taken by Adam. This is repeated to infinity.
- The result of the game is an infinite sequence of states (and transitions), basically corresponding to $\rho(\pi)$ in the above statement. Eva wins if this sequence satisfies the parity condition, otherwise Adam wins.

We now describe this parity game $\mathcal{G}_\mathcal{A}$ for PTA emptiness more formally. Note that the choice of the labels for t is done implicitly by choosing the transitions for ρ.

- The positions of Eva are the states Q of the automaton.
- The positions of Adam are the transitions Δ of the automaton.
- The initial position of the game is q^{i}.
- From a state q Eva can move to all transitions of the form (q, a, q_0, q_1).
- From a transition (q, a, q_0, q_1) Adam can move to either q_0 or q_1.
- As priority function we simply take the priority function of $\mathcal{A}$ and extend it to transitions by setting $c(q, a, q_0, q_1) = c(q)$.

We can now relate the existence of winning strategies in $\mathcal{G}_\mathcal{A}$ to the emptiness of $T(\mathcal{A})$ (cf. [128, 118]).

PROPOSITION 1.8 *Eva has a winning strategy in the emptiness game $\mathcal{G}_\mathcal{A}$ iff $T(\mathcal{A}) \neq \emptyset$.*

Proof. If $T(\mathcal{A}) \neq \emptyset$, then we can choose a tree t and an accepting run ρ of $\mathcal{A}$ on t. In a play prefix, the choices of Adam determine a position in the tree. The strategy of Eva is to choose the transition of ρ at this position. In this way she ensures that a branch of ρ is played, and as ρ is accepting she wins.

If Eva has a winning strategy in $\mathcal{G}_\mathcal{A}$, then we construct a run as follows. Each node u in $\{0,1\}^*$ determines a sequence of choices for Adam in $\mathcal{G}_\mathcal{A}$. We play the game according to these choices and according to Eva's strategy. This determines a transition, which we use at u. The rules of the game ensure that this indeed results in a run, and as the strategy is winning for Eva, the run is accepting. □

The parity game $\mathcal{G}_\mathcal{A}$ is polynomial in the size of $\mathcal{A}$, hence we can apply Theorem 1.4.

THEOREM 1.9 *The emptiness problem for parity tree automata is in NP∩co-NP.*

For later application (in Chapter 4) we also consider the emptiness game for the intersection of two languages given by PTAs. The game is similar to $\mathcal{G}_\mathcal{A}$ but now it is played on pairs of states and transitions, one from each automaton. To prove that there is a tree in the intersection of the two languages, Eva has to construct two runs in parallel using the same input letters when choosing the transitions of the two automata.

Let $\mathcal{A} = (Q_\mathcal{A}, \Sigma, q^{\mathsf{i}}_\mathcal{A}, \Delta_\mathcal{A}, c_\mathcal{A})$ and $\mathcal{B} = (Q_\mathcal{B}, \Sigma, q^{\mathsf{i}}_\mathcal{B}, \Delta_\mathcal{B}, c_\mathcal{B})$ be PTAs. The *intersection emptiness game* $\mathcal{G}_{\mathcal{A}\times\mathcal{B}}$ for $\mathcal{A}$ and $\mathcal{B}$ is defined as follows.

1. The vertex set of Eva is $Q_\mathcal{A} \times Q_\mathcal{B}$, the vertex set of Adam is $\Delta_\mathcal{A} \times \Delta_B$.
2. The game starts in $(q^{\mathsf{i}}_\mathcal{A}, q^{\mathsf{i}}_\mathcal{B})$.
3. From a position $(p,q) \in Q_\mathcal{A} \times Q_\mathcal{B}$, Eva picks two transitions from the respective states, both using the same input letter, i.e., she picks $(p, a, p_0, p_1) \in \Delta_\mathcal{A}$ and $(q, a, q_0, q_1) \in \Delta_\mathcal{B}$ for some $a \in \Sigma$. The game is now in position $((p, a, p_0, p_1), (q, a, q_0, q_1))$.
4. Adam chooses a direction $i \in \{0,1\}$ and the game moves to (p_i, q_i).
5. The coloring function is given by $col(p,q) = (c_\mathcal{A}(p), c_\mathcal{B}(q))$, and for transitions $\tau = (p, a, p_0, p1) \in \Delta_\mathcal{A}$ and $\tau' = (q, a, q_0, q_1)$ we also set $col(\tau, \tau') = (c_\mathcal{A}(p), c_\mathcal{B}(q))$.
6. The winning condition is the conjunction of the two parity conditions of $\mathcal{A}$ and $\mathcal{B}$ (see the end of Section 1.2).

The following result can be shown in the same way as Proposition 1.8.

PROPOSITION 1.10 *Eva has a winning strategy in $\mathcal{G}_{\mathcal{A}\times\mathcal{B}}$ iff $T(\mathcal{A})\cap T(\mathcal{B}) \neq \emptyset$.*

The winning condition of $\mathcal{G}_{\mathcal{A}\times\mathcal{B}}$ is a conjunction of two parity conditions. Using Remark 1.7 and Corollary 1.6 we obtain

COROLLARY 1.11 *If $T(\mathcal{A}) \cap T(\mathcal{B}) = \emptyset$, then Adam has a positional winning strategy in $\mathcal{G}_{\mathcal{A}\times\mathcal{B}}$.*

The emptiness game and the result on the positional determinacy of parity games can be used to show the existence of a certain kind of simple trees in each non-empty regular tree language. These trees are called regular because they can be generated by a deterministic finite automaton in the following sense: A tree $t \in \mathcal{T}_\Sigma^\omega$ is called *regular* if there is a DFA $(S, \{0,1\}, s^{\mathrm{i}}, \delta, \lambda)$ with an output function $\lambda : Q \to \Sigma$ instead of set of final states, such that for each $u \in \{0,1\}^*$ the label $t(u)$ of u in t is the same as the output $\lambda(\delta(u))$ computed by the DFA. We can also view the DFA from the above definition as a finite graph of out-degree two whose vertices are labeled by symbols from Σ. The tree it generates is simply the unfolding of this graph starting from the initial state s^{i}.

It is not difficult to see that regular trees are exactly those that have only finitely many non-isomorphic subtrees: A tree with this property can be generated by a DFA whose states correspond to the different subtrees it contains. Vice versa, a tree that is generated by a finite automaton can only contain as many non-isomorphic subtrees as the DFA has states.

From a positional strategy for Eva in the emptiness game $\mathcal{G}_\mathcal{A}$ we can construct a regular tree in the language of $\mathcal{A}$.

THEOREM 1.12 ([102]) *Every non-empty regular tree language contains a regular tree.*

Proof. Let T be a non-empty regular language accepted by some PTA $\mathcal{A} = (Q, \Sigma, q^{\mathrm{i}}, \Delta, c)$. Then Eva has a winning strategy in the emptiness game $\mathcal{G}_\mathcal{A}$. By Theorem 1.3 this strategy can be chosen positionally. A positional strategy in $\mathcal{G}_\mathcal{A}$ can be seen as a mapping $\sigma_\circ : Q \to \Delta$ and can directly be used to construct a DFA generating a regular tree:

- The state set of the DFA is Q.
- Let $q \in Q$ and $\sigma_\circ(q) = (q, a, q_0, q_1)$. Then we specify
  - the output function of the DFA by $\lambda(q) = a$,
  - the transition function of the DFA by $\delta(q, 0) = q_0$ and $\delta(q, 1) = q_1$.

Since $\sigma_\circ$ is a winning strategy for Eva, the regular tree defined like this is indeed in $T(\mathcal{A})$. □

We have already defined the characteristic tree $t[U]$ of a set $U \subseteq \{0,1\}^*$. The definition of regular trees by DFAs directly implies that $t[U]$ is a regular tree iff U is a regular set of words. The automaton accepting U also generates $t[U]$ if we define the output of final states to be 1, and 0 for the non-final states. This easily generalizes to tuples of sets:

REMARK 1.13 For $U_1, \ldots, U_n \subseteq \{0,1\}^*$ the tree $t[U_1, \ldots, U_n]$ is regular iff all the sets $U_1, \ldots, U_n$ are regular sets of words.

1.4 Complementation of parity tree automata

In this section we give a complementation construction for parity tree automata. The construction follows the one given in [118] but we present the details here because in Chapter 4 we rely on them for the construction of guidable automata.

Given a PTA $\mathcal{A} = (Q, \Sigma, q^{\mathsf{i}}, \Delta, c)$, the goal is to construct an automaton that can check for a tree t that there is *no* successful run of $\mathcal{A}$ on t. For this purpose we first describe a membership game for $\mathcal{A}$ and t in which Adam has a winning strategy iff $t \notin T(\mathcal{A})$. The strategies are such that they can be represented by annotating the input tree t. We then construct a tree automaton accepting those trees that are annotated with a winning strategy of Adam (and hence are not in $T(\mathcal{A})$). The desired automaton for the complement is obtained by simply removing the annotations from the labels.

We start by describing the membership game. It is similar to the emptiness game with the difference that the input symbols are not chosen by Eva but are fixed by the tree t. For this reason the arena of the membership game is infinite (because t is).

If we are given a tree t and a PTA $\mathcal{A}$, then t is accepted by $\mathcal{A}$ if

$$\exists \text{ run } \rho \; \forall \text{ branch } \pi \; : \; \rho(\pi) \text{ satisfies the parity condition.}$$

This can be captured by a game in which Eva tries to prove the existence of such a ρ, and Adam tries to refute it by building a path π on which the parity condition is not satisfied. To implement this idea, Eva starts at the root in the initial state and chooses a transition that is compatible with the label of the root. Based on the choice of Eva, Adam can now decide in which direction he wants to build the path π, i.e., he decides to move to the left or to the right successor. This determines the next node and the state from which to continue. This whole procedure essentially results in an infinite sequence of states corresponding to the labeling of a branch through a run. Eva wins if this sequence satisfies the parity condition, otherwise Adam wins.

The formal definition of the *membership game* $\mathcal{G}_{\mathcal{A},t}$ is as follows:

- The set of vertices of Eva is $\{0,1\}^* \times Q$ (pairs of tree nodes and states).
- The set of vertices of Adam is $\{0,1\}^* \times \Delta$ (pairs of tree nodes and transitions).
- The initial vertex is $(\varepsilon, q^{\mathrm{i}})$
- From a position (u, q) Eva can move to all positions of the form $(u, (q, t(u), q_0, q_1))$ with $(q, t(u), q_0, q_1) \in \Delta$.
- From a position $(u, (q, t(u), q_0, q_1))$ Adam can move to $(u0, q_0)$ or to $(u1, q_1)$.
- The priority function is adapted from the one of $\mathcal{A}$ by $c_{\mathcal{A},t}(u, q) = c(q)$, and $c_{\mathcal{A},t}(u, (q, t(u), q_0, q_1)) = c(q)$.

It is not difficult to see that Eva indeed has a winning strategy in $\mathcal{G}_{\mathcal{A},t}$ iff t is accepted by $\mathcal{A}$. By determinacy of parity games we obtain that Adam has a winning strategy iff $t \notin T(\mathcal{A})$, and furthermore this strategy can be chosen positionally (Theorem 1.3).

LEMMA 1.14 *Adam has a positional winning strategy in $\mathcal{G}_{\mathcal{A},t}$ iff $t \notin T(\mathcal{A})$.*

We now explain how to represent positional strategies of Adam by annotating the tree t. A positional winning strategy of Adam can be described by a function $\sigma_\Box : \{0,1\}^* \times \Delta \to \{0,1\}$, or equivalently by a function $\sigma_\Box : \{0,1\}^* \to (\Delta \to \{0,1\})$ assigning to each node of the tree a function from transitions to directions. There are only finitely many such functions and therefore we can annotate each node u of t with the additional information $\sigma_\Box(u)$.

A *tree representation of a strategy* of Adam is a tree $s \in \mathcal{T}^{\omega}_{\Sigma\times(\Delta\to\{0,1\})}$. Note that t is not fixed anymore. Such a tree s represents a strategy in the game $\mathcal{G}_{\mathcal{A},t}$ for the tree t obtained by projecting s to the Σ-component of the labels. The central lemma for the complementation construction is that we can construct a PTA accepting exactly the representations of winning strategies of Adam. For later use in Chapter 4 it is important that this PTA is even *top-down deterministic*, i.e., for each state q and each input letter a there is at most one transition (q, a, q_0, q_1).

LEMMA 1.15 *There is a top-down deterministic PTA accepting the tree representations of winning strategies of Adam.*

Proof. Let s be a tree representation of a strategy of Adam. Let t be the corresponding tree in $\mathcal{T}^{\omega}_{\Sigma}$ obtained by projection to the Σ-component, and let $\sigma_\Box$ be the strategy of Adam in $\mathcal{G}_{\mathcal{A},t}$ represented by s.

The PTA we want to construct has to verify that all plays played according to $\sigma_\Box$ are winning for Adam. Each play naturally induces a branch

through the tree. Hence, verifying whether all plays according to $\sigma_\Box$ are winning for Adam can be done by checking on each branch whether all plays that induce this branch are winning for Adam. This is a task that can be done by a deterministic parity word automaton. We then run this word automaton on all branches of the tree, thus obtaining the desired PTA. This idea is formalized in the following.

Each branch in s corresponds to an infinite sequence over $(\Sigma \times (\Delta \to \{0,1\}) \times \{0,1\})$, where the last $\{0,1\}$-component indicates the direction the branch takes in each step. We say that such a sequence

$$(a_0, f_0, i_0)(a_1, f_1, i_1) \cdots \in (\Sigma \times (\Delta \to \{0,1\}) \times \{0,1\})^\omega$$

is $\mathcal{A}$-accepting if there is a transition sequence $\tau_0\tau_1 \cdots \in \Delta^\omega$ such that

- for each j the transition τ_j is of the form $(q_j, a_j, q_j^{(0)}, q_j^{(1)})$ with $f_j(\tau_j) = i_j$ and $q_j^{(i_j)} = q_{j+1}$,
- $q_0 = q^{\mathsf{i}}$, and
- the acceptance condition of $\mathcal{A}$ is satisfied by $q_0 q_1 \cdots$.

The first condition states that the direction i_j corresponds to the direction taken by the strategy as described in f_j when applied to the transition τ_j, and that the next transition starts with the correct state.

Assume that s contains a branch corresponding to an $\mathcal{A}$-accepting sequence. Then Eva can play the witnessing transition sequence in $\mathcal{G}_{\mathcal{A},t}$. If Adam applies $\sigma_\Box$, then the resulting play corresponds to the $\mathcal{A}$-accepting sequence, and hence Eva wins. Therefore, $\sigma_\Box$ is not a winning strategy.

Vice versa, assume that $\sigma_\Box$ is not winning for Adam. Then Eva can play a sequence of transitions such that the resulting play according to $\sigma_\Box$ is winning for her. In the tree representation s of $\sigma_\Box$ this corresponds to an $\mathcal{A}$-accepting sequence.

We obtain that s is a tree representation of a winning strategy for Adam iff s does not contain a branch corresponding to an $\mathcal{A}$-accepting sequence. We have to construct an automaton checking this property.

It is easy to see that the set of all $\mathcal{A}$-accepting sequences over $(\Sigma \times (\Delta \to \{0,1\}) \times \{0,1\})$ is a regular language of infinite words (a non-deterministic Büchi automaton can guess the transition sequence of $\mathcal{A}$ and verify the local properties). Hence, the set of all sequences that are not $\mathcal{A}$-accepting is also a regular language, and there is a deterministic parity word automaton for this language. From this word automaton one easily constructs a top-down deterministic PTA $\mathcal{A}' = (Q', \Sigma \times (\Delta \to \{0,1\}), q_0', \Delta', c')$ accepting all $(\Sigma \times (\Delta \to \{0,1\}))$-trees in which all branches correspond to non-$\mathcal{A}$-accepting sequences. □

The complementation result is a direct consequence of this lemma.

THEOREM 1.16 ([102]) *For each PTA one can construct a PTA accepting the complement language.*

Proof. Given $\mathcal{A}$ we apply Lemma 1.15 and obtain a PTA $\mathcal{A}'$ accepting the tree representations of winning strategies of Adam. Omitting the $(\Delta \to \{0,1\})$ part from the labels in the transitions yields a (nondeterministic) PTA that accepts all trees $t \in \mathcal{T}_\Sigma^\omega$ on which Adam has a winning strategy in $\mathcal{G}_{\mathcal{A},t}$. According to Lemma 1.14 this is exactly the complement of the language of $\mathcal{A}$. □

The closure under complement of regular languages of infinite trees is a strong result. We have seen that the proof is based on the determinization theorem for automata on infinite words, and on the positional determinacy of parity games. Since the proof is constructive, we can also conclude that many algorithmic questions for PTAs are decidable as, e.g., the equivalence and inclusion problem, as well as the universality problem, i.e., the question whether a given PTA accepts all trees.

Chapter 2

Structures and logic

In the introduction we have mentioned the intimate relationship between logic and automata. In this chapter we introduce the basic terminology on logic and structures, and in Section 2.2 we formalize the connections to automata theory. One of the most central and important theorems from this area states that the monadic second-order theory of the infinite binary tree is decidable. Starting from this result, many other decidability results have been derived. In Section 2.3 we present the methods of interpretation and unfolding that allow to construct structures with decidable theories from given ones. The combination of these two methods has lead to one of the richest classes of structures with decidable monadic second-order theory, the so called Caucal hierarchy, which is also presented in Section 2.3. In Section 2.4 we consider finite set interpretations, a generalization of interpretations that allows to construct structures with decidable first-order theory out of structures with decidable monadic second-order theory. The definitions from this section are the basis for the material presented in Chapter 6. In Section 2.5 we present the classes of automatic and tree automatic structures and their relationship to finite set interpretations.

2.1 Basic terminology

A *signature* is a ranked set of symbols τ, where for all $R \in \tau$, $|R|$ denotes the arity (which is ≥ 1) of the symbol R. A *relational structure* $\mathcal{S}$ over τ is given by a tuple $(D, (R^{\mathcal{S}})_{R \in \tau})$ where D is the *domain* (or the *universe*) of $\mathcal{S}$ and where for all $R \in \tau$, $R^{\mathcal{S}}$ (which is also called the interpretation of R in $\mathcal{S}$) is a subset of $D^{|R|}$. When $\mathcal{S}$ is clear from the context, we simply write R instead of $R^{\mathcal{S}}$.

We use $\cong$ to denote that two structures are isomorphic. Usually, we do not distinguish isomorphic structures but sometimes we refer to representations of structures over a specific domain, and in these cases we use $\cong$ instead of $=$.

To every tree t labeled by $\Sigma = \{a_1, \ldots, a_n\}$, we associate a canonical structure over the signature $\{E_0, E_1, P_{a_1}, \ldots, P_{a_n}\}$ where E_0 and E_1 are binary symbols and the P_{a_i} are *predicates*, i.e., unary symbols. The universe of this structure is $\{0,1\}^*$. The symbols E_0 and E_1 are respectively interpreted as $\{(w, w0) \mid w \in \{0,1\}^*\}$ and $\{(w, w1) \mid w \in \{0,1\}^*\}$. Finally for all $i \in [1, n]$, P_{a_i} is interpreted as $\{u \in \Sigma^* \mid t(u) = a_i\}$. In the following, we will not distinguish between a tree and its canonical relational structure.

The unlabeled binary tree, i.e., the structure $(\{0,1\}^*, \{E_0, E_1\})$ with the above interpretation of E_0 and E_1 is denoted by $\mathfrak{t}_2$. The structure with $\mathbb{N}$ as domain and the successor relation on $\mathbb{N}$ can be viewed as a tree with unary branching. Hence we denote it by $\mathfrak{t}_1$.

A *graph* is a structure over a signature only containing unary predicates (node labels) and binary relations (labeled edges). We usually write such signatures as $\{(E_a)_{a\in\Gamma}, (P_b)_{b\in\Sigma}\}$ for an alphabet Γ of edge labels, and an alphabet Σ of node labels. For example, the infinite binary tree $\mathfrak{t}_2$ is a graph with edge labels $\{0,1\}$ and without node labels. As usual, we call the elements of the universe of a graph vertices. In Section 1.2 we have introduced arenas, which are specific types of graphs. We can view an arena $G = (V_\circ, V_\Box, E, col)$ as a relational structure in the above sense, where $V_\circ$ and $V_\Box$ are unary predicates, and the coloring function $col : V \to Cols$ is also represented by unary predicates $(P_i)_{i\in Cols}$. For example, the coloring function of the arena from Figure 1.3 on page 18 is represented by the predicates $P_0 = \{v_5\}$, $P_1 = \{v_7\}$, $P_2 = \{v_1, v_2, v_4\}$, $P_3 = \{v_0\}$, and $P_4 = \{v_6\}$, and the partition by $V_\circ = \{v_0, v_5, v_7\}$ and $V_\Box = \{v_1, v_2, v_3, v_4, v_6\}$.

In the definition from Section 1.2 arenas do not have edge labels, i.e., there is only one binary relation E. In Section 3.2 we use game arenas with edge labels when considering logical definability of winning strategies.

We are mainly interested in *monadic second-order logic* (MSO) over relational structures with the standard syntax and semantics (see e.g. [42] for a detailed presentation). MSO-formulas use first-order variables, which are interpreted by elements of the structure, and monadic second-order variables, which are interpreted as sets of elements. First order variables are usually denoted by small letters (e.g. x, y), and monadic second-order variables are denoted by capital letters (e.g. X, Y). First-order logic (FO) is the fragment of MSO that does not use set variables.

Atomic MSO-formulas are of the form

- $R(x_1, \ldots, x_{|R|})$ for a relation symbol from the signature and first-order variables $x_1, \ldots, x_{|R|}$, or
- $x = y$, $X = Y$, or $x \in X$ for first-order variables x, y, and monadic second-order variables X, Y

with the obvious semantics. Complex formulas are built as usual from atomic ones by the use of boolean connectives, and quantification.

We write $\phi(X_1, \ldots, X_n, y_1, \ldots, y_m)$ to indicate that the free variables of the formula ϕ are among $X_1, \ldots, X_n$ (monadic second-order) and $y_1, \ldots, y_m$ (first-order) respectively. A formula without free variables is called a *sentence*.

For a relational structure $\mathcal{S}$ and a sentence ϕ, we write $\mathcal{S} \models \phi$ if $\mathcal{S}$ *satisfies* the formula ϕ. The *MSO-theory* of $\mathcal{S}$ is the set of sentences satisfied by $\mathcal{S}$. For every formula $\phi(X_1, \ldots, X_n, y_1, \ldots, y_m)$, all subsets $U_1, \ldots, U_n$ of the universe of $\mathcal{S}$ and all elements $v_1, \ldots, v_m$ of the universe of $\mathcal{S}$, we write $\mathcal{S} \models \phi[U_1, \ldots, U_n, v_1, \ldots, v_m]$ to express that ϕ holds in $\mathcal{S}$ when X_i is interpreted as U_i for all $i \in [1, n]$ and y_j is interpreted as v_j for all $j \in [1, m]$.

A variant of MSO only differs in the way set quantifications are interpreted: Formulas of *weak monadic second-order logic* (WMSO) have the same syntax as MSO-formulas but the set quantifiers only range over finite sets. Free variables of a WMSO-formula are still interpreted by arbitrary (finite and infinite) sets.

Given a structure $\mathcal{S}$ we call a relation $R \subseteq D^n$ for some $n \geq 1$ *MSO-definable* if there is an MSO-formula $\phi(x_1, \ldots, x_n)$ with n free element variables such that $(u_1, \ldots, u_n) \in R$ iff $\mathcal{S} \models \phi[u_1, \ldots, u_n]$. Correspondingly, we say that a relation is FO- or WMSO-definable.

A single element u of a structure is called MSO-definable if there is a formula $\phi(x)$ such that u is the only element with $\mathcal{S} \models \phi[u]$.

2.2 Automata and MSO

We are particularly interested in MSO over the infinite binary tree $\mathfrak{t}_2$, which we refer to as MSO[$\mathfrak{t}_2$]. The MSO-theory of the infinite binary tree $\mathfrak{t}_2$ is also referred to as S2S, the second-order theory of two successors [102]. Correspondingly, we refer to the MSO-theory of $\mathfrak{t}_1$, the natural numbers with successor, as S1S [17].

An MSO[$\mathfrak{t}_2$]-formula $\phi(X_1, \ldots, X_n)$ with n free variables defines a tree language $T(\phi) \subseteq \mathcal{T}^{\omega}_{\{0,1\}^n}$ as follows:

$$T(\phi) = \{t[U_1, \ldots, U_n] \mid \mathfrak{t}_2 \models \phi[U_1, \ldots, U_n]\}.$$

Using this correspondence, we also write $t[U_1, \ldots, U_n] \models \phi$ if $t[U_1, \ldots, U_n] \in T(\phi)$. A theorem of Rabin states that the languages definable by MSO[$\mathfrak{t}_2$]-formulas are precisely those that can be accepted by PTAs [102]. For MSO over $\mathfrak{t}_1$ and Büchi automata this has been shown by Büchi [17].

We state here the difficult direction, the proof of which is based on the closure properties of PTAs.

THEOREM 2.1 ([102]) *For every MSO[$\mathfrak{t}_2$]-formula $\phi(X_1, \ldots, X_n)$ there is a parity tree automaton $\mathcal{A}_\phi$ such that $T(\mathcal{A}_\phi) = T(\phi)$.*

The connection between automata and logic allows to develop decision procedures for the logic based on algorithms for automata. The satisfiability problem for MSO[$\mathfrak{t}_2$], i.e., the question whether for a given MSO[$\mathfrak{t}_2$] formula there is an interpretation of the free variables such that the formula is satisfied in $\mathfrak{t}_2$, is reduced to the emptiness problem for PTAs. The latter is decidable according to Theorem 1.9.

THEOREM 2.2 *The satisfiability problem for MSO[$\mathfrak{t}_2$] is decidable.*

Another consequence of Theorem 2.1 is obtained by combining it with the results on regular trees from Theorem 1.12 and Remark 1.13.

COROLLARY 2.3 *Let $\phi(X_1, \ldots, X_n)$ be a satisfiable MSO[$\mathfrak{t}_2$]-formula. There are regular sets $U_1, \ldots, U_n$ such that $t[U_1, \ldots, U_n] \models \phi$.*

Above we have introduced the notion of MSO-definability. We conclude from Corollary 2.3 that on $\mathfrak{t}_2$ each MSO-definable set is regular: Starting from a formula $\phi(x)$ defining a set, we construct the formula $\phi'(X) = x \in X \leftrightarrow \phi(x)$. Then there is exactly one set satisfying ϕ' and this set is regular according to Corollary 2.3. Vice versa, each regular set can be defined in MSO by describing an accepting run of the DFA defining the set. We make use of this in Chapter 3.

PROPOSITION 2.4 *A set $U \subseteq \{0,1\}^*$ is definable in MSO[$\mathfrak{t}_2$] iff it is regular.*

As a consequence we can add regular predicates to $\mathfrak{t}_2$ without affecting the decidability result for MSO. An MSO-formula having access to regular predicates $P_1, \ldots, P_n$ can be turned in a standard MSO-formula over $\mathfrak{t}_2$ by using formulas $\phi_{P_1}, \ldots, \phi_{P_n}$ defining these predicates. In the next section we introduce interpretations as a generalization of this concept.

2.3 Interpretations and unfoldings

We have seen in the previous section that the MSO-theory of $\mathfrak{t}_2$ is decidable. The proof is based on automata theoretic concepts and cannot be adapted if we are interested in more general structures than trees. In this section we consider operations for transforming structures that allow to transfer the decidability result from Theorem 2.2 to a large class of structures.

Let τ and $\hat{\tau}$ be two signatures. An *MSO-interpretation* from τ-structures to $\hat{\tau}$-structures is given as a list $\mathcal{I} = \langle \phi_{dom}, (\phi_{\hat{R}})_{\hat{R} \in \hat{\tau}} \rangle$ of MSO-formulas over the signature τ. The formula ϕ_{dom} has one free first-order variable and is called the *domain formula*. For each $\hat{R} \in \hat{\tau}$ the formula $\phi_{\hat{R}}$ has $|\hat{R}|$ many free first-order variables.

Applied to a τ-structure $\mathcal{S} = (D, (R^{\mathcal{S}})_{R \in \tau})$, the interpretation defines a $\hat{\tau}$-structure $\mathcal{I}(\mathcal{S}) = (\hat{D}, (\hat{R}^{\mathcal{I}(\mathcal{S})})_{\hat{R} \in \hat{\tau}})$, where the domain formula defines the

domain of the new structure (all elements of $\mathcal{S}$ satisfying ϕ_{dom}), and the formulas $\phi_{\hat{R}}$ define the relations of $\hat{\mathcal{S}}$:

- $\hat{D} = \{u \in D \mid \mathcal{S} \models \phi_{dom}[u]\}$,
- $\hat{R}^{\mathcal{I}(\mathcal{S})} = \{(u_1, \ldots, u_{|\hat{R}|}) \in \hat{D}^{|\hat{R}|} \mid \mathcal{S} \models \phi_{\hat{R}}[u_1, \ldots, u_n]\}$.

If we use FO instead of MSO, then we speak of *FO-interpretations*, and similarly of *WMSO-interpretations*.

We illustrate this operation by an example that is shown in Figure 2.1. It transforms $\mathfrak{t}_2$ over the signature $\tau = \{E_0, E_1\}$ into the graph depicted at the bottom of Figure 2.1 over the signature $\hat{\tau} = \{E_a, E_b, E_b, E_d\}$. The gray area indicates the part that is selected by the domain formula. The dashed edges are the ones defined by the interpretation. The formulas of the interpretation $\mathcal{I}$ are

$$\begin{array}{lcl}
\phi_{dom}(x) & = & \neg\exists y, z : y \sqsubset x \wedge E_0(z, y) \\
\phi_{E_a}(x, y) & = & E_1(x, y) \\
\phi_{E_b}(x, y) & = & E_0(x, y) \\
\phi_{E_c}(x, y) & = & \phi_{E_b}(x, y) \\
\phi_{E_d}(x, y) & = & \exists x', y' : E_0(x', x) \wedge E_0(y', y) \wedge E_1(y', x')
\end{array}$$

Now assume that we are given an MSO-formula over $\mathcal{I}(\mathcal{S})$ and we want to know whether it is satisfiable. We can replace each atomic formula $\hat{R}(x_1, \ldots, x_{|\hat{R}|})$ by its definition $\phi_{\hat{R}}(x_1, \ldots, x_n)$, and restrict the range of all variables to the set $\hat{D}$ defined by ϕ_{dom}. In this way we obtain an MSO-formula over $\mathcal{S}$ that is satisfiable iff the original formula is.

Proposition 2.5 *If $\mathcal{I}$ is an MSO-interpretation and if the MSO-theory of $\mathcal{S}$ is decidable, then the MSO-theory of $\mathcal{I}(\mathcal{S})$ is decidable. The same holds for WMSO and FO instead of MSO.*

Since the MSO-theory of $\mathfrak{t}_2$ is decidable (Theorem 2.2), we can, e.g., conclude that also the MSO-theory of the graph at the bottom of Figure 2.1 is decidable.

The second operation we consider is the unfolding of graphs. Let Γ be a set of edge labels and let $G = (D, (E_a)_{a\in\Gamma}, (P_b)_{b\in\Sigma})$ be a graph over Γ. The *unfolding of G from a vertex $v_0 \in D$* is a graph whose vertices are paths in G starting from v_0. The edges correspond to path extensions by one edge in G, and the node labels are inherited from the last element of the path. Formally, we let $\mathsf{Unf}(G, v_0) = (D^{\mathsf{Unf}}, (E_a^{\mathsf{Unf}})_{a\in\Gamma}, (P_b^{\mathsf{Unf}})_{b\in\Sigma})$ be defined by

$$\begin{array}{lcl}
D^{\mathsf{Unf}} & = & \{v_0 a_1 v_1 \cdots a_n v_n \in v_0(\Gamma D)^* \mid (v_{i-1}, v_i) \in E_{a_i} \text{ for all } 1 \leq i \leq n\}, \\
E_a^{\mathsf{Unf}} & = & \{(w, wav) \mid w = v_0 a_1 v_1 \cdots a_n v_n \in D^{\mathsf{Unf}} \text{ and } (v_n, v) \in E_a\}. \\
P_b^{\mathsf{Unf}} & = & \{w \in D^{\mathsf{Unf}} \mid w = v_0 a_1 v_1 \cdots a_n v_n \text{ and } v_n \in P_b\}.
\end{array}$$

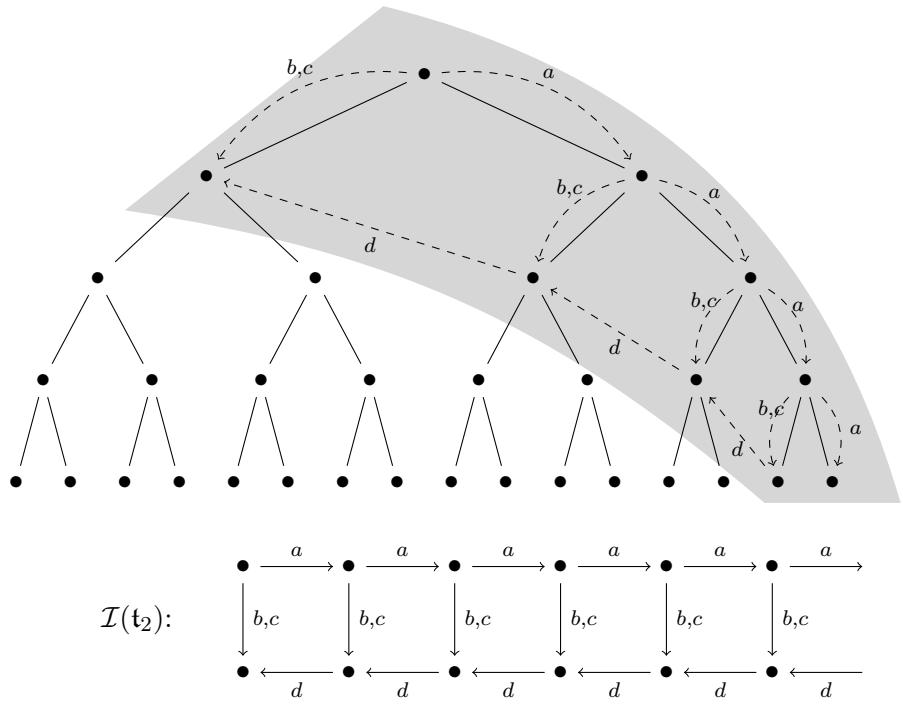


Figure 2.1: An interpretation $\mathcal{I}$ applied to $\mathfrak{t}_2$

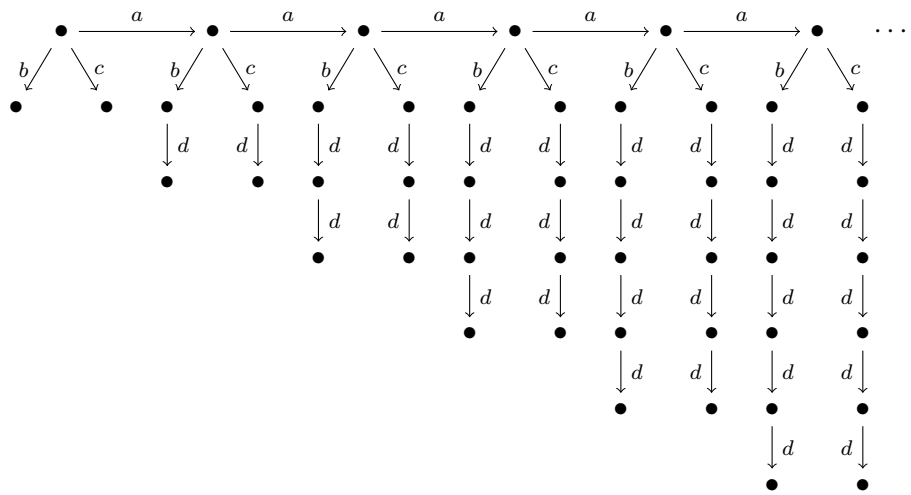


Figure 2.2: Unfolding of the graph from Figure 2.1

For example, by unfolding the graph at the bottom of Figure 2.1 from the vertex at the upper left corner, we obtain the graph shown in Figure 2.2. In [38] it is shown that the unfolding operation is compatible with MSO.

THEOREM 2.6 ([38]) *Let G be a graph with a decidable MSO-theory and let v be an MSO-definable vertex of G. Then the MSO-theory of $\mathsf{Unf}(G, v)$ is decidable.*

A more general operation that can be applied to all structures and not only graphs is the iteration. The unfolding of a graph can be reconstructed from its iteration by an MSO-interpretation. For our purpose it is enough to only consider the unfolding operation. Details on the iteration operation and its properties can be found in [8].

Using these two operations we can build up classes of graphs with decidable MSO-theory. We start with the class $\mathfrak{C}_0$ of all finite graphs. Obviously, each finite graph has a decidable MSO-theory because one can check by an exhaustive search whether an MSO-formula is satisfied. Given class $\mathfrak{C}_i$ we take as class $\mathfrak{C}_{i+1}$ all graphs that can be obtained from a graph in $\mathfrak{C}_i$ by an unfolding operation followed by an MSO-interpretation. For example, the graph at the bottom of Figure 2.1 is in $\mathfrak{C}_1$ because it is obtained by an MSO-interpretation from $\mathfrak{t}_2$, which itself can be obtained by unfolding the one-element structure with self-loops labeled 0 and 1.

Accordingly, the graph in Figure 2.2 is in $\mathfrak{C}_2$ because it is the unfolding of the graph that we have just seen to be in $\mathfrak{C}_1$.

The resulting hierarchy of graphs is called the *Caucal hierarchy* because it has been introduced by Caucal in [26]. From Proposition 2.5 and Theorem 2.6 we directly obtain that we only create graphs with decidable MSO-theory in the Caucal hierarchy.

THEOREM 2.7 ([26]) *Every graph in the Caucal hierarchy has a decidable MSO-theory.*

For general structures, a decidable MSO-theory does not imply a decidable WMSO-theory. However, on deterministic trees one can express in MSO that a set is finite. A deterministic tree is an edge labeled graph that is a tree such that each vertex has at most one successor for each edge label. A set of vertices of a deterministic tree is finite if its closure under predecessors does not contain an infinite path (by König's Lemma). This property can be expressed in MSO. Hence, on deterministic trees we can rewrite every WMSO-formula into an MSO formula. This shows that the deterministic trees in the Caucal hierarchy have a decidable WMSO-theory.

Now we use a result from [20] stating that every graph on level n of the Caucal hierarchy can be obtained from a deterministic tree of level n by an inverse rational mapping. Roughly speaking, an inverse rational mapping h^{-1} applied to some graph G defines an edge with label a between two

vertices u and v if u and v are connected in G by a path that is labeled by a word in a regular language specified by the mapping h. More details on inverse rational mappings can be found in [26] and [20]. But it is not difficult to see that an inverse rational mapping can equivalently be defined by a WMSO-interpretation. This means that every graph on level n in the Caucal hierarchy can be obtained by a WMSO-interpretation from a deterministic tree on level n. We obtain the following theorem as a consequence of Theorem 2.7 and the result from [20].

THEOREM 2.8 *Every graph in the Caucal hierarchy has a decidable WMSO-theory.*

In [20] it has been shown that the hierarchy of the classes $\mathfrak{C}_n$ is strict. The proof uses a characterization of the graphs in level $\mathfrak{C}_n$ as the configuration graphs of higher-order pushdown automata.

THEOREM 2.9 ([20]) *For each n there is a graph in $\mathfrak{C}_{n+1}$ that is not in $\mathfrak{C}_n$.*

In Section 2.5 we use the Caucal hierarchy to construct a hierarchy of structures with decidable first-order theory. Based on Theorem 2.9 and the technique developed in Chapter 6 we are able to show that this new hierarchy is also strict.

2.4 Finite set interpretations

The interpretations introduced in the previous section define new relations on the domain of the given structure. In this section we introduce a more powerful kind of interpretation that builds a new structure whose universe consists of finite sets of elements of the given structure. This is achieved by using formulas with free set variables instead of free element variables. Otherwise, the definition of set interpretations is the same as the one for standard interpretations. Since we are interested in finite subsets of the given domain, we use the formalism WMSO in which quantifications only range over finite sets.

Let τ and $\hat{\tau}$ be two signatures. A *finite set interpretation* from τ-structures to $\hat{\tau}$-structures is given as a list $\mathcal{I} = \langle \phi_{dom}, (\phi_{\hat{R}})_{\hat{R}\in\hat{\tau}} \rangle$ of WMSO-formulas over the signature τ. The domain formula ϕ_{dom} has one free set variable, and for each $\hat{R} \in \hat{\tau}$ the formula $\phi_{\hat{R}}$ has $|\hat{R}|$ many free set variables.

The application of a finite set interpretation to a structure is defined in the same way as for standard interpretations with the difference that the elements of the obtained structure are finite subsets of the universe of the original structure instead of elements of the original structure. Formally, given a structure $\mathcal{S} = (D, (R^{\mathcal{S}})_{R\in\tau})$ and a finite set interpretation $\mathcal{I} =$

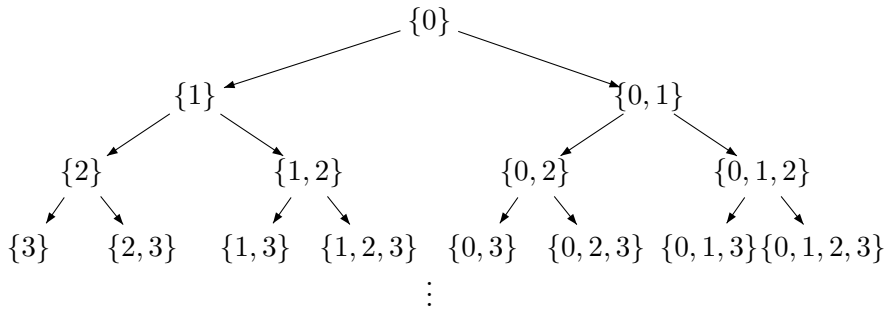


Figure 2.3: The nodes of the infinite binary tree $\mathfrak{t}_2$ coded by sets of natural numbers

$\langle \phi_{dom}, (\phi_{\hat{R}})_{\hat{R} \in \hat{\tau}} \rangle$, the structure $\mathcal{I}(\mathcal{S})$ has the universe

$$\hat{D} = \{U \subseteq D^{\mathcal{S}} \mid U \text{ finite}, \mathcal{S} \models \phi_{dom}[U]\},$$

and the relations are defined by

$$\hat{R} = \{(U_1, \ldots, U_{|\hat{R}|}) \in \hat{D}^{|\hat{R}|} \mid \mathcal{S} \models \phi_{\hat{R}}[U_1, \ldots, U_{r_i}]\}.$$

One can note at this point that set interpretations can be defined in a similar way, simply by removing the finiteness hypothesis on sets and using MSO instead of WMSO. At the end of this section we briefly comment on such possible variants for the definition.

The following example already appears in [44]. We show how to obtain the infinite binary tree extended with the prefix and equal level relations, i.e., the structure $\mathfrak{t}_2^{el} = \langle \{0,1\}^*, E_0, E_1, \sqsubseteq, el \rangle$, by a finite set interpretation from the infinite unary tree $\mathfrak{t}_1$, i.e., from the natural numbers with successor $\mathfrak{t}_1 = (\mathbb{N}, succ)$. To realize this we have to code the nodes of the tree by finite sets of natural numbers and to describe the relations E_0 (for left successor), E_1 (for right successor), $\sqsubseteq$ (for prefix), and el (for equal level) by means of WMSO formulas.

The coding of the nodes is depicted in Figure 2.3. A node $u \in \{0,1\}^*$ is represented by the set of positions corresponding to letter 1 in u and additionally by its length. For example, the node 100 is coded by $\{0, 3\}$ because its length is 3 and only position 0 is labeled 1. We now define the finite set interpretation $\mathcal{I} = (\delta, \phi_{E_0}, \phi_{E_1}, \phi_{\sqsubseteq}, \phi_{el})$ such that $\mathcal{I}(\mathfrak{t}_1)$ yields the binary tree depicted in Figure 2.3 together with the relations $\sqsubseteq$ and el. In the formulas we use abbreviations like $<$ and max that can easily be defined by WMSO-formulas in $\mathfrak{t}_1$.

- $\delta(X) := \exists x(x \in X)$ (all finite sets except $\emptyset$ are used in the coding).

- $\phi_{\sqsubseteq}(X_1, X_2) := \max(X_1) < \max(X_2) \wedge \forall x(x < \max(X_1) \rightarrow (x \in X_1 \leftrightarrow x \in X_2))$.
- $\phi_{E_0}(X_1, X_2) := \phi_{\sqsubseteq}(X_1, X_2) \wedge \max(X_2) = \max(X_1)+1 \wedge \max(X_1) \notin X_2$.
- $\phi_{E_1}(X_1, X_2) := \phi_{\sqsubseteq}(X_1, X_2) \wedge \max(X_2) = \max(X_1)+1 \wedge \max(X_1) \in X_2$.
- $\phi_{el}(X_1, X_2) := \max(X_1) = \max(X_2)$

Let us proceed with some elementary considerations. Similarly to standard interpretations that preserve the decidability of the corresponding logic, one can show that finite set interpretations map structures with decidable WMSO-theory to structures with decidable FO-theory. The reason is that elements of the new structure are finite sets of elements of the old structure and hence one can transfer FO-formulas on the new structure to WMSO-formulas on the old structure by replacing the atomic formulas by their definition in the interpretation.

PROPOSITION 2.10 *If $\mathcal{I}$ is a finite set interpretation and $\mathcal{S}$ is a structure with decidable WMSO-theory, then $\mathcal{I}(\mathcal{S})$ has a decidable FO-theory.*

Obviously, finite set interpretations are not closed under composition. But, as stated in the following proposition, applying an FO-interpretation after, or a WMSO-interpretation before a finite set interpretation, does not give more expressive power. The proof of this proposition is straightforward.

PROPOSITION 2.11 *Let $\mathcal{I}_1$ be a FO-interpretation, $\mathcal{I}_2$ be a WMSO-interpretation, and $\mathcal{I}$ be a finite set interpretation. Then $\mathcal{I}_1 \circ \mathcal{I} \circ \mathcal{I}_2$ is effectively a finite set interpretation.*

To finish our elementary considerations on finite set interpretations, we present Proposition 2.12, which is a form of converse to Proposition 2.11. It states that every finite set interpretation can be described as the composition of a specific one, called the weak powerset interpretation, and a first-order interpretation. Let $\mathcal{P}^W$ be the finite set interpretation that sends every structure $\mathcal{S}$ of signature τ onto a structure of signature $\tau \cup \{\preceq\}$ (where $\preceq$ is a new binary symbol) such that

- the universe is the set of finite subsets of the universe of $\mathcal{S}$,
- each symbol R in τ has the same interpretation as in $\mathcal{S}$ but over singletons instead of elements,
- the interpretation of $\preceq$ corresponds to the subset ordering.

The interpretation $\mathcal{P}^W$ is called the *weak powerset interpretation*, and we call $\mathcal{P}^W(\mathcal{S})$ the *weak powerset structure* of $\mathcal{S}$.

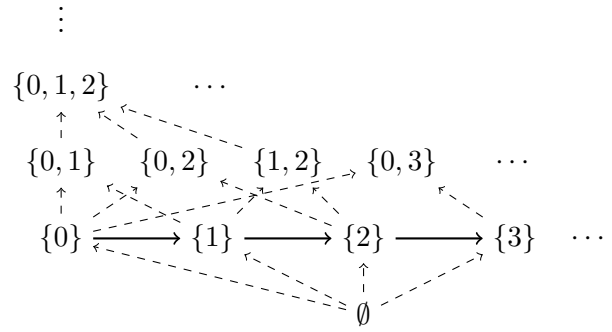


Figure 2.4: Weak powerset generator of $\mathfrak{t}_1$

Figure 2.4 depicts the weak powerset structure of $\mathfrak{t}_1$. The solid arrows correspond to the successor relation and the dashed arrows indicate the (transitive reduction of the) inclusion relation.

The weak powerset interpretation allows to reconstruct all other finite set interpretations as stated in the following proposition, which is obtained by a simple syntactic translation of formulas.

PROPOSITION 2.12 *For each finite set interpretation $\mathcal{I}$ there exists a FO-interpretation $\mathcal{I}_1$ such that $\mathcal{I}_1 \circ \mathcal{P}^W = \mathcal{I}$.*

Possible variants

The definition of finite set interpretations that we provide uses WMSO and one might wonder why we restrict ourselves to this logic. At least two modifications of the definition are very natural:

- replace WMSO logic with MSO logic in the formulas but still only consider finite sets for the free variables,
- use full MSO and produce a structure over the powerset of the original structure instead of its finite subsets.

If we use the first extension, then Proposition 2.12 fails. Proposition 2.10 also becomes problematic because transferring FO-formulas over $\mathcal{I}(\mathcal{S})$ back to $\mathcal{S}$ yields MSO-formulas in which we need to express that sets are finite (because the FO quantifiers over $\mathcal{I}(\mathcal{S})$ only range over finite sets). In general, this is not possible in MSO. Therefore, WMSO is a natural choice for the definition of finite set interpretations.

The second extension leads to what can be naturally called *set interpretation*. The formulas are MSO and the free variables can be interpreted by arbitrary sets when defining the new structure.

All the results presented in this section remain valid for this variant when we replace WMSO by MSO and the weak powerset interpretation by the powerset interpretation with its natural definition. However, all the results developed in Chapter 6 make explicit use of the finiteness of the sets representing elements of the new structure. In particular, Theorem 6.2 fails for general set interpretations. This has been shown in the particular case of ω-automatic structures in [60] (for the connection of set interpretations and automatic structures see Section 2.5).

2.5 Automatic structures

In the previous sections we have seen how it is possible to construct structures by applying operations on given structures such that the decidability of certain theories can be transferred to the new structures. A different approach to obtain structures with decidable theories is to represent them symbolically in some finite way using a formalism that can be manipulated algorithmically.

When following this approach, a natural way is to use finite words to represent the elements of the universe of a structure, and to define the relations by finite automata. Since automata have strong closure properties and can easily be manipulated algorithmically, one obtains structures with a decidable first-order theory.

Classical variants of those structures consider universes consisting of infinite words, finite trees, or infinite trees. One obtains the four classes of word-automatic, ω-word-automatic, tree-automatic, and ω-tree-automatic structures. As the most basic class, word-automatic structures are usually just called automatic structures.

Historically, automatic as well as ω-automatic structures have been introduced by Hodgson [53]. Khoussainov and Nerode introduce the notion of automatically presentable theories [61], starting the study of definability in automatic structures. The extension to tree-automatic structures can be traced back, in a different framework, to the work of Dauchet and Tison [39]. Blumensath and Grädel [9, 11] then formalize the notion of tree-automatic structure and add to it the family of ω-tree-automatic structures.

Our main interest in (tree-)automatic structures comes from their close correspondence to to finite set interpretations. For this reason we briefly present the main definitions in this section and illustrate the relationship to finite set interpretations. The main goal is to relate the results obtained in Chapter 6 to the field of automatic structures. For more detailed expositions on automatic structures we refer the reader to [106, 11, 9].

In the case of words, a relation $R \subseteq (\Sigma^*)^r$ is automatic if there is a finite automaton accepting exactly the tuples $(w_1, \ldots, w_r) \in R$, where the automaton reads all the words in parallel with the shorter words padded

with a dummy symbol $\diamond$. Formally, for $w_1, \ldots, w_r \in \Sigma^*$ we define

$$w_1 \otimes \cdots \otimes w_r = \begin{bmatrix} a'_{11} \\ \vdots \\ a'_{r1} \end{bmatrix} \cdots \begin{bmatrix} a'_{1n} \\ \vdots \\ a'_{rn} \end{bmatrix} \in (\Sigma_\diamond^r)^*$$

where $\Sigma_\diamond = \Sigma \cup \{\diamond\}$, n is the maximal length of one of the words w_i, and a_{ij} is the jth letter of w_i if $j \leq |w_i|$ and $\diamond$ otherwise. A language $L \subseteq ((\Sigma \cup \{\diamond\})^r)^*$ defines a relation $R_L \subseteq (\Sigma^*)^r$ in the obvious way: $(w_1, \ldots, w_r) \in R_L$ iff $w_1 \otimes \cdots \otimes w_r \in L$. A tuple $(L, L_1, \ldots, L_n)$ of languages $L \subseteq \Sigma^*$ and $L_i \subseteq (\Sigma_\diamond^{r_i})^*$ defines a structure of universe L with the relations R_{L_i} of arity r_i.

A structure $\mathcal{S} = (D, R_1, \ldots, R_n)$ is *automatic* if it is isomorphic to a structure of the above kind for regular languages $L, L_1, \ldots, L_n$.

A typical example for an automatic structure is $(\mathbb{N}, +)$, consisting of the natural numbers with addition. A possible automatic representation of this structure uses the binary coding of numbers. It is easy to verify that the relation $+(x, y, z)$ in the meaning of $x + y = z$ can be defined by a finite automaton.

As already mentioned at the beginning of this section, the strong closure properties of finite automata imply that each FO-formula over an automatic structure can inductively be translated into a finite automaton. The satisfiability test then reduces to an emptiness test of the resulting automaton, which is decidable.

THEOREM 2.13 ([53]) *Automatic structures have a decidable FO-theory.*

This yields a simple proof of the decidability of Presburger arithmetic [101], i.e., that FO-theory of the natural numbers with addition.

To define tree-automatic structures we need a way to code tuples of finite trees, i.e., we need an operation $\otimes$ for finite trees. For a tree $t : dom(t) \to \Sigma$ let $t^\diamond : \{0,1\}^* \to \Sigma_\diamond$ be defined by $t^\diamond(u) = t(u)$ if $u \in dom(t)$, and $t^\diamond(u) = \diamond$ otherwise. For finite Σ-labeled trees $t_1, \ldots, t_r$ we define the $\Sigma_\diamond^r$-labeled tree $t = t_1 \otimes \cdots \otimes t_r$ by $dom(t) = dom(t_1) \cup \cdots \cup dom(t_r)$ and $t(u) = (t_1^\diamond(u), \ldots, t_r^\diamond(u))$. When viewing words as unary trees, this definition corresponds to the operation $\otimes$ as defined for words. As in the case of words a set T of finite $\Sigma_\diamond^r$-labeled trees defines the relation R_T by $(t_1, \ldots, t_r) \in R_T$ iff $t_1 \otimes \cdots \otimes t_r \in T$.

A structure is called *tree-automatic* if it is isomorphic to a structure given by a tuple $(T, T_1, \ldots, T_n)$ of regular tree languages in the same way as for words.

A natural example for a tree-automatic structure that is not word-automatic is $(\mathbb{N}, \cdot)$, i.e., the natural numbers with the multiplication. To code a natural number n we consider its prime decomposition $n = p_1^{i_1} \cdot p_2^{i_2} \cdots p_n^{i_n}$, where p_i denotes the ith prime number. Then we code the exponents using

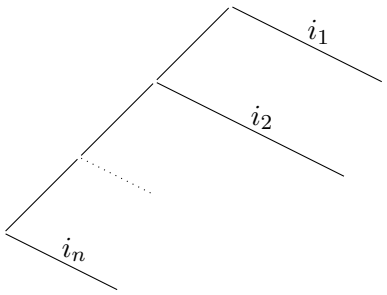


Figure 2.5: Coding a natural number by arranging the powers of its prime decomposition in a tree

their binary representations and arrange them in a tree as shown in Figure 2.5. Using this coding the multiplication of two numbers reduces to the addition of the exponents on the branches. This can be done by a tree automaton that works on each branch similar to the word automaton used for the addition in the automatic presentation of $(\mathbb{N}, +)$.

This example can be found in Section 5.3 of [9] where such a coding is used in a more general framework of weak direct products of structures. The idea is that $(\mathbb{N}, \cdot)$ is such a weak direct product of $(\mathbb{N}, +)$ because we use finitely many copies of the automata for the addition to represent the multiplication. That $(\mathbb{N}, \cdot)$ is not word-automatic is also shown in [9].

In the same way as for word-automatic structures we obtain the decidability of the FO-theory by the closure properties of the automata representing the structure.

THEOREM 2.14 ([9, 11]) *Tree-automatic structures have a decidable FO-theory.*

Using infinite words or trees leads in a natural way to the definitions of ω-automatic structures and ω-tree automatic structures. We are not going to use structures of this kind and hence we do not detail the definitions here.

One should note that we only consider so called injective presentations of automatic structures. A more general definition as, e.g., in [9] additionally uses a regular language $L_\sim \subseteq (\Sigma_\diamond^2)^*$ defining an equivalence relation identifying words representing the same element of the structure (and similarly for the other variants of automatic structures). It is known that injective presentations are sufficient for automatic structures [61] meaning that all structures that are automatic in the more general sense are also automatic according to our definition. The corresponding result for tree-automatic structures has been established in the context of finite set interpretations in [30] and is presented in Chapter 6 in Corollary 6.3.

The connection between (tree-)automatic structures and finite set inter-

pretations that allows such a transfer of results between the two worlds is presented below.

Recall that $\mathfrak{t}_1$ is the (unlabeled) infinite unary tree, i.e., the natural numbers with successor, and that $\mathfrak{t}_2$ is the (unlabeled) infinite binary tree. The following fact is a straightforward consequence of the definition of automatic structures and of the equivalences between WMSO-logic and finite automata. The first claim also appears in [106, Thm. C.2.11, page 50].

PROPOSITION 2.15 *The following holds up to isomorphism*

- *A structure is automatic iff it is finite set interpretable in* $\mathfrak{t}_1$.
- *A structure is tree-automatic iff it is finite set interpretable in* $\mathfrak{t}_2$.

Proof. As already mentioned, the first claim is shown in [106].

We describe here how to proceed for tree-automatic structures, starting by an explanation how to obtain a tree-automatic presentation from a finite set interpretation $\mathcal{I}$ in $\mathfrak{t}_2$. A finite subset X of $\mathfrak{t}_2$ is coded as a finite tree as follows: the tree has the smallest domain that contains all elements from X and is labeled 0 at positions that are not in X and 1 at positions that are in X. For each formula of $\mathcal{I}$ there is an equivalent parity automaton over $\mathfrak{t}_2$. This automaton can easily be turned into an automaton over finite trees accepting the corresponding relation over the codings as just described.

For the other direction we start from a tree-automatic presentation of a structure. A first thing to note is that a singleton alphabet of tree labels is sufficient because each symbol from a larger alphabet can be encoded in a finite pattern; in this construction, we use the leaf/non-leaf nature of each node for coding information. Since now the alphabet is a singleton, one naturally encodes a tree t by the finite set $dom(t)$. Now pick an automaton from the tree-automatic presentation accepting a relation and pick a tuple of such sets coding finite trees. Using standard techniques for translating automata to logic (cf. [118]), one describes in WMSO that the corresponding tuple of finite trees is accepted by the automaton. □

Proposition 2.15 shows that applying finite set interpretations to $\mathfrak{t}_2$ yields exactly the tree-automatic structures. But using finite set interpretations there is no reason to limit ourselves to the very simple structure $\mathfrak{t}_2$. Starting from any tree with decidable WMSO-theory we obtain a class of structures with decidable FO-theory when applying finite set interpretations. In Section 2.3 we have defined a rich class of structures with decidable WMSO-theory, namely the Caucal hierarchy, containing $\mathfrak{t}_2$ on the first level.

From this hierarchy it is easy to construct a corresponding tree-automatic hierarchy. The tree-automatic structures of level n are the image of the structures of level n of the Caucal hierarchy by finite set interpretations. Let us denote the nth level of this *tree-automatic hierarchy* by TAUT_n. Since

the trees on the first level of the Caucal hierarchy are regular and therefore can be obtained from $\mathfrak{t}_2$ by WMSO-interpretations, we can deduce from Proposition 2.15 and Proposition 2.11 that TA_{UT_1} coincides with the class of tree-automatic structures.

Furthermore, since all structures in the Caucal hierarchy have a decidable WMSO-theory (Theorem 2.8) we produce structures of decidable FO-theory by finite set interpretations (Proposition 2.10).

THEOREM 2.16 *For each n, every structure in TA_{UT_n} has a decidable FO-theory.*

Although the Caucal hierarchy is strict (see Theorem 2.9) we can not directly conclude that the tree-automatic hierarchy is also strict. To obtain such a result we need techniques to show that certain structures are not definable by finite set interpretations from a class of given structures. In Chapter 6 we develop a powerful method that allows to prove such statement.

Chapter 3

Choice functions and well-orderings

The main goal of this chapter is to present a simple proof for the fact (first shown by Gurevich and Shelah in [49]) that there is no MSO-definable choice function on the infinite binary tree $\mathfrak{t}_2$. A choice function on $\mathfrak{t}_2$ is a mapping assigning to each nonempty set of nodes one element from this set, i.e., the function chooses for each set one of its elements. Such a function is MSO-definable if there is an MSO-formula with one free set variable X and one free element variable x such that for each nonempty set U of nodes there is exactly one element $u \in U$ such that the formula is satisfied if X is interpreted as U and x is interpreted as u.

The question of the existence of an MSO-definable choice function over the infinite binary tree can be seen as a special instance of the more general uniformization problem, which asks, given a relation that is defined by a formula with free variables, whether it is possible to define by another formula a function that is compatible with this relation. More precisely, given a formula $\phi(\bar{X}, \bar{Y})$ with vectors $\bar{Y}, \bar{X}$ of free variables, such that $\forall \bar{X} \exists \bar{Y} \phi(\bar{X}, \bar{Y})$ is satisfiable, uniformization asks for a formula $\phi^*(\bar{X}, \bar{Y})$ such that

1. ϕ^* implies ϕ (each interpretation of $\bar{Y}, \bar{X}$ making ϕ^* true also makes ϕ true),

2. and ϕ^* defines a function in the sense that for each interpretation of $\bar{X}$ there is exactly one interpretation of $\bar{Y}$ making ϕ^* true.

The question of the existence of a choice function is the uniformization problem for the formula $\phi(X, y) = X \neq \emptyset \to y \in X$.

The infinite line $\mathfrak{t}_1$ is known to have the uniformization property for MSO [112]. On the infinite binary tree MSO is known to be decidable (see Theorem 2.2) but it does not have the uniformization property. This was conjectured in [112] and proved in [49] where it is shown that there is no MSO-definable choice function on the infinite binary tree. The proof in [49]

uses complex set theoretical arguments, whereas it appears that the result can be obtained by much more basic techniques. We show that this is indeed true and present a proof that only relies on the equivalence of MSO and automata over infinite trees and otherwise only uses basic techniques from automata theory. Besides its simplicity, another advantage of the proof is that we provide a concrete family of sets (parameterized by natural numbers) such that each formula fails to make a choice for those sets with the parameters chosen big enough. We use this fact when we discuss two applications of the result concerning the definability of strategies in infinite games (Section 3.2) and unambiguous tree automata (in Chapter 4).

The subject of MSO-definability of choice functions on trees has been studied in more depth in [72], where the authors consider more general trees not only the infinite binary tree. They show the following dichotomy: for a tree it is either not possible to define a choice function in MSO, or it is possible to define a well-ordering on the domain of the tree.

Note that it is very easy to define a choice function if one has access to a well-ordering of the domain: for each set one chooses the minimal element according to the well-ordering. In particular, the non-existence of an MSO-definable choice function on the infinite binary tree implies that there is no MSO-definable well-ordering on the infinite binary tree. We strengthen this result by showing that extending the infinite binary tree by any well-ordering leads to a structure with undecidable MSO-theory. As a consequence we obtain that each structure in which we can MSO-define a well-ordering and MSO-interpret the infinite binary tree must have an undecidable MSO-theory.

The chapter is structured as follows. In Section 3.1 we give the proof that there is no MSO-definable choice function on the infinite binary tree. In Section 3.2 we use the result to show the undefinability of winning strategies in certain games. In Section 3.3 the undecidability of the MSO-theory of the infinite binary tree augmented by any well-ordering is shown.

The results in this chapter have been obtained in joint work with Arnaud Carayol and are published in [21].

3.1 MSO-definable choice functions

As described in the introduction of this chapter, an MSO-definable choice function is given by an MSO-formula $\phi(X, x)$ such that

$$\forall X \exists x.\ X \neq \emptyset \rightarrow (x \in X \wedge \phi(X, x) \wedge \forall y.\ \phi(X, y) \rightarrow x = y)$$

is true over the infinite binary tree. This section is devoted to the proof of the following theorem of Gurevich and Shelah.

THEOREM 3.1 ([49]) *There is no MSO-definable choice function on the infinite binary tree.*

The technical formulation of the result we prove is given in Theorem 3.5 (on page 52), where we concretely provide counter examples for which a given formula cannot choose a unique element. As a machinery for the proof we use tree automata, which are easier to manipulate (at least for our purpose) than formulas. In the following we present a slight modification of the standard automaton model that we use in the proof.

An MSO[$\mathfrak{t}_2$]-formula with free variables defines a relation between subsets of $\{0,1\}^*$. In this section we consider formulas with two free variables, and a natural view in the context of choice functions is that the first variable represents the input, and the second variable represents the output. A choice function is the special case that the second variable is an element and not a set, and that for each nonempty input set there is a unique output which is inside the input set. To adapt this general view of inputs and outputs on the automata theoretic level we introduce automata with output, called transducers. These transducers are very simple: An output symbol is associated to each transition, which gives a natural correspondence between runs and output trees.

A *parity tree automaton with output* is a tuple $\mathcal{A} = (Q, \Sigma, q^{\mathrm{i}}, \Delta, c, \lambda)$, where $(Q, \Sigma, q^{\mathrm{i}}, \Delta, c)$ is a standard PTA, and $\lambda : \Delta \to \Gamma$ is an output function assigning to each transition a symbol from the output alphabet Γ. When ignoring the output function we can use all the terminology of standard PTAs, in particular the notion of accepting run and the notion of $\mathcal{A}$-equivalence (see page 21).

In general, a PTA with output defines for each input tree $t \in \mathcal{T}_\Sigma^\omega$ a set of output trees $\mathcal{A}(t) \subseteq \mathcal{T}_\Gamma^\omega$ defined by

$$\mathcal{A}(t) = \{\lambda(\rho) \mid \rho \text{ is an accepting run of } \mathcal{A} \text{ on } t\},$$

where $\lambda(\rho)$ denotes the tree in $\mathcal{T}_\Gamma^\omega$ that is obtained by taking at each node u the output produced by λ applied to the transition used at u in ρ.

As already indicated above we are interested in such transducers for representing formulas with two free variables, i.e., the input and the output alphabet are both equal to $\{0,1\}$. The following theorem can easily be shown by a simple argument based on Theorem 2.1.

THEOREM 3.2 *For each MSO[$\mathfrak{t}_2$]-formula $\phi(X,Y)$ there is a PTA with output $\mathcal{A}_\phi$ such that $t[U,U'] \models \phi$ iff $t[U'] \in \mathcal{A}(t[U])$.*

We are interested in PTAs with output that represent possible candidates for choice formulas. i.e., the case where the second free variable of the formula represents a single element. This motivates the following definition.

A *weak choice automaton* is a PTA with output such that the input and the output alphabet are equal to $\{0,1\}$, and for each $U \subseteq \{0,1\}^*$ there is at most one output tree in $\mathcal{A}(t[U])$, and this output tree is of the form $t[u]$ for some $u \in U$. A weak choice automaton chooses for each set U at most one

element from this set. It is called weak because it does not have to choose an element. A weak choice automaton that chooses one element from each nonempty set represents a choice formula. Our goal is to show that such an automaton cannot exist. Note that there are weak choice automata: for example the automaton that rejects all inputs is a weak choice automaton because it does not choose an element for any set.

We show in Lemma 3.4 that for each weak choice automaton $\mathcal{A}$ we can find a set that is complex enough such that $\mathcal{A}$ cannot choose an element of U because then we could construct another run outputting something different, contradicting the property of a weak choice automaton. More precisely, we define a family $(U_{M,N})_{M,N}$ of sets such that for each weak choice automaton $\mathcal{A}$ we can find M and N such that $\mathcal{A}$ cannot choose an element of $U_{M,N}$. To achieve this, we "hide" the elements of the set very deep in the tree such that weak choice automata up to a certain size are not able to uniquely choose an element.

For $M, N \in \mathbb{N}$ the set $U_{M,N} \subseteq \{0,1\}^*$ is defined by the following regular expression

$$U_{M,N} = \{0,1\}^*(0^N 0^* 1)^M \{0,1\}^*.$$

In Figure 3.1 a deterministic finite automaton for the set $U_{M,N}$ is shown, where the dashed edges represent transitions for input 1 that lead back to the initial state x_M. The chains of 0-edges between x_{k+1} and x_k have length N. The final state is x_0. It is easy to verify that x_0 is reachable from x_M by exactly those paths whose sequence of edge labels is in the set $U_{M,N}$. Let $t_{M,N} = t[U_{M,N}]$ and let $t_{k,M,N} = t[U_{k,M,N}]$, where $U_{k,N,M}$ is the set that we obtain by making x_k the initial state.

We now fix a weak choice automaton $\mathcal{A} = (Q, \{0,1\}, q_0, \Delta, c, \lambda)$ on $\{0,1\}$-labeled trees and take $M = 2^{|Q|} + 1$ and $N = |Q| + 1$. For these fixed parameters we simplify the notation by letting $t_k = t_{k,M,N}$. In particular, $t_M = t_{M,M,N} = t_{M,N}$. We say that a subtree (of some tree t) that is isomorphic to t_k for some k is of type t_k.

Our aim is to trick the automaton $\mathcal{A}$ to show that it cannot choose a unique element from the set $U_{M,N}$. This is done by modifying a run that outputs an element u of $U_{M,N}$ such that we obtain another run outputting something different.

To understand the general idea, consider the path between x_{k+1} and x_k for some k. If we take a 1-edge before having reached the end of the 0-chain, i.e., one of the dashed edges leading back to the initial state x_M, then we reach a subtree of type t_M. But if we walk to the end of the 0-chain and then move to the right using a 1-edge, then we arrive at a subtree of type t_k. If we show that there is $\ell < M$ such that t_M and t_ℓ are $\mathcal{A}$-equivalent, then this means that $\mathcal{A}$ has no means to identify when it enters the part where taking a 1-edge leads to subtree of type t_ℓ. We then exploit this fact by pumping the run on this part of the tree such that we obtain another

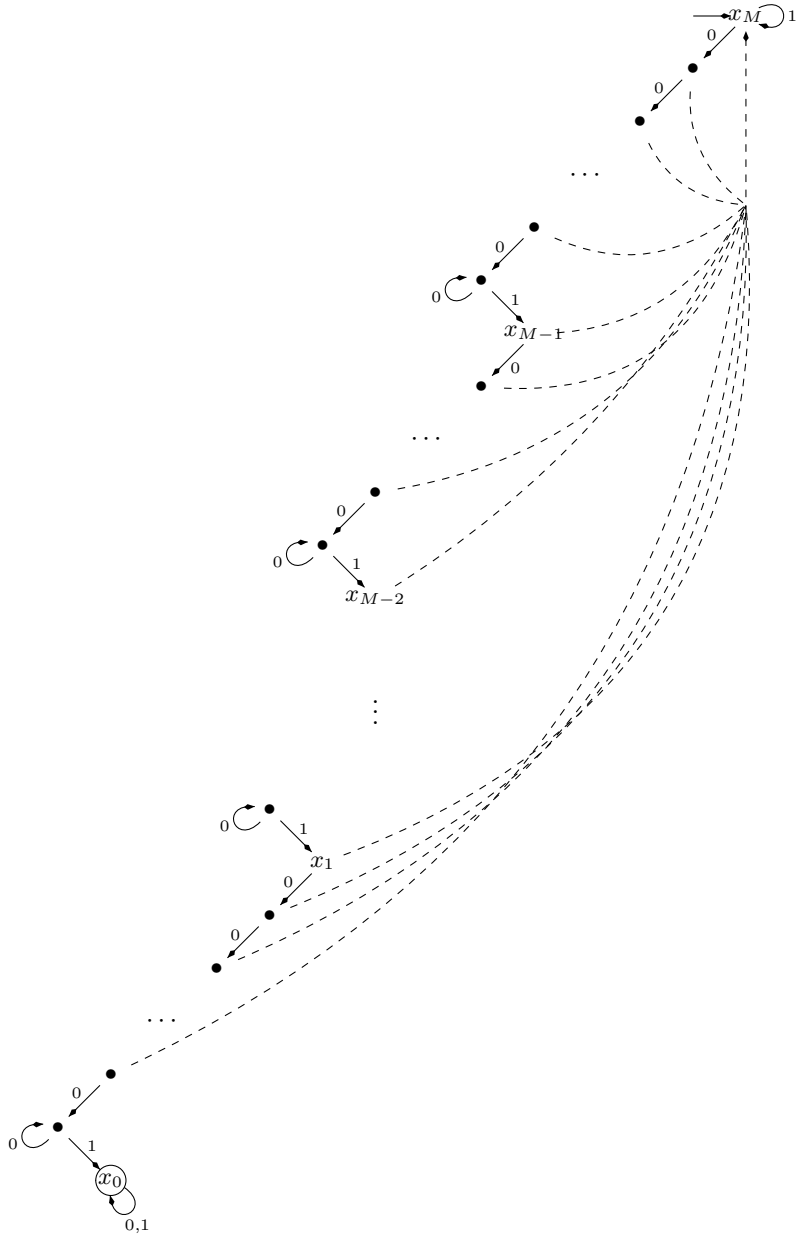


Figure 3.1: A representation of the regular tree $t_{M,N}$ by a DFA accepting $U_{M,N}$.

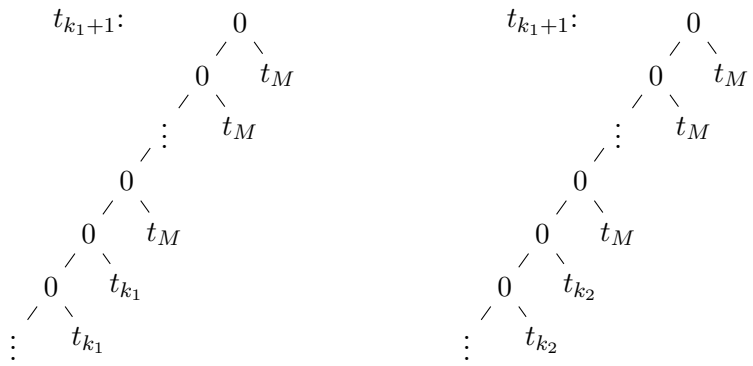


Figure 3.2: The shape of the trees t_{k_1+1} and t_{k_2+1} in the proof of Lemma 3.3

run marking something different.

LEMMA 3.3 *There exists $\ell < M$ such that $t_M \equiv_{\mathcal{A}} t_\ell$.*

Proof. We consider for each tree $t \in \mathcal{T}^{\omega}_{\{0,1\}}$ the function $f_t : Q \to \{a, r\}$ with $f_t(q) = a$ if there is an accepting run of $\mathcal{A}$ from q on t, and $f_t(q) = r$ otherwise. By definition, two trees t, t' are $\mathcal{A}$-equivalent if $f_t = f_{t'}$. There are at most $2^{|Q|}$ different such functions. By the choice of M there are $1 \leq k_1 < k_2 \leq M$ such that $t_{k_1} \equiv_{\mathcal{A}} t_{k_2}$.

We now show that $t_{k_1} \equiv_{\mathcal{A}} t_{k_2}$ implies $t_{k_1+1} \equiv_{\mathcal{A}} t_{k_2+1}$ (in case $k_2 < M$, otherwise we choose $\ell = k_1$). This allows us to lift the equivalence step by step to t_ℓ and t_M for $\ell = k_1 + (M - k_2)$.

If we only look at the left-most branches and the types of the trees going off to the right, then t_{k_1+1} and t_{k_2+1} look as shown in Figure 3.2. From this picture it should be clear that $t_{k_1+1} \equiv_{\mathcal{A}} t_{k_2+1}$ because an accepting run on t_{k_1+1} can easily be turned into an accepting run on t_{k_2+1}, and vice versa, using the equivalence of t_{k_1} and t_{k_2}. □

The following lemma states that it is impossible for $\mathcal{A}$ to choose an element of $U_{M,N}$ (and hence the set of outputs computed by $\mathcal{A}$ on $t_{M,N}$ is empty).

LEMMA 3.4 *The weak choice automaton $\mathcal{A}$ cannot choose an element from $U_{M,N}$, i.e., $\mathcal{A}([t_{M,N}]) = \emptyset$.*

Proof. Assume that that there is an accepting run ρ of $\mathcal{A}$ on $t_M = t_{M,N}$. Since $\mathcal{A}$ is a weak choice automaton this means that the output of this run must be an element of $U_{M,N}$, i.e., $\lambda(\rho) = t[u]$ for some $u \in U_{M,N}$.

From ρ we construct another accepting run with a different output tree, contradicting the definition of weak choice automaton.

Since u is in $U_{M,N}$, we know that the path from the root to u must end with the pattern as defined by the automaton in Figure 3.1. Hence we can make the following definitions. For $0 \leq k \leq M$ let u_k denote the maximal prefix of u such that the subtree at u_k is of type t_k. Let ℓ be as in Lemma 3.3. For $i \geq 0$ we let $v_i = u_{\ell+1}0^i$ and $v'_i = v_i 1$. Note that $v_0 = u_{\ell+1}$ and that for $0 \leq i < N$ the subtrees at v'_i are of type t_M, and for $i \geq N$ the subtrees at v'_i are of type t_ℓ.

From Lemma 3.3 we know that $t_\ell \equiv_{\mathcal{A}} t_M$. Hence, for each accepting run ρ_q of $\mathcal{A}$ from q on t_M we can pick an accepting run ρ'_q of $\mathcal{A}$ from q on t_ℓ.

By the choice of N there are $0 \leq j < j' < N$ such that $\rho(v_j) = \rho(v_{j'})$. For the moment, consider only the transitions taken in ρ on the sequence $v_0, v_1, \ldots$, i.e., on the infinite branch to the left starting from v_0. We now simply repeat the part of the run between v_j and $v_{j'}$ once. The effect is that some of the states that were at a node v'_i for $i < N$ are pushed to nodes v'_i for $i \geq N$, i.e., the states are moved from subtrees of type t_M to subtrees of type t_ℓ. But for those states q we can simply plug the runs ρ'_q that we have chosen above.

More formally, we define the new run ρ' of $\mathcal{A}$ on t_M as follows. On the part that is not in the subtree below v_0 the run ρ' corresponds to ρ. In the subtree at v_0 we make the following definitions, where $h = j' - j$.

- For $i < j'$ let $\rho'(v_i) = \rho(v_i)$ and $\rho'(v'_i) = \rho(v'_i)$.
- For $i \geq j'$ let $\rho'(v_i) = \rho(v_{i-h})$ and $\rho'(v'_i) = \rho(v'_{i-h})$.
- For the subtrees at v'_i for $i < j'$ we take the subrun of ρ at v'_i.
- For the subtrees at v'_i for $j' \leq i < N$ or $i \geq N + h$ we take the subrun of ρ at v'_{i-h}. This is justified because in these cases $\rho'(v'_i) = \rho(v'_{i-h})$ and the subtrees at v'_i and v'_{i-h} are of the same type (both of type t_M or both of type t_ℓ).
- For the subtrees at v'_i for $N \leq i < N + h$ we take the runs ρ'_{q_i} for $q_i = \rho'(v'_i)$. This is justified as follows. From $q_i = \rho'(v'_i)$ and the definition of ρ' we know that $\rho(v'_{i-h}) = q_i$. Hence, there is an accepting run of $\mathcal{A}$ from q_i on t_M. Thus, ρ'_{q_i} as chosen above is an accepting run of $\mathcal{A}$ from q_i on t_ℓ.

This run ρ' is accepting. Furthermore, the transition producing output 1 in ρ has been moved to another subtree: There are $n \geq N$ and $w \in \{0,1\}^*$ such that $u = u_{\ell+1}0^n w$. In ρ' the transition that is used at u in ρ is used at $u' = u_{\ell+1}0^{n+h}w$. Hence, we have constructed an accepting run whose output tree is different from $t[u]$, a contradiction. □

Of course, the statement is also true if we increase the value of M or N, e.g., if we let $N = M = 2^{|Q|+1}$. Thus, combining Theorem 3.2 and Lemma 3.4 we obtain the following.

THEOREM 3.5 *Let $\phi^*(X,x)$ by an MSO-formula. There exists $m \in \mathbb{N}$ such that for all $n \geq m$ and for each $u \in U_{n,n}$ with $t[U_{n,n},u] \models \phi^*$ there is $u' \neq u$ with $t[U_{n,n},u'] \models \phi^*$.*

Proof. Choose $n = 2^{|Q|+1}$. Assume that there is $u \in U_{n,n}$ such that $t[U_{n,n},u] \models \phi^*$. Consider the formula

$$\phi(X,x) := \phi^*(X,x) \wedge x \in X \wedge \neg\exists y(y \neq x \wedge \phi^*(X,y)).$$

Applying Theorem 3.2 to this formula yields a weak choice automaton $\mathcal{A}$. According to Lemma 3.4 we have $\mathcal{A}(t_{n,n}) = \emptyset$. Hence, because $\mathcal{A}$ and ϕ are equivalent, $t[U_{n,n},u] \not\models \phi$. By definition of ϕ this means that there must be $u' \neq u$ such that $t[U_{n,n},u'] \models \phi^*$. □

A direct consequence is the theorem of Gurevich and Shelah. The advantage of our proof is that we obtain a rather simple family of counter examples (the sets $U_{M,N}$).

An easy reduction allows us to extend the non-existence of an MSO-definable choice function to the case where we allow a finite number of fixed predicates as parameters. This result has already been shown in [73] in an even more general context, but again relying on the methods employed in [49].

COROLLARY 3.6 *Let $P_1,\ldots,P_n \subseteq \{0,1\}^*$ be arbitrary predicates. There is no MSO-formula $\phi^*(X_1,\ldots,X_n,X,x)$ such that for each nonempty set U there is exactly one $u \in U$ with $t[P_1,\ldots,P_n,U,u] \models \phi^*$.*

Proof. Assume that there are $P_1,\ldots,P_n \subseteq \{0,1\}^*$ and $\phi^*(X_1,\ldots,X_n,X,x)$ such that for each set U there is exactly one $u \in U$ with $t[P_1,\ldots,P_n,U,u] \models \phi^*$. Then the formula

$$\exists X_1,\ldots,X_n \forall X \exists x\ \phi^*(X_1,\ldots,X_n,X,x) \wedge \forall y\ \phi^*(X_1,\ldots,X_n,X,y) \rightarrow x = y$$

is satisfiable. Hence, there are regular predicates $P_1,\ldots,P_n$ (see Corollary 2.3 on page 32) such that

$$t[P_1,\ldots,P_n] \models \forall X \Big(\exists x\big(\phi^*(X_1,\ldots,X_n,X,x) \wedge \forall y\ \phi^*(X_1,\ldots,X_n,X,y) \rightarrow x = y\big)\Big).$$

Regular predicates are MSO-definable (see Proposition 2.4), and therefore we can find formulas $\psi_1(X_1),\ldots,\psi_n(X_n)$ defining $P_1,\ldots,P_n$, respectively. Then the formula $\phi'(X,x)$ defined as

$$\exists X_1,\ldots,X_n\ \phi^*(X_1,\ldots,X_n,X,y) \wedge \bigwedge_{i=1}^{n} \psi_i(X_i)$$

describes a choice function, contradicting Theorem 3.5. □

We point out here that this method only relies on the fact that the property of being a choice function is MSO-definable. This way of reducing the case with parameters to the parameter free case can be applied whenever the properties of the object under consideration are MSO-definable (in Proposition 3.18 on page 64 we also use this technique).

3.2 MSO-definable winning strategies

In Section 1.2 we have introduced games, and in Chapter 2 we have mentioned how to view arenas as relational structures. In this section we consider the question how to define strategies by MSO-formulas. Based on Theorem 3.1 we show that there are game arenas on which it is not possible to define winning strategies in MSO. We restrict ourselves to positional strategies. First of all, the winning conditions we are interested in all admit positional strategies, and furthermore we consider game arenas whose underlying graph is a tree. On such arenas each vertex contains the full information on its history, and hence each strategy can be seen as a positional strategy.

We start by defining MSO-definability of positional strategies. In this setting it is reasonable to consider arenas with edge labels to make it easier to address the individual edges by formulas. Let Γ be an alphabet for the edge labels and $G = (V, (E_a)_{a\in\Gamma}, V_\circ, V_\Box, (P_i)_{i\in Cols})$ be an arena. We say that a positional strategy $\sigma_\circ$ for Eva is MSO-definable if there is a formula $\phi(x, y)$ such that for all $v \in V_\circ$ there is exactly one $v' \in V$ such that $G \models \phi[v, v']$, namely $v' = \sigma_\circ(v)$.

We call G a *game tree* if the underlying graph is isomorphic to a labeled binary tree, i.e., $\Gamma = \{0, 1\}$, and V corresponds to the set $\{0, 1\}^*$ where E_0, E_1 are interpreted as usual (see Chapter 2, page 30). For game trees we always consider the root as initial vertex, i.e., if we say that a player has a winning strategy in a game played on a game tree, then we always refer to plays starting at the root.

On game trees there is an alternative way of representing positional strategies by a set U of vertices that contains the root, for each vertex of Adam that is in U also all its successors are in U, and for each vertex of Eva in U there is one of its successors in U. Hence, we can define a positional strategy in MSO using a formula $\psi(X)$ with one free set variable such that there is exactly one set U with $G \models \psi[U]$ and U satisfies the above properties. It is easy to see that these two ways of defining positional strategies are equivalent on game trees:

- Starting from a formula $\phi(x, y)$ we can define $\psi(X)$ to be satisfied by the smallest set U that contains the root, for each vertex $v \in U$ of

Adam also all its successors are in U, and for each vertex $v \in U$ of Eva also the unique vertex v' with $G \models \phi[v, v']$ is in U.

- Given $\psi(X)$ defining a strategy by a set U, we define $\phi(x, y)$ such that $G \models \phi[v, v']$ for $v \in V_\circ$ if v' is the successor of v that is contained in U (if v is not in U, then we can choose an arbitrary successor).

The advantage of representing strategies by sets is that a set corresponds to an annotation of the game tree and we can use automata to accept or reject such annotations.

Let us consider the very simple case of *reachability games*, where the coloring function assigns colors 1 or 2 to the vertices, and Eva wins a play if it contains a vertex of color 2. These games are called reachability games because the goal of Eva is to reach a vertex of color 2. If Eva has a winning strategy in a reachability game tree, then all plays played according to this strategy are decided after a finite time. For this reason strategies are definable in MSO: The formula only has to describe the choices of Eva up to a certain depth, the later choices do not matter.

Remark 3.7 For each reachability game tree G on which Eva has a winning strategy there is a formula $\psi_G(X)$ defining a winning strategy for Eva.

Note that the formula ψ_G depends on G. From Theorem 3.1 we can conclude that there it is not possible to eliminate this dependency.

Theorem 3.8 *There is no formula $\psi(X)$ over reachability game trees such that ψ defines a winning strategy of Eva for all reachability game trees on which Eva has a winning strategy.*

Proof. We only consider game trees for which $V_\Box = \emptyset$. On such game trees Eva has a winning strategy iff there is at least one node with color 2. A winning strategy describes a unique path from the root to such a vertex. Hence, a formula $\psi(X)$ that defines winning strategies of Eva for all reachability game trees on which Eva has a winning strategy can easily be turned into a formula defining a choice function, contradicting Theorem 3.1. □

We now consider a more complex winning condition. Büchi games are the special case of parity games, where only priorities 1 and 2 are used. Eva wins a play if it contains infinitely many vertices of priority 2. We show that for these games an analogue of Remark 3.7 is not possible. For this purpose we now construct a Büchi game tree on which, intuitively, Adam can choose to move to a tree of the form $t[U_{n,n}]$ for some n, and then Eva has to move to an element of $U_{n,n}$ to win (see page 3.1 for the definition of $U_{n,n}$). An MSO-definable winning strategy for Eva in this game would allow us to define a choice function on the sets $U_{n,n}$ contradicting Theorem 3.5.

Formally, we define the Büchi game tree G_{choice} over $V = \{0, 1\}^*$ by

- $V_\Box = \{1^n \mid n \geq 0\}$,
- $V_\circ = V \setminus V_\Box$,
- $P_2 = \bigcup_{n\geq 0}(\{1^n\} \cup (1^n 0 \cdot U_{n,n}))$,
- $P_1 = V \setminus P_2$.

The vertices of Adam are the vertices 1^n on the rightmost branch. All these vertices have priority 2. This forces Adam to move to the left at some point because otherwise he loses. The left subtree below 1^n corresponds to the tree $t[U_{n,n}]$, where the nodes in the set $U_{n,n}$ are labeled 2, and the other nodes are labeled 1.

It is easy to see that Eva has a winning strategy in G_{choice}: Either Adam stays on the rightmost branch, then she wins. If Adam moves to $1^n 0$ for some n Eva moves to a node in $1^n 0 \cdot U_{n,n} \subset P_2$. Once she has reached such a node all the following nodes are also in P_2, hence she wins.

We now show that there is no MSO-definable winning strategy for Eva on this game tree.

THEOREM 3.9 *There is no MSO-formula $\psi(X)$ defining a winning strategy for Eva on the Büchi game tree G_{choice}.*

Proof. Assume that there is a winning strategy on G_{choice} that is MSO-definable by a formula $\psi(X)$.

We can view G_{choice} as a tree $t_{\text{choice}} : \{0,1\}^* \to \Sigma$ where Σ contains letters to describe $V_\circ$, $V_\Box$, P_1, and P_2 (for example $\Sigma = \{0,1\}^4$). According to Theorem 2.1 there a parity automaton $\mathcal{A}_\psi$ equivalent to ψ (we view the predicates $V_\circ$, $V_\Box$, P_1, and P_2 as free variables). This automaton accepts Σ-trees together with an annotation for the interpretation of the variable X.

Since ψ defines a winning strategy on G_{choice}, there is exactly one annotation of t_{choice} that is accepted by $\mathcal{A}_\psi$. We say that $\mathcal{A}_\psi$ accepts exactly one winning strategy for t_{choice}. Let ρ be an accepting run of this winning strategy for t_{choice}.

Using $\mathcal{A}_\psi$ we can construct a formula $\phi(X, x)$ that chooses exactly one $u \in U_{n,n}$ for each $n \in \mathbb{N}$, contradicting Theorem 3.5. For this we fix an arbitrary order on the states of $\mathcal{A}_\psi$. For each n there is at least one state q of $\mathcal{A}_\psi$ such that $\mathcal{A}_\psi$ accepts exactly one winning strategy on the subtree of t_{choice} at node $1^n 0$ (namely the state $\rho(1^n 0)$ assumed at this node in the accepting run fixed above). The formula ϕ picks the smallest state q with this property and then chooses the element of $U_{n,n}$ that is described by the unique winning strategy accepted by $\mathcal{A}_\psi$ from q on the subtree at $1^n 0$. It is not difficult to verify that this is indeed possible in MSO. □

One should note here that the game tree G_{choice} is not too complex.

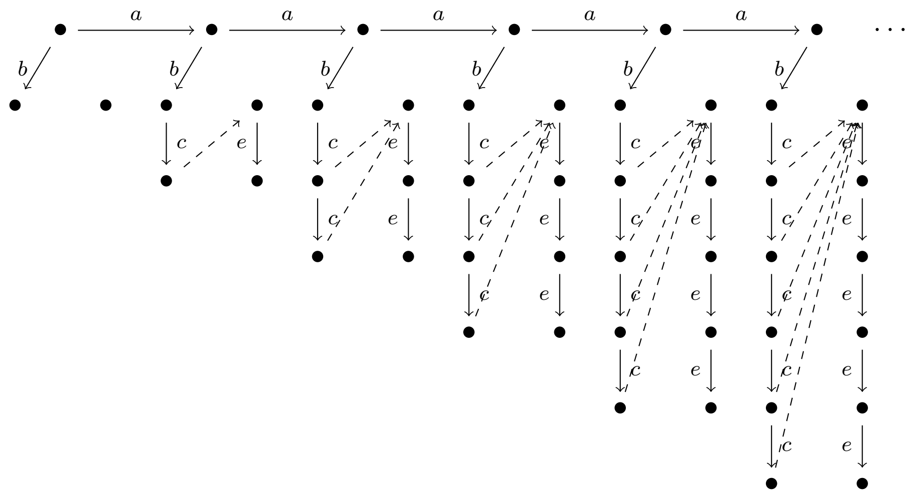


Figure 3.3: MSO-interpretation of the graph from Figure 2.2

REMARK 3.10 The graph G_{choice} belongs to the Caucal hierarchy. In particular it has a decidable MSO-theory.

We briefly show how it can be obtained by using unfoldings and MSO-interpretations. The main difficulty is to obtain the sequences of the form $(0^n1)^n$ for each n. We sketch how to obtain theses chains. The remaining details of the construction are left to the reader.

We start from the graph in Figure 2.2 on page 34, which is in $\mathfrak{C}_2$. By an MSO-interpretation we can add the dashed edges shown in Figure 3.3, delete the old c-edges, and relabel some edges. We assume the dashed edges to be labeled d but omit the labels in the picture. Unfolding this graph yields the graph depicted in Figure 3.4. For readability we only show the part below the vertex number 3 of the a-chain. In general, below vertex number n there is a b-edge followed by n c-edges. Additionally, after each c-edge there is another chain branching off consisting of one d-edge and n e-edges. Below node number n of the a-chain we basically have n chains of length n. By concatenating these chains and adding some more edges we can obtain below node number n of the a-chain the graph representing the set $U_{n,n}$ as shown in Figure 3.1. This can be done by an MSO interpretation. Another unfolding operation yields the desired graph G_{choice}.

3.3 Well-orderings

A *well-ordering* is a total order relation with the property that there are no infinite descending chains or, equivalently, every non-empty subset of the domain of the relation has a smallest element. A simple example is the natural ordering on the natural numbers $\mathbb{N}$. The natural ordering on the

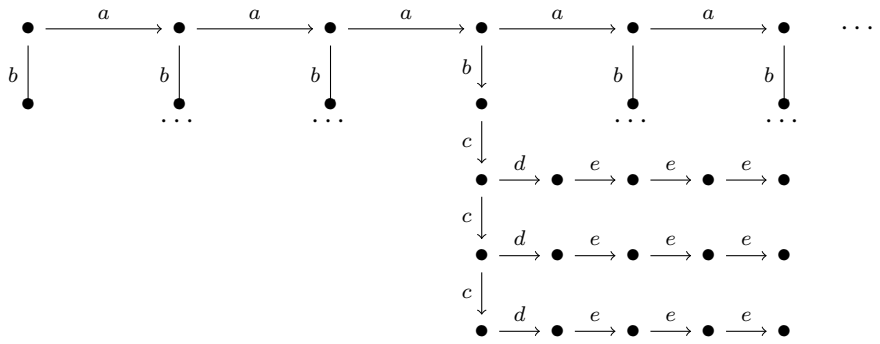


Figure 3.4: Unfolding the graph from Figure 3.3

integers $\mathbb{Z}$ is not a well-ordering because, e.g., the whole set $\mathbb{Z}$ does not have a minimal element w.r.t. to this ordering. If we define an ordering $\preceq$ on $\mathbb{Z}$ by first comparing the absolute values of the numbers and in case of equality letting the negative one be smaller, i.e., $n \preceq m$ iff $|n| < |m|$, or $|n| = |m|$ and $n \leq m$, then we obtain a well-ordering.

In this section we are interested in well-orderings on the tree domain $\{0,1\}^*$. A typical ordering of $\{0,1\}^*$ is the *lexicographic ordering* $\leq_{\text{lex}}$ defined by $u_1 \leq_{\text{lex}} u_2$ if u_1 is a prefix of u_2, or $u_1 = u0u_1'$ and $u_2 = u1u_2'$ for the greatest common prefix u of u_1 and u_2, and some $u_1', u_2' \in \{0,1\}^*$. From the definition of $\leq_{\text{lex}}$ one can easily see that it is MSO-definable in $\mathfrak{t}_2$. But it is not a well-ordering because, e.g., the set of nodes of the form 0^*1 does not have a minimal element.

In a similar way as we changed the standard ordering on $\mathbb{Z}$ into a well-ordering, we can obtain a well-ordering on $\{0,1\}^*$ by first comparing the length of the elements and in case of equality take the lexicographic ordering. We obtain the *length-lexicographic ordering* $\leq_{\text{llex}}$ formally defined by $u_1 \leq_{\text{llex}} u_2$ iff $|u_1| < |u_2|$, or $|u_1| = |u_2|$ and $u_1 \leq_{\text{lex}} u_2$. This defines a well-ordering but $\leq_{\text{llex}}$ is not MSO-definable in $\mathfrak{t}_2$ because it compares the length of the elements.

More generally, as a direct consequence of Theorem 3.1 we obtain that there exists no MSO-definable well-ordering on the nodes of the infinite binary tree. In fact, from an MSO-formula $\phi_{\leq}(x,y)$ defining a well-ordering in $\mathfrak{t}_2$, a choice function

$$\phi_{\text{choice}}(x,X) := x \in X \wedge \forall y \ : \ y \in X \rightarrow \phi_{\leq}(x,y)$$

is easily defined by taking the smallest element of the set, contradicting Theorem 3.1. This naturally rises the question whether there is a well-ordering that we can add to $\mathfrak{t}_2$ while preserving the decidability of MSO. In this section, we prove the following negative result.

THEOREM 3.11 *The MSO-theory of the full-binary tree together with any well-ordering is undecidable.*

In the particular case of t_{llex}, the infinite binary tree with length-lexicographic order, i.e., the structure $t_{\text{llex}} = (\{0,1\}^*, E_0, E_1, \leq_{\text{llex}})$, this result is well-known [11].

We show that t_{llex} can be MSO-interpreted in the infinite binary tree with any well-ordering.

THEOREM 3.12 *There exists an MSO-interpretation $\mathcal{I}$ such that for every well-ordered infinite binary tree t, $\mathcal{I}(t)$ is isomorphic to t_{llex}*

As MSO-interpretations preserve the decidability of MSO (see Proposition 2.5), Theorem 3.11 follows from the undecidability of the MSO-theory of t_{llex}. The rest of this section is dedicated to the proof of Theorem 3.12.

Well-ordered trees

We consider structures over the binary signature $\tau = \{E_0, E_1, \leq\}$. We say that a τ-structure t is a *well-ordered (infinite binary) tree* if it is isomorphic to a well-ordered tree with universe $\{0,1\}^*$ where E_0 and E_1 are respectively interpreted as $\{(u, u0) \mid u \in \{0,1\}^*\}$ and $\{(u, u1) \mid u \in \{0,1\}^*\}$, and $\leq$ is interpreted as a well-ordering on $\{0,1\}^*$. Such a τ-structure is referred to as a *canonical well-ordered tree*. Up to isomorphism, a well-ordered tree is entirely characterized by the well-ordering on the set of words over $\{0,1\}$. Above, we have already defined t_{llex} to be the well-ordered tree with $\leq_{\text{llex}}$ as ordering relation.

The key property of t_{llex} is that it is MSO-definable (up to isomorphism) in the class of well-ordered trees[1].

PROPOSITION 3.13 *There exists an MSO-sentence ϕ_{llex} such that $t \models \phi_{llex}$ iff $t \cong t_{\text{llex}}$ for every well-ordered tree t.*

Proof. Let t be a well-ordered tree with the order relation $\leq$. We describe the properties that we need to express by ϕ_{llex} to ensure that t is isomorphic to t_{llex}. In the description we use the notation $\text{Succ}(u)$ to denote the successor of a node u for the ordering $\leq$.

Now consider an MSO-sentence ϕ_{llex} over τ expressing the following:

(1) The root ε is the smallest element for $\leq$.

(2) For all nodes $u \in \{0,1\}^*$ and $v \in 1^*$, $\text{Succ}(u0v)$ is the smallest element for the prefix order of the set $u10^* \setminus \{\text{Succ}(u0v') \mid v' \sqsubset v\}$. This is illustrated in Figure 3.5 by the dashed lines for $u = \varepsilon$.

[1]The class of well-ordered trees is itself MSO-definable in the class of τ-structures.

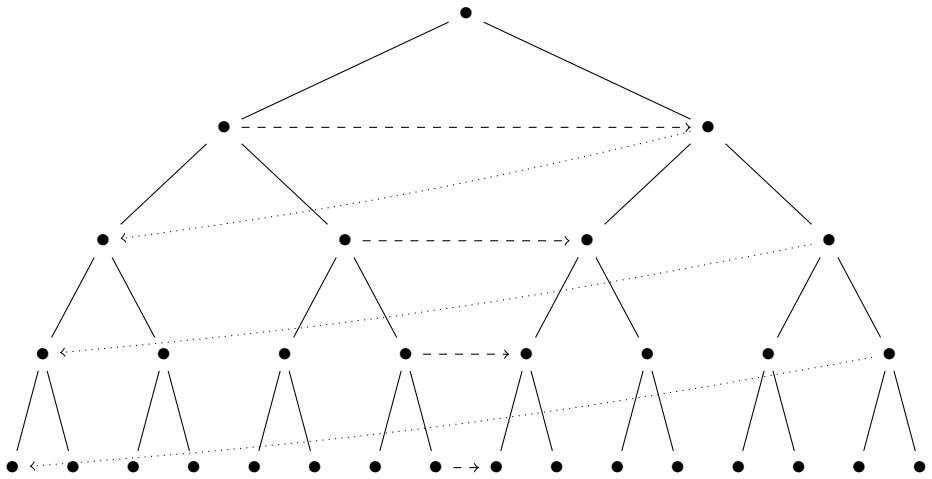


Figure 3.5: The successor relation on nodes of the form $u01^*$ and $u10^*$

(3) For all nodes $v \in 1^*$, the successor of v is the smallest element for the prefix order of the set $0^+ \setminus \{\mathrm{Succ}(v') \mid v' \sqsubset v\}$. This is illustrated in Figure 3.5 by the dotted lines.

These properties can be defined in MSO. It is easy to see that t_{llex} satisfies the formula ϕ_{llex}. It remains to show that for every well-ordered tree t satisfying ϕ_{llex} is isomorphic to t_{llex}.

Let t be a well-ordered tree satisfying ϕ_{llex}. We can assume w.l.o.g that t is a canonical well-ordered tree. It is therefore enough to show that $t = t_{\mathrm{llex}}$.

For all nodes $u \in \{0,1\}$, we write $\mathrm{Succ}_{\mathrm{llex}}(u)$ for the successor of u for the order $\leq_{\mathrm{llex}}$. As by condition (1) the root ε of t is the minimal element of $\leq$, it is enough to establish that for all $u \in \{0,1\}^*$, $\mathrm{Succ}(u) = \mathrm{Succ}_{\mathrm{llex}}(u)$.

Assume by contradiction that this property is not satisfied. Let u_0 be the smallest node u for the order $\leq_{\mathrm{llex}}$ such that $\mathrm{Succ}(u) \neq \mathrm{Succ}_{\mathrm{llex}}(u)$. We distinguish two cases depending whether u_0 contains an occurrence of 0 or not.

If u_0 contains an occurrence of 0 then u_0 can be uniquely written as $u0v$ with $u \in \{0,1\}^*$ and $v \in 1^*$. The successor of u_0 in the order $\leq_{\mathrm{llex}}$ is $\mathrm{Succ}_{\mathrm{llex}}(u_0) = u10^{|v|}$. By condition (2) of the definition of ϕ_{llex}, $\mathrm{Succ}(u_0)$ is the smallest element for the prefix order of the set $u10^* \setminus \{\mathrm{Succ}(u0v') \mid v' \sqsubset v\}$. By minimality (for the order $\leq_{\mathrm{llex}}$) of u_0, we have for all $v' \sqsubset v$ that $\mathrm{Succ}(u0v') = \mathrm{Succ}_{\mathrm{llex}}(u0v') = u10^{|v'|}$. Therefore, $\mathrm{Succ}(u_0)$ is the minimal element for the prefix order of the set $u10^* \setminus \{u10^{|v'|} \mid v' \sqsubset v\}$. This implies that $\mathrm{Succ}(u_0) = u10^{|v|} = \mathrm{Succ}_{\mathrm{llex}}(u_0)$ which contradicts the definition of u_0.

If u_0 does not contain an occurrence of 0 then $u_0 \in 1^*$. The successor of u_0 for the order $\leq_{\mathrm{llex}}$ is $0^{|u_0|+1}$. By condition 3 of the definition of ϕ_{llex}, $\mathrm{Succ}(u_0)$ is the smallest element for the prefix order of the set

$0^+ \setminus \{\mathrm{Succ}(u) \mid u \sqsubset u_0\}$. By minimality (for the order $\leq_{\mathrm{llex}}$) of u_0, we have for all $u \sqsubset u_0$ that $\mathrm{Succ}(u) = \mathrm{Succ}_{\mathrm{llex}}(u) = 0^{|u|+1}$. Therefore, $\mathrm{Succ}(u_0)$ is the minimal element for the prefix order of the set $0^+ \setminus \{0^{|u|+1} \mid u \sqsubset u_0\}$. This implies that $\mathrm{Succ}(u_0) = 0^{|u_0|+1} = \mathrm{Succ}_{\mathrm{llex}}(u_0)$ which contradicts the choice of u_0. □

Interpreting t_{llex}

We now define the notion of *induced well-ordered tree*. Consider a canonical well-ordered tree t and a set $U \subseteq \{0,1\}^*$ of nodes which

- is closed under greatest common prefix ($u \in U \wedge v \in V \rightarrow u \wedge v \in U$),
- has the property that for all $u \in U$, $u0\{0,1\}^* \cap U \neq \emptyset$ and $u1\{0,1\}^* \cap U \neq \emptyset$, i.e., from every node we can go to the left and to the right and find another element of U below.

$t|_U$ induced by U in t has universe U and its relations are interpreted as

$$\begin{array}{lcl} E_0^{t|_U} & = & \{(u,v) \in U \times U \mid v \text{ is the } \sqsubseteq\text{-smallest element of } u0\{0,1\}^* \cap U\} \\ E_1^{t|_U} & = & \{(u,v) \in U \times \ \mid v \text{ is the } \sqsubseteq\text{-smallest element of } u1\{0,1\}^* \cap U\} \\ \leq^{t|_U} & = & \{(u,v) \in U \times U \mid u \leq^t v\}. \end{array}$$

It is easy to check that $t|_U$ is a well-ordered tree. The plan is to show that in each well-ordered tree we can find a subset U inducing a well-ordered tree that is isomorphic to t_{llex}. As a first step we show that we can express each MSO-property ϕ of $t|_U$ by an MSO-formula ϕ^* on t that takes U as a parameter.

Lemma 3.14 *For every MSO-formula ϕ over τ there exists a formula $\phi^*(X)$ such that for every canonical well-ordered tree t and set U, $t \models \phi^*[U]$ iff the set U induces a well-ordered tree $t|_U$ on t and $t|_U \models \phi$.*

Proof. Consider an MSO-formula ϕ over τ. Let $\phi_{\mathrm{ind}}(X)$ be an τ-formula expressing that X satisfies the conditions to induce a full binary tree and let $\phi'(X)$ be the formula obtained from ϕ by relativizing the quantifications to X and by replacing $E_i(x,y)$ with $y \in xi\{0,1\}^* \cap X \wedge \forall z, z \in xi\{0,1\}^* \cap X \rightarrow y \sqsubseteq z$ for $i \in \{0,1\}$. It is easy to check that the formula $\phi^*(X) := \phi_{\mathrm{ind}}(X) \wedge \phi'(X)$ satisfies the property stated in the lemma. □

Applying this lemma to the formula ϕ_{llex} from Proposition 3.13 yields that $t \models \phi^*_{\mathrm{llex}}[U]$ iff U induces a well-ordered tree isomorphic to t_{llex}.

We now show that for every canonical well-ordered tree t there exists a subset $U \subseteq \{0,1\}^*$ such that $t|_U$ is isomorphic to t_{llex}. Later we refine this result and show that there is in fact an MSO-definable such set (Lemma 3.17).

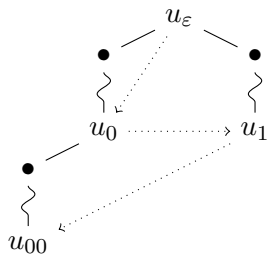


Figure 3.6: Construction of a set inducing t_{llex}

To construct such a set U we built up a sequence of nodes indexed by the elements from $\{0,1\}^*$. We start with a node u_ε representing the root of $t|_U$. Then we define u_0 to be a node in the left subtree of u_ε such that $u_\varepsilon < u_0$. We continue by finding a node u_1 in the right subtree of u_ε such that $u_0 < u_1$. Then we take a node u_{00} in the left subtree of u_0 such that $u_1 < u_{00}$, and so on. This construction is illustrated in Figure 3.6, where the dashed arrows represent the order relation $<$ on the sequence $u_\varepsilon, u_0, u_1, u_{00}, \ldots$ by always pointing to the next bigger element from this sequence.

For this construction to work we need to ensure that we always can find nodes as required, e.g., that there is a node u_{00} in the left subtree of u_0 such that $u_1 < u_{00}$. For this purpose we make the definition of a mixed node and later choose u_ε to be a node with this property. Intuitively, a node u is mixed if there are no node that is bigger than all nodes in some subtree below u, i.e., the order is completely mixed below u.

Formally, a node $u \in \{0,1\}^*$ of a canonical well-ordered tree t is *mixed* if for all $v \sqsupseteq u, v' \sqsupseteq u \in \{0,1\}^*$ there exists a $w \in \{0,1\}^*$ such that $v < v'w$. We need to show that we can indeed find a node with this property.

Lemma 3.15 *Every canonical well-ordered tree t contains a mixed node.*

Proof. Let t be a canonical well-ordered tree. Assume by contradiction that t does not have any mixed nodes. We construct by induction two sequences of nodes $(u_i)_{i\in\mathbb{N}}$ and $(v_i)_{i\in\mathbb{N}}$ such that $u_i \geq v_i w$ for all $w \in \{0,1\}^*$, and $v_i \sqsubseteq u_{i+1}$. This implies that the sequence $(u_i)_{i\in\mathbb{N}}$ is an infinite strictly decreasing sequence which contradicts the fact that $\leq$ is a well-ordering.

As ε is not mixed there exist two nodes u and v such that $u \geq vw$ for all $w \in \{0,1\}^*$. We take $u_0 = u$. It might happen that v is a prefix of u_0. Hence, to ensure that $u_0 > v_0 w$ for all $w \in \{0,1\}^*$, we pick as v_0 an element such that $v \sqsubseteq v_0 \not\sqsubseteq u_0$.

Assuming that both sequences are constructed up to rank $i \geq 0$ we define u_{i+1} and v_{i+1}. As v_i is not mixed, there exists two nodes $u \sqsupseteq v_i$ and $v \sqsupseteq v_i$ such that $u \geq vw$ for all $w \in \{0,1\}^*$. We take u_{i+1} equal to u and v_{i+1} an

element such that $v \sqsubseteq v_{i+1} \not\sqsubseteq u_{i+1}$ thus ensuring that for all $w \in \{0,1\}^*$, $u_{i+1} > v_{i+1}w$. By construction we obtain that $v_i \sqsubseteq u_{i+1}$. □

Now we can formalize the construction of the set U inducing t_{llex} as indicated above and in Figure 3.6.

LEMMA 3.16 *For every canonical well-ordered tree t, there exists a set of nodes U inducing a well-ordered tree $t|_U$ isomorphic to t_{llex}.*

Proof. Let t be a canonical well-ordered tree. We construct a sequence of nodes $(u_w)_{w\in\{0,1\}^*}$ indexed by the set of words over $\{0,1\}^*$ such that:

- for all $w, w' \in \{0,1\}^*$, $w \leq_{\text{llex}} w'$ implies $u_w \leq u_{w'}$,
- and for all $w \in \{0,1\}^*$ and $i \in \{0,1\}$, $u_{wi} \in u_w i\{0,1\}^*$.

If we assume that this sequence has been constructed and we take $U := \{u_w \mid w \in \{0,1\}^*\}$, it is easy to check that U is closed by greatest common prefix and hence U induces a full binary tree on t. Furthermore the mapping from $\{0,1\}^*$ to U associating w to u_w is an isomorphism from t_{llex} to $t|_U$.

We now construct the sequence $(u_w)_{w\in\{0,1\}^*}$ by induction on the length-lexicographic order $\leq_{\text{llex}}$. By Lemma 3.15, the tree t has a mixed node. We take u_ε to be a mixed node of t. Assume that the sequence has been constructed up to $w \in \{0,1\}^*$. We construct the element u_v where v is the successor of w for the length-lexicographic order. Since $v \neq \varepsilon$ we can pick its parent node v', i.e., $v = v'i$ for some $i \in \{0,1\}$. The definition of $\leq_{\text{llex}}$ implies that $v' \leq_{\text{llex}} w$ and hence $u_{v'}$ is already defined. As u_ε is mixed, there exists a z such that $u_w < u_{v'}iz$. We take $u_v = u_{v'}iz$ and obtain that $u_w < u_v$. □

Note that we can now already derive Theorem 3.11: For every formula ϕ over τ, consider the formula $\phi^*(X)$ obtained from ϕ by Lemma 3.14 and $\phi^*_{\text{llex}}(X)$ obtained from the formula ϕ_{llex} of Proposition 3.13. By Lemma 3.16, for every well-ordered tree t,

$$t \models \exists X \ : \ \phi^*_{\text{llex}}(X) \wedge \phi^*(X)$$

iff $t_{\text{llex}} \models \phi$. As the formula $\phi^*(X)$ can be effectively constructed from the formula ϕ, it follows that the MSO-theory of t_{llex} is recursive in the MSO-theory of any well-ordered tree t.

We now strengthen the result of Lemma 3.16 by showing that in every well-ordered tree t there exists an MSO-definable set of nodes inducing a well-ordered tree isomorphic to t_{llex}.

LEMMA 3.17 *For every canonical well-ordered tree, there exists an MSO-definable set of nodes U_0 inducing a well-ordered tree isomorphic t_{llex}.*

Proof. Consider the following MSO-formula $\psi(X)$ defined by:

$$\phi^*_{\text{llex}}(X) \ \wedge \ \forall x \in X \ \forall Z \ : \ (X_{<x} < Z \wedge \phi^*_{\text{llex}}(X_{<x} \cup Z)) \ \rightarrow \ x \leq \min Z$$

where

- $X_{<x} = \{x' \in X \mid x' < x\}$,
- $X < Y$ stands for $\forall x \in X \ \forall y \in Y \ : \ x < y$, and
- $\min Z$ designates the smallest element of the set Z for the order $\leq$.

Let t be a well-ordered tree. We claim that $t \models \exists^{=1} X : \psi(X)$ (where $\exists^{=1}$ stands for there exists a unique), which establishes the MSO-definability of a set of nodes U_0 inducing on t a well-ordered tree isomorphic to t_{llex}.

The first step is to show that $t \models \exists X : \psi(X)$. For this we define a sequence of nodes $(u_w)_{w \in \{0,1\}^*}$ of t by induction on the order $\leq_{\text{llex}}$. The node u_ε is the smallest element of $\{\min Z \mid Z \subseteq \{0,1\}^* \wedge t \models \phi^*_{\text{llex}}[Z]\}$. Lemma 3.16 guarantees that this set is not empty. Assume that the sequence has been constructed up to the element u_{w_0}. We define u_{w_1} where w_1 is the successor of w_0 for the order $\leq_{\text{llex}}$. We take u_{w_1} as the smallest element for $\leq$ of the set $\{\min Z \mid w_0 < Z \wedge \phi^*_{\text{llex}}(\{u_w \mid w \leq w_0\} \cup Z)\}$. This set is not empty by definition of u_{w_0}.

Consider the set $U_0 := \{u_w \mid w \in \{0,1\}^*\}$. It is straightforward to show that U_0 induces a well-ordered tree on t isomorphic to t_{llex}. It follows that $t \models \psi[U_0]$. Hence $t \models \exists X : \psi(X)$.

We now show that $t \models \exists^{=1} X : \psi(X)$. Let U, U' be two sets with $t \models \psi[U]$ and $t \models \psi[U']$. We show by induction on n that the first n elements of U and U' w.r.t. $\leq$ are the same. The induction base for $n = 0$ is clear. Assume that the $n-1$ first elements of U and U' are the same, and let $u \in U$ and $u' \in U'$ be the nth element of U and U', respectively. From the induction hypothesis we obtain that $U_{<u} = U'_{<u'}$ and therefore we can write the sets U and U' as

$$U = U'_{<u'} \cup U_{\geq u} \text{ and } U' = U_{<u} \cup U'_{\geq u'} \ .$$

From the conditions in ψ we conclude that $u' \leq \min U_{\geq u} = u$ and $u \leq \min U'_{\geq u'} = u'$. Hence, $u = u'$. □

Using the previous results it is easy to prove Theorem 3.12.

Proof(of Theorem 3.12). The interpretation $\mathcal{I} = (\phi_{\text{dom}}, \phi_{E_0}, \phi_{E_1}, \phi_\leq)$ such that $\mathcal{I}(t) \cong t_{\text{llex}}$ for each well-ordered tree t is defined as follows. As domain formula ϕ_{dom} we take the formula ψ defining the set U_0 from Lemma 3.17. The formulas ϕ_{E_0}, ϕ_{E_1}, and $\phi_\leq$ just define the induced successor relations and the induced ordering from the definition of induced well-ordered tree. □

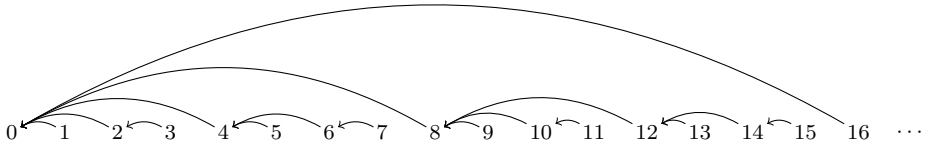


Figure 3.7: Graph of the flip function

An immediate consequence of this result is that the infinite binary tree cannot be MSO-interpreted in $\mathfrak{t}_1$ (the natural numbers with successor). Based on this we can use the same technique as in Corollary 3.6 to obtain the same result for $\mathfrak{t}_1$ extended with unary predicates.

PROPOSITION 3.18 *Let $P_1, \ldots, P_n$ be unary predicates for $\mathfrak{t}_1$. There is no interpretation $\mathcal{I}$ such that $\mathcal{I}((\mathfrak{t}_1, P_1, \ldots, P_n)) \cong \mathfrak{t}_2$.*

Proof. As already mentioned there is no interpretation $\mathcal{I}$ such that $\mathcal{I}(\mathfrak{t}_1) \cong \mathfrak{t}_2$. Otherwise we could extend this interpretation by a formula transferring the ordering on $\mathfrak{t}_1$ to $\mathfrak{t}_2$, thus obtaining a well-ordered tree with decidable MSO-theory (contradicting Theorem 3.11). Now assume that there is an interpretation $\mathcal{I}$ such that $\mathcal{I}((\mathfrak{t}_1, P_1, \ldots, P_n)) \cong \mathfrak{t}_2$. We first construct a formula $\phi_{\text{tree}}(X_1, \ldots, X_n)$ such that

$$\mathfrak{t}_1 \models \phi_{\text{tree}}[U_1, \ldots, U_n] \text{ iff } \mathcal{I}((\mathfrak{t}_1, U_1, \ldots, U_n)) \cong \mathfrak{t}_2.$$

Such a formula only needs to express that the formulas from the interpretation describe a structure in which all nodes have exactly one E_0 successor, exactly one E_1 successor, and exactly one predecessor except for one node which is the root. This can easily be done by an MSO-formula.

The formula $\exists X_1, \ldots, X_n : \phi_{\text{tree}}(X_1, \ldots, X_n)$ is satisfiable (interpreting $X_1, \ldots, X_n$ by $P_1, \ldots, P_n$). Hence, there are regular interpretations $U_1, \ldots, U_n$ of $X_1, \ldots, X_n$ such that $\mathfrak{t}_1 \models \phi_{\text{tree}}[U_1, \ldots, U_n]$ (see Theorem 2.3). By construction of ϕ_{tree} this means that $\mathcal{I}((\mathfrak{t}_1, U_1, \ldots, U_n)) \cong \mathfrak{t}_2$. Since regular sets can be defined in MSO we can directly refer to $U_1, \ldots, U_n$ in the interpretation and obtain in interpretation $\mathcal{I}'$ such that $\mathcal{I}'(\mathfrak{t}_1) = \mathfrak{t}_2$, which is not possible. □

In general, Theorem 3.11 implies that $\mathfrak{t}_2$ is not MSO-interpretable in any structure that has a decidable MSO-theory and admits an MSO-definable well-ordering.

Consider, for example, the function *flip* on the natural numbers. It maps a natural number n to the number that is obtained by changing the least significant bit that is 1 in the binary representation of n to 0. The graph of the *flip* function is shown in Figure 3.7. The structure $(\mathfrak{t}_1, \mathit{flip})$ of the

natural numbers with successor extended by the *flip* function has a decidable MSO-theory [82]. We conclude that $\mathfrak{t}_2$ cannot be MSO-interpretable in this structure because otherwise we could also interpret a well-ordered tree in it.

PROPOSITION 3.19 *There is no MSO-interpretation $\mathcal{I}$ with $\mathcal{I}(\mathfrak{t}_1, \mathit{flip}) \cong \mathfrak{t}_2$.*

3.4 Discussion

In this chapter we have studied questions on MSO-definability in the binary tree and on decidability of MSO in extensions of the binary tree by well-orderings. As a result we obtain a rather simple and elementary proof of the theorem of Gurevich and Shelah stating that there is no MSO-definable choice function on the infinite binary tree. A simple consequence is that there is also no MSO-definable well-ordering on the domain of the infinite binary tree. We obtain the even stronger result that adding any well-ordering to the infinite binary tree yields a structure with undecidable MSO-theory. These two main results from this chapter can be used to derive non-definability results for MSO as we have illustrated with some examples.

One natural question that remains open is whether there exists a choice function that can be added to the infinite binary tree such that the resulting structure has a decidable MSO-theory.

As mentioned at the beginning of this chapter, MSO-definability of a choice function is a very special instance of the uniformization problem, namely for the formula $\phi(X, y) = y \in X$. The result from Section 3.1 shows that uniformization is not possible in general on the infinite binary tree. This leaves the question whether there are other types of formulas that allow uniformization. In [111] it is mentioned that uniformization is possible for formulas of the form $\phi(x, Y)$. Here, uniformization means that the relation between elements and sets defined by $\phi(x, Y)$ can be turned into an MSO-definable function associating to each element exactly one set from the relation.[2]

Another type of question related to this is the one of decidability. In the introduction of this thesis we have mentioned Church's synthesis problem. An instance of this problem is given by an MSO-formula $\phi(\bar{X}, \bar{Y})$ over the infinite line $\mathfrak{t}_1$ such that for each input sequence $\bar{X}$ there is at least one output sequence $\bar{Y}$. A solution is a very specific function compatible with this relation: It should be implementable by a finite state automaton that

[2]Such a result can be shown by transforming the formula into an equivalent automaton and then constructing a formula that selects for each x a unique run that accepts the tree annotated with x and some set Y. For this purpose it is enough to select a (say lexicographically) smallest run on the path to the element x that can be completed to an accepting run. In this finite part of the run we can plug for each 'dangling' state a fixed MSO-definable run. The details of this construction are left to the reader.

reads the input sequence at produces in each step one element of the output sequence. The task is now to decide if such an automaton exists (and to construct one if possible). Similarly, one can study the decision variant of uniformization on the binary tree: Given an MSO-formula $\phi(\bar{X}, \bar{Y})$ over $\mathfrak{t}_2$, does there exist a formula $\phi^*(\bar{X}, \bar{Y})$ that defines a function compatible with $\phi(\bar{X}, \bar{Y})$. In its full generality this question is certainly too difficult but one could study specific instances of it for simple classes of formulas.

Chapter 4

Unambiguous and guidable tree automata

Normal forms and canonical representations play an important role in automata theory. For regular languages of finite words various such representations are known. The most prominent ones are the minimal deterministic automaton (see [55]) and the syntactic monoid (see [96]). Another canonical object associated to a regular language is the universal automaton (see [77]), which is a kind of most general nondeterministic automaton accepting the language. Besides their use for algorithmic purposes, canonical representations have the advantage that they do not exhibit a strange behavior but represent the language in a reasonable and natural way. Consider, for example, the language of all finite words over some alphabet. One way to represent it is to use a nondeterministic automaton that guesses the last letter of the input word and then checks if this guess is correct. This automaton is certainly a representation of the language of all words but it is a rather unnatural way of representing it.

The absence of canonical representations makes it difficult to work with the objects under consideration because one has to deal with all kinds of strange or unnatural representations, as illustrated above. Certainly, asking for canonical representations is a very strong requirement and one can relax this by simply looking for representations with good properties excluding strange behaviors. For example, when considering regular languages of infinite words, we do not know any canonical form of automaton. But we know at least that each such language can be accepted by a deterministic parity automaton (see Theorem 1.2), and in many situations it is much more convenient to work with deterministic automata than with nondeterministic ones.

For languages of infinite trees one can easily observe that there is no reasonable notion of deterministic automaton that would capture all the regular languages (see Section 4.1 below). Hence, we study in this chapter two other

possible concepts: unambiguous and guidable automata. Unambiguous automata are nondeterministic automata that have at most one accepting run for each input. We show that there are languages that are inherently ambiguous. Guidable automata are, intuitively, automata that can simulate all other automata for the same language. This is loosely related to the concept of universal automaton mentioned above. We prove in Section 4.3 that each regular language of infinite trees can be accepted by a guidable automaton.

The results in this chapter are taken from joint work with Arnaud Carayol [21] and from joint work with Thomas Colcombet [32].

4.1 Determinism

For tree automata there are two different notions of determinism: top-down and bottom-up. Recall that an automaton is *top-down deterministic* if for each state q and each input letter a there is at most one transition (q, a, q_0, q_1) (see page 26). It is very easy to see that not every regular language $T \subseteq \mathcal{T}_\Sigma^\omega$ can be accepted by a top-down deterministic tree automaton.

For the bottom-up view the situation is different: It is required that for all states q_0, q_1 and all letters a there is at most one q such that (q, a, q_0, q_1) is a transition of the automaton. This condition only ensures that, once we fix the states at the successors of a node the state at this node is uniquely determined. We call such automata *bottom-up deterministic*. Note that bottom-up determinism does not ensure a unique run. In [83] Mostowski gives a construction that yields bottom-up deterministic automata.

THEOREM 4.1 ([83]) *Every regular language of infinite trees can be accepted by a bottom-up deterministic automaton.*

Proof. The proof is based on the fact that we are working on binary trees (no unary branching). Starting from a PTA $\mathcal{A} = (Q, \Sigma, q^{\mathrm{i}}, \Delta, c)$ we use pairs of states, where the first component codes a run of $\mathcal{A}$, and the second component codes the future of the run (in the bottom-up view). This second component ensures the bottom-up determinism.

Formally, let $\mathcal{A}'$ be defined by the following components:

- The state set is $Q' = Q \times Q$.
- The transitions in Δ' are $((q, p), a, (q_0, q), (q_1, p))$ for all $q, p, q_0, q_1 \in Q$ such that $(q, a, q_0, q_1) \in \Delta$.
- The priority function is defined by $c'(q, p) = c(q)$.

The automaton $\mathcal{A}'$ is obviously bottom-up deterministic because the state (q, p) is uniquely determined by the states $(q_0, q), (q_1, p)$. Furthermore, each run of $\mathcal{A}'$ projected to the first components of the states is a run of $\mathcal{A}$. Hence

$T(\mathcal{A}') \subseteq T(\mathcal{A})$. Vice versa, each run ρ of $\mathcal{A}$ can be turned into a run ρ' of $\mathcal{A}'$ by letting $\rho'(\varepsilon) = (\rho(\varepsilon), p)$ for some $p \in Q$, and if $\rho'(u) = (q, p)$, then we let $\rho'(u0) = (\rho(u0), q)$ and $\rho'(u1) = (\rho(u1), p)$.

This defines a run of $\mathcal{A}'$ such that the projection to the first component yields ρ. Hence $T(\mathcal{A}) \subseteq T(\mathcal{A}')$. □

The above proof shows that the property is not very helpful. The construction yields a bottom-up deterministic automaton that has even more behaviors than the given automaton.

Nevertheless, on finite trees a similar technique has been used in [100] to show that all regular languages can be captured by anti-chain logic or star-free expressions.

4.2 Unambiguous automata

In the previous section we have seen that top-down deterministic automata are not enough to accept all regular languages and that bottom-up deterministic automata do not have the desired properties on infinite trees because they can have several runs for each input. The property of unambiguity requires accepting runs to be unique for each input, i.e., an automaton is called unambiguous if for each object that it accepts there is exactly one accepting run.

Unambiguous automata have been introduced on finite words because they allow more efficient algorithms for equivalence and inclusion testing [113]. This has been generalized to automata on finite trees in [110]. On finite words and trees it is clear that every regular language can be accepted by an unambiguous automaton because deterministic automata are unambiguous.

For Büchi automata on infinite words the situation is more difficult. Although determinization is possible (see Theorem 1.2), it requires the model of deterministic parity (or Muller) automata with acceptance conditions that are more powerful than Büchi conditions. However, from the determinization result one can infer that all regular languages of infinite words can be accepted by an unambiguous Büchi automaton [4]. In [59] a construction for unambiguous automata is presented that does not rely on deterministic automata.

In this section we show that the situation is different for infinite trees: There are regular languages that cannot be accepted by an unambiguous automaton. The proof is based on the main result from the previous chapter: the undefinability of a choice function in MSO. The underlying idea (taken from [92]) is rather simple: We consider the language

$$T_{\exists 1} = \{t \in \mathcal{T}^{\omega}_{\{0,1\}} \mid \exists u \in \{0,1\}^* \ : \ t(u) = 1\}$$

of trees with at least one node labeled 1. Each tree in $T_{\exists 1}$ can be viewed as the set of nodes labeled 1. An automaton accepting $T_{\exists 1}$ has to verify that the input tree contains a 1, i.e., an accepting run has to identify some position labeled by 1. If the automaton is unambiguous, then there is exactly one run for each tree in $T_{\exists 1}$, and hence the automaton identifies exactly one position from the set coded by the 1-positions. This can be turned into a formula defining a choice function.

We show the following more technical result because it can be used to prove different statements on the ambiguity of automata.

LEMMA 4.2 *Let $\mathcal{A}$ be a parity tree automaton over the alphabet $\{0,1\}$ that does not accept the tree that is completely labeled by 0. Then there is a formula $\psi_{\mathcal{A}}(X,x)$ such that for each $U \subseteq \{0,1\}^*$ for which there is a unique accepting run of $\mathcal{A}$ on $t[U]$, there is a unique element $u \in U$ such that $t[U,u] \models \psi_{\mathcal{A}}$.*

Proof. We consider the following game which can be seen as the emptiness game for the automaton $\mathcal{A}$ intersected with the trivial automaton that accepts only the tree $t[\emptyset]$ (i.e., the tree labeled 0 everywhere). This intersection corresponds to removing all transitions from the game that use a letter different from 0, i.e., all transitions for 1. Then the game is as follows:

- The initial position is the initial state of $\mathcal{A}$.
- From position q Eva chooses a transition $(q,0,q_0,q_1)$ of $\mathcal{A}$.
- From position $(q,0,q_0,q_1)$ Adam chooses a state q_0 or q_1.

Eva wins a play if she can always choose a transition and if the resulting sequence of states chosen by Adam satisfies the acceptance condition of $\mathcal{A}$.

As already mentioned, this game corresponds to the emptiness game for $T(\mathcal{A}) \cap \{t[\emptyset]\}$ (see Section 1.3). By the assumption on $\mathcal{A}$ we have that $t[\emptyset] \notin T(\mathcal{A})$ and therefore Adam has a winning strategy. Since the winning condition is a parity condition, Adam even has a positional winning strategy f that picks for each transition of the form $(q,0,q_0,q_1)$ one of the states q_0, q_1.

We now construct the formula $\psi_{\mathcal{A}}(X,x)$ based on this strategy f. Evaluated on $t[U,u]$ the formula states that there exists an accepting run of $\mathcal{A}$ on $t[U]$ such that the path leading to node u corresponds to the choices of Adam according to the strategy f. Note that if we apply the strategy of Adam to the transitions in an accepting run, then this results in a path ending in a node labeled 1 because otherwise the resulting infinite play would be winning for Eva.

Assuming that the states of $\mathcal{A}$ are $\{1,\dots,n\}$ the formula $\psi_{\mathcal{A}}(X,x)$ looks as follows:

$$\begin{aligned} \exists X_1,\dots,X_n: \quad & \mathit{AccRun}(X_1,\dots,X_n,X) \wedge X(x) \wedge \\ & \forall y \sqsubset x : \mathit{strat}_f(X,x,X_1,\dots,X_n,y) \end{aligned}$$

where

- the formula *AccRun* expresses that $X_1, \ldots, X_n$ describe an accepting run of $\mathcal{A}$ on the characteristic tree of X, and
- $strat_f$ states that the strategy f applied at node y in the run coded by $X_1, \ldots, X_n$ moves into the direction of x. In case the strategy allows to move to both directions (when the states q_0 and q_1 are the same), the formula picks the left move:

$$\neg X(y) \wedge \bigwedge_{q,q_0,q_1 \in \{1,\ldots,n\}} \quad \begin{array}{l} X_q(y) \wedge X_{q_0}(y0) \wedge X_{q_1}(y1) \rightarrow \\ \big[f(q,0,q_0,q_1) = q_0 \leftrightarrow (y0 \sqsubseteq x)\big] \end{array}$$

If we fix the interpretations for X and $X_1, \ldots, X_n$, then there is exactly one interpretation of x such that $\forall y \sqsubset x.strat_f(X, x, X_1, \ldots, X_n, y)$ is satisfied (if there are two positions, then we obtain a contradiction when interpreting y as the greatest common ancestor of these two positions).

Now assume that there is exactly one accepting run of $\mathcal{A}$ on $t[U]$. Then the interpretations of $X_1, \ldots, X_n$ are fixed by U and hence the formula $\psi_{\mathcal{A}}(X, x)$ has the claimed property. □

Using this lemma it is easy to obtain the following theorem.

THEOREM 4.3 ([92]) *There is no unambiguous parity tree automaton accepting the language $T_{\exists 1}$ consisting of exactly those $\{0,1\}$-labeled trees in which at least one node is labeled* 1.

Proof. Assume $\mathcal{A}$ is an unambiguous automaton for $T_{\exists 1}$ and consider the formula $\psi_{\mathcal{A}}(X, x)$ from Lemma 4.2. As there is a unique accepting run of $\mathcal{A}$ on $t[U]$ for each non-empty set U, $\psi_{\mathcal{A}}(X, x)$ defines a choice function. This contradicts Theorem 3.1. □

Using Lemma 4.2 we can also show that there are regular languages whose ambiguity is witnessed by a single tree. The definition of this tree is similar to the definition of the game arena G_{choice} in Section 3.2.

THEOREM 4.4 ([21]) *There is a regular language $T \subseteq \mathcal{T}^{\omega}_{\{0,1\}}$ and a tree $t \in T$ such that there is no parity automaton accepting T that has a unique accepting run for t.*

Proof. Consider the language T of trees with the property that each subtree rooted at a node of the form 1^*0 contains a node labeled 1. Obviously this is a regular language.

The tree t is defined to have all nodes of the form 1^* labeled 0, and for each n we plug the tree $t_{n,n}$ as the subtree rooted at the node 1^n0 (see

page 48 for the definition of $t_{n,n}$). As each $t_{n,n}$ contains a node labeled 1, we have $t \in T$.

Assume that there is parity automaton $\mathcal{A}$ accepting T that has a unique run on t. Let q be a state of $\mathcal{A}$ that occurs at infinitely many nodes of the form 1^*0 in this run. Let $\mathcal{A}'$ be the automaton $\mathcal{A}$ with initial state q. As $\mathcal{A}$ accepts T it is clear that $\mathcal{A}'$ does not accept the tree $t[\emptyset]$. Furthermore, as the run of $\mathcal{A}$ on t is unique, there are infinitely many n such that $\mathcal{A}'$ has a unique run on $t_{n,n}$. For the formula $\psi_{\mathcal{A}'}(X, x)$ from Lemma 4.2 this yields a contradiction to Theorem 3.5. □

4.3 Guidable automata

Besides minimal deterministic automata, there is another canonical automaton associated to each regular language of finite words: the universal automaton (see [77] for an overview). The universal automaton of a language is a nondeterministic automaton that contains a morphic image of each nondeterministic automaton for the same language. In this section we introduce the concept of guidable automata on infinite trees. This concept is different from the universal automaton because guidable automata are not unique. But nevertheless there is a similarity because a guidable automaton is one that can simulate each automaton accepting the same language. Simulating another automaton on some input can also be seen as the other automaton acting as a guide (and therefore the name guidable). Intuitively, a PTA $\mathcal{B}$ can guide a PTA $\mathcal{A}$ if for a given accepting run of $\mathcal{B}$ on some input tree t we can deterministically construct an accepting run of $\mathcal{A}$ on t.

Formally, we say that a parity automaton $\mathcal{A} = (Q_\mathcal{A}, \Sigma, q^{\mathsf{i}}_\mathcal{A}, \Delta_\mathcal{A}, c_\mathcal{A})$ *can be guided by* a parity automaton $\mathcal{B} = (Q_\mathcal{B}, \Sigma, q^{\mathsf{i}}_\mathcal{B}, \Delta_\mathcal{B}, c_\mathcal{B})$ if there is a mapping $g : Q_\mathcal{A} \times \Delta_\mathcal{B} \to \Delta_\mathcal{A}$ with the following properties:

- $g(p, (q, a, q', q'')) = (p, a, p', p'')$ for some $p', p'' \in Q_\mathcal{A}$.
- For every accepting run ρ of $\mathcal{B}$ over a tree t, $g(\rho)$ is an accepting run of $\mathcal{A}$ over t, where $g(\rho) = \rho'$ is the unique run such that $\rho'(\varepsilon) = q^{\mathsf{i}}_\mathcal{A}$, and for all $u \in \{0,1\}^*$:

$$(\rho'(u), t(u), \rho'(u0), \rho'(u1)) = g(\rho'(u), (\rho(u), t(u), \rho(u0), \rho(u1))) \ .$$

The first condition ensures that the transition chosen by g is compatible with the given state of $\mathcal{A}$ and the input letter of the given transition of $\mathcal{B}$. The second condition states that accepting runs of $\mathcal{B}$ are translated into accepting runs of $\mathcal{A}$ when applying g. If such a mapping g exists, then we say that $(\mathcal{B}, g)$ *guides* $\mathcal{A}$.

Obviously, these conditions can only be satisfied if $T(\mathcal{B}) \subseteq T(\mathcal{A})$. We say that $\mathcal{A}$ is *guidable* if it can be guided by every $\mathcal{B}$ such that $T(\mathcal{B}) \subseteq T(\mathcal{A})$.

A first remark is that in the definition of guidable automaton it is sufficient to require that $\mathcal{A}$ can be guided by every automaton $\mathcal{B}$ that accepts the same language as $\mathcal{A}$ (instead of a sublanguage).

REMARK 4.5 If $\mathcal{A}$ can be guided by every $\mathcal{B}$ with $T(\mathcal{B}) = T(\mathcal{A})$, then $\mathcal{A}$ is guidable.

Proof. Assume that $\mathcal{A}$ can be guided by every $\mathcal{B}$ with $T(\mathcal{B}) = T(\mathcal{A})$. Let $\mathcal{C}$ be a PTA such that $T(\mathcal{C}) \subseteq T(\mathcal{A})$. We have to show that $\mathcal{A}$ can be guided by $\mathcal{C}$. For this it is enough to consider the disjoint union $\mathcal{A} \cup \mathcal{C}$ of $\mathcal{A}$ and $\mathcal{C}$ resulting in an automaton equivalent to $\mathcal{A}$. By assumption there is a mapping g such that $(\mathcal{A} \cup \mathcal{C}, g)$ guides $\mathcal{A}$. From g we can easily construct a mapping g' by restricting g to the states of $\mathcal{C}$ such that $(\mathcal{C}, g')$ guides $\mathcal{A}$. (To be precise, there is a small technical issue because our automata only have a single initial state. Hence we have to introduce a fresh initial state for the automaton $\mathcal{A} \cup \mathcal{C}$. This requires some technical adjustments for the definition of g' but these are rather straightforward.) □

If $(\mathcal{B}, g)$ guides $\mathcal{A}$, then the mapping g can also be seen as defining a simulation relation between tree automata (see [1] for simulation relations on automata for finite trees) with an additional fairness constraint for the acceptance condition (see [51]). In that sense a guidable automaton can simulate every equivalent automaton as mentioned at the beginning of this section.

Note that the concept of guidability is not tied to infinite trees. The notion of guidable automaton can directly be transferred to automata for finite words or trees or infinite words. Let us consider some examples to get a better understanding of guidable automata and their relation to unambiguous automata.

We explain the first example for automata on finite words. Consider the automaton $\mathcal{A}$ from the left-hand side of Figure 4.1 that accepts all words over the alphabet $\{a, b\}$ by guessing in the first step of the run if the last letter is a or b and then proceeds to two subautomata, one accepting words ending with a, the other accepting words ending with b. This automaton is not guidable because the automaton $\mathcal{B}$ on the right-hand side of Figure 4.1 accepts the same language, but any mapping g such that $(\mathcal{B}, g)$ guides $\mathcal{A}$ would have to choose exactly one a-transition from q_0 and exactly one b-transition from q_0. Assume that $g(q_0, (p_0, a, p_0)) = (q_0, a, q_1)$. Then g would not translate the accepting run of $\mathcal{B}$ on the word ab to an accepting run of $\mathcal{A}$. A similar argument for the other cases shows that $\mathcal{B}$ can indeed not guide $\mathcal{A}$. However, $\mathcal{A}$ is an unambiguous automaton since there is exactly one accepting run for each input. This shows that already on finite words the concepts of unambiguity and guidability are different.

The above example supports the intuition that a guidable automaton does not make any "unnecessary guesses" because then an equivalent au-

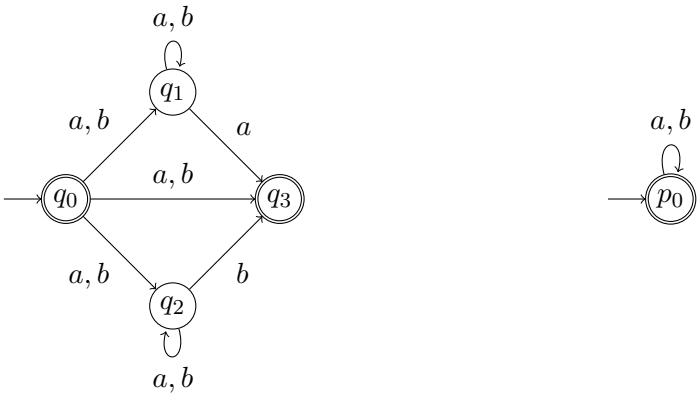
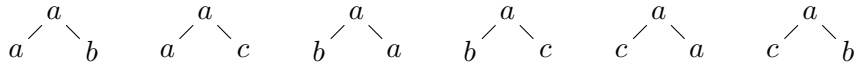


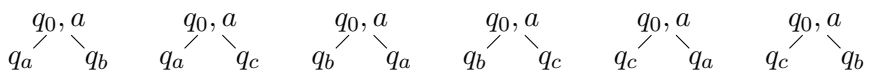
Figure 4.1: Two automata accepting all finite words over $\{a, b\}$. The automaton on the left-hand side is not guidable because the automaton on the right hand side cannot guide it.

tomaton that does not make such a guess would not be able to guide it. However, this does not imply that a guidable automaton uses as few nondeterminism as possible. We illustrate this with the following example taken from [22].

Consider the language T of infinite trees over the alphabet $\{a, b, c\}$ containing all trees whose root is labeled a and whose successors of the root have different labels, i.e., the trees starting in one of the following ways:

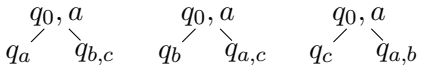


It is easy to see that T can be accepted by a parity automaton only using priority 0. The automaton checks whether the tree starts correctly with one of the above patterns and then moves to a looping state. But there are various ways to check for the initial patterns. The most direct way of doing it is to use an initial state q_0 and three states q_a, q_b, q_c (the index indicates for which letter the automaton can proceed from the corresponding state), and to cover the above patterns by transitions:



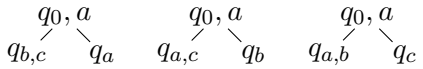
Let us refer to this automaton as $\mathcal{A}_1$ (of course one has to add transitions to a looping state from q_a, q_b, q_c for the correct combinations of input letter and state).

But it is also possible to use more states and less transitions. For example, an a at the left successor allows b or c at the right successor. Hence, we can use states q_a and $q_{b,c}$ for these combinations, and similarly for the other patterns. This results in the following transitions:



Let us refer to this automaton as $\mathcal{A}_2$. We observe that $\mathcal{A}_1$ is not guidable because it cannot be guided by $\mathcal{A}_2$. For example, if $\mathcal{A}_2$ uses the transition $(q_0, a, q_a, q_{b,c})$, then there are two possible transitions of $\mathcal{A}_1$ and none of them can be excluded.

But the automaton $\mathcal{A}_2$ is also not guidable. If we take an automaton $\mathcal{A}_3$ that works in the same way as $\mathcal{A}_2$ but with left and right successors exchanged, then we obtain an automaton the cannot guide $\mathcal{A}_2$:



And of course $\mathcal{A}_3$ is not guidable because it cannot be guided by $\mathcal{A}_2$. To obtain a guidable automaton for the language T we have to take the transitions from $\mathcal{A}_2$ and the transitions from $\mathcal{A}_3$. This shows that it might be necessary to add nondeterminism to automata to make them guidable. And obviously we obtain a guidable automaton that is not unambiguous.

Before we proceed to the main result of this section we collect some remarks on guidable automata.

Remark 4.6 We have the following observations on guidable automata, which are either direct consequences of the definition or of the examples presented in this section.

- Deterministic automata are guidable.
- There are unambiguous automata that are not guidable.
- There are guidable automata that are not unambiguous.

The main result of this section is that for each regular tree language we can construct a guidable automaton.

Theorem 4.7 *For each regular tree language T there exists a guidable parity automaton that can be constructed from a given parity automaton for T.*

Proof. We show that the complementation procedure presented in Section 1.4 yields a guidable automaton.

We start from a PTA $\mathcal{A} = (Q_\mathcal{A}, \Sigma, q^{\mathrm{i}}_\mathcal{A}, \Delta_\mathcal{A}, c_\mathcal{A})$ for the complement of T. Recall that in the complementation procedure we first construct a deterministic PTA $\mathcal{C}'$ over the alphabet $\Sigma \times (\Delta_\mathcal{A} \to \{0, 1\})$ accepting the tree representations of winning strategies for Adam in the membership game (Lemma 1.15). The complement automaton $\mathcal{C}$ is then obtained by omitting the component of the labels describing the strategy. Obviously $\mathcal{C}$ accepts T because $\mathcal{C}$ accepts the complement of $T(\mathcal{A})$. We have to show that $\mathcal{C}$ is a

guidable automaton. The idea is that a strategy for Adam in the emptiness game for the intersection of $\mathcal{B}$ and $\mathcal{A}$ can be used to construct the mapping g to guide $\mathcal{C}$.

Let $\mathcal{B} = (Q_\mathcal{B}, \Sigma, q^{\mathrm{i}}_\mathcal{B}, \Delta_\mathcal{B}, c_\mathcal{B})$ be a PTA with $T(\mathcal{B}) \subseteq T = T(\mathcal{C})$. The mapping g is constructed from a strategy in the intersection emptiness game $\mathcal{G}_{\mathcal{B}\times\mathcal{A}}$ (see Section 1.3, page 23). According to Corollary 1.11, Adam has a positional winning strategy in this game. Such a positional winning strategy is a mapping $\sigma_\Box : \Delta_\mathcal{B} \times \Delta_\mathcal{A} \to \{0,1\}$ assigning to each pair of transitions a direction. It can be equivalently written as a mapping $\sigma_\Box : \Delta_\mathcal{B} \to (\Delta_\mathcal{A} \to \{0,1\})$ assigning to each $\mathcal{B}$-transition a mapping from the set of $\mathcal{A}$-transitions to $\{0,1\}$.

To construct the mapping g guiding $\mathcal{C}$ we use the automaton $\mathcal{C}'$ mentioned above. Note that $\mathcal{C}'$ and $\mathcal{C}$ are essentially the same: they have the same set of states $Q_\mathcal{C}$, the same priority function $c_\mathcal{C}$, and the transitions of $\mathcal{C}$ are those of $\mathcal{C}'$ where the $(\Delta_\mathcal{A} \to \{0,1\})$ component of the labels is omitted. We denote the transitions of $\mathcal{C}$ by $\Delta_\mathcal{C}$ and those of $\mathcal{C}'$ by $\Delta_{\mathcal{C}'}$.

We define $g : Q_\mathcal{C} \times \Delta_\mathcal{B} \to \Delta_\mathcal{C}$ as follows. Let $(q, a, q', q'') \in \Delta_\mathcal{B}$ be a transition of $\mathcal{B}$. We can apply $\sigma_\Box$ to this transition, giving us a mapping $f : \Delta_\mathcal{A} \to \{0,1\}$, i.e., $f = \sigma_\Box(q, a, q', q'')$. Taking the pair (a, f) we get an input for $\mathcal{C}'$, and from a state $p \in Q_\mathcal{C}$ there is exactly one transition $(p, (a, f), p', p'') \in \Delta_{\mathcal{C}'}$ with this input (a, f). The transition $(p, a, , p', p'')$ is in $\Delta_\mathcal{C}$, and we let $g(p, (q, a, q', q'')) = (p, a, p', p'')$.

We now need to show that g translates accepting $\mathcal{B}$-runs into accepting $\mathcal{C}$-runs. Let $\rho_\mathcal{B}$ be an accepting run of $\mathcal{B}$ on some tree t. Using $\sigma_\Box$ we can construct a tree representation of a strategy for Adam in the membership game by applying $\sigma_\Box$ to all the transitions in $\rho_\mathcal{B}$. Formally, let $s \in \mathcal{T}^\omega_{\Sigma\times(\Delta_\mathcal{A}\to\{0,1\})}$ be defined by

$$s(u) = (t(u), \sigma_\Box(\rho_\mathcal{B}(u), t(u), \rho_\mathcal{B}(u0), \rho_\mathcal{B}(u1))) \ .$$

This tree s is a representation of a winning strategy for Adam in the membership game because $\sigma_\Box$ is a winning strategy for Adam in $\mathcal{G}_{\mathcal{B}\times\mathcal{A}}$. Hence, s is accepted by $\mathcal{C}'$. This run is unique, and by construction of g and s one obtains that the corresponding run of $\mathcal{C}$ on t is the run $g(\rho_\mathcal{B})$. Therefore g has the desired property. □

Concerning the complexity, we obtain a doubly exponential construction when starting from an automaton for T because we have to apply the complementation construction twice. (If we get an automaton for the complement of T as input, then the construction is singly exponential.) This raises the question whether the doubly exponential blow-up is necessary. We show that it indeed cannot be avoided, by extending the idea underlying the example (starting on page 74) that we used to explain why guidable automata might contain more nondeterminism than necessary for accepting a language.

PROPOSITION 4.8 *For each $n \geq 1$ there is a language $T_n \subseteq \mathcal{T}^{\omega}_{\{a,b\}}$ such that T_n can be accepted a PTA with $2n+2$ states, and each guidable PTA for T_n needs at least $2^{2^n} - 2$ states.*

Proof. To define the language T_n we consider the left-most branches starting in the two successors of the root, i.e., nodes of the form 0^+ and 10^*. We let T_n be the language of all trees whose labels on these two branches differ at one of the first n positions:

$$T_n = \{t \in \mathcal{T}^{\omega}_{\{a,b\}} \mid t(00^i) \neq t(10^i) \text{ for some } i \in \{0, \ldots, n-1\}\}.$$

We can easily construct an automaton with $2n+2$ states accepting T_n. The automaton has an initial state q_0 and guesses at the beginning the position i on which the two branches differ using transitions of the form $(q_0, *, q_{a,i}, q_{b,i})$ or $(q_0, *, q_{b,i}, q_{a,i})$ (where $* \in \{a, b\}$) . Then it counts down the i on the two branches, checking for the labels a or b when reaching the states $q_{a,0}$ or $q_{b,0}$, respectively. Additionally there is one accepting state used on the remaining part of the tree.

We now show that a guidable automaton needs at least $2^{2^n} - 2$ states for accepting T_n by constructing an automaton $\mathcal{A}_n$ that accepts T_n and cannot guide an automaton with less than $2^{2^n} - 2$ states.

The automaton $\mathcal{A}_n$ works as follows. It has states of the form q_L, where $L \subseteq \{a, b\}^n$ is a non-empty set of words of length n (possible label sequences of the two branches to be compared). In the following we denote by $\bar{L}$ for $L \subseteq \{a, b\}^n$ the complement $\bar{L} = \{a, b\}^n \setminus L$ of L within the set of words of length n. In its first transition $\mathcal{A}_n$ guesses a set L and moves to the states q_L and $q_{\bar{L}}$ at the two successors of the root, i.e., it has transitions of the form $(q^{\mathrm{i}}_n, *, q_L, q_{\bar{L}})$, where q^{i}_n is the initial state of $\mathcal{A}_n$ and $* \in \{a, b\}$. Then it verifies that the corresponding label sequences of the two branches are in L and in $\bar{L}$, respectively. This proves that there is a position on which the labels differ, as required in the definition of T_n. Note that there are $2^{2^n} - 2$ states of the form q_L (the -2 because L cannot be the empty or the full set). To verify from state q_L that the label sequence is in L the automaton needs further states. But for our reasoning we do not need to specify them, we are only interested in the states q_L.

Assume that $\mathcal{A}$ is a guidable automaton for T_n with less than $2^{2^n} - 2$ states and initial state q^{i}. Let g be such that $(\mathcal{A}_n, g)$ guides $\mathcal{A}$. Then there are two states q_L and q_M such that $L \neq M$, and for the two transitions $(q^{\mathrm{i}}_n, a, q_L, q_{\bar{L}})$ and $(q^{\mathrm{i}}_n, a, q_M, q_{\bar{M}})$ the mapping g chooses two transitions $(q^{\mathrm{i}}, a, q, q')$ and $(q^{\mathrm{i}}, a, q, q'')$ with the same state at the left successor. Now pick a word w in the symmetric difference of L and M and consider the case that $w \in L \setminus M$ (the other case is symmetric). Then $w \in L$ and $w \in \bar{M}$. Define the tree t_w such that both initial segments of the branches we consider

are labeled w (and a for the rest of the tree):

$$t_w(u) = \begin{cases} w(i) & \text{if } u \in \{00^i, 10^i\} \text{ for } i \in \{0, \dots, n-1\} \\ a & \text{otherwise.} \end{cases}$$

As g chooses the transition $(q^{\mathrm{i}}, a, q, q')$ for $(q_n^{\mathrm{i}}, a, q_L, q_{\bar{L}})$ and because $w \in L$ we know that there is an accepting run of $\mathcal{A}$ from q on the left subtree of t_w (we can construct one using $\mathcal{A}_n$ as a guide from state q_L). Similarly, there is an accepting run of $\mathcal{A}$ from q'' on the right subtree of t_w. Hence we can construct an accepting run of $\mathcal{A}$ on t_w that starts with the transition $(q^{\mathrm{i}}, a, q, q'')$. But t_w is not in T_n contradicting the choice of $\mathcal{A}$. □

We would like to point out that we have presented the above result for infinite trees but the languages T_n can easily be adapted to languages of finite trees. Hence, the doubly exponential blow-up is also unavoidable if we consider guidable automata on finite trees.

A further question is whether it is decidable if a given automaton $\mathcal{A}$ is guidable. To settle this question it is helpful to view guidability in terms of a game. A mapping g such that $(\mathcal{B}, g)$ guides $\mathcal{A}$ can be seen as a witness for the inclusion $T(\mathcal{B}) \subseteq T(\mathcal{A})$ because g translates accepting runs of $\mathcal{B}$ into accepting runs of $\mathcal{A}$. We can define a game, which we call weak inclusion game for $\mathcal{A}$ and $\mathcal{B}$, such that g corresponds to a positional winning strategy in this game. It is called "weak" because the game does not characterize the language inclusion for the two automata. From a winning strategy for Eva in the game we can conclude that $T(\mathcal{B}) \subseteq T(\mathcal{A})$ but not the other way round.

The *weak inclusion game for $\mathcal{A}$ and $\mathcal{B}$* is defined as follows.

- The positions of Adam are $Q_{\mathcal{A}} \times Q_{\mathcal{B}}$ and $\Delta_{\mathcal{A}} \times \Delta_{\mathcal{B}}$.
- The positions of Eva are $Q_{\mathcal{A}} \times \Delta_{\mathcal{B}}$.
- From (p, q) Adam can move to $(p, (q, a, q_0, q_1))$ for $(q, a, q_0, q_1) \in \Delta_{\mathcal{B}}$.
- From $(p, (q, a, q_0, q_1))$ Eva can move to $((p, a, q_0, q_1), (q, a, q_0, q_1))$ for some $(p, a, q_0, q_1) \in \Delta_{\mathcal{A}}$.
- From $((p, a, q_0, q_1), (q, a, q_0, q_1))$ Adam can choose a direction $i \in \{0, 1\}$ and move to (p_i, q_i).
- An infinite play in this sequence generates an infinite sequence of state pairs (the chosen transitions are not important for the winning condition). Eva wins if either the sequence of $\mathcal{B}$ states does not satisfy the parity condition of $\mathcal{B}$, or if the sequence of $\mathcal{A}$ states satisfies the parity condition of $\mathcal{A}$. In other words, Eva has to ensure that the acceptance condition of $\mathcal{A}$ is satisfied if the one of $\mathcal{B}$ is satisfied.

PROPOSITION 4.9 *The weak inclusion game for $\mathcal{A}$ and $\mathcal{B}$ is a Rabin game and Eva has a winning strategy iff $\mathcal{A}$ can be guided by $\mathcal{B}$.*

Proof. The winning condition is a disjunction of parity conditions (the negation of the parity condition of $\mathcal{B}$ can easily be expressed as a parity condition by adding 1 to all the priorities of $\mathcal{B}$). The disjunction of two parity conditions is a Rabin condition (the dual of Remark 1.7).

If $(\mathcal{B}, g)$ guides $\mathcal{A}$, then g directly corresponds to a positional winning strategy for Eva. Vice versa, if Eva has a winning strategy, then she has a positional winning strategy (Theorem 1.5), and a positional winning strategy for Eva can directly be transformed into a mapping g such that $(\mathcal{B}, g)$ guides $\mathcal{A}$. □

Note that Proposition 4.9 directly implies that $T(\mathcal{B}) \subseteq T(\mathcal{A})$ if Eva has a winning strategy because only in this case $\mathcal{B}$ can guide $\mathcal{A}$.

Using Proposition 4.9 we first note that for two automata $\mathcal{A}$ and $\mathcal{B}$ it is decidable whether $\mathcal{A}$ can be guided by $\mathcal{B}$. This is a simple consequence of the fact that the problem of deciding whether Eva has a winning strategy in a given Rabin game is NP-complete [45].

PROPOSITION 4.10 *It is decidable, given two PTAs $\mathcal{A}$ and $\mathcal{B}$, whether $\mathcal{A}$ can be guided by $\mathcal{B}$.*

Deciding whether a given PTA $\mathcal{A}$ is guidable requires to check whether $\mathcal{A}$ can be guided by every $\mathcal{B}$ with $T(\mathcal{B}) \subseteq T(\mathcal{A})$. But we can observe that the relation "can be guided by" between automata is transitive, and hence an automaton is guidable iff it can be guided by an automaton which is guidable itself.

PROPOSITION 4.11 *It is decidable, given a PTA $\mathcal{A}$, whether it is guidable.*

Proof. According to Theorem 4.7 we can construct a guidable automaton $\mathcal{A}'$ for the language $T(\mathcal{A})$. Using Proposition 4.10 we can decide whether $\mathcal{A}$ can be guided by $\mathcal{A}'$. If $\mathcal{A}$ cannot be guided by $\mathcal{A}'$, then $\mathcal{A}$ obviously is not guidable.

So assume that $(\mathcal{A}', g)$ guides $\mathcal{A}$. Let $\mathcal{B}$ be some automaton with $T(\mathcal{B}) \subseteq T(\mathcal{A})$. Since $\mathcal{A}'$ is guidable there is a mapping g' such that $(\mathcal{B}, g')$ guides $\mathcal{A}'$. We can view g and g' as positional winning strategies for Eva in the weak inclusion games for $\mathcal{A}$ and $\mathcal{A}'$, and $\mathcal{A}'$ and $\mathcal{B}$, respectively. These two strategies can be composed to a strategy h for Eva in the weak inclusion game for $\mathcal{A}$ and $\mathcal{B}$. This strategy is not positional because it has to recall the current state of $\mathcal{A}'$: Given a pair $(p, (q, a, q_0, q_1))$ of a state of $\mathcal{A}$ and a transition of $\mathcal{B}$, the strategy h uses the current state r of $\mathcal{A}'$ that it has constructed along the play (starting from the initial state of $\mathcal{A}'$), and applies g' to $(r, (q, a, q_0, q_1))$ resulting in a transition (r, a, r_0, r_1) of $\mathcal{A}'$. Then it applies

g to $(p, (r, a, r_0, r_1))$ giving a transition (p, a, p_0, p_1), which determines the move Eva plays.

The subsequent choice of a direction $i \in \{0, 1\}$ of Adam determines the next game position (p_i, q_i) as well as the next state r_i of $\mathcal{A}'$ that serves as memory for the strategy h.

Because g and g' define winning strategies it is easy to conclude that h is also a winning strategy. Hence, Eva has a winning strategy in the weak inclusion game for $\mathcal{A}$ and $\mathcal{B}$. According to Proposition 4.9 $\mathcal{A}$ can be guided by $\mathcal{B}$. □

One should mention that the decision procedure used to prove Proposition 4.10 requires to construct a guidable automaton that is equivalent to the given one. We leave open the question whether there are other, maybe more efficient ways to decide if an automaton is guidable.

4.4 Discussion

We have considered the properties of bottom-up determinism, unambiguity, and guidability for parity tree automata. The notion of bottom-up determinism does not seem to be very useful for automata on infinite trees because bottom-up deterministic automata can still have many different runs on a given input and there are no other helpful properties known to be implied by bottom-up determinism.

Unambiguity has proved to be an interesting concept for automata on finite words and trees because it allows more efficient decision procedures in some cases. We have shown that unambiguous automata on infinite trees do not capture all regular languages. The question whether it is decidable if a given tree language is unambiguous remains open. It would also be interesting to obtain a deeper understanding of the precise connection between the existence of choice functions and unambiguity of automata.

Guidable automata can be seen as a most general way of accepting a language. We have shown that each regular tree language can be accepted by a guidable automaton. In the next chapter we present a construction that requires to be applied to guidable automata for being correct. This is a first illustration of the use of the concept of guidable automata. We hope to find further applications in the future.

The notion of guidable automata has not been investigated in the setting of words or finite trees. In particular, it would be interesting to understand the connection to the universal automaton for languages of finite words [77], and to analyze whether there are unique minimal guidable automata for languages of finite trees. These questions are subject to future research.

Chapter 5

The parity index problem

An important topic in automata theory is the simplification of automata. On finite words and finite trees the main parameter for simplification is the number of states. For automata on infinite objects another parameter becomes interesting: the complexity of the acceptance condition. In this chapter we consider the *parity index problem* of finding a minimal range of priorities needed for a parity tree automaton to accept a given regular language. The number of priorities used in a parity condition is important because algorithms for solving parity games (and hence the emptiness problem of parity tree automata) are exponential only in this parameter, whereas they are polynomial in the size of the automaton (see [48] for an overview).

Several special cases of the parity index problem have been solved [93, 121, 94] but in its full generality the problem remains open. Unfortunately, we cannot present a solution here. Instead we give a reduction to a different problem opening a new way of attacking the problem.

The reduction is inspired from the proof of decidability of the (restricted) star-height problem due to Kirsten [64] (the problem was originally solved by Hashiguchi [50]). The star-height problem is the following: given a regular language of finite words and a natural number k, is it possible to describe the language by a regular expression using at most k nesting of Kleene stars?

For showing the decidability of the star-height problem, Kirsten introduces a new class of automata: nested distance desert automata. Those are non-deterministic finite automata running on finite words while executing instructions on counters that do not influence the control structure. The semantics of such automata is either to reject a word, or to compute a natural number for it that we can see as the price to pay for accepting it. Hence a nested distance desert automaton defines a partial mapping from words to natural numbers. The proof then goes in two steps.

- Reduce the star-height problem to the limitedness problem for nested distance desert automata (the limitedness is the problem of determining if the partial mapping defined by the automaton is bounded).

- Solve the limitedness problem for nested distance desert automata.

We want to follow exactly the same scheme to solve the parity index problem. We introduce the family of distance parity automata running over infinite trees. Those automata combine the features of nested distance desert automata and of parity automata. If we restrict a distance parity automaton to run on finite words, we fall back to the class of automata defined by Kirsten. If we restrict those automata to infinite words, we get a family of automata equivalent to the hierarchical ωB-automata in [13].

In this chapter we prove an analogue of the first proof step. We reduce the parity index problem to the uniform universality problem for distance parity automata (a problem related to the limitedness problem). This reduction part is very different from the one in the proof of Kirsten. We use guidable automata as introduced in Chapter 4 and show that we can optimize a guidable automaton in the sense that we can reduce its range of priorities (if this is possible at all for the accepted language) but thereby turning it into a distance parity automaton.

To solve the parity index problem it remains to show the decidability of the uniform universality problem for distance parity automata. This problem is still open at present.

The chapter is structured as follows. In Section 5.1 we formally introduce the parity index problem and the corresponding terminology. In Section 5.2 we introduce the model of distance parity automaton, and in Section 5.3 we give the reduction of the parity index problem to the uniform universality problem.

The results in this chapter have been obtained in collaboration with Thomas Colcombet and were published in [32].

5.1 The Mostowski hierarchy

Given a parity tree automaton $\mathcal{A}$, each state is assigned a priority from some finite interval $[i, j]$. This interval $[i, j]$ is called the index of $\mathcal{A}$, and we say that a language T is $[i, j]$-*feasible* if there exists a PTA of index $[i, j]$ accepting T. For the acceptance condition only the order of the priorities and their parities are relevant. Hence we can easily normalize each parity automaton such that the minimal priority used is either 0 or 1. We obtain in a natural way a hierarchy of language classes as depicted in Figure 5.1, where an entry i, j denotes the class of all languages that are $[i, j]$-feasible. We refer to this hierarchy as the *parity index hierarchy* or *Mostowski hierarchy* because Mostowski was the first one to use an acceptance condition of the shape of a parity condition [84].

Note that the level $(1, 1)$ of the hierarchy does not make sense if we only consider infinite trees because no PTA that uses only priority 1 can

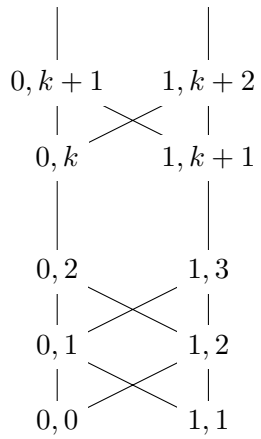


Figure 5.1: The structure of the Mostowski hierarchy

accept an infinite tree. In a setting where we allow finite and infinite trees, a language would be $(1, 1)$-feasible iff it is a regular language of finite trees.

The Mostowski hierarchy does not only exist for PTAs, i.e., nondeterministic parity automata on infinite trees, but for all kinds of underlying automata that use a parity acceptance condition, and is known to be strict in most cases: for deterministic automata on infinite words and trees [125], for nondeterministic automata on infinite trees [91], and for alternating automata on infinite trees [15, 71] (see also [5]). For nondeterministic automata over infinite words the hierarchy is easily seen to collapse to the Büchi level $(1, 2)$ because Büchi automata recognize all regular languages of infinite words. In the case of alternating automata on infinite words it even collapses to the intersection of the levels $(0, 1)$ and $(1, 2)$ (see [88, 74, 68]).

Since we are interested in the hierarchy for PTAs, we use the term Mostowski hierarchy in the following to refer to this case. The problem that we study in this chapter is the problem of deciding for a given regular language of infinite trees and a level of the Mostowski hierarchy, whether the language is on this level. Expressed in terms of $[i, j]$-feasibility we obtain the following decision problem, which we call the *parity index!problem*:

Given: A regular tree language T, and natural numbers $i < j$. [1]
Question: Is T $[i, j]$-feasible?

We have already mentioned in the introduction to this chapter that several partial results on this problem have been obtained by restricting the types of languages that are allowed as input to the problem. In the simplest case, the language T is a *path language*. For a given language L of infinite words the path language $Path(L)$ of L is the set of all trees t such that the

[1] As explained before we can always assume that $i \in \{0, 1\}$.

label sequence along each branch of t yields a word in L. In [67] it is shown that $Path(L)$ can be accepted by a (nondeterministic) Büchi tree automaton iff L can be accepted by a deterministic Büchi word automaton. This result has been generalized in [93] to the full hierarchy of deterministic parity word automata: $Path(L)$ can be accepted by a (nondeterministic) PTA of index $[i, j]$ iff L can be accepted by a deterministic parity word automaton of index $[i, j]$. Therefore, to solve that parity index problem for path languages it suffices to solve the parity index problem for deterministic parity word automata. In [93] it is shown that for a given deterministic parity word automaton one can compute in polynomial time its location in the Mostowski hierarchy of deterministic word languages. This result was obtained independently in [24], and it also follows from [66] when applying their results on Rabin and Streett automata to the particular case of parity automata.

A larger class than path languages is defined by deterministic top-down PTAs. In [121] it has been shown that the parity index problem is decidable if the input automaton $\mathcal{A}$ is deterministic, and $(i, j) = (1, 2)$, i.e., it is decidable for a given deterministic PTA if its language can also be accepted by a *nondeterministic* Büchi tree automaton. This has been generalized to arbitrary values for i and j in [94], i.e., it is decidable for a given deterministic PTA if its language can be accepted by a nondeterministic PTA of index $[i, j]$.

In [86] the problem of deciding for a given deterministic tree automaton (with Muller acceptance condition) whether its language is on a certain level of the Borel hierarchy is studied. Since there is no direct correspondence between the levels of the Borel hierarchy and the levels of the nondeterministic Mostowski hierarchy, this problem is of a different nature.

For the general case of nondeterministic automata the problem of deciding the levels of the Mostowski hierarchy remains open and the methods developed for the special cases cannot be generalized. We propose in this chapter a reduction of the parity index problem to a different type of problem for an extended automaton model that we introduce in the next section.

Before presenting the reduction we first show that we can decide the lowest level of the hierarchy, i.e., whether a language is $(0, 0)$-feasible. These are languages that are closed in the standard prefix topology on infinite trees (see [56] for characterizations of $(0, 0)$-feasible languages). We consider the following decidability result folklore and do not attribute it to a specific paper.

THEOREM 5.1 *Given a PTA $\mathcal{A}$ it is decidable in exponential time whether $T(\mathcal{A})$ is $(0, 0)$-feasible.*

Proof. The idea is to construct the closure of the language $T(\mathcal{A})$. This closure contains all trees t such that each finite prefix of t, i.e., an initial part of t up to some depth n, can be extended to some tree in $T(\mathcal{A})$. A

language is $(0,0)$-feasible iff it is equal to its closure. Hence, it suffices to construct the closure and to test equivalence to decide $(0,0)$-feasibility of a language.

An automaton $\mathcal{A}_{\mathrm{cl}}$ for the closure of $T(\mathcal{A})$ is constructed as follows.

1. Remove all states from $\mathcal{A}$ that are not productive, i.e., from which no accepting run exists. This can be done in exponential time according to Theorem 1.9.

2. Set the priority of all remaining states to 0.

We show that $T(\mathcal{A})$ is $(0,0)$-feasible iff $T(\mathcal{A}) = T(\mathcal{A}_{\mathrm{cl}})$. It is clear that $T(\mathcal{A})$ is $(0,0)$-feasible if $T(\mathcal{A}) = T(\mathcal{A}_{\mathrm{cl}})$ because $\mathcal{A}_{\mathrm{cl}}$ is of index $(0,0)$. For the other direction it suffices to show that $T(\mathcal{A}_{\mathrm{cl}}) \subseteq T(\mathcal{A})$ if $T(\mathcal{A})$ is $(0,0)$-feasible because the other inclusion $T(\mathcal{A}) \subseteq T(\mathcal{A}_{\mathrm{cl}})$ always holds.

If $T(\mathcal{A})$ is $(0,0)$-feasible, then there is a PTA $\mathcal{B}$ of index $(0,0)$ with $T(\mathcal{A}) = T(\mathcal{B})$. Let $t \in T(\mathcal{A}_{\mathrm{cl}})$ and let ρ be an accepting run of $\mathcal{A}_{\mathrm{cl}}$ on t. By construction of $\mathcal{A}_{\mathrm{cl}}$ we know that for each n there is a tree t_n that is equal to t up to depth n and that is accepted by $\mathcal{A}$ (take the initial part of ρ up to depth n and add below each state at the frontier a tree accepted from this state). Since $\mathcal{B}$ is equivalent to $\mathcal{A}$ there is for each n an accepting run ρ_n of $\mathcal{B}$ on each t_n. From the sequence of the runs ρ_n we can now extract a run of $\mathcal{B}$ on t. Since $\mathcal{B}$ is of index $(0,0)$ this run is accepting and hence $t \in T(\mathcal{B}) = T(\mathcal{A})$.

Equivalence of $T(\mathcal{A})$ and $T(\mathcal{A}_{\mathrm{cl}})$ can be decided in exponential time by complementing $\mathcal{A}$ and then checking the emptiness of the intersection with $\mathcal{A}_{\mathrm{cl}}$. □

By a simple reduction for the universality problem for tree automata we can also show a lower complexity bound for the problem.

THEOREM 5.2 *The problem of deciding for a given PTA $\mathcal{A}$ whether $T(\mathcal{A})$ is $(0,0)$-feasible is complete for exponential time.*

Proof. We show how to reduce the universality problem for PTAs to the problem of $(0,0)$-feasibility. The universality problem for PTAs is to decide for a given PTA $\mathcal{A}$ whether $T(\mathcal{A}) = \mathcal{T}_\Sigma^\omega$. This problem is complete for exponential time already for automata on finite trees (cf. [33]).

We fix the alphabet $\Sigma = \{a, b\}$ and let $\mathcal{A}$ be a PTA over Σ. Let $\mathcal{B}$ be a PTA that accepts all trees for which the left subtree of the root is in $T(\mathcal{A})$ or the right subtree of the root contains infinitely many b on each branch.

We use the fact that the language of all trees containing infinitely many b is not $(0,0)$-feasible. This implies that $T(\mathcal{B})$ is $(0,0)$-feasible iff $T(\mathcal{A}) = \mathcal{T}_\Sigma^\omega$. One direction of this claim is easy. If $T(\mathcal{A}) = \mathcal{T}_\Sigma^\omega$, then $T(\mathcal{B}) = \mathcal{T}_\Sigma^\omega$, and hence $T(\mathcal{B})$ is $(0,0)$-feasible.

Assume that $T(\mathcal{A}) \neq \mathcal{T}_\Sigma^\omega$ and that $T(\mathcal{B})$ is $(0,0)$-feasible. Let $\mathcal{B}'$ be a PTA of index $(0,0)$ that is equivalent to $\mathcal{B}$. Let t be a regular tree that is not in $T(\mathcal{A})$. Such a tree exists according to Theorem 1.12 and Theorem 1.16. From $\mathcal{B}'$ we can construct a PTA $\mathcal{B}''$ of index $(0,0)$ that accepts all trees in $T(\mathcal{B})$ which have t has left subtree of the root. From $\mathcal{B}''$ we can easily extract a PTA of index $(0,0)$ that accepts those trees that have infinitely many b on each path. As mentioned above, this language is not $(0,0)$-feasible, a contradiction. □

As a final remark let us mention that Mostowski introduced parity automata (under the name of *standard form automata*) in [84] and showed that they capture all regular languages. He also shows the strictness of the parity index hierarchy (Theorem 2 of [84]) and mentions that the problem that we refer to as parity index problem is decidable (Theorem 3 in [84]). We would like to point out that the model originally introduced by Mostowski is slightly different from our model: it only assigns priorities to some of the states but no priority is assigned to two different states. One can easily transfer our model into this one by renaming priorities but this of course changes the hierarchy. And in the model used in [84], given a fixed interval of priorities, there is only a bounded number of states that is relevant for the acceptance condition, which makes hierarchy questions much simpler.

5.2 Distance parity automata

Roughly speaking, distance parity automata are parity tree automata with additional instructions for manipulating counters. The counters are never tested, they can only be incremented or reset to 0, and their only use is to determine a value for a given input. Such automata have been introduced in several papers with different terminology and notations. While the name "distance" is borrowed from the work of Hashiguchi [50] and Kirsten [64], we use the notation from [13] for the instructions on the counters.

Let $D = \{I_1, R_1, \ldots, I_k, R_k\}$ be a set of counter instructions, where I stands for increment, and R for reset. Intuitively, we want to use the counters to measure the distances between some hierarchically ordered events. For this reason we consider the following order on the instructions: $I_1 < R_1 < \cdots < I_k < R_k$.

One way to determine the value of a (finite or infinite) sequence of these instructions is to use k counters numbered from 1 to k. When seeing I_j the counter j is incremented and all the counters below are reset. When seeing R_j, all the counters up to j are reset. The value of a sequence is the supremum of all values of counters seen during this process (starting from 0). Note that for infinite sequences this value can be infinite, which we

denote by the value ω. This is illustrated with a few examples:

$$\begin{aligned} val((I_1R_1)^\omega) &= 1 \\ val((I_1I_2I_1I_2I_1R_2)^\omega) &= 2 \\ val((I_1I_2)^\omega) &= \omega \\ val(I_1R_1I_1I_1R_1I_1I_1I_1R_1\cdots) &= \omega \end{aligned}$$

An alternative way to define the value of a sequence considers all finite factors of the given sequence and the maximal instruction on these factors. To formalize this we denote by $|w|_a$ the number of occurrences of a in w, where $w \in D^*$ and $a \in D$. Given an infinite sequence $\alpha \in D^\omega$, its value $val(\alpha)$ is the supremum of $|w|_{I_i}$ where w ranges over all finite factors of α such that the maximal instruction in w is I_i, and i ranges over $\{1, \ldots, k\}$. This definition is equivalent to the one using counters described above.

We naturally extend this value from infinite sequence to infinite trees by letting $val(t)$ for $t \in \mathcal{T}_D^\omega$ be the supremum over the value of all branches

$$val(t) = \sup\{val(\alpha) \mid \alpha \in D^\omega \text{ is a branch of } t\}.$$

Based on this function we can now define our automaton model. A *distance parity tree automaton* (distance PTA) $\mathcal{D} = (Q, \Sigma, q^{\mathsf{i}}, \Delta, c, d)$ is a PTA augmented with a function $d : Q \to D$, where D is a set of ordered counter instructions as defined above. A run ρ of $\mathcal{D}$ on a tree $t \in \mathcal{T}_\Sigma^\omega$ induces via the mapping d a tree $\rho_d \in \mathcal{T}_D^\omega$ (and via the mapping c a tree ρ_c of priorities). As usual we define a run ρ to be accepting if it is accepting w.r.t. the parity condition. The value computed by $\mathcal{D}$ on t is defined as

$$\mathcal{D}(t) = \min\{val(\rho_d) \mid \rho \text{ is an accepting run of } \mathcal{D} \text{ on } t\}.$$

If there is no accepting run of $\mathcal{D}$ on t, then the minimum from the definition is ω.

In other words, the value of an input is the minimal value of an accepting run on this input, where the value of a run is the supremum over the values of all paths. Note that the value of a tree can be ω even though the tree is accepted.

For each natural number $N < \omega$ we define the language $T^{(N)}(\mathcal{D}) := \{t \in \mathcal{T}_\Sigma^\omega \mid \mathcal{D}(t) \leq N\}$. In this way $\mathcal{D}$ defines a non-decreasing ω-sequence of languages, i.e., an ω-chain of languages. It is easy to observe that for each N, $T^{(N)}(\mathcal{D})$ is a regular language: For a fixed N the counters can be coded in the states of the automaton. This construction gives a parity automaton of index $[i, j]$ for $T^{(N)}(\mathcal{D})$. Hence, we define the *parity index* of a distance PTA to be the parity index of the underlying parity condition. From the above explanation we easily conclude the following.

Remark 5.3 Given a distance PTA $\mathcal{D}$ of parity index $[i, j]$ and a natural number N, the language $T^{(N)}(\mathcal{D})$ is $[i, j]$-feasible.

For automata on finite words as considered in [64], the value for all accepted inputs is finite, and it is ω for all inputs that are not accepted. In this setting, the *limitedness problem* is the question whether there is a bound on the values of the accepted words. On infinite trees (also on infinite words) there is a new phenomenon: the value of an accepted input can be ω as well. In that sense, a distance PTA has two ways of rejecting an input: by the parity condition or by value ω. In this situation, the limitedness problem in its above formulation becomes less interesting. However, the general question we are interested in is still whether a given distance PTA is bounded on a certain set. In the above formulation of the limitedness problem this set is implicitly defined by the given automaton itself. Here we provide this set as an extra input to the problem. To distinguish it from the original limitedness problem we call it the *uniform inclusion problem*:

Given: A distance PTA $\mathcal{D}$ and a regular tree language T.
Question: Does there exists N with $T \subseteq T^{(N)}(\mathcal{D})$?

If the inclusion $T \subseteq T^{(N)}(\mathcal{D})$ holds for some N, then we say that T is uniformly included in $\mathcal{D}$. It is obvious why we call this problem an inclusion problem. We call it *uniform* inclusion problem because we are looking for a uniform bound on the values of the trees in T (instead of just asking whether each tree in T has a finite value). A special case of this problem is the *uniform universality problem*, where the language T is the set of all trees. This can be formulated in the following way:

Given: A distance PTA $\mathcal{D}$ (over alphabet Σ).
Question: Does there exists N with $T^{(N)}(\mathcal{D}) = \mathcal{T}^{\omega}_{\Sigma}$?

It is clear that the uniform universality problem is a special case of the uniform inclusion problem. But we can also give a reduction in the other direction.

Proposition 5.4 *The uniform inclusion problem can be reduced to the uniform universality problem.*

Proof. Let $\mathcal{D}$ be a distance PTA and T be a regular language given by some PTA $\mathcal{A}$. We want to construct a distance PTA $\mathcal{D}'$ such that $\mathcal{D}'$ is uniformly universal iff $T \subseteq T^{(N)}(\mathcal{D})$ for some N.

Let $\mathcal{A}'$ be a PTA for the complement of T. We turn $\mathcal{A}'$ into a distance PTA by assigning to each state some reset for the counter instructions used in $\mathcal{D}$. Hence, $\mathcal{A}'$ computes value 0 for each input tree it accepts. Now take the disjoint union of $\mathcal{A}$ and $\mathcal{D}$ and obtain $\mathcal{D}'$. It follows that

$$\mathcal{D}'(t) = \begin{cases} \mathcal{D}(t) & \text{if } t \in T \\ 0 & \text{otherwise} \end{cases}$$

Hence, $\mathcal{D}'$ is uniformly universal iff $T \subseteq T^{(N)}(\mathcal{D})$ for some N. □

5.3 Reduction to uniform universality

In this section we describe how to reduce the problem of deciding the parity index problem (whether a regular tree language is $[i,j]$-feasible) to the problem of deciding the uniform universality of a distance parity automaton. We fix a regular language T and an interval $[i,j]$ of natural numbers, i.e., an instance of the parity index problem.

The general idea is to construct a generic distance PTA that "optimizes" the runs of a PTA for T by mapping the priorities to the interval $[i,j]$. This optimization is only possible if the PTA for T has some good properties. Here we rely on the results presented in Section 4.3. According to Theorem 4.7 there exists a guidable PTA $\mathcal{A} = (Q_\mathcal{A}, \Sigma, q^{\mathrm{i}}_\mathcal{A}, \Delta_\mathcal{A}, c_\mathcal{A})$ with $T(\mathcal{A}) = T$. Let $P \subseteq \mathbb{N}$ be the interval of priorities that $\mathcal{A}$ is using, i.e., $c_\mathcal{A} : Q_\mathcal{A} \to P$.

The technical core of the reduction consists in constructing a distance PTA $\mathcal{D}^P_{i,j}$ of parity index $[i,j]$ reading P-trees such that:

(Correctness) For all P-trees t, if $\mathcal{D}^P_{i,j}(t) < \omega$ then t is accepting (stated in Lemma 5.6).

(Completeness) If T is $[i,j]$-feasible, then there exists $N < \omega$ such that for all trees $t \in T$, there is an accepting run ρ of $\mathcal{A}$ on t with $\mathcal{D}^P_{i,j}(c_\mathcal{A}(\rho)) \leq N$ (stated in a more general form in Lemma 5.7).

Then we can cascade the automata $\mathcal{D}^P_{i,j}$ and $\mathcal{A}$ into a single one denoted $\mathcal{D}$: this automaton guesses a run of $\mathcal{A}$ and applies the automaton $\mathcal{D}^P_{i,j}$ on the coloring of this run. The priority and distance mappings are inherited from $\mathcal{D}^P_{i,j}$ (this corresponds to a standard cascade or series composition of automata as it can be found, e.g., in Chapter 8 of [3]). This resulting distance PTA $\mathcal{D}$ is such that

- $T^{(N)}(\mathcal{D}) \subseteq T$ for all N (correctness), and
- $T \subseteq T^{(N)}(\mathcal{D})$ for some N iff T is $[i,j]$-feasible (completeness).

Hence, to decide the $[i,j]$-feasibility of T it suffices to decide the uniform inclusion problem for $\mathcal{D}$ and T. From this and Proposition 5.4 we directly get the main result of this chapter.

THEOREM 5.5 *The problem of deciding the $[i,j]$-feasibility of a regular tree language is reducible to the uniform universality of distance PTAs.*

We now describe the automaton $\mathcal{D}^P_{i,j}$ in detail. Intuitively, $\mathcal{D}^P_{i,j}$ maps each priority in P to a priority in $[i,j]$. When reading a priority $k \in P$ it "outputs" the priority associated to k. To implement this idea, the main objects used by $\mathcal{D}^P_{i,j}$ are partial mappings from P to $[i,j]$ (for technical

reasons we allow some priorities to be undefined). To ensure correctness of $\mathcal{D}^P_{i,j}$ (in the sense mentioned above), these mappings have to respect some conditions. First of all, odd priorities of P should be mapped to odd priorities of $[i, j]$. Second, the ordering of the priorities should be respected in the following sense: the image of every odd priority is required to dominate the image of all even priorities below it.

Formally, let S be the set of partial mappings $s : P \to [i, j]$ such that for all k for which $s(k)$ is defined:

1. If k is odd, then $s(k)$ is odd.
2. If k is even, then for all odd $l > k$ such that $s(l)$ is defined, $s(l) \geq s(k)$.

The transitions of $\mathcal{D}^P_{i,j}$ allow to change the mapping s in a safe way. It is not very difficult to observe that on reading priority $k \in P$, $\mathcal{D}^P_{i,j}$ can safely change the values for P-priorities strictly below k. Additionally, we also allow a bounded number of arbitrary changes of the values. This bound is not fixed a priori and is controlled by the distance condition.

We now proceed to the formal definition of the distance PTA $\mathcal{D}^P_{i,j} = (S \times P \times S, P, S_I, \delta, c, d)$. For simplicity we use a set of initial states.

- The set of initial states is $S_I = \{\langle s_\perp, k, s\rangle \mid s \in S,\ k \in P\}$, where $s_\perp$ is the mapping that is undefined everywhere.
- For all $s', s, s_0, s_1 \in S$, $k, k_0, k_1 \in P$ we allow the transition

$$\big(\langle s', k, s\rangle,\ k,\ \langle s, k_0, s_0\rangle, \langle s, k_1, s_1\rangle\big)\ .$$

  In fact, every transition is allowed, provided that the first component of the states at the successors is equal to the last component of the state at the parent node, and the second component of a state at some node is equal to the label of the P-tree at this node. As a consequence, a run of $\mathcal{D}^P_{i,j}$ on a given input tree is completely determined by the tree and the last components of its states.

- The priority mapping $c : S \to \{i, \ldots, j\}$ is defined by

$$c(\langle s', k, s\rangle) = \begin{cases} s(k) & \text{if } s(k) \text{ is defined,} \\ i & \text{if } s(k) \text{ is undefined.} \end{cases}$$

- Each priority $k \in P$ induces two counter instructions, a reset and an increment, where the reset induced by k is below the increment induced by k. If we assume that P contains priorities from m to n (with $m < n$), then the counter instructions are $R_{m-1} < I_m < R_m <$

$I_{m+1} < \cdots < R_{n-1} < I_n$, and we let D be the set containing these instructions. The mapping $d : S \to D$ is defined by

$$d(\langle s', k, s\rangle) = \begin{cases} R_{k-1} & \text{if } s(k) \text{ is defined and } s'(l) = s(l) \text{ for all } l \geq k \\ I_k & \text{if } s(k) \text{ is undefined and } s'(l) = s(l) \text{ for all } l \geq k \\ I_l & \text{for the maximal } l \text{ with } s'(l) \neq s(l) \text{ otherwise.} \end{cases}$$

We now establish that $\mathcal{D}^P_{i,j}$ satisfies the correctness condition mentioned at the beginning of this section.

LEMMA 5.6 (CORRECTNESS) *For all P-trees t, if $\mathcal{D}^P_{i,j}(t) < \omega$ then t satisfies the parity condition of P on each branch.*

Proof. Let t be a P-tree and let ρ be a run of $\mathcal{D}^P_{i,j}$ on t. Consider a branch π and let k be the maximal P-priority that appears infinitely often on π in t. We show that if k is odd, then either π is rejecting in ρ for the parity condition, or π has value ω.

In the following, the s-value of a priority in P refers to the value it is mapped to by the sates of $\mathcal{D}^P_{i,j}$. An immediate consequence of the definition of d and the choice of k is that R_{k-1} is the maximal reset that can occur infinitely often on π in ρ. Hence, if the s-value of some $l \geq k$ changes infinitely often along π, then I_l occurs infinitely often along π and the value of π is ω. Otherwise, the s-value of k becomes stable eventually on π. This value is odd because k is odd (property 1 in the definition of S), and furthermore it is bigger than the s-values of the smaller even priorities (property 2 in the definition of S). Hence, the maximal priority assumed infinitely often in ρ along π is odd and thus π is rejecting in ρ. □

Note that this correctness is a generic property of $\mathcal{D}^P_{i,j}$ that does not depend on the automaton $\mathcal{A}$.

The statement of the following lemma is more general than the statement needed for the completeness. It shows that each $[i, j]$-feasible sublanguage of T is uniformly included in $\mathcal{D}$ (the cascade of $\mathcal{A}$ and $\mathcal{D}^P_{i,j}$). If T itself is $[i, j]$-feasible this gives the desired completeness argument.

LEMMA 5.7 (COMPLETENESS) *For every $[i, j]$-feasible language $\hat{T} \subseteq T$ there exists $N \in \mathbb{N}$ such that for all trees $t \in \hat{T}$ there exists an accepting run ρ of $\mathcal{A}$ over t with $\mathcal{D}^P_{i,j}(c_{\mathcal{A}}(\rho)) \leq N$.*

The remainder of this section is devoted to the proof of Lemma 5.7. Let $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma, q^{\mathrm{i}}_{\mathcal{B}}, \Delta_{\mathcal{B}}, c_{\mathcal{B}})$ be some parity automaton of index $[i, j]$ such that $T(\mathcal{B}) \subseteq T$.

Our aim is to show that there exists N such that for all trees $t \in T(\mathcal{B})$ there is an accepting run $\rho_{\mathcal{A}}$ of $\mathcal{A}$ over t with $\mathcal{D}^P_{i,j}(c_{\mathcal{A}}(\rho_{\mathcal{A}})) \leq N$. Here we use that $\mathcal{A}$ is guidable and therefore there is a mapping g such that $(\mathcal{B}, g)$ guides

$\mathcal{A}$. From an accepting run $\rho_\mathcal{B}$ of $\mathcal{B}$ we can use g to obtain an accepting run $\rho_\mathcal{A} := g(\rho_\mathcal{B})$ of $\mathcal{A}$. The run $\rho_\mathcal{B}$ only uses priorities from i to j and we exploit this fact to construct a run of $\mathcal{D}^P_{i,j}$ on $\rho_\mathcal{A}$ that is accepting and has bounded cost. At the end of this construction we establish Lemma 5.16 providing the bound. From this we directly obtain Lemma 5.7.

We first need some more definitions. Let $t \in T(\mathcal{B})$ and as indicated above let $\rho_\mathcal{B}$ be an accepting run of $\mathcal{B}$ on t, and let $\rho_\mathcal{A} := g(\rho_\mathcal{B})$ be the induced guided run of $\mathcal{A}$. First To simplify notation, for a position $x \in \{0,1\}^*$ we write $c_\mathcal{A}(x)$ and $c_\mathcal{B}(x)$ to denote the respective priorities of the states in the runs $\rho_\mathcal{A}$ and $\rho_\mathcal{B}$ at position x, i.e., $c_\mathcal{A}(\rho_\mathcal{A}(x))$ and $c_\mathcal{B}(\rho_\mathcal{B}(x))$.

Let $y, z \in \{0,1\}^*$ be two nodes with $y \sqsubset z$. For $\ell \in [i,j]$ we say that (y,z) is an *ℓ-loop* if

- $\rho_\mathcal{A}(y) = \rho_\mathcal{A}(z)$,
- $\rho_\mathcal{B}(y) = \rho_\mathcal{B}(z) =: q \in Q_\mathcal{B}$ with $c_\mathcal{B}(q) = \ell$, and
- $c_\mathcal{B}(x) \leq \ell$ for all x with $y \sqsubset x \sqsubset z$.

An ℓ-loop is a loop for $\mathcal{A}$ and $\mathcal{B}$ (in the sense that they are in the same state at the entry and the exit), and the $\mathcal{B}$-priority ℓ at the entry and exit is the maximal one on the loop.

We say that an ℓ-loop (y,z) is *dominated* by a priority $k \in P$ (a priority of $\mathcal{A}$) if k is the highest $\mathcal{A}$ priority appearing in the loop (z excluded), i.e., if

$$k = \max\{c_\mathcal{A}(x) \ : \ y \sqsubseteq x \sqsubset z\}.$$

The following lemma is easy to prove but it is the central reason why we use guidable tree automata. It basically states that an accepting loop of $\mathcal{B}$ has to be dominated by an accepting priority of $\mathcal{A}$.

Lemma 5.8 *Let (y,z) be an ℓ-loop dominated by k. Then k is even if ℓ even.*

Proof. Assume that ℓ is even. The proof is a simple pumping argument. Let $\rho_\mathcal{B}^{[y,z)^*}$ be the run obtained from $\rho_\mathcal{B}$ by iterating the part between y and z. Formally, let $u \in \{0,1\}^*$ be such that $z = yu$. Then $\rho_\mathcal{B}^{[y,z)^*}$ is defined by

$$\rho_\mathcal{B}^{[y,z)^*}(x) = \begin{cases} \rho_\mathcal{B}(x) & \text{if } y \not\sqsubseteq x, \\ \rho_\mathcal{B}(yv) & \text{if } x = yu^n v \text{ for some } n \geq 0 \text{ and } v \text{ with } u \not\sqsubseteq v. \end{cases}$$

Because $\rho_\mathcal{B}(y) = \rho_\mathcal{B}(z)$ this indeed defines a run of $\mathcal{B}$, namely on the tree $t^{[y,z)^*}$ (which is defined in the same way). Furthermore, $\rho_\mathcal{B}^{[y,z)^*}$ is accepting because the maximal priority seen on the branch yu^ω is ℓ. All the other infinite branches in $\rho_\mathcal{B}^{[y,z)^*}$ correspond from some point onwards to a branch in $\rho_\mathcal{B}$ and are therefore accepting.

Now consider $g(\rho_{\mathcal{B}}^{[y,z)^*})$. Because g only locally translates the transitions of $\mathcal{B}$ into transitions of $\mathcal{A}$, and because $\rho_{\mathcal{A}}(y) = \rho_{\mathcal{A}}(z)$, one can observe that $g(\rho_{\mathcal{B}}^{[y,z)^*}) = (g(\rho_{\mathcal{B}}))^{[y,z)^*} = \rho_{\mathcal{A}}^{[y,z)^*}$, i.e., applying g to $\rho_{\mathcal{B}}^{[y,z)^*}$ yields the same as applying g to $\rho_{\mathcal{B}}$ and then iterating the loop (y, z).

In $\rho_{\mathcal{A}}^{[y,z)^*}$ the highest priority seen infinitely often on the branch yu^ω is k. As $\rho_{\mathcal{B}}^{[y,z)^*}$ is accepting, $\rho_{\mathcal{A}}^{[y,z)^*}$ must also be accepting and therefore k must be even. □

Having established this connection between priorities of $\mathcal{A}$ and $\mathcal{B}$ on loops, we now start the construction of an accepting run of $\mathcal{D}_{i,j}^P$ on $c_{\mathcal{A}}(\rho_{\mathcal{A}})$. Recall that the runs of $\mathcal{D}_{i,j}^P$ are in one-to-one correspondence with mappings that assign to each node an element of S. We are now going to specify such a mapping that associates to each node $x \in \{0,1\}^*$ an element $s_x \in S$ and show that it defines an accepting run of $\mathcal{D}_{i,j}^P$ on $\rho_{\mathcal{A}}$.

The essential idea is the following: In an ℓ-loop we map (more precisely the elements of S used by the run in the loop map) the highest priority for $\mathcal{A}$ on this loop (i.e., the priority dominating this loop) to the highest priority of $\mathcal{B}$ on this loop (i.e., to ℓ). Then Lemma 5.8 ensures that we indeed map odd priorities of $\mathcal{A}$ to odd priorities of $\mathcal{B}$. To make this simple idea work we have to tune the definitions a bit.

We say that a position x is *covered* by k in an ℓ-loop (y, z) if this loop is dominated by k, and there is an x' with $y \sqsubseteq x' \sqsubseteq x \sqsubset z$ such that $c_{\mathcal{A}}(x') = k$. That is, a position with priority k has to appear before x in the loop. We just say that x is covered by k if it is covered by k in some ℓ-loop.

We let

$$cov_x(k) = \max\{\ell \in [i, j] \mid x \text{ is covered by } k \text{ in an } \ell\text{-loop}\}$$

where the maximum is undefined in case x is not covered by k.

We intend to derive the mappings s_x in the run of $\mathcal{D}_{i,j}^P$ from cov_x. We have to take care that we satisfy the conditions imposed on the elements in S (see page 90). This is handled by the next lemma.

Lemma 5.9 *The function cov_x is nondecreasing (over its domain). Furthermore, for all k such that $cov_x(k)$ is defined, $cov_x(k)$ is odd if k is odd.*

Proof. Let $k < k'$ be such that $cov_x(k) = \ell$ and $cov_x(k') = \ell'$ are defined. By definition, x is k-covered in some ℓ-loop (y, z) and k'-covered in some ℓ'-loop (y', z'). As (y, z) is k-dominated and $k < k'$ we conclude that no position inside (y, z) can have priority k'. This implies that $y' \sqsubset y \sqsubseteq x \sqsubset z'$. Since (y', z') is an ℓ'-loop, ℓ' is the highest $\mathcal{B}$-priority on (y', z') and in particular $\ell = c_{\mathcal{B}}(y) \leq \ell'$.

The second statement is a direct consequence of Lemma 5.8. □

Using cov_x we now define $s_x : P \to [i, j]$ as follows:

$$s_x(k) = \begin{cases} \bot \text{ if } cov_x(k) \text{ is undefined,} \\ \max\{\ell \leq cov_x(k) \ : \ \ell = i \text{ or } \ell \equiv k \pmod 2\} \text{ otherwise.} \end{cases}$$

Note in this definition the special treatment of the priority i to prevent the use of the disallowed value $i - 1$. Note also that $s_x(k) = cov_x(k)$ if both values have the same parity (which is in particular the case if k is odd, see Lemma 5.9). If the two values are of different parity, then $s_x(k) = cov_x(k) - 1$.

A very important thing to remark is that, though cov_x is monotonic over its domain, this is not the case for s_x: Consider, e.g., the case $cov_x(2) = cov_x(1) = 1$. This mapping is monotonic, but we get $s_x(2) = 0$ and $s_x(1) = 1$, which is not monotonic (and this case can happen for real). That is why condition 2 in the definition of S is weaker than monotonicity.

LEMMA 5.10 *For each x, the mapping s_x belongs to S.*

Proof.

1. If k is odd, then $cov_x(k)$ is odd (Lemma 5.9). The second line in the definition of s_x has to be taken, and hence $s_x(k) = cov_x(k)$ is odd.

2. Let $k' > k$, k' odd such that $s_x(k)$ and $s_x(k')$ are defined. From the definition of s_x, $s_x(k) \leq cov_x(k)$. Still from the definition, and by Lemma 5.9, $s_x(k') = cov_x(k')$ is odd. Furthermore by Lemma 5.9, $cov_x(k') \geq cov_x(k)$. Hence $s_x(k') = cov_x(k') \geq cov_x(k) \geq s_x(k)$. □

Thus, we have assigned to each node x some $s_x \in S$, defining a unique run $\rho_{\mathcal{D}^P_{i,j}}$ of $\mathcal{D}^P_{i,j}$ on $c_{\mathcal{A}}(\rho_A)$. It remains to show that this run of $\mathcal{D}^P_{i,j}$ is accepting and that its value is bounded by some N that can depend on $\mathcal{A}$ and $\mathcal{B}$ but not on the specific run under consideration (nor the underlying tree). Our first step is to show that $\rho_{\mathcal{D}^P_{i,j}}$ is accepting for the parity condition.

LEMMA 5.11 *The run $\rho_{\mathcal{D}^P_{i,j}}$ is accepting for the parity condition.*

Proof. Recall that the priority function of $\mathcal{D}^P_{i,j}$ is defined as

$$c(\langle s', k, s\rangle) = \begin{cases} s_x(k) & \text{if } s(k) \text{ is defined,} \\ i & \text{if } s(k) \text{ is undefined.} \end{cases}$$

This means that the priority assumed by $\rho_{\mathcal{D}^P_{i,j}}$ is $s_x(c_{\mathcal{A}}(x))$ if defined, and i otherwise.

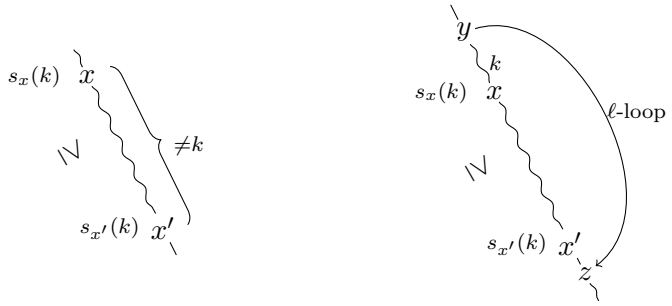


Figure 5.2: Illustration of Lemma 5.12 (on the left) and Lemma 5.13 (on the right)

Take a branch π and let k and ℓ be the maximal priorities in $\rho_{\mathcal{A}}$ and $\rho_{\mathcal{B}}$, respectively, that appear infinitely often on this branch. Both k and ℓ are even since $\rho_{\mathcal{A}}$ and $\rho_{\mathcal{B}}$ are accepting. Then there is a suffix π' of π on which k and ℓ are the maximal priorities of $\mathcal{A}$ and $\mathcal{B}$, and furthermore each position on π' is k-covered in some ℓ-loop. Then $cov_x(k) \geq \ell$ by definition of $cov_x(k)$. Since k is even, $s_x(k)$ is the maximal even priority below $cov_x(k)$. Since ℓ is even we conclude $s_x(k) \geq \ell$. This priority $s_x(k)$ is seen infinitely often on π'. It remains to show that no higher odd priority is seen infinitely often.

Assume that x is a node on π' such that $c_{\mathcal{A}}(x) = k'$ and $s_x(k') = n$ is odd. Two cases can happen. Either $n = i$ is odd, and in this case $n < \ell$, a contradiction. Otherwise, by definition of s_x we obtain that k' is odd. Because k is the maximal priority on π' and k is even, we get $k' < k$. Since $s_x(k') = n$ is defined, k' dominates some n-loop (y', z'). By the choice of π' we know that x is also k-covered in an ℓ-loop (y, z). Because (y', z') is dominated by k' and not by k we conclude that $y \sqsubset y' \sqsubseteq x$. Thus, $n \leq \ell$ (definition of an ℓ-loop). And for parity reasons, we get $n < \ell$. Hence, all odd priorities of $\rho_{\mathcal{D}^P_{i,j}}$ on π' are smaller than $s_x(k)$. □

We now turn to the distance value. Recall that each $k \in P$ induces a reset R_{k-1} and an increment I_k. We want to show that there is an N such that for each k the number of times that I_k occurs without any $I_{k'}$ or $R_{k'-1}$ for $k' > k$ in-between is bounded by N. We start by presenting Lemmas 5.12 and 5.13 that are concerned with the evolution of the value of $s_x(k)$ when x changes. The statements of the two lemmas are illustrated in Figure 5.2. They show that the value of $s_x(k)$ cannot increase along a path under certain conditions. Hence we know that if this value changes, which causes an increment I_k, then it can only decrease (under the conditions specified in the lemmas). This can only happen a bounded number of times often and is the basis for finding the desired bound N.

In the following we always assume that the value "undefined" is smaller than all other values.

LEMMA 5.12 *Let $x \sqsubset x'$ be such that $c_{\mathcal{A}}(u) \neq k$ for all $x \sqsubseteq u \sqsubseteq x'$. Then $s_x(k) \geq s_{x'}(k)$.*

Proof. If $s_x(k) = s_{x'}(k)$, then the claim holds. So assume that $s_x(k) \neq s_{x'}(k)$.

If $s_x(k)$ is undefined, then x' is k-covered in an ℓ'-loop (y', z') in which x is not k-covered (or that does not even contain x). Then the position k-covering x' must be between x and x', contradicting the assumption that $c_{\mathcal{A}}(u) \neq k$ for all $x \sqsubseteq u \sqsubseteq x'$. Hence $s_x(k)$ must be defined. If $s_{x'}(k)$ is undefined, then the claim obviously holds.

Let $\ell = cov_x(k)$ and $\ell' = cov_{x'}(k)$. Then x is covered by k in an ℓ-loop (y, z) and x' is covered by k in an ℓ'-loop (y', z'). Assume that $\ell' > \ell$. Then y' cannot be between y and z since this would contradict the fact that (y, z) is an ℓ-loop. If y' is above y, then x would also be k covered in the ℓ'-loop (y', z') and hence $cov_x(k) \geq \ell'$. If y' is below y, then it has to be below x because otherwise it would be between y and z. But then the position covering x' in the loop (y', z') must be between x and x', contradicting the assumption that $c_{\mathcal{A}}(u) \neq k$ for all $x \sqsubseteq u \sqsubseteq x'$.

We obtain that $cov_x(k) \geq cov_{x'}(k)$, and because $s_x(k) \neq s_{x'}(k)$ implies that $cov_x(k) \neq cov_{x'}(k)$ we have $cov_x(k) > cov_{x'}(k)$. Hence we get $s_x(k) \geq cov_x(k) - 1 \geq cov_{x'}(k) \geq s_{x'}(k)$. □

LEMMA 5.13 *Let (y, z) be a k-dominated ℓ-loop and $x \sqsubset x'$ positions that are k-covered in (y, z). Then $s_x(k) \geq s_{x'}(k)$.*

Proof. Assume that $s_x(k) < s_{x'}(k)$. Then x' must be k-covered in an ℓ'-loop (y', z') with $\ell' > \ell$ such that x is not k-covered in this ℓ'-loop. From $\ell' > \ell$, we deduce that y' cannot be between y and z and hence $y' \sqsubset y$. But then x is also k-covered in (y', z') because it is in (y, z). □

Based on these properties we now want to bound the number of each I_k without intermediate occurrences of higher counter instructions. By $d(x)$ we denote the counter instruction at node x in the run $\rho_{\mathcal{D}^P_{i,j}}$, i.e., $d(x) = d(\langle s_u, c_{\mathcal{A}}(x), s_x\rangle$ where u is the parent node of x (see the definition of the distance function d of $\mathcal{D}^P_{i,j}$ on page 91).

Let β be a finite segment of a branch such that $\max\{d(x) \mid x \in \beta\} = I_k$, i.e., I_k is the maximal counter instruction occurring on this segment. By 'segment of a branch' we mean that β is linearly ordered, and $x, x' \in \beta$ with $x \sqsubset x'$ implies that $u \in \beta$ for all $x \sqsubset u \sqsubset x'$. Let $X_k = \{x \in \beta \mid d(x) = I_k\}$ be the positions on β with counter instruction I_k. We want to find a uniform bound on the size of X_k. We order the positions in X_k as $x_1 \sqsubset \cdots \sqsubset x_{|X_k|}$. Our goal is to prove that $|X_k|$ is bounded by a value that depends solely on $\mathcal{A}$ and $\mathcal{B}$ but not on the runs or the input tree. We first prove a bound for

segments inside ℓ-loops in the following lemma. The general case is treated in Lemma 5.15.

LEMMA 5.14 *Let (y, z) be an ℓ-loop with $y, z \in \beta$. There cannot be more than $2(j - i + 2)$ elements from X_k between y and z .*

Proof. We show the claim of the lemma by considering two parts of the loop. The first part is an initial segment that only contains $c_{\mathcal{A}}$-values smaller than k. We show that on both parts the s-values assigned to the positions in X_k by $\mathcal{D}^P_{i,j}$ must strictly decrease, which can only happen $j - i + 1$ times.

Let x with $y \sqsubseteq x \sqsubset z$ be such that $c_A(u) < k$ for all $y \sqsubseteq u \sqsubset x$. Consider two positions x_m, x_{m+1} of X_k with $y \sqsubseteq x_m \sqsubset x_{m+1} \sqsubset x$. We show that $s_{x_m}(k) \neq s_{s_{m+1}}(k)$ and then apply Lemma 5.12. Because the positions in X_k are those with distance value I_k and because I_k is the maximal distance value on β, we know that the s-value of k can only change at positions in X_k. Furthermore, by the choice of x, $c_{\mathcal{A}}(x_m) < k$ and $c_{\mathcal{A}}(x_{m+1}) < k$. Hence, the only way to obtain distance value I_k is a change of the s-value for k. We conclude that $s_{x_m}(k) \neq s_{x_{m+1}}(k)$ and by Lemma 5.12 that $s_{x_m}(k) > s_{x_{m+1}}(k)$. There can be at most $j - i + 1$ such decreases. Thus, if (y, z) contains more than $j - i + 2$ positions from X_k, we can choose x such that $c_{\mathcal{A}}(x) = k$ and the part between y and x contains at most $j - i + 2$ positions from X_k. All the positions from X_k that are between x and z are thus k-covered (no priority higher than k can occur in β because $c_{\mathcal{A}}(x) > k$ implies that $d(x) > I_k$). Hence, we can use Lemma 5.13 to infer that the values assigned to these X_k positions by $\mathcal{D}^P_{i,j}$ cannot increase. They even strictly decrease because all positions are k-covered, and thus the only way to obtain distance value I_k is a change of the s-value of k. Thus, on the second part of the loop (y, z) there can also be at most $j - i + 2$ positions from X_k. □

Finally we can give a bound in all cases.

LEMMA 5.15 $|X_k| \leq (2(j - i + 2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}| + 1)^{j-i+1}$.

Proof. Let $\ell = \max\{c_{\mathcal{B}}(x) \mid x \in \beta\}$ be the maximal $\mathcal{B}$-priority on β. For each $m \in \{1, \ldots, |X_k| - 1\}$ let $\ell_m = \max\{c_{\mathcal{B}}(x) \mid x_m \sqsubset x \sqsubseteq x_{m+1}\}$. Assume that ℓ appears more than $2(j - i + 2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}|$ times among the ℓ_m. Then there are two positions y', z' in β that form an ℓ-loop such that this ℓ-loop contains more than $2(j - i + 2)$ positions from X_k. This contradicts Lemma 5.14. Thus, there are at most $2(j - i + 2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}|$ of the ℓ_m that are equal to ℓ.

If $\ell = i$, then this implies that $|X_k|$ is bounded by $2(j-i+2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}|+1$. For $\ell = i + h$ with $h > 0$, we can split β in at most $2(j - i + 2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}| + 1$ pieces on which the maximal $\mathcal{B}$-priority is less than ℓ. By induction hypothesis, each of these pieces contains at most $(2(j-i+2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}|+1)^h$ elements from X_k. In total we get that $|X_k| \leq (2(j - i + 2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}| + 1)^{j-i+1}$. □

Putting together Lemma 5.11 and Lemma 5.15, we obtain the following.

LEMMA 5.16 $val(\rho_{\mathcal{D}^P_{i,j}}) \leq (2(j-i+2)|Q_\mathcal{A}||Q_\mathcal{B}|+1)^{j-i+1}$.

Summing up, there is an N such that for each tree $t \in T(\mathcal{B})$ we can find a run $\rho_\mathcal{A}$ of $\mathcal{A}$ on t and a run $\rho_{\mathcal{D}^P_{i,j}}$ on $c_\mathcal{A}(\rho_\mathcal{A})$ such that $\rho_{\mathcal{D}^P_{i,j}}$ is accepting (Lemma 5.11), and $val(\rho_{\mathcal{D}^P_{i,j}}) \leq N$ (Lemma 5.16). This concludes the proof of Lemma 5.7 and thus of Theorem 5.5.

5.4 Discussion

In this chapter we have shown that we can reduce the parity index problem to the uniform universality problem for distance PTAs. Unfortunately, we have to leave open the question whether the latter problem is decidable. But we would like to point out that the problem is decidable for models of distance automata on finite words, infinite words, and finite trees. For finite words it has been solved by Kirsten in [64] in the context of deciding the star-height of regular languages. For infinite words the decidability can be derived from results in [13] on hierarchical B- and S-automata. In [2] the uniform universality problem has been solved for finite and infinite words for the case that the counters are not ordered, i.e., the instructions I_j and R_j only manipulate the counter number j.

The reason why it is difficult to lift these results to trees is that the techniques for solving the limitedness or uniform universality problem on words use algebraic methods and factorizations of sequences. The semantics of the automaton is captured by a semigroup and all computations are carried out in this algebraic object. For tree automata there is no corresponding algebraic object to capture its behavior. Nevertheless, the results on the decidability of the star-height of regular languages are lifted to finite trees in [31]. To be able to use algebraic methods, the tree automaton is first converted into a special form that almost behaves like a word automaton and therefore can be captured by a semigroup. This special form of automata is called purely nondeterministic in [31]. A purely nondeterministic automaton can only send a state to one of the successors instead of both successors. The other subtree is not considered in the run. Thus, each run of such an automaton corresponds to a path. Obviously, this is a rather strong restriction and hence it might be surprising that it is possible to convert each automaton into a purely nondeterministic one. But the transformation does not result in an equivalent automaton, it only preserves uniform universality.

The technique used in the transformation is based on games. In Chapter 1 we have seen that acceptance of a tree by an automaton can be captured by a membership game (see Lemma 1.14). This also works in the

setting of finite trees, with the only difference that the resulting game is of finite duration and the last state of the game determines the winner. The key argument for the transformation is a positional determinacy result for membership games that result from distance automata on finite trees. This allows to annotate trees with strategies (in a similar way as presented in Section 1.4). The strategy annotation determines at each node in the tree for each chosen transition of the original automaton a direction in the tree, and the state that is finally reached at the leaf determines the winner of the game. Hence, to check if a strategy is winning on a particular run of the original automaton, it suffices to check a single path. This can be done by a purely nondeterministic automaton.

To lift this technique to infinite trees a positional (or finite memory) determinacy result for distance parity conditions is needed. It turns out that the presence of the parity condition makes the problem much harder.

Chapter 6

Finite set interpretations on trees

In Section 2.4 we have introduced finite set interpretations as a method for constructing structures with decidable FO-theory from structures with decidable WMSO-theory (see Proposition 2.10). This is a very strong method that already allows to construct structures like $(\mathbb{N}, +)$ and $(\mathbb{N}, \cdot)$ from the very basic trees $\mathfrak{t}_1$ and $\mathfrak{t}_2$. In fact, from these two trees we already obtain the classes of all automatic and tree-automatic structures as we have explained in Section 2.5. The power of finite set interpretations makes it difficult to obtain results for the general case where such interpretations are applied to arbitrary structures. For this reason, in this chapter we only consider the special case of finite set interpretations applied to deterministic trees. This restriction can be justified in two ways. The first justification is that on trees there are specific tools suitable for treating WMSO questions, namely tree automata. The second justification is that the main interest in finite set interpretation is the construction of structures with decidable FO-theory using Proposition 2.10. If Seese's conjecture [109] holds, stating that all structures of decidable WMSO theory are WMSO-interpretations of trees, then the only structures that we can prove to have a decidable FO-theory using Proposition 2.10 are finite set interpretations of trees (for recent work on Seese's conjecture see [37]).

We give here two results concerning finite set interpretations applied to deterministic trees. The first one shows that finite set interpretations on deterministic trees followed by a quotient are simply finite set interpretations (Theorem 6.2). The second result (Theorem 6.5) concerns finite set interpretations producing powerset lattices when applied to trees. We demonstrate the use of Theorems 6.2 and 6.5 by showing some non-definability results, the strictness of the hierarchy mentioned above, and a result on intrinsic definability of relations related to similar questions studied for automatic structures (cf. [7]).

6.1 Notations and terminology

The trees introduced in Section 1.3 always have the full domain $\{0,1\}^*$. In this chapter we consider trees t that can have finite branches, i.e., whose domain $dom(t)$ is a prefix closed subset of $\{0,1\}^*$.

Of course it is possible to code such trees by a tree over the full domain by adding a new label, say $\perp$, and labeling all nodes outside of $dom(t)$ by $\perp$. But when applying interpretations to trees to generate new structures, it makes a difference whether we consider a tree over smaller domain or its coding over the full domain: Applied to the coding over the full domain interpretations are more powerful because they can access the elements that are not in the domain of the original tree. Consider, for example, the tree $\mathfrak{t}_1$ corresponding to the natural numbers with successor. In Section 2.5 we have seen that finite set interpretations can generate exactly the word-automatic structures from $\mathfrak{t}_1$, whereas from $\mathfrak{t}_2$ all tree-automatic structures can be generated. Hence, coding $\mathfrak{t}_1$ by a corresponding labeling of $\mathfrak{t}_2$ makes a difference concerning the class of structures that can be generated by finite set interpretations.

However, for technical reasons we assume that each node u of $dom(t)$ has either two successors or no successor. We use this restriction to keep the notations and definitions for automata running on these trees simple. All the results can also be transferred to the setting without this restriction.

We have to adapt the model of tree automaton to be able to deal with leafs in the tree. This is achieved by adding a new type of transitions to the transition relation. The transition relation of a parity tree automaton (PTA) $\mathcal{A} = (Q, \Sigma, q^{\mathsf{i}}, \Delta, c)$ is now of the form $\Delta \subseteq (Q \times \Sigma \times Q \times Q) \cup (Q \times \Sigma)$.

A run of $\mathcal{A}$ on a tree t is a mapping $\rho : dom(t) \to Q$ such that

- $(\rho(u), t(u), \rho(u0), \rho(u1)) \in \Delta$ for all inner nodes u of $dom(t)$, and
- $(\rho(u), t(u)) \in \Delta$ for all leaves u of $dom(t)$.

A run is accepting if the parity condition is satisfied an all infinite branches of t (see Section 1.3, page 21).

We are interested here in automata running on a fixed underlying tree $t : dom(t) \to \Sigma$ with additional markings representing (tuples of) subsets of its domain. To mark a certain subset U of $dom(t)$ we can put additional labels on the tree. This definition is similar to the one of characteristic tree of a set (see page 21) with the difference that we now start from a labeled tree and put additional labels to code the set (the characteristic tree only had labels for coding the set).

Formally, the tree t annotated by U is the tree with the same domain as t and labels from $\Sigma \times \{0,1\}$, where a node u is labeled by the pair $(t(u), 0)$ if $u \notin U$ and $(t(u), 1)$ if $u \in U$. In the same way one can also annotate a

tree with tuples of subsets of its domain using a separate $\{0,1\}$-component for each set.

If t is fixed and $U_1, \ldots, U_n$ are subsets of $dom(t)$, then we say that an automaton accepts the tuple $(U_1, \ldots, U_n)$ if it accepts t with the additional labelings corresponding to the tuple $(U_1, \ldots, U_n)$ as explained above. If we consider all the tuples accepted by an automaton, we obtain a relation over the subsets of $dom(t)$. We call this relation the relation recognized or accepted by the automaton.

We write $(t, U_1, \ldots, U_n)$ for the tree that is annotated with $U_1, \ldots, U_n$.

A WMSO-formula with free set variables $X_1, \ldots, X_n$ also defines a relation over the subsets of $dom(t)$. Throughout the paper we make use of the fact that for each WMSO-formula one can construct a PTA that for each tree accepts precisely those relations that are defined by the formula. In comparison to Theorem 2.1 there are some differences: We now use WMSO instead of MSO, and we consider trees that can have finite branches. Since we are only interested in the translation of formulas into automata, the use of WMSO does not pose any problems because on deterministic trees each WMSO-formula can be translated into an equivalent MSO-formula and hence we are only talking about a subset of MSO. The use of more general trees is not problematic because in the standard inductive translation of formulas into automata one can easily handle trees with finite branches.

Because this result can be obtained by minor variations of the arguments for proving Theorem 2.1 we also attribute it to [102].

THEOREM 6.1 ([102]) *For each WMSO-formula ϕ there is a PTA $\mathcal{A}$ such that for each tree t the relation over subsets of $dom(t)$ defined by ϕ is the same as the one accepted by $\mathcal{A}$.*

6.2 Quotient structures

We show here that if a structure is finite set interpretable in a deterministic tree containing a symbol interpreted as a congruence on the structure, then it is possible to directly obtain the quotiented structure by a finite set interpretation.

A *congruence* on a structure $\mathcal{S}$ is an equivalence relation $\sim$ such that for every symbol R of arity n and all elements $x_1, \ldots, x_n, y_1, \ldots, y_n$ of $\mathcal{S}$: if $x_1 \sim y_1, \ldots, x_n \sim y_n$, then $R^{\mathcal{S}}(x_1, \ldots, x_n)$ iff $R^{\mathcal{S}}(y_1, \ldots, y_n)$. We say that a symbol is a congruence if its interpretation in the structure is a congruence. For a congruence $\sim$ over a structure $\mathcal{S}$, we denote by $\mathcal{S}/_\sim$ the *quotient structure*, i.e., the structure which has as elements the equivalence classes of $\sim$, and the relations of which are the images of the relations on $\mathcal{S}$ under the canonical surjection induced by $\sim$: If we denote the equivalence class of an element x of $\mathcal{S}$ by $[x]_\sim$, then the relation $R^{\mathcal{S}/\sim}$ contains all the tuples $([x_1]_\sim, \ldots, [x_{|R|}]_\sim)$ such that $(x_1, \ldots, x_{|R|}) \in R^{\mathcal{S}}$.

Note that the operation of quotienting preserves the decidability of the FO-theory: Given an FO-formula on the structure $\mathcal{S}/_\sim$ we can transfer it to a formula on $\mathcal{S}$ by replacing all atomic formulas of the kind $x = y$ by $x \sim y$ and all formulas of the form $x \neq y$ by $x \not\sim y$.

For this reason we may wonder if constructing a structure by a finite set interpretation followed by a quotient is more powerful than solely using a finite set interpretation. Theorem 6.2 below shows that it is not the case when the original structure is a tree. A tempting approach to show such a result would be to select one representative from each equivalence class of $\sim$ by means of a WMSO-formula. This would require to choose from each $\sim$-class one of the finite sets representing the elements of the $\sim$-classes. The results from Chapter 3 on the undefinability of choice functions suggest that such an approach is bound to fail.

The proof of the following theorem uses a different method to code each equivalence class of finite sets by a single finite set.

THEOREM 6.2 *Given a finite set interpretation $\mathcal{I}$, there exists a finite set interpretation $\mathcal{I}'$ such that for every tree t, if the symbol $\sim$ is a congruence in $\mathcal{I}(t)$, then $\mathcal{I}'(t)$ is isomorphic to $\mathcal{I}(t)/_\sim$.*

Proof. Let t be a tree and let $\phi_\sim(X, Y)$ be the formula defining the relation $\sim$, which evaluates to a congruence in $\mathcal{I}(t)$. Our aim is to associate to each $\sim$-class c a single finite set U_c of nodes in a WMSO-definable way (such that the sets for different classes are different).

We start by associating to each set a WMSO-definable description in such a way that for sets from one class we only have finitely many different descriptions. Then we can pick for each class the smallest of these descriptions for a suitable WMSO-definable ordering.

For a $\sim$-class c we let its border $Bor(c)$ be the set of all nodes u with the following properties:

1. There is a set $U \in c$ that does not contain any nodes from the subtree rooted at u.

2. The subtree rooted at u is infinite.

The set of nodes that are not in the subtree of one of the border elements is called the common prefix of the class c and denoted by $CP(c)$. Note that $Bor(c)$ as well as $CP(c)$ are finite sets. The common prefix can also be obtained by taking the prefix closure of all sets in c, intersecting all these prefix closures, and then closing the set by finite subtrees rooted in the set. Since all members in c are finite sets, the common prefix is also finite. And the border of c is the set of all minimal nodes that are not in the common prefix.

We can easily construct a formula $\phi_{Bor}(X, Z)$ that evaluates to true if Z is interpreted as the border of the class of X, and a formula $\phi_{CP}(X, Z)$

that evaluates to true if Z is interpreted as the common prefix of the class of X. Note that expressing that a subtree below some node is infinite can be done in WMSO by stating that there is no finite set covering the whole subtree.

The description of a finite set U consists of the border of its class, the set U restricted to the common prefix of its class, and a function f that describes the behavior of an automaton for the relation $\sim$ on the set U below the border elements.

This is the only place in this chapter where we need to describe the behavior of some automaton by means of a WMSO-formula. For nondeterministic parity automata this is not possible in general (one needs full MSO for this). But there are automata theoretic characterizations of weakly definable tree languages: Rabin [103] showed that these are exactly the languages that can be accepted by a Büchi tree automaton and whose complement can also be accepted by a Büchi tree automaton. In [88] it is shown that weakly definable languages are those that can be accepted by weak alternating automata. The details of the definition of weak alternating automata is not needed for this proof. We simply use the fact that this model characterizes the weakly definable languages.

So let $\mathcal{A}$ be a weak alternating automaton for the formula $\phi_\sim(X, Y)$. We are interested in the behavior of $\mathcal{A}$ on the tree t with additional annotations for the sets X and Y.

Now we can specify the description of a set U that is in some $\sim$-class c. It is given by the triple $(Bor(c), U \cap CP(c), f)$, where f is a function that associates to each node $u \in Bor(c)$ the set of states of $\mathcal{A}$ for which $\mathcal{A}$ has an accepting run on the subtree of $(t, U, \emptyset)$ rooted at u. That is, we look at the subtree at u annotated with U and $\emptyset$. The reason is that we know that for each node u on the border of c there exists an element U' of c that does not contain any elements below this node. So the function f describes how the automaton $\mathcal{A}$ behaves below u on the tree (t, U, U').

Note that for each class there are only finitely many descriptions possible for its elements: The border is finite and fixed for the class, and the common prefix of the class is finite thus only allowing finitely many different sets as second component of the description.

We now show that if two sets U and V have the same description, then they are in the same class. Let c be the class of U and let $u_1, \ldots, u_n$ be the nodes in $Bor(c)$ below which U and V differ. We show the claim by induction on n. If $n = 0$, then U and V are the same: They have the same description, therefore the borders of their classes coincide. Hence, the common prefixes of their classes also coincide. The only part where U and V can differ is on the common prefixes of their classes, but the second components of the descriptions contain exactly this information.

If $n \geq 1$, then obtain the set W by changing U as follows: Remove all nodes below u_1 from U and add those of V that are below u_1. Since

$u_1 \in Bor(c)$, there is some set U' in c that does not have any nodes below u_1, and because U' is in c there is an accepting run ρ of $\mathcal{A}$ on (t, U, U'). It is now easy to construct an accepting run ρ' of $\mathcal{A}$ on (t, W, U'): Let t_1 be the subtree of t at u_1. If ρ contains a subrun on t_1 that starts in some state q, then we substitute it by an accepting run of $\mathcal{A}$ from q on the subtree corresponding to t_1 in $(t, V, \emptyset)$. This is legal since U' does not contain any nodes below u, and W coincides with V on t_1. The other parts of the run remain unchanged since U and W coincide on the corresponding parts of the tree. Since $\mathcal{A}$ accepts (t, W, U'), we obtain that $W \sim U'$ and by transitivity $U \sim W$. This implies that W is in the same class as U. By construction of W from U and V we can conclude that W has the same description as U (and V). Since W differs from V only below the nodes $u_2, \ldots, u_n$ we obtain by induction that $W \sim V$ and thus $U \sim V$.

Let us remark now that a description $(Bor(c), U \cap CP(c), f)$ can be encoded uniquely by a set of nodes: This set is

$$Bor(c) \cup (U \cap CP(c)) \cup Coding(f)$$

where $Coding(f)$ contains exactly one element for each element U of the border, and this element is located in a place uniquely describing the value of $f(u)$ (e.g. the leftmost node at distance $g(f(u))$ below u, where g is a numbering of 2^Q, the power set of the state set of $\mathcal{A}$). This is possible since the subtree rooted in u is infinite.

Note that associating to a set U the coding of its description is doable by means of a WMSO-formula. We already mentioned that given a class c there is only a finite number of descriptions for the elements it contains. Hence we can choose the smallest description (for a suitable total order) as unique representative for the class. A suitable total order can, e.g., use the lexicographically smallest node that is in the symmetric difference of the two sets coding the two descriptions.

We have shown that we can associate to each $\sim$-class a unique finite set in a WMSO-definable way. Based on this it is not difficult to rewrite the interpretation $\mathcal{I}$ to obtain $\mathcal{I}(t)/_\sim$. □

In combination with Proposition 2.15 we obtain the following.

COROLLARY 6.3 *Tree-automatic structures are effectively closed under quotient.*

In the terminology of [9], this result is rephrased as "every tree-automatic structure admits an injective presentation." Let us remark that this result is announced in [9], but unfortunately the proof proposed there contains an unrecoverable error.

6.3 Powerset lattices

The result presented in the previous section is a positive result in the sense that it allows to prove that certain structures can be defined by finite set interpretations. In this section we develop a technique that can be used to show that it is not possible to obtain a certain structure by a finite set interpretation from a given structure. This technique reduces the question of finite set interpretability to a question of WMSO-interpretability. Questions of the latter type are usually easier to handle.

Consider, for example, the question whether it is possible to obtain the binary tree $\mathfrak{t}_2$ by a WMSO-interpretation from $\mathfrak{t}_1$. Using automata-theoretic techniques it is quite easy to show that this is not possible. Now we lift this question to the level of finite set interpretations by asking whether it is possible to obtain the weak powerset structure $\mathcal{P}^W(\mathfrak{t}_2)$ of $\mathfrak{t}_2$ by a finite set interpretation from $\mathfrak{t}_1$. Such an interpretation would have to produce the structure of $\mathfrak{t}_2$, and additionally all finite subsets of the domain of $\mathfrak{t}_2$ together with the inclusion relation. We have already seen that we can easily built $\mathfrak{t}_2$ (even extensions of it) from $\mathfrak{t}_1$ by a finite set interpretation (see Figure 2.3 on page 37). So it might be possible that we can use a similar coding that allows to construct $\mathcal{P}^W(\mathfrak{t}_2)$.

On the other hand, the intuition says that we already use the power of finite sets for coding the structure of $\mathfrak{t}_2$, and there is no possibility to additionally code the complex structure of the finite power set of $\mathfrak{t}_2$. And in fact, we show that this intuition is correct. Applied to this very special situation, we show that if there is a finite set interpretation $\mathcal{I}$ such that $\mathcal{I}(\mathfrak{t}_1) = \mathcal{P}^W(\mathfrak{t}_2)$, then there is also a WMSO-interpretation $\mathcal{I}_2$ such that $\mathcal{I}_2(\mathfrak{t}_1) = \mathfrak{t}_2$. As already mentioned, the latter statement can easily be falsified.

Our result can be applied for arbitrary trees t instead of $\mathfrak{t}_1$ and arbitrary structures $\mathcal{S}$ instead of $\mathfrak{t}_2$. This version of the result is presented in Theorem 6.5 below. The technical core of the result is given in Lemma 6.4. It basically states that every finite set interpretation constructing a so called finite powerset lattice from a tree can be rewritten in such a way that it maps singleton sets of the tree to singleton sets of the lattice. Here, a finite powerset lattice is a powerset structure with all relations but $\subseteq$ removed.

We start by stating the main technical result and the main theorem and variants of it. Then we turn to the proof of the technical result.

Let $\mathcal{S}$ be a structure of signature $\preceq$. We say that $\mathcal{S}$ is a *finite powerset lattice* if it is isomorphic to $(\mathcal{P}^F(E), \subseteq)$ for some set E, where $\mathcal{P}^F(E)$ represents the finite subsets of E. Such a finite powerset lattice can be seen as a particular case of weak powerset generator applied to a vocabulary-free structure. We call the elements corresponding to singletons in this isomorphic structure *atoms*, i.e., those elements which have exactly one element strictly smaller with respect to $\preceq$.

LEMMA 6.4 *For every finite set interpretation $\mathcal{I} = (\phi_{dom}(X), \phi_{\preceq}(X, Y))$, there exists a WMSO-formula $Code(X, x)$ such that, whenever for some tree t, $\mathcal{I}(t)$ is a finite powerset lattice, then $Code(X, x)$ evaluates on t to an injection that maps the atoms of $\mathcal{I}(t)$ to nodes of t.*

Before we come to the rather long and involved proof of this result, we give some consequences of it.

THEOREM 6.5 *For every finite set interpretation $\mathcal{I}$, there exists a WMSO-interpretation $\mathcal{I}_2$ such that whenever for some structure $\mathcal{S}$ and some tree t, $\mathcal{I}(t)$ is isomorphic to $\mathcal{P}^W(\mathcal{S})$ then $\mathcal{I}_2(t)$ is isomorphic to $\mathcal{S}$.*

Proof. If we remove all relations other than $\preceq$, the weak powerset generator is nothing but a finite powerset lattice. Hence we can obtain a formula $Code(X, x)$ by application of Lemma 6.4. It is then easy to transfer all relations defined on singletons to their image by *Code*.

Formally, we define the WMSO-interpretation $\mathcal{I}_2 = (\phi_{dom}, \phi_{R_1}, \ldots, \phi_{R_l})$ as follows:

- $\phi_{dom}(x) = \exists X. Code(X, x)$,
- for each symbol R of arity r from the signature, $\phi_R(x_1, \ldots, x_r)$ is defined as

$$\exists X_1, \ldots, X_r. \psi_R(X_1, \ldots, X_r) \wedge \bigwedge_{i=1}^{r} Code(X_i, x_i)$$

  where each ψ_R is the WMSO-formula in $\mathcal{I}$ defining the interpretation of the symbol R.

As *Code* maps each atom of $\mathcal{I}(t)$ to a unique node, this interpretation indeed maps t to a structure isomorphic to $\mathcal{S}$. □

A weaker, yet more readable formulation of the above statement is provided in the following corollary. The situation is depicted in Figure 6.1.

COROLLARY 6.6 *If for a structure $\mathcal{S}$ and a tree t the class of structures that can be obtained by finite set interpretations from $\mathcal{S}$ is contained in the class of structures that can be obtained by finite set interpretations from t, then $\mathcal{S}$ is WMSO-interpretable in t.*

Proof. Assume that the hypothesis of the corollary holds. In particular, the structure $\mathcal{P}^W(\mathcal{S})$ is isomorphic to $\mathcal{I}(t)$ for some finite set interpretation $\mathcal{I}$. By applying Theorem 6.5, the structure $\mathcal{S}$ is WMSO-interpretable in t. □

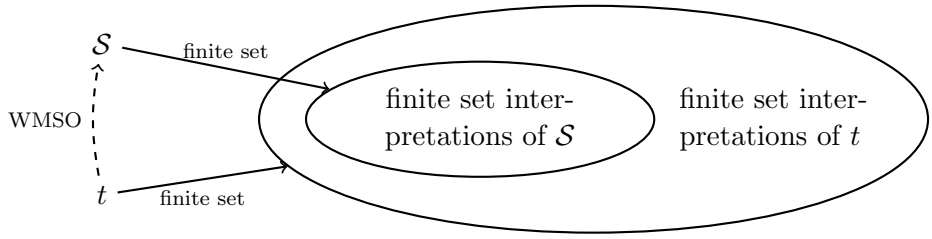


Figure 6.1: The situation from Corollary 6.6

Proof of Lemma 6.4

The proof of Lemma 6.4 is rather complex and split into several parts. We start by introducing the key notions used throughout the proof, and we make the scheme of the proof more precise.

We fix a finite set interpretation $\mathcal{I} = (\phi_{dom}(X), \phi_{\preceq}(X, Y))$. Along the whole proof we use a tree $t : dom(t) \to \Sigma$ together with a set E and the isomorphism f that are assumed to satisfy the equality

$$f(\mathcal{P}^F(E)) = \mathcal{I}(t) \ .$$

The reader must keep in mind that none of the constructions we perform makes use of t, E, or f. Hence, the result will hold for any such tree, set, and isomorphism. This lightens the presentation of the proof by avoiding to systematically quantify over those objects.

We consider the set *Atoms* of finite subsets of t representing atoms of the powerset lattice, i.e.,

$$Atoms = \{f(\{u\}) \ : \ u \in E\} \ .$$

The set *Atoms* can be defined as the set of finite subsets of t which are minimal — for the $\phi_{\preceq}$ formula seen as an ordering — and distinct from the minimal element itself (which is $f(\emptyset)$). This description can be done in weak monadic second-order logic. Hence *Atoms* is regular in t and there exists an automaton

$$\mathcal{A}_{Atoms} = (Q_{Atoms}, \Sigma \times \{0,1\}, q^{\mathrm{i}}_{Atoms}, \Delta_{Atoms}, c_{Atoms})$$

accepting the language *Atoms*. We also consider the binary relation *Mem* over $\mathcal{I}(t)$ defined as the image under f of the $\in$ relation in $\mathcal{P}^F(E)$, i.e.,

$$Mem = \{(f(\{u\}), f(V)) \ : \ u \in V \subseteq E \text{ and } V \text{ finite}\} \ .$$

This *Mem* relation is also definable in weak monadic second-order logic, and consequently is regular. We fix

$$\mathcal{A}_{Mem} = (Q_{Mem}, \Sigma \times \{0,1\}^2, q^{\mathrm{i}}_{Mem}, \Delta_{Mem}, c_{Mem})$$

to be an automaton recognizing the relation *Mem*.

Recall that the statement we want to prove claims the existence of a formula $Code(X, x)$ such that the corresponding relation is an injection from *Atoms* into $dom(t)$.

Our goal in the construction of *Code* is to uniquely attach to each X in *Atoms* an element in $dom(t)$ in a WMSO-definable way. As a first approximation we define a mapping *Index* which assigns to each X in *Atoms* a node in $dom(t)$. Though the *Index* mapping is not in general an injection from *Atoms* into $dom(t)$, it does not either concentrate a lot of indices in the same area of the tree t. Formally, if we set $D(x)$ for $x \in dom(t)$ to be the cardinality of $Index^{-1}(x)$, then by Lemmas 6.12 and 6.14, D happens to be a sparse distribution (see Definition 6.7 below).

In the part "car parking" we study sparse distributions. The central lemma of this part, Lemma 6.19, establishes that, given elements concentrated according to a sparse distribution, we can uniformly redistribute them in $dom(t)$ in a unique WMSO-definable way. Applied to our case, this means that the *Index* mapping can be transformed into an injection by use of WMSO-formulas. And this last step is used at the end of this section to finish the proof of Lemma 6.4.

The key definition connecting the two main parts of the proof (definition of the *Index* mapping and turning it into an injection) is the notion of sparsity. This definition requires the notion of zone. A *zone* Z in t is a connected subset of $dom(t)$, where t is seen as an non-oriented graph. That is, Z contains a minimal element w.r.t. to the prefix ordering, and whenever $x \sqsubseteq y \sqsubseteq z$ for $x, z \in Z$, then also $y \in Z$. A zone Z is completely characterized by its least element x, and by the minimal elements that are below x and not in Z. We call these elements the *frontier* of the zone.

Given nodes $x, x_1, \ldots, x_n$ of t such that the x_i are pairwise incomparable and $x \sqsubseteq x_i$ for all i, we define $Z_t^{x,x_1,\ldots,x_n}$ to be the set of nodes y such that $x \sqsubseteq y$ and $x_i \not\sqsubseteq y$ for all $i \in [n]$ where $[n]$ denotes the set $\{1, \ldots n\}$. By construction, $Z_t^{x,x_1,\ldots,x_n}$ is the only zone which has frontier $\{x, x_1, \ldots, x_n\}$.

The following two special cases of zones appear in several places and thus we mention them here.

- For $x \in dom(t)$ the zone Z_t^x has x as minimal element and there are no elements below x that are not in Z_t^x. Hence, Z_t^x contains precisely the nodes in the subtree of t rooted at x.

- The zone $Z_t^{\epsilon,x}$ has the root of t as minimal element, and the only nodes not in $Z_t^{\epsilon,x}$ are those that are in the subtree rooted at x. Hence $Z_t^{\epsilon,x}$ contains all nodes that are *not* in the subtree of t rooted at x.

We now define the notion of sparse distribution.

DEFINITION 6.7 A *distribution* D is a mapping from $dom(t)$ to $\mathbb{N}$. For a finite zone Z we write $D(Z)$ for $\sum_{x \in Z} D(x)$. A distribution D is *K-sparse* for some $K \in \mathbb{N}$ if for every finite zone Z with frontier F, $D(Z) \leq |Z| + K|F|$. A distribution is *strongly K-sparse* if for any finite zone Z of frontier F, $D(Z) \leq K|F|$.

If we view $D(x)$ as a number of tokens placed on the node x, then sparsity means that each finite zone contains at most as many tokens as the size of the zone plus a factor linearly depending on the size of the frontier. This means that it is enough to push at most K tokens through each exit at the frontier of the zone to be left with a number of tokens that is at most the size of the zone.

Strong sparsity means that we can empty the zone by pushing at most K tokens through each exit at the frontier.

Clearly, a distribution needs to be bounded to be sparse. If t is a complete tree (i.e., there are no finite branches), then boundedness of a distribution is enough for it being sparse because the size of the frontier of each zone is of the same order as the size of the zone. In the case of incomplete trees boundedness and sparsity differ because there may be arbitrarily large zones with a small number of frontier nodes.

As mentioned above, we first construct a mapping *Index* that maps atoms to elements of $dom(t)$ such that the resulting distribution is sparse. Then we show how to turn a mapping inducing a sparse distribution into an injection.

Important nodes

In order to construct the mapping *Index*, given an element X of *Atoms*, we first define the set $Imp(X) \subseteq dom(t)$ of its important nodes via combinatorial constraints. Essentially, we try to locate the places where "important coding decisions" are made by the automaton $\mathcal{A}_{Atoms}$ when reading X.

In the following, we provide the key combinatorial lemmas concerning important nodes. Then, depending on the shape of the set $Imp(X)$ two cases are separated and two distinct definitions of *Index* are given. The first kind of index is called standard index, denoted $SIndex(X)$, and the other kind is called branch index, denoted $BIndex(X)$.

Let us first introduce a convenient notation for studying the behavior of the automata $\mathcal{A}_{Atoms}$ and $\mathcal{A}_{Mem}$ over zones: For a zone $Z = Z_t^{x,x_1,\ldots,x_n}$ and states $q, q_1, \ldots, q_n \in Q_{Atoms}$, we denote by $Atoms(q, Z, q_1, \ldots, q_n)$ the set of all $X \subseteq Z$ that can be extended to an atom X' that is accepted by $\mathcal{A}_{Atoms}$ with the corresponding states at the frontier of Z:

$X \subseteq Z$ is in $Atoms(q, Z, q_1, \ldots, q_n)$ if there exists $X' \subseteq dom(t)$ with

- $X = X' \cap Z$, and

- X' is accepted by $\mathcal{A}_{Atoms}$ with a run ρ such that $\rho(x) = q$ and $\rho(x_i) = q_i$ for all $i \in [n]$.

Similarly, for $q, q_1, \ldots, q_n \in Q_{Mem}$ we denote by $Mem(q, Z, q_1, \ldots, q_n)$ the set of all pairs (X, Y) with $X, Y \subseteq Z$ such that there are $X', Y' \subseteq dom(t)$ with

- $X = X' \cap Z$, $Y = Y' \cap Z$, and
- (X', Y') is accepted by $\mathcal{A}_{Mem}$ with a run ρ such that $\rho(x) = q$ and $\rho(x_i) = q_i$ for all $i \in [n]$.

The definition of important nodes is then the following, where the constant K_{im} is chosen to make the combinatorial arguments in the subsequent lemmas work. A node x is important for X if there are many — i.e., more than K_{im} — ways to modify X below x while remaining in $Atoms$. Intuitively, without knowing how X looks like below x, we cannot say much about which atom is coded because there are too many possibilities left. The formal definition looks as follows.

Definition 6.8 Let $K_{\text{im}} = (2|Q_{Mem}| + 1)|Q_{Atoms}|$. Given $X \in Atoms$, a node $x \in dom(t)$ is called *important* for X if

$$|\{Y \subseteq Z_t^x \ : \ (X - Z_t^x) \cup Y \in Atoms\}| > K_{\text{im}} \ .$$

We denote by $Imp(X)$ the set of important nodes for X.

Remark that the set $Imp(X)$ is by definition prefix closed. The fundamental property that we show in Lemma 6.10 is that for an important node x of X, the part of X that is not below x comes from a set of small size. To prove this lemma we need its combinatorial core stated in the following lemma.

Lemma 6.9 *Let $K_{\text{c}} = 2|Q_{Mem}| + 1$. For any two disjoint zones Z and Z' of respective frontiers $F = \{x, x_1, \ldots, x_n\}$ and $F' = \{x', x'_1, \ldots, x'_m\}$, and all accepting runs ρ of $\mathcal{A}_{Atoms}$*

$$\begin{aligned} \textit{either} \quad & |Atoms(\rho(x), Z, \rho(x_1), \ldots, \rho(x_n))| < K_{\text{c}} \ , \\ \textit{or} \quad & |Atoms(\rho(x'), Z', \rho(x'_1), \ldots, \rho(x'_m))| < K_{\text{c}} \ . \end{aligned}$$

Proof. It is sufficient for us to prove the result for two complementary zones. This comes from the fact that increasing a zone also increases the number of possible projections w.r.t. a fixed run, i.e.,

$$|Atoms(\rho(x), Z, \rho(x_1), \ldots, \rho(x_n))| \leq |Atoms(\rho(y), Z'', \rho(y_1), \ldots, \rho(y_\ell))|$$

for a zone Z'' of frontier $\{y, y_1, \ldots, y_\ell\}$ with $Z \subseteq Z''$. Hence, we will assume Z to be $Z_t^{\epsilon,x}$ and Z' to be Z_t^x for some node x.

Assume that for some $K \geq K_{\mathrm{c}}$ we have distinct sets $X_1, \ldots, X_K$ in $Atoms(\rho(\epsilon), Z, \rho(x))$ and distinct sets $X'_1, \ldots, X'_K$ in $Atoms(\rho(x), Z')$. Then, for every $i, j \in [K]$, let $Y_{i,j}$ be $X_i \cup X'_j$. As the union of Z and Z' gives the whole domain of t, we have $Y_{i,j} \in Atoms$ for all $i, j \in [K]$.

Let us now consider the set $Comb$ of possible combinations of the $Y_{i,j}$, combination in the sense of the relation Mem. More precisely, $A \subseteq dom(t)$ is in $Comb$ if whenever $(Y, A) \in Mem$ holds for some atom Y, then $Y = Y_{i,j}$ for some i, j. The cardinality of $Comb$ is 2^{K^2}. We now show by a combinatorial argument that $\mathcal{A}_{Mem}$ cannot distinguish all the elements from $Comb$ because the amount of information that can be passed between the two zones Z and Z' is limited by the number of states in Q_{Mem}.

For this purpose, we define for each $A \in Comb$, $f_A : [K] \times Q_{Mem} \to \{0, 1\}$ and $g_A : Q_{Mem} \times [K] \to \{0, 1\}$ by

$$f_A(i, q) = \begin{cases} 1 & \text{if } (X_i, A \cap Z) \in Mem(q^{\mathrm{i}}_{Mem}, Z, q), \\ 0 & \text{else,} \end{cases}$$

$$g_A(q, j) = \begin{cases} 1 & \text{if } (X'_j, A \cap Z') \in Mem(q, Z'), \\ 0 & \text{else.} \end{cases}$$

It is obvious that if two sets $A, B \in Comb$ are such that $f_A = f_B$ and $g_A = g_B$, then $(Y_{i,j}, A) \in Mem$ iff $(Y_{i,j}, B) \in Mem$ for all $i, j \in [K]$. This means, by definition of $Comb$, that $A = B$.

However, there are only $2^{2|Q_{Mem}|K}$ different possible pairs f_A, g_A of functions. Hence we obtain $|Comb| \leq 2^{2|Q_{Mem}|K}$. This contradicts $|Comb| = 2^{K^2}$. □

The following lemma shows that the possibilities to code an atom 'above' an important node are bounded.

Lemma 6.10 *For each node x we have*

$$|\{X \cap Z_t^{\epsilon,x} \ : \ X \in Atoms \text{ and } x \in Imp(X)\}| < K_{\mathrm{im}}.$$

Proof. We are aiming at a contradiction to Lemma 6.9 for $Z = Z_t^{\epsilon,x}$ and $Z' = Z_t^x$.

For each X with $x \in Imp(X)$ there are more than K_{im} many $Y \subseteq Z_t^x$ such that $X_{Y,x} := (X - Z_t^x) \cup Y$ is in $Atoms$. Since $K_{\mathrm{im}} = |Q_{Atoms}| \cdot K_{\mathrm{c}}$ (with K_{c} from Lemma 6.9), we can choose a state $q_{X,x} \in Q_{Atoms}$ such that more than K_{c} of these $X_{Y,x}$ are accepted by $\mathcal{A}_{Atoms}$ with a run that labels x with $q_{X,x}$. This means that $|Atoms(q_{X,x}, Z_t^x)| \geq K_{\mathrm{c}}$.

Now, assume that there are K_{im} different $X \in Atoms$ with $x \in Imp(X)$ that differ on $Z_t^{\epsilon,x}$. Then there are at least K_{c} such sets $X_1, \ldots, X_{K_{\mathrm{c}}}$ with $q_{X_1,x} = \cdots = q_{X_{K_{\mathrm{c}}},x} =: q$. In particular, we obtain $|Atoms(q^{\mathrm{i}}_{Atoms}, Z_t^x, q)| \geq$

K_c. Together with $|Atoms(q, Z_t^x)| \geq K_c$ from above we obtain the desired contradiction. □

Standard indices

We now address the problem of computing $Index(X)$ for some atom X under the assumption that $Imp(X)$ is *not an infinite branch* (the case when $Imp(X)$ is an infinite branch is treated subsequently). Since we call this case the standard case, we will denote the index defined for such atoms X by $SIndex(X)$. The simplest case is that $Imp(X)$ is totally ordered by $\sqsubseteq$, i.e., $Imp(X)$ is a finite path starting from the root. We define $SIndex(X)$ to be the last node on this path. The other case corresponds to $Imp(X)$ not being a finite path nor an infinite branch. This corresponds to $Imp(X)$ not being totally ordered by $\sqsubseteq$. In this situation, we define $SIndex(X)$ to be the first node at which $Imp(X)$ splits into two paths. Those two cases are unified in the following definition.

Definition 6.11 For $X \in Atoms$ such that $Imp(X)$ is not an infinite branch, the *index* of X, denoted by $SIndex(X)$, is the maximal element of $Imp(X)$ which is comparable to every element of $Imp(X)$.

As already mentioned, the intention of this definition is that $SIndex(X)$ roughly locates in the tree where the main information concerning the atom coded by X lies. This location is far from being precise, and many elements of $Atoms$ may have the same index. However, we will see that it is possible to obtain a good understanding of the repartition of the standard indices. The following lemma gives precise bounds on the quantity of indices that may occur in a zone, i.e., it states that the distribution assigning to each node x the number of X such that $SIndex(X) = x$ is sparse.

Lemma 6.12 *There is a constant K_s such that $|SIndex^{-1}(Z)| \leq |Z| + K_s|F|$ for every finite zone Z of frontier F.*

Proof. Denote the elements of the frontier of Z by x and $x_1, \ldots, x_n$, i.e., $Z = Z_t^{x,x_1,\ldots,x_n}$. The proof of the lemma consists of two steps. We first show that for atoms X such that $SIndex(X)$ is inside Z, the amount of information located outside Z is bounded. More precisely, we first show for $M := K_{im} \cdot |Q_{Atoms}|$

(a) $|\{X \cap Z_t^{\epsilon,x} \ : \ X \in Atoms \text{ and } SIndex(X) \in Z\}| < M$ and

(b) $|\{X \cap Z_t^{x_i} \ : \ X \in Atoms \text{ and } SIndex(X) \in Z\}| < M$ for all $i \in [n]$.

For (a) note that from $SIndex(X) \in Z$, the definition of $SIndex$, and the prefix closure of $Imp(X)$ we obtain that $x \in Imp(X)$. Therefore,

$$\begin{aligned}&\{X \cap Z_t^{\epsilon,x} \ : \ X \in Atoms \text{ and } SIndex(X) \in Z\}\\ &\subseteq \{X \cap Z_t^{\epsilon,x} \ : \ X \in Atoms \text{ and } x \in Imp(X)\}\end{aligned}$$

and (a) follows from Lemma 6.10.

For (b) we show that for each $X \in SIndex^{-1}(Z)$ and each x_i there is a state $q \in Q_{Atoms}$ such that $X \cap Z_t^{x_i} \in Atoms(q, Z_t^{x_i})$ and $|Atoms(q, Z_t^{x_i})| < K_{\mathrm{im}}$. From this, (b) follows because each $X \cap Z_t^{x_i}$ comes from one of at most $|Q_{Atoms}|$ many sets of size less than K_{im}. We distinguish two cases.

If $x_i \notin Imp(X)$, then we take q to be the state at x_i in an accepting run of $\mathcal{A}_{Atoms}$ on X. From the definition of $Imp(X)$ we immediately obtain the desired property.

Else, if $x_i \in Imp(X)$, by definition of standard indices there must some $y \in Imp(X)$ incomparable to x_i, the index of X being the deepest common ancestor of x_i and y. From the definition of important nodes for X and from $K_{\mathrm{im}} = |Q_{Atoms}| \cdot K_{\mathrm{c}}$ we obtain that there exists a set $Y \subseteq Z_t^y$ such that $(X - Z_t^y) \cup Y$ is in $Atoms$ and is accepted with a run of $\mathcal{A}_{Atoms}$ that labels y by a state q' such that $|Atoms(q', Z_t^y)| \geq K_{\mathrm{c}}$. Let q be the state assumed at node x_i by this run. From Lemma 6.9 applied to the zones $Z_t^{x_i}$, Z_t^y, and to the aforementioned run, we can conclude that $|Atoms(q, Z_t^{x_i})| < K_{\mathrm{c}}$. The desired property follows from $K_{\mathrm{c}} \leq K_{\mathrm{im}}$. This finishes the proof of (b).

After these preliminary considerations, we come back to the claim of the lemma. We denote the elements from $\{X \cap Z_t^{\epsilon,x} \ : \ SIndex(X) \in Z\}$ by $X^1, \ldots, X^M$ and the elements from $\{X \cap Z_t^{x_i} \ : \ SIndex(X) \in Z\}$ by $X_i^1, \ldots, X_i^M$ (the same element can be represented more than once, what is important is that all elements are represented).

Now, consider the set $Comb$ of all combinations of atoms whose index falls inside Z, i.e., the atoms from $SIndex^{-1}(Z)$ (combinations in the same sense as in the proof of Lemma 6.9). A combination $A \in Comb$ is entirely characterized by the following objects

- the set $A \cap Z$,
- the mapping $f_{A,x} : [M] \times Q_{Mem} \to \{0, 1\}$ with

$$f_{A,x}(j, q) = \begin{cases} 1 & \text{if } \ (X^j, A \cap Z_t^{\epsilon,x}) \in Mem(q_{Mem}^{\mathrm{i}}, Z_t^{\epsilon,x}, q), \\ 0 & \text{else,} \end{cases}$$

- and the mapping $f_{A,x_i} : Q_{Mem} \times [M] \to \{0, 1\}$ with

$$f_{A,x_i}(q, j) = \begin{cases} 1 & \text{if } \ (X_i^j, A \cap Z_t^{x_i}) \in Mem(q, Z_t^{x_i}), \\ 0 & \text{else.} \end{cases}$$

Thus, $|Comb| \leq 2^{|Z|} \cdot 2^{M|Q_{Mem}|} \cdot \prod_{i=1}^{n} 2^{|Q_{Mem}|M}$. For $K_s = |Q_{Mem}|M$ we obtain

$$2^{|SIndex^{-1}(Z)|} = |Comb| \leq 2^{|Z|+|F|K_s}$$

and hence $|SIndex^{-1}(Z)| \leq |Z| + K_s|F|$. □

Treatment of infinite branches

It is possible that for some $X \in Atoms$ the set $Imp(X)$ of important nodes is an infinite branch. For these X we also develop a notion of index, called $BIndex(X)$, and show that the distribution obtained in this way is strongly sparse. Since the sum of a K-sparse distribution and of a strongly K'-sparse distribution is a $K + K'$-sparse distribution, we can add the indices corresponding to infinite branches to the other indices without affecting the sparsity of the induced distribution.

We call the infinite branches that are equal to $Imp(X)$ for some atom X *important branches*. We start with the helpful observation that the number of elements of $Atoms$ corresponding to the same important branch is bounded.

LEMMA 6.13 *For every important branch B, $|I^{-1}(B)| < K_{im}$.*

Proof. If there are K_{im} different sets in $I^{-1}(B)$, then we can pick a node x on B such that all these sets differ on the zone $Z_t^{\epsilon,x}$. Since x is important for all X in $I^{-1}(B)$, we obtain a contradiction to Lemma 6.10. □

This observation is already the key for defining $BIndex(X)$. If $Imp(X) = B$, then we consider all atoms that have B as set of important nodes. Then there is some node x on B such that all these atoms are coded above x, i.e., they are contained in $Z_t^{\epsilon,x}$. We take $BIndex(X)$ as the minimal such x. Formally, for $X \in Atoms$ with $Imp(X) = B$ we let

$$BIndex(X) = \min\{x \in B \mid \forall Y \in Atoms \ : \ Imp(Y) = B \rightarrow Y \subseteq Z_t^{\epsilon,x}\}.$$

It remains to show that the distribution induced by $BIndex$ is indeed strongly sparse:

LEMMA 6.14 *For Z a finite zone of frontier F, $|BIndex^{-1}(Z)| \leq K_{im}^2|F|$.*

Proof. The node $BIndex(X)$ by definition only depends on B where $B = Imp(X)$. Hence it is legal to define the index of an important branch B as $BIndex(X)$ for some X with $Imp(X) = B$. Let $x, x_1, \ldots, x_n$ be the nodes on the frontier of Z. For x_i consider the important branches $B_1, \ldots, B_m$ whose index is inside Z and that leave Z through x_i. Let y be the maximal index of the branches $B_1, \ldots, B_m$. Then all atoms Y with $Imp(Y) = B_j$ for some

j are completely contained in $Z_t^{\epsilon,y}$. By Lemma 6.10 we obtain that there are less than K_{im} many such branches, i.e., $m < K_{\text{im}}$.

Furthermore, according to Lemma 6.13, for each important branch, there are at most K_{im} many atoms having this branch as set of important nodes.

We obtain that for each frontier node of the zone there are at most $|K_{\text{im}}|^2$ many atoms whose *BIndex* is inside Z and whose important branch exits Z through this frontier node. Hence we obtain that $|BIndex^{-1}(Z)| \leq K_{\text{im}}^2|F|$. □

Now we let $Index(X)$ be either $SIndex(X)$ or $BIndex(X)$ depending on the shape $Imp(X)$. According to Lemma 6.14 and Lemma 6.12 this yields a sparse distribution. In the next part we study such distributions. At the end of this section we combine all the results to finish the proof of Lemma 6.4.

Car parking

Our goal is to spread the indices around the tree such that each index ends in exactly one position. This can be seen as parking vehicles. In the beginning there are cars (indices) placed in the nodes of the tree, possibly more than one at the same position, and we aim at parking each of them in one node, i.e., attaching a single node to each of those cars. This is obviously not possible in general but we shall prove that, under a sparsity constraint on the distribution of vehicles, it is possible to attach a single parking place to each car, and furthermore that the mapping that, given a car, tells where to park it, can be described by a WMSO-formula.

For this purpose, we have to describe distributions and other kinds of mappings that involve integers in their domain or image by WMSO-formulas. These integers will always be bounded by some constant K and hence we can split the WMSO-definition into several formulas, one for each number that may be involved. Formally, we say that a relation $R \subseteq dom(t)^r \times I$ for some finite $I \subseteq \mathbb{Z}$ is WMSO-definable if there are WMSO-formulas $\phi_i(x_1, \ldots, x_r)$ for each $i \in I$ such that $(u_1, \ldots, u_r, i) \in R$ iff $t \models \phi_i[u_1, \ldots, u_r]$. Note that a K-sparse distribution can be seen as a relation of this kind since $0 \leq D(x) \leq 3K$ for every $x \in dom(t)$.

Definition 6.15 Given a distribution D, a *placement* P for D is an injective partial mapping from $dom(t) \times \mathbb{N}$ to $dom(t)$ such that $P(x, i)$ is defined iff $i \in [D(x)]$.

A flow, defined below, can be seen as a kind of instruction on how to spread the values of a distribution to obtain a placement. In the vehicles description, this is the number of cars which will have to cross an edge in order to reach the final placement.

Recall that, for simplicity reasons, we assume that all the nodes of a tree t have either 2 successors or no successors, i.e., for all nodes u we have

$u0 \in dom(t)$ iff $u1 \in dom(t)$. This assumption is not essential but simplifies the definitions and allows to avoid case distinctions.

DEFINITION 6.16 A *flow* is a mapping f from the nodes of t to $\mathbb{Z}$. A flow f is *compatible* with a distribution D if for all inner nodes x, $D(x) + f(x) \leq 1 + f(x0) + f(x1)$ and for every leaf x, $D(x) + f(x) \leq 1$. A flow f is *bounded* by a constant K if $|f(x)| \leq K$ for each node x.

In this definition, $f(x)$ is interpreted as the number of cars crossing the edge from the ancestor of x to x. In case of a negative value, $-f(x)$ cars are driving from x to its ancestor. The condition of f being compatible with D states that after distributing all the cars according to the flow there is at most one car remaining at each node. One should note here that, according to our definition, it is possible that $f(\epsilon) < 0$. With the above intuition this would mean that one has to send $-f(\epsilon)$ cars to the (non-existing) ancestor of the root. We need this case when constructing flows on finite subtrees of a given infinite tree.

In the following we show that for a K-sparse distribution there is a compatible flow that is bounded. From this flow we then compute a placement for the distribution. We start by defining a flow on finite trees (which can then also be used to deal with finite subtrees of a given infinite tree).

LEMMA 6.17 *For every finite tree t and every WMSO-definable K-sparse distribution D over t, there exists a WMSO-definable flow f that is compatible with D, bounded by $2K + 1$, and such that for each node x there is a zone Z_x rooted in x of frontier F_x with*

(1) $D(Z_x) + f(x) = |Z_x| + K(|F_x| - 1)$,

(2) and $f(y) \geq K$ *for all* $y \in F_x$ *different from* x.

Proof. First note that (1) implies $f(x) \geq -K$ for each x since D is K-sparse. We define the values $f(x)$ and the zones Z_x inductively starting at the leaves. These definitions directly imply that f is compatible with D.

The base case of a leaf x is straightforward: we set $f(x)$ to be $1 - D(x)$ and $Z_x = \{x\}$. By the hypothesis of sparsity, we have that $D(x) \leq K + 1$ and hence $|f(x)|$ is bounded by $2K + 1$.

Let x be an inner node and assume that the values $f(x0)$, $f(x1)$ and the zones Z_{x0}, Z_{x1} are already defined. We set $f'(x0) = \min(K, f(x0))$ and $f'(x1) = \min(K, f(x1))$. Let us now define $f(x)$ to be $1 + f'(x0) + f'(x1) - D(x)$ and Z_x to contain the node x, the nodes of Z_{x0} if $f(x0) < K$, and the nodes of Z_{x1} if $f(x1) < K$. By the hypothesis of induction, we indeed obtain that $D(Z_x) + f(x) = |Z_x| + K(|F_x| - 1)$. We illustrate this only for the case $f'(x0) < K$ and $f'(x1) = K$, the other cases are similar. In this

case $Z_x = Z_{x0} \cup \{x\}$, $|F_x| = |F_{x0}| + 1$, and $f(x) = 1 + f(x0) + K - D(x)$. From this we get the following sequence of equalities:

$$\begin{aligned} D(Z_x) + f(x) &= D(Z_{x0}) + D(x) + 1 + f(x0) + K - D(x) \\ &= D(Z_{x0}) + f(x0) + 1 + K \\ &= |Z_{x0}| + K(|F_{x0}| - 1) + 1 + K \\ &= |Z_{x0}| + 1 + K|F_{x0}| \\ &= |Z_x| + K(|F_x| - 1). \end{aligned}$$

From the definition of $f(x)$ it is clear that $f(x) \leq 2K + 1$. As mentioned before, $f(x) \geq -K$ and thus $|f(x)|$ is bounded by $2K + 1$.

It is obvious that this flow, which has only a bounded number of possible values and is defined inductively, is WMSO-definable. This definition can be done by requiring the existence of sets $X_{-K}, \ldots, X_{2K+1}$ such that a node x is in X_i iff $f(x) = i$. This can be directly expressed if x is a leaf. Otherwise it is a simple statement on the membership of the successors $x0$ and $x1$ of x. □

Lemma 6.18 *For every infinite tree t and every WMSO-definable K-sparse distribution D, there exists a WMSO-definable flow f bounded by $7K$ that is compatible with D such that $f(\epsilon) = 0$.*

Proof. As a K-sparse distribution restricted to a finite subtree of t is also K-sparse on this subtree, we can apply Lemma 6.17 to define the values of $f(x)$ for the nodes x that are not on infinite branches of t.

Let B be the set of nodes appearing in some infinite branch. We define inductively for any node $x \in B$ a flow $f(x)$ such that $0 \leq f(x) \leq 7K$. We only consider non-negative values since on infinite branches we never reach a leaf and hence there is no need for an upward flow.

We start by setting $f(\epsilon) = 0$. For $x \neq \epsilon$ let $y \in B$ be the father of x. Three cases may happen. If at node y only one of its children is in B (case (a)), we forward everything to this node. Otherwise (cases (b) and (c)), we forward at most $5K$ to the left child and the rest to the right child. The formal definitions are given below, where the max operator is only used to avoid negative flows.

(a) If x is the only child of y in B, then we set

$$f(x) = \max(0, f(y) + D(y) - f(x') - 1)$$

where x' is the other child of y.

(b) If the two children of y are in B and x is the left child, then we set

$$f(x) = \max(0, \min(5K, f(y) + D(y) - 1)).$$

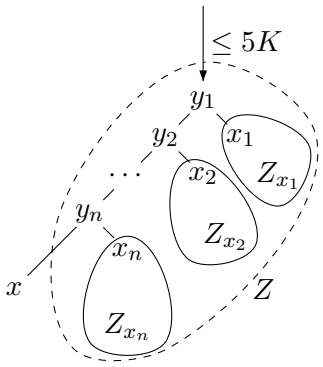


Figure 6.2: Proof of Lemma 6.18: The flow in nodes of type (a) is bounded by $7K$.

(c) If the two children of y are in B and x is the right child, then we set

$$f(x) = \max(0, f(y) + D(y) - 1 - 5K).$$

Obviously, $f(x) \geq 0$ in all cases. We show that if x is of type (b) or (c), then $f(x) \leq 5K$, and if x is of type (a), then $f(x) \leq 7K$.

In case (b), $f(x) \leq 5K$ follows directly from the definition and in case (c) from $f(y) \leq 7K$ (by induction) and $D(y) \leq 3K + 1$ (D is K-sparse).

For x as in (a) we cannot use a local argument but we have to go upwards until we reach a node that has a flow of at most $5K$. Such a node must exist because we eventually meet a node of type (b) or (c), or the root, which has flow 0. All nodes we met before must be of type (a).

The following definitions are illustrated in Figure 6.2. The choice of the y_i being the left successors in the figure is arbitrary and only for matters of presentation. Let $y_n, \ldots, y_1$ be such that y_n is the father of x, y_{i-1} is the father of y_i for all $i \in \{2, \ldots, n\}$, $f(y_1) \leq 5K$, and $f(y_2), \ldots, f(y_n) > 5K$. As mentioned above, $y_2, \ldots, y_n$ are of type (a).

Let $x_1, \ldots, x_n$ be such that y_i is the father of x_i, $x_n \neq x$, and $x_i \neq y_{i+1}$ for $i \in \{1, \ldots, n-1\}$, i.e., x_i is the brother of y_{i+1}. For the x_i the flow is defined using Lemma 6.17 because they are not lying on an infinite branch. Hence there are zones Z_{x_i} rooted at x_i of frontier F_{x_i} such that

$$D(Z_{x_i}) + f(x_i) = |Z_{x_i}| + K(|F_{x_i}| - 1). \tag{6.1}$$

Let $Z = \bigcup_{i=1}^{n}(\{y_i\} \cup Z_{x_i})$ and let F be the frontier of Z. Then

$$|Z| = n + \sum_{i=1}^{n} |Z_{x_i}| \tag{6.2}$$

$$|F| = 2 + \sum_{i=1}^{n} (|F_{x_i}| - 1) \tag{6.3}$$

$$\sum_{i=1}^{n} D(y_i) = D(Z) - \sum_{i=1}^{n} D(Z_{x_i}) \tag{6.4}$$

Since $y_2, \ldots, y_n$ are of type (a) and furthermore their flow is bigger than $5K$ (and hence bigger than 0), we get

$$f(x) = f(y_1) + \sum_{i=1}^{n} (D(y_i) - 1 - f(x_i)).$$

We know that $f(y_1) \leq 5K$ and hence it remains to be shown that

$$\sum_{i=1}^{n} (D(y_i) - 1 - f(x_i)) \leq 2K.$$

This can be deduced as follows:

$$\begin{aligned}
\sum_{i=1}^{n} (D(y_i) - 1 - f(x_i)) &\overset{(6.4)}{=} D(Z) - n - \sum_{i=1}^{n} (D(Z_{x_i}) + f(x_i)) \\
&\overset{(6.1)}{=} D(Z) - n - \sum_{i=1}^{n} (|Z_{x_i}| + K(|F_{x_i}| - 1)) \\
&\overset{(6.3)}{=} D(Z) - n - K(|F| - 2) - \sum_{i=1}^{n} |Z_{x_i}| \\
&\overset{K\text{-sparse}}{\leq} |Z| + K|F| - n - K(|F| - 2) - \sum_{i=1}^{n} |Z_{x_i}| \\
&\overset{(6.2)}{=} 2K
\end{aligned}$$

That this flow f is compatible with D and that it is WMSO-definable can easily be deduced from the definitions. □

We are now ready to establish our placement Lemma.

Lemma 6.19 (car parking) *For every tree t and every WMSO-definable K-sparse distribution D, there exists a WMSO-definable placement for D. If t is finite, we additionally require that $D(dom(t)) \leq |dom(t)|$.*

Proof. According to Lemmas 6.17 and 6.18 we know that there is a WMSO-definable flow f that is compatible with D. For simplicity, we first assume that $f(\epsilon) = 0$, which is always the case for infinite trees (Lemma 6.18). If $f(\epsilon) > 0$ for finite trees, then we can simply redefine $f(\epsilon) = 0$ without changing the property of f being compatible with D. If t is finite and

$f(\epsilon) < 0$, then we cannot simply set $f(\varepsilon) = 0$ because this would affect the compatibility of f with D. At the end of the proof we briefly explain how to treat this case.

The general strategy for defining the placement is the following:

- From each node we send all the cars except one to its neighbors. The number of cars sent to each neighbor is described by the flow.
- If we follow this strategy, then each edge in t is crossed by the cars in only one direction. Hence, a car cannot visit the same node twice. This means that it might be sent up in the tree for some steps and from some point onwards it is only sent downwards.
- To be sure that each car will be parked after finite time we order all the cars that cross a node (as described by D and f) according to a fixed strategy and we also fix a scheme for distributing the cars to the neighboring nodes.
- This ordering will ensure that the index of a car decreases each time it is sent down in the tree. As described above, each car is sent up in the tree only a finite number of times. Hence, if we always park the car that is first in the ordering at a specific node, then each car will eventually be parked.

To show that this strategy can be realized by WMSO-formulas we first describe the ordering of cars that we use and then define formulas

- $\text{send}_i(x, y)$ meaning that the ith car at node x is sent to node y.
- $\text{drive}_{i,j}(x, y)$ meaning that the ith car at node x is sent to y and is car number j at y.
- $\text{start}_i(x)$ meaning that the ith car at node x does not come from another node.
- $\text{itinerary}_i(x, X_1, \ldots, X_K, y)$ meaning that the ith car at node x will be parked at node y using an itinerary that is described by the sets $X_1, \ldots, X_K$.

To define these formulas we first have to introduce some notation. To avoid case distinctions we define $f^+(x) = \max(f(x), 0)$ and $f^-(x) = \min(f(x), 0)$. Furthermore, we assume by convention that for a leaf x the values $f^+(x0)$, $f^+(x1)$, $f^-(x0)$, $f^-(x1)$ are defined, and are all set to 0.

Then the number of cars crossing a node x is $f^+(x) + f^-(x0) + f^-(x1) + D(x)$. Since all the values involved in this expression are bounded and since we can increase K without affecting the K-sparsity of D, we can assume that $f^+(x) + f^-(x0) + f^-(x1) + D(x) \leq K$ for all x.

Since f is WMSO-definable, we can also assume that there are formulas $\phi_i^+(x)$ and $\phi_i^-(x)$ defining f^+ and f^-. Then expressions of the form $i_1 + f^+(x) + f^-(y) = i_2$ for $i_1, i_2 \in [K]$ can easily be expressed as Boolean combinations of the formulas ϕ_i^+ and ϕ_i^-. The use of expressions of this kind simplifies the presentation of the formulas.

We start by giving the orderings used in the definitions of the formulas send_i and $\text{drive}_{i,j}$. The cars that cross a node x will be distributed in the following order that we refer to as the distribution order:

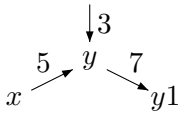
$f^+(x)$	$f^-(x0)$	$f^-(x1)$	$D(x)$

That is, we first distribute the cars that come from the father of x, then the cars that come from the left child of x, and so on. It remains to fix where to send the cars.

1	$f^-(x)$	$f^+(x0)$	$f^+(x1)$

This means that the first car in the distribution order is parked in the node x. The next cars are sent to the father of x if $f(x)$ is negative. The following cars in the distribution order are send to the left child of x and the remaining cars to the right child of x.

To illustrate this, consider the following example with $x = y0$, $f^+(y) = 3$, $f^-(x) = 5$, and $f^+(y1) = 7$.



Let us first see how the cars at y are ordered according to the distribution order. The first three are the ones coming from the father of y. The next 5 are those coming from x. There are no cars coming from $y1$, and the last cars are those from $D(y)$. Now, we would like to know what happens to the 4th car at x. The first car at x stays at x. The next 5 cars are sent to y, that is, the fourth car at x is the third car sent from x to y. According to the distribution order at y described before, this car becomes car number 6 at y.

This is expressed by the following formulas, where the first two are only defined for $2 \leq i \leq K$ because the first car is always kept at the current position.

- For $2 \leq i \leq K$:

$$\begin{aligned} \text{send}_i(x,y) := \; & [(x = y0 \vee x = y1) \wedge 1 < i \leq C_1] \\ & \vee [x0 = y \wedge C_1 < i \leq C_2] \\ & \vee [x1 = y \wedge C_2 < i \leq C_3] \end{aligned}$$

Here $C_1 = 1 + f^-(x)$, $C_2 = 1 + f^-(x) + f^+(x0)$, and $C_3 = 1 + f^-(x) + f^+(x0) + f^+(x1)$.

- For $2 \leq i \leq K$:

$$\begin{aligned}\mathrm{drive}_{i,j}(x,y) := \;& \mathrm{send}_i(x,y) \wedge \\ & \big([x = y0 \wedge j = i - 1 + f^+(y)] \\ & \vee [x = y1 \wedge j = i - 1 + f^+(y) + f^-(y0)] \\ & \vee [x0 = y \wedge j = i - 1 - f^-(x)] \\ & \vee [x1 = y \wedge j = i - 1 - f^-(x) - f^+(x0)]\big)\end{aligned}$$

- For $1 \leq i \leq K$:

$$\begin{aligned}\mathrm{start}_i(x) := \;& f^+(x) + f^-(x0) + f^-(x1) < i \wedge \\ & i \leq f^+(x) + f^-(x0) + f^-(x1) + D(x)\end{aligned}$$

- For $1 \leq i \leq K$:

  $\mathrm{itinerary}_i(x, X_1, \ldots, X_K, y) :=$

$$\begin{aligned}& \mathrm{disjoint}(\mathrm{X}_1, \ldots, \mathrm{X}_\mathrm{K}) \wedge \\ & x \in X_i \wedge \mathrm{start}_i(x) \wedge X_1 = \{y\} \\ & \wedge \bigwedge_{j=2}^{K} (\forall z \in X_j \bigvee_{j' \in [K]} \exists z' \in X_{j'} \; : \; \mathrm{drive}_{j,j'}(z,z'))\end{aligned}$$

  This formula states that the ith car at x starts there. The free set variables describe the set of positions that this car crosses, where a position is included in X_m if the car is the mth one at this position. Finally, it states that y is the only position where the car is first in the ordering. Hence it will stop at y.

Then the formulas $\psi_i(x,y)$ defining the placement are given by

$$\psi_i(x,y) = \exists X_1, \ldots X_K(\mathrm{itinerary}_i(x, X_1, \ldots, X_K, y)).$$

As mentioned at the beginning of the proof we now discuss how to treat the case $f(\epsilon) < 0$. Recall that this may only happen for finite trees. In general, simply redefining $f(\epsilon) = 0$ may lead to a flow that is not compatible with D anymore. Therefore, we have to use a different strategy.

The strategy for distributing the cars described above would lead to $f^-(\epsilon)$ cars that "get stuck" at the root because, following the flow, they should be sent upwards, and this is not possible. This means that there are at most K cars (for simplicity assume exactly K cars) that start at some node but never arrive at some destination, i.e., there are nodes $x_1, \ldots, x_K$ and $i_1, \ldots, i_K \in [K]$ such that

$$\mathrm{start}_{i_j}(x_j) \wedge \neg\exists y(\psi_{i_j}(x_j, y))$$

is satisfied. From the assumption $D(dom(t)) \leq |dom(t)|$ we can conclude that there remain at least K nodes where no car is parked, i.e., K nodes which are not image of the function defined by the ψ_i's. Let $y_1, \ldots, y_K$ be the K first such nodes for some WMSO-definable order (this is possible since the tree is finite). We can now extend this function by ordering the $x_1, \ldots, x_K$ and map the i_jth car from x_j to y_j. Note that these definitions are expressible in WMSO. In this way, we obtain a modification of the function defined by the ψ_i's into a placement for D.

Combining the results

We can now prove the main Lemma 6.4 of this section by combining the previous results.

As already mentioned previously, for $X \in Atoms$ we let

$$Index(X) = \begin{cases} BIndex(X) \text{ if } Imp(X) \text{ is an infinite branch,} \\ SIndex(X) \text{ otherwise.} \end{cases}$$

We construct a formula $\phi_{\text{ind}}(X, y)$ that associates to each $X \in Atoms$ its index $y = Index(X)$. From the definitions of $Imp(X)$, $SIndex(X)$, and $BIndex(X)$ it is clear that we can construct such a formula. Note that in the definition of this formula, we do not have to explicitly represent infinite sets (though $Imp(X)$ may be infinite) because we can construct a WMSO formula $\phi_{imp}(X, x)$ that associates to $X \in Atoms$ its important nodes. From this one can construct WMSO definitions of $SIndex(X)$ and $BIndex(X)$, and hence the formula $\phi_{\text{ind}}(X, y)$ is also WMSO.

Then, we compute the distribution D defined by $D(x) = |Index^{-1}(x)|$. Since this distribution is $(K_{\text{s}} + K_{\text{im}}^2)$-sparse by Lemmas 6.12 and 6.14, it is also WMSO-definable using the formula ϕ_{ind}. Let K be a constant such that $D(x) \leq K$ for all nodes x. Applying Lemma 6.19, we obtain a WMSO-definable placement P for D. One should note that for finite t the assumption $D(dom(t)) \leq |dom(t)|$ is satisfied because $D(dom(t))$ is the number of elements in the set E, and $\mathcal{I}(t)$ being isomorphic to $\mathcal{P}^F(E)$ implies that t has at least as many elements as E.

Let $\psi_i(x_1, x_2)$ for $i \in [K]$ be the formulas defining P. Now, we order all the $X \in Atoms$ with the same index. A possible definition for such an ordering is $X < Y$ if the lexicographically smallest node that is not in $X \cap Y$ is in X. Then one can construct WMSO-formulas $\theta_i(X)$ stating that X is the ith set in the ordering among those that have the same index as X.

The WMSO-formula $Code(X, x)$ that attaches X to its final position x is then defined as follows:

$$Code(X, x) = \bigvee_{i \in [K]} (\theta_i(X) \wedge \exists y(\phi_{\text{ind}}(X, y) \wedge \psi_i(y, x)).$$

(Proof of Lemma 6.4) □

One should note here that without the results on sparse distributions we can obtain a weaker version of Corollary 6.5 by replacing the interpretation $\mathcal{I}_2$ by a transduction, i.e., an interpretation that can use a fixed number K of copies of the given structure. Such a transduction can be realized using the formula ϕ_{ind} instead of *Code*. In particular one could use this weaker version of the result in all the applications, but at the price of some notational and technical overheads.

From this point of view, Lemma 6.19 can also be seen as a result on the question under which conditions a (W)MSO-transduction is equivalent to a (W)MSO-interpretation on binary trees. Namely, if the distribution defined by the transduction, i.e., the function assigning to each node the number of times it is used in the result of the transduction, is K-sparse (for some K) on each tree. And this presentation can be used either for WMSO-transductions or MSO-transductions.

6.4 Applications

We present here several applications of the results obtained in this chapter. The first one, showing that the hierarchy of tree-automatic structures introduced in Section 2.5 is strict, is a straightforward application of Theorem 6.5. The second one, showing that the free monoid is not obtainable by a finite set interpretation of a tree, is a combination of Theorem 6.2 and Theorem 6.5. These two applications are typical examples illustrating how our result can be used to easily reduce questions on the non-definability by finite set interpretations to the non-definability by WMSO-interpretations.

The third application requires a more involved argument. It establishes that the random graph is not finite set interpretable in a tree, extending the known result for automatic structures [63].

Strictness of the tree-automatic hierarchy

A simple application of our result is the strictness of the tree-automatic hierarchy introduced at the end of Section 2.5.

THEOREM 6.20 *For each $n \geq 0$ the class TA_{UT_n} of structures on the nth level of the automatic hierarchy is strictly contained in $TA_{UT_{n+1}}$.*

Proof. We know that each level n of the Caucal hierarchy contains a tree generator G_n, i.e., each structure of level n is WMSO-interpretable in G_n [20]. Let us suppose that the automatic hierarchy collapses at some level n, i.e., $\text{TA}_{\text{UT}_n} = \text{TA}_{\text{UT}_{n+1}}$. This would imply that the structure $\mathcal{P}^W(G_{n+1})$ can be obtained by a finite set interpretation from G_n. Then, by Theorem 6.5, we obtain G_{n+1} as a WMSO-interpretation of G_n and hence G_{n+1}

is in the nth level of the hierarchy. This contradicts the strictness of the Caucal hierarchy. □

The free monoid

We consider the free monoid as a structure $(\{a,b\}^*, \cdot, a, b)$ — the set of words over $\{a,b\}$ together with the ternary relation corresponding to the concatenation and the two words a and b identified by unary predicates — and want to answer the question whether this structure is isomorphic to a finite set interpretation of a tree.

One should note that the FO-theory of the free monoid is undecidable and hence we can directly conclude that it cannot be obtained by a finite set interpretation from a tree with a decidable WMSO-theory. However, this reasoning does not include trees with an undecidable WMSO-theory.

The negative answer we give here to the above question is the simplest and in some sense the purest application of the results presented above and should be considered for this reason as a key example.

PROPOSITION 6.21 *The free monoid over a two letter alphabet is not isomorphic to any finite set interpretation of a tree.*

Proof. We first show how to obtain $\mathcal{P}^W(\mathbb{N},+)$ from the free monoid by an FO-interpretation followed by a quotient. Then, assuming that our claim is false, we invoke the two results from the previous section and obtain a contradiction.

Let f be the function which to each word of the form $ba^{n_1}ba^{n_2}b\dots ba^{n_k}b$ over $\{a,b\}$ associates the set of naturals $\{n_1,n_2,\dots,n_k\}$. The domain of f is the set of elements satisfying $dom(x) = \exists y.x = byb$. The relation of inclusion is also FO-definable: $f(u) \subseteq f(v)$ iff $sub(u,v)$ holds, where $sub(u,v)$ is defined by the following formula

$$\forall x \in a^*.\exists y,z.\ u = ybxbz \rightarrow \exists y',z'.v = y'bxbz' \ ,$$
$$\text{where } x \in a^* \text{ stands for } \forall\ y,z.x \neq ybz \ .$$

Let $x \sim y$ be the formula $sub(x,y) \wedge sub(y,x)$. Naturally, for every u,v, $u \sim v$ iff $f(u) = f(v)$. Finally the addition over singletons is definable. More precisely, $f(u) = \{i\}$, $f(v) = \{j\}$ and $f(w) = \{i+j\}$ iff $add(u,v,w)$ holds with $add(u,v,w) =$

$$\exists x \in a^*, y \in a^*.u \sim bxb \wedge v \sim byb \wedge w \sim bxyb.$$

Using those formulas, one can FO-interpret in the free monoid a structure which, when quotiented by $\sim$, is isomorphic to $\mathcal{P}^W(\mathbb{N},+)$.

Assume now that the free monoid can be finite set interpreted in some tree t. This implies that $\mathcal{P}^W(\mathbb{N},+)$ is finite set interpretable in t since

structures obtainable by finite set interpretations from t are closed under FO-interpretations (Proposition 2.11) and quotient (Theorem 6.2). By Theorem 6.5 we deduce that $(\mathbb{N}, +)$ is WMSO-interpretable in t. This yields a contradiction since $(\mathbb{N}, +)$ is not WMSO-interpretable in a tree. Let us provide a direct argument for proving this last argument.

Assume that $(\mathbb{N}, +)$ is WMSO-interpretable in t and let U denote the set of nodes of t that represent $\mathbb{N}$ in a corresponding interpretation. We first note that for each node of t at most one of its subtrees contains infinitely many elements from U. Otherwise, if the two subtrees of a node u contain both infinitely many elements from U, the successor relation on $\mathbb{N}$ (which is addition with one argument fixed to 1) would infinitely often jump between these two subtrees. If $\mathcal{A}$ is an automaton with n transitions accepting the successor relation, and if $x_0, \ldots, x_n \in U$ are in the left subtree of u, $y_0, \ldots, y_n \in U$ are in the right subtree of u, and all x_i, y_i are in the successor relation, then $\mathcal{A}$ also accepts a pair x_i, y_j with $i \neq j$ by a simple counting argument on the transitions used at u in accepting runs of $\mathcal{A}$.

By starting at the root and always proceeding to the unique subtree containing infinitely many elements from U we obtain an infinite branch B of t.

Now let $\mathcal{A}_+$ be an automaton with n transitions accepting the relation $+$ on t and let $x_0, \ldots, x_n \in U$. For each x_i there are infinitely many y_i, z_i such that the triple (x_i, y_i, z_i) is accepted by $\mathcal{A}_+$. Choose a node v on B such that none of the x_i is below v and for each i choose y_i, z_i as above that are below v. Counting the possible transitions that are used at v in accepting runs of $\mathcal{A}_+$ on the tuples (x_i, y_i, z_i) we obtain that $\mathcal{A}_+$ also accepts (x_i, y_j, z_j) for some $i \neq j$. This gives a contradiction. □

Random graph

The *random graph* (also called *Rado graph*) is a non-oriented unlabeled countable graph with the following fundamental property: for any two disjoint finite sets E and F of vertices, there exists a vertex v that is connected to all the elements of E and to none of the elements of F. A graph with this property can be constructed as follows: The set of vertices are the natural numbers, and there is an edge between n and m if the nth bit in the binary expansion of m is 1. Now, given two finite disjoint sets E and F of numbers, we can pick a number that has all bits corresponding to numbers in E set to 1 and all bits corresponding to numbers F set to 0.

In fact, this graph is unique up to isomorphism, and it has a decidable FO-theory (it admits quantifier elimination). These facts and more information on the random graph can be found in, e.g., [52] and [40]. Since the random graph has a decidable FO-theory and since finite set interpretations define a large number of structures also having this property, it is interesting

to consider the question whether the random graph can be obtained by a finite set interpretation from a tree. A partial answer to this question has been studied: one knows that the random graph is not isomorphic to any word-automatic structure [63]. We show that there is no tree from which the random graph can be generated by a finite set interpretation.

A fundamental property that is required for the proof is that the random graph contains every finite (non-oriented) graph as an induced subgraph. This easily follows from the property used to define the random graph. Given a finite graph G, one can construct by induction on the size of G a set of vertices that induces this graph: By removing one vertex v of G one obtains a smaller graph G' that can be found as induced subgraph by induction. The vertices of G' can be partitioned into two sets, those connected to v and those not connected to v. Hence we can also find G as induced subgraph by definition of the random graph.

THEOREM 6.22 *The random graph is not finite set interpretable in a tree.*

Proof. Heading for contradiction, let us assume that there exists a finite set interpretation $\mathcal{I}_R = (\phi_{dom}(X), \psi(X,Y))$ and a binary tree t_R such that $\mathcal{I}_R(t_R)$ is (isomorphic to) the random graph.

The basic idea is to prove that, under this assumption, "the random graph is WMSO-interpretable in a tree". However, this statement is uncomfortable to handle since the properties of the random graph do only refer to its finite induced subgraphs. Instead, we show a similar interpretability result for every finite induced subgraph of the random graph; i.e. for every finite graph. Formally we establish the following claim.

Claim: There exists a finite set interpretation $\mathcal{I}'$ such that for any finite non-oriented graph G there exists a tree t_G such that $\mathcal{I}'(t_G)$ is isomorphic to $\mathcal{P}^W(G)$.

Before we prove this claim, let us demonstrate how to use it to show Theorem 6.22. Let us apply Theorem 6.5 on the interpretation $\mathcal{I}'$. We obtain a WMSO-interpretation $\mathcal{I}''$ with the property that $\mathcal{I}''(t_G) \cong G$ for any non-oriented unlabeled graph G and a suitably chosen tree t_G. As trees have bounded clique-width and WMSO-interpretations applied to a class of graphs of bounded clique-width yield also a class of graphs of bounded clique-width (see e.g. [35][1]), we obtain a contradiction to the fact that there exists non-oriented graphs of arbitrary high clique-width (the $(n \times n)$-grids yield such a family of graphs, see [79]).

[1] To be precise the result in [35] applies to classes of finite graphs, whereas our trees t_G may be infinite. But given an interpretation that produces a class of finite graphs from a class of infinite trees one can modify the interpretation such that one obtains the same resulting class of graphs from a class of finite trees.

Proof of the claim: Let us show first how we can encode any finite set of elements of $\mathcal{I}_R(t_R)$ by a pair of finite subsets of $dom(t_R)$ in such a way that the membership relation is "definable".

Let E be a finite set of vertices of $\mathcal{I}_R(t_R)$. Each vertex of $\mathcal{I}_R(t_R)$ is a finite set of nodes of t_R and therefore it makes sense to define D_E as the union of all the elements in E. The set D_E does not characterize E because we do not know which subsets of D_E correspond to elements in E. We now use the definition of the random graph to add additional information allowing us to uniquely identify E.

Let F be the set of vertices of $\mathcal{I}_R(t_R)$ that correspond to subsets of D_E, and that are not in E. Since D_E is finite, F is finite. Since we assume that $\mathcal{I}_R(t_R)$ is isomorphic to the random graph, there is an element I_E of $\mathcal{I}_R(t_R)$ which is connected to all elements of E and to none of the elements of F. From D_E and I_E one can easily reconstruct the set E. More precisely, let X be an element of $\mathcal{I}_R(t_R)$. Then X belongs to E if and only if t_R models $\phi_{dom}(X) \wedge X \subseteq D_E \wedge \psi(X, I_E)$.

Let now G be a non-oriented finite graph. Our goal is to find a tree t_G and a finite set interpretation $\mathcal{I}'$ (which is independent of G) such that $\mathcal{I}'(t_G)$ is isomorphic to $\mathcal{P}^W(G)$. The tree t_G is obtained from t_R by some additional markings. Hence, we have to identify sets of vertices of G with sets of vertices of t_R. We do this using a similar trick as above.

Since $\mathcal{I}_R(t_R)$ is the random graph, the graph G appears as an induced subgraph of $\mathcal{I}_R(t_R)$. Let V be the set of vertices of $\mathcal{I}_R(t_R)$ inducing this subgraph.

For each subset H of V, one can construct an element v_H of $\mathcal{I}_R(t_R)$ which is connected to all the vertices in H and to no vertex in $V \setminus H$. Knowing V, this element v_H completely characterizes H. Let now W be the set of all the v_H for all subsets H of V.

Let t_G be the tree t_R extended with markings describing D_V, I_V, D_W and I_W. This allows to define the formula $X \in V$ (similarly $X \in W$) by

$$\phi_{dom}(X) \wedge X \subseteq D_V \wedge \psi(X, I_V).$$

We now want to finite set interpret $\mathcal{P}^W(G)$ in t_G. Obviously, we can identify the elements of $\mathcal{P}^W(G)$ with the elements of W. Pursuing this idea, we define the interpretation $\mathcal{I}' = (\phi'_{dom}(X), \psi'(X, Y), \phi_{\subseteq}(X, Y))$ in the following way:

- The universe is defined by $\phi'_{dom}(X) = X \in W$.
- The subset relation is defined by:

$$\phi_{\subseteq}(X, Y) = \forall Z \in V. \psi(Z, X) \rightarrow \psi(Z, Y)$$

- Finally the edge relation is defined by:

$$\psi'(X,Y) = \mathit{Singleton}(X) \wedge \mathit{Singleton}(Y) \\ \wedge \exists X', Y' \in V.\psi(X',X) \wedge \psi(Y',Y) \wedge \psi(X',Y')$$

  where $\mathit{Singleton}(Z)$ stands for $Z \in W \wedge \exists! Y \in V.\psi(Y,Z)$ and $\exists!$ abbreviates "there exists one and only one".

Using the definitions of V and W, it is not difficult to see that $\mathcal{I}'(t_G)$ is (up to isomorphism) $\mathcal{P}^W(G)$. Furthermore, $\mathcal{I}'$ does not depend on G. This finishes the proof of the claim and hence the proof of the theorem. □

6.5 Discussion

In this chapter we have studied finite set interpretations on trees. The main results state that

1. a finite set interpretation followed by a quotient (w.r.t. to a congruence defined in the interpretation) can directly be expressed as a finite set interpretation,
2. if finite set interpretations applied to a tree t allow to construct everything that is obtained by finite set interpretations from a structure $\mathcal{S}$, then $\mathcal{S}$ can already be interpreted in t by a WMSO-interpretation.

These two results are important tools allowing to study the expressiveness of finite set interpretations. We have illustrated their use by showing that the free monoid and the Rado graph are not definable by finite set interpretations from a tree.

We have also shown that the tree-automatic hierarchy that was introduced in Section 2.5 is infinite. This raises the problem of identifying natural structures located on the levels of the hierarchy. This seems to be a rather difficult problem because already very little is known about the higher levels of the Caucal hierarchy, which serves as basis for generating the tree-automatic hierarchy. This demonstrates even more the power of our method because it allows to separate the levels of such a general hierarchy.

Another promising direction would be to study restricted versions of finite set interpretations that allow to obtain decidability results for stronger logics than just FO in the resulting structure. For example, in [76] it is shown that the Caley graph of an inverse monoid has a decidable FO theory with regular path predicates. Their technique can be viewed as a special kind of finite set interpretation applied to a tree, where the elements of the new structure are only special sets (so called Munn trees). A general formulation of such restricted finite set interpretations could lead to interesting classes of structures where reachability problems are decidable.

Conclusion

In this thesis we have presented some recent work on automata and logic on infinite trees. The material is based on joint publications with Arnaud Carayol [21] and Thomas Colcombet [32, 30]. The main results can be summarized as follows:

1. We have given an automata theoretic proof that it is not possible to define a choice function on the nodes of the infinite binary tree in MSO logic. The advantage of this new proof is that it only uses elementary facts from automata theory and also provides simple counter examples for candidate formulas defining choice functions (Theorem 3.5 on page 52). This allows, e.g., to define infinite game trees of rather simple structure on which no winning strategy can be defined in MSO logic (Theorem 3.9 on page 55).

2. We have shown that every extension of the infinite binary tree by a well-ordering relation on its domain results in a structure with undecidable MSO-theory (Theorem 3.11 on page 57). This allows to deduce undefinability results for MSO-interpretations (Propositions 3.18 and 3.19).

3. Based on the result on choice functions we showed that there are regular languages of infinite trees that cannot be accepted by unambiguous automata (Theorems 4.3 and 4.4 on page 71).

4. We introduced the concept of guidable automaton and showed that every regular language of infinite trees can be accepted by a guidable automaton (Theorem 4.7 on page 75).

5. We have presented a reduction of the parity index problem to the uniform universality problem for distance parity automata (Theorem 5.5 on page 89). In the reduction we make use of guidable automata.

6. We have shown that finite set interpretations on trees are closed under quotient (Theorem 6.2 on page 104) and we have introduced a method that allows to reduce questions on definability by finite set interpretations (on trees) to questions on definability by WMSO-interpretations

(Theorem 6.5 on page 108). These results allow to show undefinability results for finite set interpretations on trees.

There are many open questions that remain, in particular as we have used some new concepts like guidable automata, distance parity automata, and finite set interpretations. We have already mentioned open questions in the discussions at the end of each chapter. Here we list some of them as a summary and a guide for future research.

We have settled negatively the question whether one can add a well-ordering to the infinite binary tree such that the resulting structure has a decidable MSO-theory. We can ask the same question for choice functions:

QUESTION 1 *Is it possible to add a choice function to the infinite binary tree such that the resulting structure has a decidable MSO-theory?*

More precisely, one should ask whether we can enrich $\mathrm{MSO}[\mathfrak{t}_2]$ by atomic formulas $R(X, x)$ such that the interpretation of these atomic formulas corresponds to a choice function and such that the resulting logic is decidable.

The result on unambiguous automata is obtained by invoking the undefinability result for choice functions. A natural question is whether this is also possible in the other direction, i.e., to conclude from the existence of inherently ambiguous languages to the undefinability of choice functions. More generally, we would like to understand the relationship between unambiguity and choice functions.

QUESTION 2 *What is the relationship between ambiguous languages and the undefinability of choice functions (for restricted classes of sets)?*

The concept of guidable automata is rather new, and a detailed analysis of various models (e.g., on finite trees) is still to be done. This leads to the following general question.

QUESTION 3 *What are the properties of guidable automata on finite trees?*

We have presented one application of guidable automata in the reduction of the parity index problem to the uniform universality problem for distance parity automata. To solve the parity index problem by this reduction we would need a solution of the latter problem.

QUESTION 4 *Is the uniform universality problem for distance parity automata decidable?*

Finally, there are various open issues in the area of (finite) set interpretations. For example, the techniques that we have developed work for finite set interpretations on trees. It is not clear if similar results can be obtained for other classes of structures. The heavy use of automata in our proofs

suggests that a generalization to other classes of structures requires different techniques. Another question is what happens if we drop the finiteness hypothesis on the sets in the interpretations.

QUESTION 5 *Is it possible to reduce questions on definability by set interpretations to questions on definability by MSO-interpretations similar to the technique developed for finite set interpretations in Chapter 6?*

The list of recent results and of open questions shows that there is still a lot of interesting work to do in the area of automata and logic on infinite trees.

Bibliography

[1] Parosh Aziz Abdulla, Ahmed Bouajjani, Lukás Holík, Lisa Kaati, and Tomás Vojnar. Computing simulations over tree automata. In *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008*, volume 4963 of *Lecture Notes in Computer Science*, pages 93–108. Springer, 2008. 73

[2] Parosh Aziz Abdulla, Pavel Krcál, and Wang Yi. R-automata. In *Proceedings of the 19th International Conference on Concurrency Theory, CONCUR'08*, volume 5201 of *Lecture Notes in Computer Science*, pages 67–81. Springer, 2008. 98

[3] Michael A. Arbib. *Theories of Abstract Automata*. Prentice-Hall, London, 1969. 89

[4] André Arnold. Rational ω-languages are non-ambiguous. *Theoretical Computer Science*, 26:221–223, 1983. 69

[5] André Arnold, Jacques Duparc, Filip Murlak, and Damian Niwiński. On the topological complexity of tree languages. In J. Flum, E. Grädel, and T. Wilke, editors, *Logic and automata: History and Perspectives*, pages 9–28. Amsterdam University Press, 2007. 83

[6] André Arnold and Damian Niwiński. *Rudiments of μ-calculus*, volume 146 of *Studies in Logic*. Elsevier, 2001. 4

[7] Vince Bárány. Invariants of automatic presentations and semi-synchronous transductions. In *STACS 2006*, volume 3884 of *Lecture Notes in Computer Science*, pages 289–300. Springer, 2006. 101

[8] Dietmar Berwanger and Achim Blumensath. The monadic theory of tree-like structures. In E. Grädel, W. Thomas, and T. Wilke, editors, *Automata, Logics, and Infinite Games*, number 2500 in Lecture Notes in Computer Science, chapter 16, pages 285–301. Springer, 2002. 35

[9] Achim Blumensath. Automatic structures. Diploma thesis, RWTH-Aachen, 1999. 9, 40, 42, 106

[10] Achim Blumensath. Prefix-recognizable graphs and monadic second order logic. Technical Report AIB-2001-06, RWTH Aachen, May 2001. 9

[11] Achim Blumensath and Erich Grädel. Automatic structures. In *Proceedings of the 15th IEEE Symposium on Logic in Computer Science, LICS 2000*, pages 51–62. IEEE Computer Society Press, 2000. 9, 40, 42, 58

[12] Achim Blumensath and Erich Grädel. Finite presentations of infinite structures: Automata and interpretations. *Theory of Computing Systems*, 37:641–674, 2004. 8, 9

[13] Mikołaj Bojańczyk and Thomas Colcombet. Tree-walking automata cannot be determinized. *Theor. Comput. Sci.*, 350(2–3):164–173, 2006. Conference version in ICALP'04. 82, 86, 98

[14] Yves Bontemps, Pierre Yves Schobbens, and Christof Löding. Synthesis of open reactive systems from scenario-based specifications. *Fundamenta Informaticae*, 62(2):139–169, 2004. 4

[15] Julian C. Bradfield. The modal mu-calculus alternation hierarchy is strict. *Theoretical Computer Science*, 195(2):133–153, March 1998. 83

[16] J. Richard Büchi. Weak second order logic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960. 2

[17] J. Richard Büchi. On a decision method in restricted second order arithmetic. In *International Congress on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, 1962. 2, 14, 31

[18] J. Richard Büchi. Regular canonical systems. *Archiv für Mathematische Grundlagenforschung*, 6:91–111, 1964. 8

[19] J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969. 3, 4

[20] Aranud Carayol and Stefan Wöhrle. The caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In *Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science, FST TCS 2003*, volume 2914 of *Lecture Notes in Computer Science*, pages 112–123. Springer, 2003. 35, 36, 126

[21] Arnaud Carayol and Christof Löding. MSO on the infinite binary tree: Choice and order. In *Proceedings of the 16th Annual Conference of the European Association for Computer Science Logic, CSL 2007*, volume 4646 of *Lecture Notes in Computer Science*, pages 161–176. Springer, 2007. 7, 46, 68, 71, 133

[22] Julien Carme, Rémi Gilleron, Aurélien Lemay, Alain Terlutte, and Marc Tommasi. Residual finite tree automata. In *Developments in Language Theory, 7th International Conference, DLT 2003, Proceedings*, volume 2710 of *Lecture Notes in Computer Science*, pages 171–182. Springer, 2003. 74

[23] Olivier Carton. Chain automata. *Theoretical Computer Science*, 161(1&2):191–203, 1996. 15, 16

[24] Olivier Carton and Ramón Maceiras. Computing the Rabin index of a parity automaton. *RAIRO Theoretical Informatics and Applications*, 33(6):495–506, 1999. 84

[25] Didier Caucal. On infinite transition graphs having a decidable monadic theory. In *Proceedings of the 23rd International Colloquium on Automata, Languages and Programming, ICALP '96*, volume 1099 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 1996. 9

[26] Didier Caucal. On infinite terms having a decidable monadic theory. In *MFCS'02*, volume 2420 of *Lecture Notes in Computer Science*, pages 165–176. Springer, 2002. 9, 10, 35, 36

[27] Alonzo Church. Logic, arithmetic and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35, 1962. 3

[28] Thomas Colcombet. On families of graphs having a decidable first order theory with reachability. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming, ICALP 2002*, volume 2380 of *Lecture Notes in Computer Science*, pages 98–109. Springer, 2002. 9

[29] Thomas Colcombet. Equational presentations of tree-automatic structures. In *Workshop on Automata, Structures and Logic, Auckland, New Zealand*, December 2004. 10

[30] Thomas Colcombet and Christof Löding. Transforming structures by set interpretations. *Logical Methods in Computer Science*, 3(2), 2007. 11, 42, 133

[31] Thomas Colcombet and Christof Löding. The nesting-depth of disjunctive mu-calculus for tree languages and the limitedness problem. In *Proceedings of the 17th EACSL Annual Conference on Computer Science Logic CSL 2008*, volume 5213 of *Lecture Notes in Computer Science*. Springer, 2008. 98

[32] Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, ICALP 2008*, volume 5126 of *Lecture Notes in Computer Science*, pages 398–409. Springer, 2008. 7, 8, 68, 82, 133

[33] Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Christof Löding, Denis Lugiez, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*. Available on `http://tata.gforge.inria.fr/`. Last Release: October 12, 2007. 85

[34] Bruno Courcelle. The monadic second order logic of graphs II: Infinite graphs of bounded width. *Mathematical System Theory*, 21:187–222, 1989. 11

[35] Bruno Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, editor, *Handbook of graph grammars and computing by graph transformations, Vol. 1 : Foundations*, chapter 5, pages 313–400. World Scientific, 1997. 129

[36] Bruno Courcelle. Clique-width of countable graphs: a compactness property. *Discrete Mathematics*, 276(1-3):127–148, 2004. 11

[37] Bruno Courcelle and Sang-il Oum. Vertex-minors, monadic second-order logic and a conjecture by seese. *Journal of Combinatorial Theory, Series B*, 97:91–126, 2007. 101

[38] Bruno Courcelle and Igor Walukiewicz. Monadic second-order logic, graph converings and unfoldings of transition systems. *Annals of Pure and Applied Logic*, 92(1):35–62, 1998. 35

[39] Max Dauchet and Sophie Tison. The theory of ground rewrite systems is decidable. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science, LICS '90*, pages 242–248. IEEE Computer Society Press, 1990. 9, 40

[40] Reinhard Diestel. *Graph Theory*. Springer, third edition, 2005. 128

[41] John Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4:406–451, 1970. 2

[42] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Perspectives in Mathematical Logic. Springer, Berlin, 1995. 30

[43] Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98:21–51, January 1961. 2

[44] Calvin C. Elgot and Michael O. Rabin. Decidability and undecidability of extensions of second (first) order theory of (generalized) successor. *Journal of Symbolic Logic*, 31(2):169–181, 1966. 10, 37

[45] E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs (exteded abstract). In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science, FoCS '88*, pages 328–337, Los Alamitos, California, October 1988. IEEE Computer Society Press. 4, 15, 18, 79

[46] E. Allen Emerson and A. Prasad Sistla. Deciding full branching time logic. *Information and Control*, 61(3):175–201, 1984. 4

[47] Javier Esparza, David Hansel, Peter Rossmanith, and Stefan Schwoon. Efficient algorithms for model checking pushdown systems. In *Proceedings of the 12th International Conference on Computer Aided Verification, CAV 2000*, volume 1855 of *Lecture Notes in Computer Science*, pages 232–247. Springer, 2000. 8

[48] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002. 1, 13, 19, 81

[49] Yuri Gurevich and Saharon Shelah. Rabin's uniformization problem. *J. Symb. Log.*, 48(4):1105–1119, 1983. 6, 45, 46, 52

[50] Kosaburo Hashiguchi. Algorithms for determining relative star height and star height. *Inf. Comput.*, 78(2):124–169, 1988. 81, 86

[51] Thomas A. Henzinger, Orna Kupferman, and Sriram K. Rajamani. Fair simulation. *Inf. Comput.*, 173(1):64–81, 2002. 73

[52] Wilfrid Hodges. *Model Theory*. Cambridge University Press, 1993. 128

[53] Bernard R. Hodgson. Décidabilité par automate fini. *Ann. Sci. Math. Québec*, 7(3):39–57, 1983. 9, 40, 41

[54] Gerard J. Holzmann. *The Spin Model Checker - Primer and Reference Manual*. Addison-Wesley, 2003. 3

[55] John E. Hopcroft and Jeffrey D. Ullman. *Formal Languages and their Relation to Automata*. Addison-Wesley, 1969. 67

[56] David Janin and Giacomo Lenzi. On the logical definability of topologically closed recognizable languages of infinite trees. *Computing and Informatics*, 21(3):185–203, 2002. 84

[57] Marcin Jurdziński. Small progress measures for solving parity games. In Horst Reichel and Sophie Tison, editors, *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science, STACS 2000*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer, February 2000. 4, 19

[58] Marcin Jurdziński, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms, SODA 2006*, pages 117–123. ACM/SIAM, 2006. 4, 19

[59] Detlef Kähler and Thomas Wilke. Complementation, disambiguation, and determinization of Büchi automata unified. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, ICALP 2008, Part I*, volume 5125 of *Lecture Notes in Computer Science*, pages 724–735. Springer, 2008. 4, 16, 69

[60] Łukasz Kaiser, Sasha Rubin, and Vince Bárány. Cardinality and counting quantifiers on omega-automatic structures. In *Proceedings of the 25th International Symposium on Theoretical Aspects of Computer Science, STACS 2008*, volume 08001 of *Dagstuhl Seminar Proceedings*, pages 385–396, 2008. 40

[61] Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. In *Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, Indiana, USA, 13-16 October 1994*, volume 960 of *Lecture Notes in Computer Science*, pages 367–392. Springer, 1995. 9, 40, 42

[62] Bakhadyr Khoussainov and Anil Nerode. *Automata theory and its applications*, volume 21 of *Progress in Computer Science and Applied Logic*. Birkhäuser, 2001. 1, 13

[63] Bakhadyr Khoussainov, André Nies, Sasha Rubin, and Frank Stephan. Automatic structures: Richness and limitations. In *LICS'04*, pages 44–53. IEEE Computer Society, 2004. 126, 129

[64] Daniel Kirsten. Distance desert automata and the star height problem. *RAIRO – Theoretical Informatics and Applications*, 3(39):455–509, 2005. 8, 81, 86, 88, 98

[65] Nils Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Annals of Pure and Applied Logic*, 69(2–3):243–268, 1994. 19

[66] Sriram C. Krishnan, Anuj Puri, and Robert K. Brayton. Structural complexity of ω-automata. In *12th Annual Symposium on Theoretical Aspects of Computer Science, STACS '95*, volume 900 of *Lecture Notes in Computer Science*, pages 143–156. Springer, 1995. 84

[67] Orna Kupferman, Shmuel Safra, and Moshe Y. Vardi. Relating word and tree automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science, LICS '96*, pages 322–332, Los Alamitos, California, July 1996. IEEE Computer Society Press. 84

[68] Orna Kupferman and Moshe Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. Comput. Log.*, 2(3):408–429, 2001. 83

[69] Orna Kupferman and Moshe Y. Vardi. Safraless decision procedures. In *46th Annual IEEE Symposium on Foundations of Computer Science, FOCS'05, Proceedings*, pages 531–542. IEEE Computer Society, 2005. 4

[70] Dietrich Kuske and Markus Lohrey. On the theory of one-step rewriting in trace monoids. In *ICALP'02*, volume 2380 of *Lecture Notes in Computer Science*, pages 752–763. Springer, 2002. 9

[71] Giacomo Lenzi. A hierarchy theorem for the mu-calculus. In F. Meyer auf der Heide and B. Monien, editors, *Proceedings of the 23rd International Colloquium on Automata, Languages and Programming, ICALP '96*, volume 1099 of *Lecture Notes in Computer Science*, pages 87–97. Springer, July 1996. 83

[72] Shmuel Lifsches and Saharon Shelah. Uniformization, choice functions and well orders in the class of trees. *J. Symb. Log.*, 61(4):1206–1227, 1996. 46

[73] Shmuel Lifsches and Saharon Shelah. Uniformization and Skolem functions in the class of trees. *J. Symb. Log.*, 63(1):103–127, 1998. 52

[74] Peter A. Lindsay. On alternating omega-automata. *J. Comput. Syst. Sci.*, 36(1):16–24, 1988. 83

[75] Christof Löding. Optimal bounds for the transformation of omega-automata. In C. Pandu Rangan, V. Raman, and R. Ramanujam, editors, *Foundations of Software Technology and Theoretical Computer Science*, volume 1738 of *Lecture Notes in Computer Science*, pages 97–109. Springer, December 1999. 15

[76] Markus Lohrey and Nicole Ondrusch. Inverse monoids: Decidability and complexity of algebraic questions. In *Mathematical Foundations of Computer Science 2005, 30th International Symposium, MFCS'05, Proceedings*, volume 3618 of *Lecture Notes in Computer Science*, pages 664–675. Springer, 2005. 131

[77] Sylvain Lombardy and Jacques Sakarovitch. The universal automaton. In J. Flum, E. Grädel, and T. Wilke, editors, *Logic and automata: History and Perspectives*, pages 457–504. Amsterdam University Press, 2007. 67, 72, 80

[78] P. Madhusudan. Model-checking trace event structures. In *LICS'03*, pages 371–380. IEEE Computer Society, 2003. 9

[79] Johann A. Makowsky and Udi Rotics. On the clique-width of graphs with few P_4's. *International Journal of Foundations of Computer Science*, 10(3):329–348, 1999. 129

[80] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521–530, 1966. 3, 15

[81] Max Michel. Complementation is more difficult with automata on infinite words. CNET, Paris, 1988. 15

[82] Angelo Monti and Adriano Peron. Systolic tree omega-languages: the operational and the logical view. *Theor. Comput. Sci.*, 233(1–2):1–18, 2000. 65

[83] Andrzej W. Mostowski. Determinacy of sinking automata on infinite trees and inequalities between various pairs of rabin's indices. *Information Processing Letters*, 15(4):159–163, 1982. 68

[84] Andrzej W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Computation Theory*, volume 208 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 1984. 4, 8, 15, 82, 86

[85] Andrzej W. Mostowski. Games with forbidden positions. Technical Report 78, Uniwersytet Gdański, Instytut Matematyki, 1991. 4, 15, 18

[86] Andrzej W. Mostowski, Jerzy Skurczyński, and Klaus Wagner. Deterministic automata on infinite trees and the Borel hierarchy. In *Proceedings of the 4th Hungarian Conference on Computer Science*, pages 103–115, 1985. 84

[87] David E. Muller. Infinite sequences and finite machines. In *Proceedings of the 4th IEEE Symposium on Switching Circuit Theory and Logical Design*, pages 3–16, 1963. 3, 15

[88] David E. Muller, Ahmed Saoudi, and Paul E. Schupp. Alternating automata, the weak monadic theory of the tree, and its complexity. In *Proceedings of the 13th International Colloquium on Automata, Languages and Programming, ICALP '86*, volume 226 of *Lecture Notes in Computer Science*, pages 275–283. Springer, 1986. 83, 105

[89] David E. Muller and Paul E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, 37:51–75, 1985. 8, 9

[90] David E. Muller and Paul E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141(1&2):69–107, 1995. 16

[91] Damian Niwiński. On fixed-point clones (extended abstract). In Laurent Kott, editor, *Proceedings of the 13th International Colloquium on Automata, Languages and Programming, ICALP '86*, volume 226 of *Lecture Notes in Computer Science*, pages 464–473. Springer, July 1986. 83

[92] Damian Niwiński and Igor Walukiewicz. Ambiguity problem for automata on infinite trees. Unpublished note. 7, 69, 71

[93] Damian Niwiński and Igor Walukiewicz. Relating hierarchies of word and tree automata. In Michel Morvan, Christoph Meinel, and Daniel Krob, editors, *Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science, STACS '98*, volume 1373 of *Lecture Notes in Computer Science*, pages 320–331. Springer, February 1998. 81, 84

[94] Damian Niwiński and Igor Walukiewicz. Deciding nondeterministic hierarchy of deterministic tree automata. *Electr. Notes Theor. Comput. Sci.*, 123:195–208, 2005. 81, 84

[95] Dominique Perrin and Jean-Éric Pin. *Infinite words*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004. 13

[96] Jean-Éric Pin. Syntactic semigroups. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Language Theory*, volume I, pages 679–746. Springer, 1997. 67

[97] Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *Proceedings of the 21st IEEE Symposium on Logic in Computer Science (LICS 2006)*, pages 255–264. IEEE Computer Society, 2006. 4, 16

[98] Amir Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981. 3

[99] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Proceedings of the Symposium on Principles of Programming Languages, POPL'89*, pages 179–190, 1989. 4

[100] Andreas Potthoff and Wolfgang Thomas. Regular tree languages without unary symbols are star-free. In *Proceedings of Fundamentals of Computation Theory, 9th International Symposium, FCT '93*, volume 710 of *Lecture Notes in Computer Science*, pages 396–405. Springer, 1993. 69

[101] Mojżcesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. *Comptes Rendus du I*$^{\text{er}}$ *Congrès des Mathématiciens des Pays Slaves, Warszawa*, pages 92–101, 1929. 41

[102] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, July 1969. 3, 19, 20, 24, 28, 31, 103

[103] Michael O. Rabin. Weakly definable relations and special automata. In Y. BarHillel, editor, *Mathematical Logic and Foundations of Set Theory*, pages 1–23. North-Holland, 1970. 105

[104] Michael O. Rabin. *Automata on Infinite Objects and Church's Problem*. American Mathematical Society, Boston, MA, USA, 1972. 3

[105] Klaus Reinhardt. The complexity of translating logic to finite automata. In *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*, pages 231–238. Springer, 2002. 3

[106] Sasha Rubin. *Automatic Structures*. PhD thesis, University of Auckland, New Zealand, 2004. 10, 40, 43

[107] Shmuel Safra. On the complexity of omega-automata. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science, FoCS '88*, pages 319–327, Los Alamitos, California, October 1988. IEEE Computer Society Press. 4, 15, 16

[108] Sven Schewe. Solving parity games in big steps. In *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science, 27th International Conference, New Delhi, India, December 12-14, 2007, Proceedings*, volume 4855 of *Lecture Notes in Computer Science*, pages 449–460. Springer, 2007. 4, 19

[109] Detlef Seese. The structure of the models of decidable monadic theories of graphs. *Annals of Pure and Applied Logic*, 53:169–195, 1991. 101

[110] Helmut Seidl. Deciding equivalence of finite tree automata. *SIAM J. Comput.*, 19(3):424–437, 1990. 69

[111] Alexei L. Semenov. Decidability of monadic theories. In *Proceedings of the 11th International Symposium on Mathematical Foundations of Computer Science, MFCS '84*, volume 176 of *Lecture Notes in Computer Science*, pages 162–175. Springer, June 1984. 65

[112] Dirk Siefkes. The recursive sets in certain monadic second order fragments of arithmetic. *Arch. für mat. Logik und Grundlagenforschung*, 17:71–80, 1975. 6, 45

[113] Richard E. Stearns and Harry B. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM Journal on Computing*, 14(3):598–611, 1985. 7, 69

[114] Colin Stirling. *Modal and Temporal Properties of Processes*. Graduate Texts in Computer Science. Springer, 2001. 4

[115] Larry J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD thesis, Dept. of Electrical Engineering, MIT, Boston, Mass., 1974. 3

[116] Robert S. Streett. Propositional dynamic logic of looping and converse is elementary decidable. *Information and Control*, 54:121–141, 1982. 19

[117] James W. Thatcher and Jesse B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, 1968. 2

[118] Wolfgang Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Language Theory*, volume III, pages 389–455. Springer, 1997. 1, 4, 13, 15, 18, 20, 22, 25, 43

[119] Boris A. Trakhtenbrot. Finite automata and the logic of one-place predicates. *Sibirian Mathematical Journal*, 3:103–131, 1962. English translation in: AMS Transl. 59 (1966) 23-55. 2

[120] Boris A. Trakhtenbrot and Y.M. Barzdin. *Finite Automata: Behavior and Synthesis*. North-Holland Publishing Company, Amsterdam, 1973. 3

[121] Tomasz F. Urbański. On deciding if deterministic Rabin language is in Büchi class. In *ICALP'00*, pages 663–674, 2000. 81, 84

[122] Moshe Y. Vardi and Thomas Wilke. Automata: from logics to algorithms. In *Logic and Automata - History and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 629–724. Amsterdam University Press, 2007. 1, 13

[123] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings, Symposium on Logic in Computer Science, 16-18 June 1986, Cambridge, Massachusetts, USA*, pages 332–344. IEEE Computer Society, 1986. 3

[124] Jens Vöge and Marcin Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *Proceedings of the 12th International Conference on Computer Aided Verification, CAV 2000*, volume 1855 of *Lecture Notes in Computer Science*, pages 202–215. Springer, 2000. 4

[125] Klaus W. Wagner. Eine topologische Charakterisierung einiger Klassen regulärer Folgenmengen. *J. Inf. Process. Cybern. EIK*, 13(9):473–487, 1977. 83

[126] Igor Walukiewicz. Pushdown processes: Games and model checking. *Information and Computation*, 164(2):234–263, January 2001. 18

[127] Qiqi Yan. Lower bounds for complementation of ω-automata via the full automata technique. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, ICALP'06*, volume 4052 of *Lecture Notes in Computer Science*, pages 589–600. Springer, 2006. 15

[128] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998. 1, 4, 13, 18, 22

Index