

University of Warsaw  
Faculty of Mathematics, Informatics and Mechanics

Szczepan Hummel

Topological Complexity of Sets  
Defined by Automata and Formulas

*PhD dissertation*

Supervisor  
prof. Damian Niwiński

Institute of Informatics  
University of Warsaw

May 2017

Author's declaration:

aware of legal responsibility I hereby declare that I have written this dissertation myself and all the contents of the dissertation have been obtained by legal means.

May 10, 2017

*date*

.....

*Szczepan Hummel*

Supervisor's declaration:

the dissertation is ready to be reviewed

May 10, 2017

*date*

.....

*prof. Damian Niwiński*

## Abstract

In this thesis we consider languages of infinite words or trees defined by automata of various types or formulas of various logics. We ask about the highest possible position in the Borel or the projective hierarchy inhabited by sets defined in a given formalism. The answer to this question is called the topological complexity of the formalism.

It is shown that the topological complexity of Monadic Second Order Logic extended with the unbounding quantifier (introduced by Bojańczyk to express some asymptotic properties) over  $\omega$ -words is the whole projective hierarchy. We also give the exact topological complexities of related classes of languages recognized by nondeterministic  $\omega$ B-,  $\omega$ S- and  $\omega$ BS-automata studied by Bojańczyk and Colcombet, and a lower complexity bound for an alternating variant of  $\omega$ BS-automata.

We present the series of results concerning bi-unambiguous languages of infinite trees, i.e. languages recognized by unambiguous parity tree automata whose complements are also recognized by unambiguous parity automata. We give an example of a bi-unambiguous tree language  $G$  that is  $\Sigma_1^1$ -complete (analytic-complete). We present an operation  $\sigma$  on tree languages with the property that  $\sigma(L)$  is topologically harder than any language in the sigma-algebra generated by the languages continuously reducible to  $L$ . If the operation is applied to a bi-unambiguous language then the result is also bi-unambiguous. We then show that the application of the operation can be iterated to obtain harder and harder languages. We also define another operation that enables a limit step iteration. Using the operations we are able to construct a sequence of bi-unambiguous languages of increasing topological complexity, of length at least  $\omega^2$ .

## ACM Computing Classification System

Theory of computation – Formal languages and automata theory – Automata over infinite objects

Theory of computation – Formal languages and automata theory – Tree languages

Theory of computation – Formal languages and automata theory – Automata extensions

Mathematics of computing – Continuous mathematics – Topology – Point-set topology

Theory of computation – Logic – Higher order logic

## Keywords

topological complexity, Borel hierarchy, projective hierarchy, infinite words, infinite trees, MSO+U,  $\omega$ BS-automata, unambiguous automata, index hierarchy

## Streszczenie

W niniejszej rozprawie rozważane są języki nieskończonych słów lub drzew definiowane poprzez automaty różnych typów lub formuły różnych logik. Pytamy o najwyższą możliwą pozycję w hierarchii borelowskiej lub rzutowej zajmowaną przez zbiory definiowane w danym formalizmie. Odpowiedź na to pytanie jest nazywana złożonością topologiczną formalizmu.

Przedstawiony został dowód, że złożonością topologiczną Logiki Monadycznej Drugiego Rzędu rozszerzonej o kwantyfikator Unbounding (wprowadzony przez Bojańczyka w celu umożliwienia wyrażania własności asymptotycznych) na słowach nieskończonych jest cała hierarchia rzutowa. Obliczone zostały również złożoności topologiczne klas języków rozpoznawanych przez niedeterministyczne  $\omega$ B-,  $\omega$ S- i  $\omega$ BS-automaty rozważane przez Bojańczyka i Colcombet'a, oraz zostało podane dolne ograniczenie złożoności wariantu alternującego  $\omega$ BS-automatów.

Zaprezentowane zostały wyniki dotyczące języków podwójnie jednoznacznych, tzn. języków rozpoznawanych przez jednoznaczne automaty parzystości na drzewach, których dopełnienia również są rozpoznawane przez jednoznaczne automaty parzystości. Podany został przykład podwójnie jednoznacznego języka drzew  $G$ , który jest  $\Sigma_1^1$ -zupełny (analityczny-zupełny). Została wprowadzona operacja  $\sigma$  na językach drzew taka, że język  $\sigma(L)$  jest topologicznie bardziej złożony niż jakikolwiek język należący do sigma-algebry generowanej przez języki redukujące się w sposób ciągły do języka  $L$ . W wyniku zastosowania powyższej operacji do języka podwójnie jednoznacznego otrzymujemy język podwójnie jednoznaczny. Zostało pokazane, że kolejne iteracje aplikacji powyższej operacji dają coraz bardziej złożone języki. Została również wprowadzona druga operacja, która umożliwia krok graniczny iteracji. Używając obydwu powyższych operacji można skonstruować ciąg długości  $\omega^2$  złożony z języków podwójnie jednoznacznych o coraz większej złożoności.

# Acknowledgements

First of all I would like to thank Damian Niwiński, my supervisor, for all the guidance and support that I have obtained from him. He was ready to devote more time to discussing scientific and editorial matters with me than I could even use. The weaknesses of my work that he was pointing out almost always occurred to be much deeper than I originally understood. This critical view has helped improve the presentation and correctness of my text in many places. I highly appreciate his patience in teaching me the basics of mathematical writing style.

I would like to express my great gratitude to my colleagues and friends, Szymon Toruńczyk, Mikołaj Bojańczyk, Filip Murlak, Henryk Michalewski and Michał Skrzypczak, who always had time to answer my questions and explain me scientific matters, without whom I would achieve nothing in science. Szymon encouraged me to enter the world of topology deeper. It is impossible to overestimate the benefits that I took from our many discussions throughout whole the studies and PhD studies. Mikołaj introduced me into automata and formal language theory, and explained many difficult issues in a very understandable way. Filip have supported me in entering the world of topology applied to formal language theory. Michał was ready to answer all the questions I asked concerning automata theory, topology and other fields. He spent hours discussing with me issues that I got stuck with. Henryk gave me priceless support in descriptive set theory. He was the one to support me when I discovered an error in one of my proofs and was desperately looking for a fix. He committed his time to understand my construction and try to find a possible weakness. He was, finally, the one to advice me to use formal verification to disprove my “hypothesis”—that occurred to be the right way to go, and which was a fun too.

I would like to thank my coauthors, Jacques Duparc, Kevin Fournier, Henryk Michalewski, Damian Niwiński, Michał Skrzypczak, and Szymon Toruńczyk, for all I have learned while working with them. I also owe gratitude to anonymous referees of my articles, who gave me broader view of the results.

I thank my office mates, Lorenzo Clemente, Nathanaël Fijalkow, Tomasz Gogacz, Eryk Kopczyński, Denis Kuperberg, Paweł Parys, Charles Paperman, Michał Skrzypczak, Joost Winter, and Martin Zimmermann, for inspiring discussions, writing style advices, and friendly atmosphere.

During my PhD studies I have attended a number of inspiring workshops and spring schools organized by GAMES program.

Last, but not least, I would like to express my gratitude to my family for all the support and encouragement that I obtained from them. This I prefer to do in Polish.

Aniu, Tosiu, Taniu, Janko, Rodzice, Babciu i cała moja Rodzino, bardzo Wam dziękuję, że przez cały czas mojej pracy nad doktoratem mnie wspieraliście poświęcając swój czas, okazując zainteresowanie i zrozumienie, że nie zawsze mam czas na inne ważne czynności.

# Contents

<b>Introduction</b>	<b>9</b>
<b>1 Topological Complexity</b>	<b>18</b>
1.1 Preliminaries . . . . .	18
1.2 Topological Complexity Classes . . . . .	27
1.2.1 Borel and Projective Hierarchy . . . . .	27
1.2.2 Wadge Hierarchy . . . . .	33
1.3 Automata, Formulas and Definability . . . . .	33
1.4 Topological Complexity of Classes of Languages . . . . .	34
1.5 Applications . . . . .	35
1.5.1 Separation of Classes . . . . .	35
1.5.2 Logics and Topological Complexity . . . . .	35
1.5.3 Automata Models and Their Limitations . . . . .	37
1.5.4 Undecidability . . . . .	42
<b>2 Infinite Words – Beyond Regularity</b>	<b>43</b>
2.1 Regular Languages of Infinite Words . . . . .	43
2.2 Complexity of Regular Languages of Infinite Words . . . . .	47
2.3 Unbounding Quantifier . . . . .	49
2.3.1 Topological Complexity of WMSO+U . . . . .	53
2.4 Topological Complexity of MSO+U . . . . .	54
2.5 $\omega$ BS-automata . . . . .	65
2.5.1 Complexity of $\omega$ B- and $\omega$ S-regular Languages . . . . .	68
2.5.2 Complexity of $\omega$ BS-regular Languages . . . . .	71
2.5.3 Alternating $\omega$ BS-automata . . . . .	74
2.6 Remarks and Open Questions . . . . .	83
<b>3 Infinite Trees – Within Regularity</b>	<b>85</b>
3.1 Parametrized Tree Languages . . . . .	86
3.2 Parity Tree Automata . . . . .	86
3.3 Complexity of Regular Tree Languages . . . . .	91

3.4	Rabin-Mostowski Index . . . . .	92
3.4.1	Relations Between Hierarchies . . . . .	96
3.5	Subclasses of Known Topological Complexity . . . . .	98
3.5.1	Deterministic Tree Languages . . . . .	98
3.5.2	Büchi Tree Languages . . . . .	100
3.5.3	Weak Tree Languages . . . . .	101
3.6	Unambiguous Tree Languages . . . . .	102
3.6.1	Basic Properties . . . . .	102
3.6.2	Topological Complexity of Unambiguous Büchi Au- tomata . . . . .	107
3.7	Analytic-Complete Unambiguous Language . . . . .	107
3.8	Sigma-Lifting Operation . . . . .	111
3.8.1	Topological Properties . . . . .	122
3.8.2	Automata Theoretic Properties . . . . .	126
3.8.3	Iteration Potential . . . . .	135
3.9	Lower Bound Pushed Up . . . . .	150
3.10	Limit Step . . . . .	154
3.10.1	Automaton . . . . .	163
3.10.2	Topological Properties . . . . .	177
3.10.3	Stretchability . . . . .	180
3.11	The Hierarchy . . . . .	188
3.12	Further Work and Discussion . . . . .	189
3.12.1	A Vain Attempt . . . . .	189
3.12.2	Difference Hierarchy . . . . .	189
3.12.3	Unambiguous vs Regular . . . . .	190
3.12.4	Wadge Hierarchy . . . . .	191
3.12.5	Bi-unambiguous vs Unambiguous . . . . .	192
	<b>Notations</b>	<b>193</b>
	<b>Bibliography</b>	<b>197</b>



# Introduction

The theory of definability is the branch of logic which studies the complexity of concepts by looking at the grammatical complexity of their definitions. To measure the complexity of a concept it is necessary first to find a definition which is as simple as possible from the standpoint of the grammatical yardstick under consideration and second to show that no simpler definition is possible. If a certain kind of definition is possible for a given concept then it is usually (but not always!) not so hard to establish that it can be described in the given grammatical form, but often it is difficult, even if true, to prove that the concept is undefinable in the given form.

J.W. Addison [Add04]

Measuring complexity of “concepts” in terms of the complexity of their definitions is the main interest of this thesis. The concepts that we consider are properties of words or labeled trees, while as “grammars” (called formalisms in the thesis) we take automata models, logics or descriptive set theoretical constructions.

A basic version of such a measurement is to determine whether a set provided through some informal description can be defined in a given formalism. Is there, for example, a finite automaton that accepts exactly those words over a one letter alphabet that have even length? The answer is yes, and it is straightforward to construct an appropriate automaton. Now one can ask: Is the same set of words definable by a formula of first order logic that quantifies over word positions and uses order relation on the positions? This question is more involved, and the answer is negative<sup>1</sup>.

A natural question arises: How the measures relate to each other? Based on the above example we already see that not every set definable using finite automata is definable in terms of first order logic. However, it occurs that the opposite is true. Thanks to the results of Büchi [Büc60] and Elgot [Elg61],

---

<sup>1</sup>See e.g. [Tho97, Proposition 4.1].

we know that the expressive power of finite automata is the same as the expressive power of monadic second order logic (an extension of first order logic).

In this thesis we concentrate on infinite words and trees. While infinite objects occur naturally in mathematics, it may require explanation why should they be considered on the ground of computer science. Infinite words are used e.g. in formal verification to model potentially infinite runs of systems, like an operating system or a server interacting with clients (see e.g. [JGP99]). In such a context system can be modeled as an automaton, while a specification of a system can be expressed in some logic. Infinite trees occur e.g. when we need to reason about all possible computations of a nondeterministic system (see e.g. branching time concept [Eme90]).

By considering infinite structures, we gain a variety of complexity measures that come from descriptive set theory. A set of all words or all trees over a finite alphabet is, from the topological point of view, equivalent to the Cantor space—one of the two main spaces of interest of descriptive set theory (the other being the Baire space). The measures provided by this discipline include: the Borel hierarchy, the projective hierarchy, and the Wadge hierarchy.

A hierarchy is here understood as an increasing, with respect to inclusion, sequence of classes of sets. Hierarchies implement a way of measuring complexity of sets, already mentioned by Addison in the paragraph cited above. Once we know that a given set is definable in a given formalism, we ask how complex is its description. For example: What is the number of alternations between existential and universal quantifiers in a formula necessary to describe the set? Or: At what level of the Borel hierarchy the set is?

Hierarchies also provide a way to measure complexity of formalisms, or more generally, to compare complexity measures. Let  $\mathbf{F}$  be a formalism and let  $\mathbf{H}$  be a complexity hierarchy defined in terms of a different formalism, e.g. the Borel hierarchy. We may ask what level of the hierarchy do the sets definable in  $\mathbf{F}$  reach. We call the answer to this question for the hierarchy being the combined Borel and projective hierarchy the **topological complexity** of  $\mathbf{F}$ . If  $\mathbf{F}$  comes with associated internal hierarchy  $\mathbf{G}$ , e.g. alternation depth hierarchy for a logic, we may additionally ask how classes of  $\mathbf{G}$  embed into classes of  $\mathbf{H}$ . The concepts are presented in Figure 1.

In his paper [Büc62], Büchi showed that MSO logic over  $\omega$ -words is equivalent to a certain model of finite automata, later called Büchi automata. Since this model, as many other automata models, came with straightforward decidability properties, the equivalence was used to prove decidability of MSO logic over  $\omega$ -words. Similar method—through introduction of an appropriate

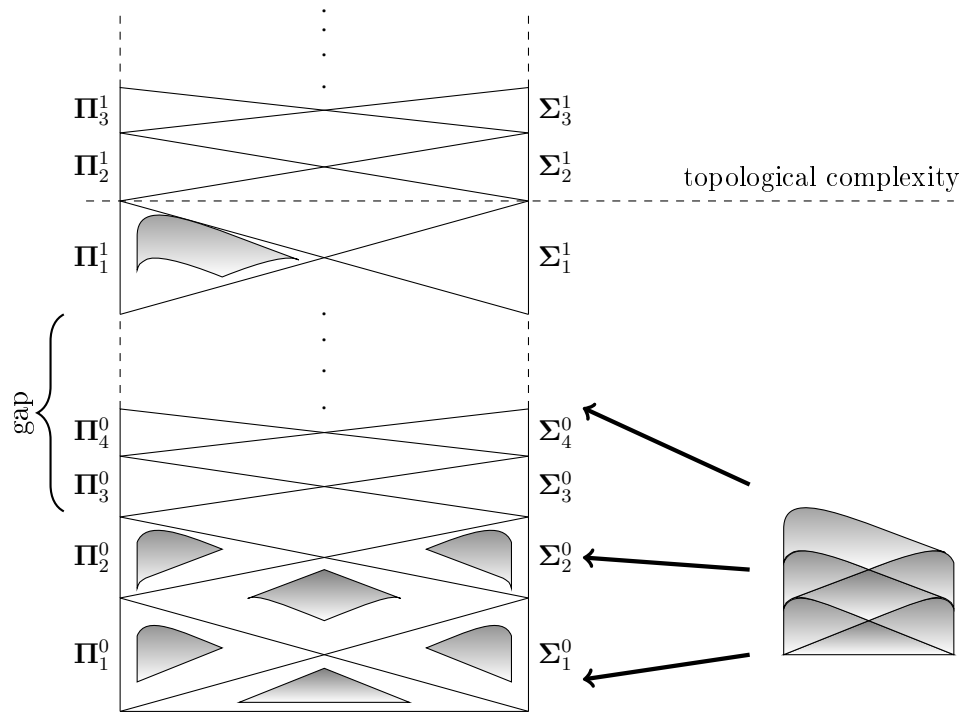


Figure 1: Embedding of the class of sets definable in a given formalism into the Borel and the projective hierarchies.

model of automata—was used by Rabin [Rab69] to prove that MSO logic is decidable also over infinite binary labeled trees. Sets definable in MSO logic are called **regular languages**, both for words and for trees.

Until now several models of automata equivalent to the Rabin’s one were developed. In this thesis, we concentrate on one of them: parity tree automata. We also consider parity automata on  $\omega$ -words. This is because those are the ones that give rise to a hierarchy, namely so called **Rabin-Mostowski index** hierarchy—another descriptive complexity measure.

Regular languages of infinite words (called  $\omega$ -regular languages) are already well understood: topologically, they reach the class of boolean combinations of  $\Pi_2^0$  (or  $G_\delta$ ) sets in the Borel hierarchy and not beyond [BL69]; the Rabin-Mostowski index hierarchy for nondeterministic automata collapses at the level of  $(1, 2)$ , i.e. each  $\omega$ -regular language is recognized by a Büchi automaton (see e.g. [PP04, Proposition 10.6]); the Rabin-Mostowski index hierarchy for deterministic automata is known to be strict and for a given language one can effectively compute its position in the index hierarchy [NW98].

Robustness and tractability of the class of  $\omega$ -regular languages have encouraged researchers, including Bojańczyk and Colcombet [Boj04, BC06, Boj11, Boj10], to look for extension of this class of sets that would maintain some of the good properties. One of the approaches [Boj04] was to add to MSO logic a quantifier, called **unbounding quantifier**, that allows for expressing properties like “the length of blocks of consecutive letters  $a$  in a word is unbounded”. The extended logic is called MSO+U. Similar quantitative extensions of Büchi automata were considered [BC06]. Automata with bounding condition, called  $\omega$ B-automata, are able to recognize languages like

$$L_B = \{a^{n_0}ba^{n_1}ba^{n_2}b \dots \mid \limsup n_i < \infty\},$$

while automata with strongly unbounding condition, called  $\omega$ S-automata, are able to recognize languages like

$$L_S = \{a^{n_0}ba^{n_1}ba^{n_2}b \dots \mid \liminf n_i = \infty\}.$$

Automata combining features of the two above models are called  $\omega$ BS-automata. Chapter 2 presents topological complexity analysis of  $\omega$ B-automata,  $\omega$ S-automata,  $\omega$ BS-automata and MSO+U logic, what is described in details in the next section.

On the other hand, the class of regular languages of infinite trees is still a bit mysterious. Its topological complexity is non-Borel, namely regular tree languages reach the level  $\Delta_2^1$  of the projective hierarchy. The computability of nondeterministic Rabin-Mostowski index is proven only if as an input

one provides a language recognized by a deterministic automaton, called a deterministic language [NW05], or a language recognized by a so-called game automaton [FMS16]. Also the shape of the embedding into the Borel and projective hierarchies (see Figure 1) is better understood for deterministic than for all regular languages. Niwiński and Walukiewicz have shown that each deterministic language is either  $\Pi_1^1$ -complete or is at most at the level  $\Pi_3^0$  of the Borel hierarchy. It is clear, for the cardinality reasons, that some gap exists also for the full class of regular languages, but the question what levels does the gap embrace remains open. Skurczyński [Sku93], Duparc and Murlak [DM07] have described the embedding of another subclass of regular tree languages into the Borel hierarchy. Namely, they have shown that languages recognized by weak automata inhabit exactly all finite levels of the Borel hierarchy.

The complexity of the class of regular languages of trees encourages one to look for subclasses that would be simpler to tackle. Since the best understood subclass corresponds to deterministic automata, we consider a natural extension of the class—unambiguous automata. An important characteristic of deterministic automata is that they admit only one run on each input. Unambiguous automata, in a way, preserve this characteristic by having at most one accepting run on each input while syntactically preserving nondeterminism. It is known that the class is a strict extension of the class of deterministic languages and a strict subclass of the class of regular languages [NW96, CL07, CLNW10].

The class of unambiguous tree languages is not well understood yet. For instance, it is not known if the following question is decidable: given a regular language of infinite trees, is the language recognized by some unambiguous automaton. Before results obtained by the author of this thesis, it was not even known whether topological complexity of this class is any greater than the complexity of deterministic languages, which are known to be co-analytic (i.e.  $\Pi_1^1$ ). It occurred to be greater [Hum12]. The thesis presents results that lift lower topological complexity bound for unambiguous languages, but no upper bound is given. In particular, it is still not known whether for each regular language there exists an unambiguous language that is topologically harder.

## Contribution

We prove that the topological complexity of the class of languages recognized by  $\omega$ B-automata ( $\omega$ S-automata,  $\omega$ BS-automata, respectively) is  $\Sigma_3^0$  ( $\Pi_3^0$ ,  $\Sigma_4^0$ , respectively) Borel class (Theorems 2.5.10, 2.5.13).

We give a sequence of languages  $L_n$  (Definition 2.5.19) hard for consecutive finite levels of the Borel hierarchy definable in MSO+U (Proposition 2.5.23). We prove that the languages are recognized by an alternating variant of  $\omega$ BS-automata (Theorem 2.5.25). The result provides a negative answer to the question stated by Bojańczyk and Colcombet [BC06, Chapter 6], whether nondeterministic  $\omega$ BS-automata are equivalent to their alternating variant.

We construct an infinite sequence of MSO+U definable languages of  $\omega$ -words hard for consecutive levels of the projective hierarchy (Theorem 2.4.1). The results conclude the subject of the topological complexity of MSO+U, since it is easy to see that each MSO+U definable language is projective.

All the mentioned results on the topological complexity of  $\omega$ B-automata,  $\omega$ S-automata,  $\omega$ BS-automata, and MSO+U were published [HMT10, HS12].

We construct an analytic-complete unambiguous language of infinite trees (language  $G$  in Section 3.7). This language witnesses that unambiguous languages are topologically more complex than the deterministic ones, that are all co-analytic. The result was published [Hum12]. We observe that the language  $G$  is **bi-unambiguous**, i.e. unambiguous, whose complement is also unambiguous.

We present an operation that applied to a given tree language  $L$  outputs a language that is topologically harder than any countable boolean combination of sets continuously reducible to  $L$  (Theorem 3.8.16). The operation preserves bi-unambiguity (Theorem 3.10.24). The operation also preserves a topological property called **stretchability**. As a result consecutive iterations of application of the operation to a stretchable language give topologically harder and harder languages (Theorem 3.8.37). We also construct another version of the operation (also preserving bi-unambiguity and stretchability), that allows for transfinite iterations (Section 3.11). Using the two operations, starting from any bi-unambiguous stretchable language, we can construct a sequence of length  $\omega^2$  of bi-unambiguous languages of increasing topological complexity. The results described in this paragraph are not published yet.

## Applications of the Results

The topological complexity of MSO+U was used by Bojańczyk, Gogacz, Michalewski and Skrzypczak [BGMS14] to prove that the logic is “almost undecidable”. Precisely, they have shown that no algorithm which decides the MSO+U theory of the full binary tree has a correctness proof in ZFC. The technique used in this proof became a model example of utilization of relation between topological complexity and decidability. One year later Bo-

jańczyk, Parys and Toruńczyk [BPT16] gave undecidability proof that did not use topology. However, the topological results have given a direction towards a solution of the decidability question that was open for a decade.

While working on the topological complexity of unambiguous languages, the author have consulted Jacques Duparc to find out how operations introduced by Wadge and Duparc, and then transferred to trees by Murlak [Wad72, Wad84, Dup01, Mur06, DM07], fit the unambiguous world. The co-operation resulted in a paper [DFH15], coauthored also by Kevin Fournier, about the Wadge hierarchy of unambiguous languages. The result used language  $G$  mentioned above.

## Structure of the Thesis

Chapter 1 serves as a kind of extended introduction. In Sections 1.1, 1.2 we recall basic definitions from set theory, topology and descriptive set theory, and provide as many facts as it is necessary to be able to apply descriptive set theory to languages of infinite words and trees. In Section 1.3 we recall what it means for an automaton or a formula to define a language. The central notion of the thesis, i.e. the topological complexity of a formalism is explained in Section 1.4. In Section 1.5 we enumerate some applications of topological complexity to formal language theory, what gives a motivation for the study in this context. A reader acquainted with descriptive set theory and its applications to formal language theory may skip this chapter, or quickly skim over it taking notice of the definition of a **topological complexity class** at the beginning of Section 1.2 and Definition 1.4.1, that may be considered not entirely standard. Note additionally that Sections 1.5.2 and 1.5.3 contain abstract statements and remarks that, formally, may be considered as proof templates to be referenced later.

Chapter 2 contains results concerning topological complexity of MSO+U definable and  $\omega$ BS-automata definable languages of  $\omega$ -words. Sections 2.1-2.3 and the initial part of Section 2.5 provide the context, basic definitions, background and former results of other researchers. Sections 2.4 and 2.5.1-2.5.3 present original results: Section 2.4 presents the topological complexity of MSO+U logic; Sections 2.5.1-2.5.2 present the topological complexity of  $\omega$ B-,  $\omega$ S-, and  $\omega$ BS-automata; Section 2.5.3 presents a lower topological complexity bound for alternating  $\omega$ BS-automata. The chapter is concluded in Section 2.6.

Chapter 3 contains results on the topological complexity of bi-unambiguous languages of infinite trees. Sections 3.1, 3.2 provide definitions of parity automata and regular tree languages, but in a bit generalized way suited to

the further needs. Also the presentation of index hierarchy in Section 3.4 may be considered nonstandard, although the notion itself is the same as in classical texts in the field. Sections 3.3, 3.5, and 3.6 present state-of-the-art results to be contrasted with author’s results. The original results by the author of this thesis are presented in Sections 3.7-3.11, where in Section 3.7 we recall the analytic-complete bi-unambiguous language presented before on a conference [Hum12], and the remaining sections introduce abstract operations on tree languages that preserve bi-unambiguity and lift topological complexity, that are not included in any publication yet. Section 3.12 contains various concluding remarks concerning unambiguous languages.

Let us explain the convention we use for theorem-like statement names.

- Fact – a well known fact or a theorem by another researcher,
- Proposition – the author’s contribution of minor importance,
- Theorem – the author’s contribution of substantial importance, or another researcher’s result that the author did not dare to name a “fact” (always signed with a name or reference),
- Lemma – a proposition of an importance only in the context of the currently considered proof of a theorem or a proposition,
- Remark – a statement that, in the author’s opinion, does not require a proof in a given context, but is stated for further reference.

## Credits

The lower complexity bounds stated in Theorems 2.5.10 and 2.5.13 were proven using the example languages provided by the authors of  $\omega$ B-automata and  $\omega$ S-automata models, Mikołaj Bojańczyk and Thomas Colcombet [BC06]. Lemma 2.5.14 giving an upper topological complexity bound for  $\omega$ BS-automata was proven by Szymon Toruńczyk.

A sequence of MSO+U definable languages  $L_n$  (Definition 2.5.19) hard for respective finite levels of the Borel hierarchy was given by Michał Skrzypczak (Proposition 2.5.23). The author of this thesis have proven that the languages are recognized by an alternating variant of  $\omega$ BS-automata (Theorem 2.5.25).

The results concerning projective topological complexity of MSO+U were obtained together with Skrzypczak (Theorem 2.4.1). The presentation of one of the parts of the proof was improved in cooperation with Toruńczyk.



At an early stage of the construction of the operation presented in Section 3.8 the author have obtained support from Skrzypczak, that made the operation simpler, clearer and more general.

The notion of stretchability (Definition 3.8.31) and the way how it can be used to prove that a set is not continuously reducible to itself, was borrowed from André Arnold and Damian Niwiński [AN07].

# Chapter 1

## Topological Complexity

### 1.1 Preliminaries

**Set Theory** By  $\mathbb{N}$  we note the set of natural numbers (including 0), by  $\omega$  the smallest infinite ordinal (the type of the natural order on  $\mathbb{N}$ ) and by  $\omega_1$  the smallest uncountable ordinal. We recall that the **cofinality** of  $\omega_1$  is  $\omega_1$ , i.e. supremum of a countable set of countable ordinals is always countable.

Let  $X$  be a set. For a subset  $A \subseteq X$  we denote by  $\bar{A}$  the complement of  $A$ , i.e.  $X \setminus A$ . Powerset of  $X$  (the set of all subsets of  $X$ ) is denoted by  $\mathbb{P}(X)$ . By  $|X|$  we denote the cardinality (number of elements) of  $X$ .

**Definition 1.1.1 (Set Algebra)** *Let  $X$  be a set. A set  $\mathcal{A} \subseteq \mathbb{P}(X)$  of subsets of  $X$  is an **algebra** if  $\emptyset \in \mathcal{A}$ ,  $\mathcal{A}$  is closed under finite intersections, finite unions, and under complements.*

For a set  $\mathcal{A}$  of subsets of some set  $X$ , we use a notation  $BC(\mathcal{A})$  for an algebra generated by  $\mathcal{A}$ , i.e. the smallest algebra containing all elements of  $\mathcal{A}$ . Note that the notion is well-defined, because  $\mathbb{P}(X)$  is an algebra and an arbitrary intersection of algebras is an algebra. Set  $BC(\mathcal{A})$  is often referred to as the set of (finite) boolean combinations of sets in  $\mathcal{A}$ .

**Definition 1.1.2 (Sigma-Algebra)** *Let  $X$  be a set. A set  $\mathcal{A} \subseteq \mathbb{P}(X)$  of subsets of  $X$  is a **sigma-algebra** if  $\emptyset \in \mathcal{A}$ ,  $\mathcal{A}$  is closed under countable intersections, countable unions, and under complements.*

For  $\mathcal{A} \subseteq \mathbb{P}(X)$ , we denote by  $\sigma(\mathcal{A})$  the sigma-algebra generated by  $\mathcal{A}$ , i.e. the smallest sigma-algebra containing all elements of  $\mathcal{A}$ . This is also a well-defined notion, because  $\mathbb{P}(X)$  is a sigma-algebra and an arbitrary intersection of sigma-algebras is a sigma-algebra. Intuitively,  $\sigma(\mathcal{A})$  is the class of all sets obtained from sets in  $\mathcal{A}$  by countable boolean operations.

## Topology<sup>1</sup>

A **topological space** is a pair  $(X, \tau)$ , where  $X$  is a set and  $\tau \subseteq \mathbb{P}(X)$  is a **topology**, i.e. such set of subsets of  $X$  that  $\emptyset \in \tau$ ,  $X \in \tau$ , and that is closed under arbitrary unions and finite intersections. Members of  $\tau$  are called **open sets**. Complements of open sets are called **closed sets**. A set  $\mathcal{B} \subseteq \tau$  is called a **basis** (of the topology  $\tau$ ) if each open set can be obtained as a union of sets from  $\mathcal{B}$  (by the convention the empty set is obtained as the union of the empty family of sets). A **subbasis** is such a set  $\mathcal{S} \subseteq \tau$  that closure of  $\mathcal{S}$  under finite intersection gives a basis.

A **subspace** of a topological space  $(X, \tau)$  is a topological space consisting of a subset  $Y \subseteq X$  and the **relative topology**  $\tau|Y = \{U \cap Y : U \in \tau\}$ .

Let  $(X, \tau)$  and  $(Y, \theta)$  be topological spaces. A function  $f : X \rightarrow Y$  is called **continuous** if an inverse image of each open set in  $Y$  is open in  $X$ . The function is **open** if an image of each open set in  $X$  is open in  $Y$ . It is a homeomorphism if it is a continuous and open bijection. Homeomorphism plays a role of equivalence in the topological context. If two spaces,  $X$  and  $Y$ , are homeomorphic, i.e. if there is a homeomorphism between them, in symbols  $X \simeq Y$ , they share all the properties of interest of the topology.

A topological space is called **discrete** if all its subsets are open.

One of the ways to define topology of a space is to give a metric.

A **metric space** is a pair  $(X, d)$ , where  $X$  is a set and  $d : X^2 \rightarrow [0, \infty)$  is a **metric**, i.e. a function satisfying:

1.  $d(x, y) = 0 \iff x = y$ ,
2.  $d(x, y) = d(y, x)$ ,
3.  $d(x, z) \leq d(x, y) + d(y, z)$ .

Given a metric space  $(X, d)$ , a **ball** (or precisely a  $d$ -ball) with center  $x$  and radius  $r$  is defined as:

$$B_d(x, r) = \{y : d(x, y) < r\}$$

Set of all balls forms a subbasis of a topology that is called **the topology defined by metric  $d$** . Topological space  $(X, \tau)$  is **metrizable** if there is such a metric  $d$  that  $\tau$  is defined by  $d$ .

We recall here a handy characterization of closed sets in metric spaces. We say that a sequence  $\{x_n\}$  in a metric space  $(X, d)$  **converges** to  $x \in X$ , if for each  $\varepsilon > 0$ , almost all elements of the sequence are in the ball  $B_d(x, \varepsilon)$ . If such an  $x$  exists, it is unique, it is called a **limit** of  $\{x_n\}$ , is denoted by  $\lim \{x_n\}$ , and sequence  $\{x_n\}$  is called **convergent**.

---

<sup>1</sup>Most of the definitions in this section are repeated after Kechris's textbook [Kec95].

**Fact 1.1.3** A subset  $Y \subseteq X$  of a metric space  $(X, d)$  is closed in the topology defined by  $d$  if and only if the limit of each convergent sequence  $\{x_n\} \subseteq Y$  of elements in  $Y$  belongs to  $Y$ .

**Fact 1.1.4** Let  $(X, d_X)$ ,  $(Y, d_Y)$  be metric spaces. Function  $f : X \rightarrow Y$  is continuous (in topologies defined by metrics  $d_X$ ,  $d_Y$ ) if and only if for each convergent sequence  $\{x_n\} \subseteq X$ ,  $\{f(x_n)\} \subseteq Y$  is also convergent and  $f(\lim \{x_n\}) = \lim \{f(x_n)\}$ .

**Definition 1.1.5** Let  $(X, d_X)$ ,  $(Y, d_Y)$  be metric spaces. Function  $f : X \rightarrow Y$  is **contracting** if there is  $c < 1$  such that for each  $x_1, x_2 \in X$ ,  $d_Y(f(x_1), f(x_2)) \leq c \cdot d_X(x_1, x_2)$ .

**Corollary 1.1.6 (of Fact 1.1.4)** Each contracting function is continuous.

Let  $(X, d)$  be a metric space. A **Cauchy sequence** is a sequence  $\{x_n\}$  of elements of  $X$  such that:

$$\forall \varepsilon > 0 \exists_N \forall_{n, m > N} d(x_n, x_m) < \varepsilon$$

We call  $(X, d)$  **complete** if every Cauchy sequence  $\{x_n\}$  has a limit in  $X$ . We call a topological space  $(X, \tau)$  **completely metrizable** if there is such a metric  $d$  that  $(X, d)$  is complete and  $\tau$  is defined by  $d$ .

A subset  $A$  of a topological space  $(X, \tau)$  is **dense** if it intersects each nonempty open set. A topological space is **separable** if it has a countable dense set.

**Fact 1.1.7** Each topological space that has a countable basis is separable. Each metrizable separable space has a countable basis.

A topological space that is separable and completely metrizable is called **Polish**.

A topological space  $(X, \tau)$  is **compact** if every its cover consisting of open sets has a finite subcover, i.e. if  $\{U_i\}_{i \in I}$  is such a family of open sets that  $X = \bigcup_{i \in I} U_i$ , then there exists a finite  $I_0 \subseteq I$  such that  $X = \bigcup_{i \in I_0} U_i$ .

For a family of topological spaces  $\{(X_i, \tau_i)\}_{i \in I}$ , the **product space**  $\prod_{i \in I} (X_i, \tau_i)$  is the Cartesian product of sets  $X_i$  with the topology (called **Tychonoff topology**) where basic open sets are of the form  $\prod_{i \in I} U_i$ , for  $U_i \in \tau_i$ , such that  $U_i \neq X_i$  for only finitely many  $i$ .

If all the spaces in the family are equal, then the product  $\prod_{i \in I} X$  is denoted by  $X^I$ .

**Theorem 1.1.8 (Tychonoff)** The product of any family of compact spaces is compact.

A point  $x$  in a topological space  $(X, \tau)$  is **isolated** if  $\{x\}$  is an open set. A topological space is **perfect** if it has no isolated points.

A **neighborhood** of a point  $x$  is an open set containing  $x$ . A topological space is **Hausdorff** if every two distinct points have disjoint neighborhoods. Note that each metrizable space is Hausdorff.

A subset of a topological space is called **clopen** if it is closed and open. A space is **connected** if it has no clopen subsets except the empty set  $\emptyset$  and the whole space. A space is **zero-dimensional** if it is Hausdorff and has a basis consisting of clopen sets.

### Word Spaces and Tree Spaces

Let  $A$  be an arbitrary set. A (finite or infinite) **word** over alphabet  $A$  is a (finite or infinite, respectively) sequence of elements (called **letters**) from set  $A$ . We denote the set of all finite words over alphabet  $A$  by  $A^*$ . The empty word (the sequence of length 0) is denoted by  $\varepsilon$ . The set of all infinite words of length  $\omega$  ( $\omega$ -words for short) is denoted by  $A^\omega$ . Subsets of  $A^*$  or  $A^\omega$  are called **languages**. We use notation  $A^{\leq \omega}$  for  $A^* \cup A^\omega$ . By  $|w|$  we denote the length of a sequence  $w$ . For a word  $w$ ,  $w_n$  denotes its  $n$ 'th element (counting from 0, unless otherwise noted). We use multiplicative notation for concatenation of sequences. Namely, the concatenation  $wv$  of  $w \in A^*$  and  $v \in A^*$  is defined as follows:

$$wv = w_0w_1 \dots w_{|w|-1}v_0v_1 \dots v_{|v|-1}$$

The concatenation is also well defined in case of the second word infinite, i.e. for  $w \in A^*$ ,  $v \in A^\omega$ :

$$wv = w_0w_1 \dots w_{|w|-1}v_0v_1 \dots$$

Now we define the concatenation of sets of words. For  $L \subseteq A^*$ ,  $M \subseteq A^{\leq \omega}$

$$LM = \{lm : l \in L, m \in M\} \quad (1.1)$$

For  $L \subseteq A^*$ , we define:

$$L^n = \underbrace{LL \dots L}_n \quad (1.2)$$

$$L^* = \bigcup_{n \geq 0} L^n \quad \text{the Kleene star operator} \quad (1.3)$$

$$L^\omega = \{w_0w_1w_2 \dots : \forall_{i \in \omega} w_i \in L\} \quad (1.4)$$

In this context, for singleton languages, we often use abbreviated notation  $v$  instead of  $\{v\}$ .

We use  $\preceq$  to designate the prefix order on words. Namely, for  $v \in A^*$  and  $w \in A^{\leq \omega}$ ,  $v \preceq w$  if and only if there exists  $u \in A^{\leq \omega}$  such that  $w = vu$ . By  $w \upharpoonright_n$  we note the prefix of length  $n$  of sequence  $\alpha \in A^{\leq \omega}$ .

We define topology  $\tau^\omega$  on set  $A^\omega$  by giving the following basis:

$$W = \{W_w\}_{w \in A^*}, \text{ where } W_w = wA^\omega = \{v \in A^\omega : w \preceq v\}.$$

We note that this is standard Tychonoff product topology on the product of  $\omega$  copies of discrete space  $A$ . Since we never use different topology on  $A^\omega$  we will simply write  $A^\omega$  to designate the topological space  $(A^\omega, \tau^\omega)$ .

Let us consider the following metric  $d$  on  $A^\omega$ , called “first difference metric”. For  $w, v \in A^\omega$ :

$$d(w, v) = \begin{cases} 0 & \text{if } w = v \\ 2^{-\min\{n: w_n \neq v_n\}} & \text{otherwise} \end{cases}$$

Since balls in this metric are exactly the elements of basis  $W$  defined above, topology  $\tau^\omega$  is defined by metric  $d$ .

Let  $B$  be a set. An **unlabeled tree** on  $B$  (or  $B$ -branching tree) is a prefix-closed set of finite sequences of elements from  $B$ . The set of all trees on  $B$  is denoted by  $T^B$ . A tree  $t$  is **full** if  $\text{dom}(t) = B^*$ . A tree  $t$  is **binary** if  $|B| = 2$  (we use  $B = \{l, r\}$  in this case).

For  $A$  being a set, a **labeled tree** over alphabet  $A$  ( $A$ -labeled tree) is a function  $t : \text{dom}(t) \rightarrow A$ , where  $\text{dom}(t)$  is an unlabeled tree. A labeled tree  $t$  is **full** if so is  $\text{dom}(t)$ . The set of all full  $B$ -branching trees over  $A$  is denoted by  $T_A^B$ . A labeled tree  $t$  is **binary** (respectively **finite**) if so is  $\text{dom}(t)$ . The set of all finite  $B$ -branching labeled trees over  $A$  is denoted by  $T_A^{B, \text{fin}}$ . Since the whole Chapter 3 is focused on binary trees we use a convention that  $T_A := T_A^{\{l, r\}}$  and  $T_A^{\text{fin}} := T_A^{\{l, r\}, \text{fin}}$ .

The elements of the domain of a tree<sup>2</sup> are called **nodes**. All nodes that are words of length  $n$  form the  $n$ 'th **level** of a tree. A **branch** of a  $B$ -branching tree  $t$  is a sequence  $\alpha \in B^\omega$  such that each prefix of  $\alpha$  is a node in  $t$ . We sometimes identify the branch with this set of nodes. For a labeled tree  $t \in T_A^B$ , by  $t(\alpha)$  we denote the sequence of labels along a branch  $\alpha \in B^\omega$  (so  $t(\alpha) \in A^\omega$ ).

We consider  $T_A^B$  as a topological space. The following family indexed by finite trees is a basis of the topology:

$$G = \{G_s\}_{s \in T_A^{B, \text{fin}}}, \text{ where } G_s = \{t \in T_A^B : \forall v \in \text{dom}(s) \ t(v) = s(v)\} \quad (1.5)$$

---

<sup>2</sup>To be able to talk about labeled and unlabeled trees at the same time whenever possible, we assume  $\text{dom}(t) = t$  for an unlabeled tree  $t$ .

Under such a definition, if  $B$  is countable and nonempty, space  $T_A^B$  can also be viewed as a product  $A^{B^*}$  of countably many copies of discrete space  $A$ . Since the structure of the set that indexes the product does not affect the product topology, we get:

**Remark 1.1.9** *If  $0 < |B| \leq \omega$ ,  $T_A^B \simeq A^\omega$ .*

The above remark suggests that the topology (or descriptive set theory) is an appropriate environment to compare complexity of languages of trees and words, what will be deeper covered in the next section.

In order to define metric on  $T_A^B$  in case of countable  $B$ , we assign natural numbers to the nodes of the full  $B$ -labeled tree. Let  $\text{num} : B^* \rightarrow \mathbb{N}$  be such a function that each value occurs only finitely many times. Let us define a metric  $d_{\text{num}}$  on  $T_A^B$  as follows. For  $t, s \in T_A^B$ :

$$d_{\text{num}}(t, s) = \begin{cases} 0 & \text{if } t=s \\ 2^{-\min\{\text{num}(x) : t(x) \neq s(x)\}} & \text{otherwise} \end{cases}$$

Note that if  $B$  is finite then the function assigning the level to each node of the full  $B$ -branching tree can be used as  $\text{num}$ .

**Fact 1.1.10** *Let  $\text{num} : B^* \rightarrow \mathbb{N}$  be such a function that each value occurs only finitely many times. Metric  $d_{\text{num}}$  defines topology with basis  $G$ .*

**Proof:**

First we note that each set  $G_s$  can be obtained as a union of balls in metric  $d_{\text{num}}$ , namely:

$$G_s = \bigcup \{ B_{d_{\text{num}}}(t, 2^{-\max\{\text{num}(y) : y \in \text{dom}(s)\}}) : t \in G_s \}$$

Now we show how to obtain each  $d_{\text{num}}$ -ball as a union of sets  $G_s$ :

$$B_{d_{\text{num}}}(t, 2^{-n}) = \bigcup \{ G_s : \forall x \in B^* (\text{num}(x) \leq n \implies x \in \text{dom}(s) \wedge t(x) = s(x)) \}$$

Therefore, basis consisting of  $d_{\text{num}}$ -balls and basis  $G$  give the same topology.  $\square$

The above fact implies, in particular, that the topology defined by metric  $d_{\text{num}}$  does not depend on function  $\text{num}$ . Therefore, since we are mainly interested in topological properties, we will simply write  $d$  and not mention any specific  $\text{num}$  function. Since the metric is, in essence, the same as on words<sup>3</sup>, we also call it “first difference metric”.

---

<sup>3</sup>This can be easiest seen when we take a bijection as  $\text{num}$ .

**Remark 1.1.11** *Balls in metric  $d$  are closed.*

**Proof:**

It is enough to note that:

$$\overline{B_{d_{\text{num}}}(t, 2^{-n})} = \bigcup \{G_s : \exists_{x \in \text{dom}(s)} (\text{num}(x) \leq n \wedge t(x) \neq s(x))\}.$$

□

**Corollary 1.1.12** *Spaces  $A^\omega$  and  $T_A^B$  are zero-dimensional.*

Since basis  $W$  of space  $A^\omega$  is countable if and only if  $A$  is countable and basis  $G$  of space  $T_A^B$  is countable if  $A$  and  $B$  are countable, by Fact 1.1.7, we get:

**Fact 1.1.13** *If  $A$  and  $B$  are countable then spaces  $A^\omega$  and  $T_A^B$  are separable.*

Metric spaces  $(A^\omega, d)$  and  $(T_A^B, d)$  are complete. Therefore, topological spaces  $A^\omega$ ,  $T_A^B$  are completely metrizable, what, together with Fact 1.1.13, gives:

**Corollary 1.1.14** *If  $B$  is countable then spaces  $A^\omega$  and  $T_A^B$  are Polish.*

By definition, a discrete space is compact if and only if it is finite. Therefore, by Tychonoff's Theorem:

**Fact 1.1.15** *If  $A$  is finite then spaces  $A^\omega$  and  $T_A^B$  are compact.*

**Fact 1.1.16** *If  $|A| > 1$  and  $|B| > 0$  then spaces  $A^\omega$  and  $T_A^B$  are perfect.*

**Proof:**

Let  $X$  be one of  $A^\omega$  and  $T_A^B$ . Assume, towards a contradiction, that there is such  $x \in X$  that  $\{x\}$  is open, i.e.  $X \setminus \{x\}$  is closed. Recall that  $X$  is metrizable, and consider the metric that is defined above. By the definition of the metric, we can obtain a point (i.e. a word or a tree) arbitrary close to  $x$  by changing a label in just one appropriately chosen position of this word or tree. Therefore we can construct a sequence in  $X \setminus \{x\}$  that converges to  $x$ , what contradicts the closedness of  $X \setminus \{x\}$ . □

If  $A = \{0, 1\}$  then  $A^\omega$  is the Cantor space  $\mathcal{C}$ , as defined e.g. by Kechris [Kec95, Section 3.A]. If  $A = \mathbb{N}$  then  $A^\omega$  is the Baire space  $\mathbb{N}^\omega$ , as defined in the same place. We now recall facts that generalize those observations.

First we recall a theorem characterizing the Cantor space.



**Theorem 1.1.17 (Brouwer)**<sup>4</sup> *Each perfect, nonempty, compact, metrizable, zero-dimensional space is homeomorphic to the Cantor space.*

Using this theorem, thanks to Fact 1.1.16, Fact 1.1.15, Fact 1.1.10 and Corollary 1.1.12, we get:

**Corollary 1.1.18** *If  $1 < |A| < \omega$  and  $0 < |B| \leq \omega$  then spaces  $A^\omega$  and  $T_A^B$  are homeomorphic to the Cantor space.*

Now we recall a theorem characterizing the Baire space.

**Theorem 1.1.19 (Alexandrov-Urysohn)**<sup>5</sup> *Each nonempty Polish zero-dimensional space for which none compact subset has a nonempty open set as a subset is homeomorphic to the Baire space.*

**Corollary 1.1.20** *If  $|A| = \omega$  and  $0 < |B| \leq \omega$  then spaces  $A^\omega$  and  $T_A^B$  are homeomorphic to the Baire space.*

**Proof:**

By Corollary 1.1.14 and Corollary 1.1.12, it is enough to show that none compact subset of considered spaces has a nonempty open set as a subset.

Thanks to Remark 1.1.9 it is enough to prove the claim for  $T_A^B$ . Let us fix a bijection  $\text{num} : B^* \rightarrow \mathbb{N}$ . Take any compact  $Y \subseteq X$ . Assume, towards a contradiction, that there is a nonempty open set  $Z \subseteq Y$ . Then there is a ball  $U = B_{d_{\text{num}}}(x, 2^{-n}) \subseteq Z$ . By Remark 1.1.11,  $U$  is closed, so, as a closed subset of a compact set  $Y$ ,  $U$  is compact (see e.g. [Kec95, Proposition 4.1.ii]). Now consider a family of balls  $\{B_{d_{\text{num}}}(t_a, 2^{-n-1})\}_{a \in A}$  such that:

$$t_a(x) = \begin{cases} a & \text{if } \text{num}(x) = n + 1 \\ t(x) & \text{otherwise} \end{cases}$$

This is an infinite family of pairwise-disjoint nonempty sets that exactly covers set  $U$  (a partitioning of  $U$ ). Therefore, there is no finite subcover, what contradicts compactness of  $U$ .  $\square$

We also consider sets of unlabeled trees as topological spaces. To define a topology on set  $T^B$ , we treat it as a subspace of  $T_{\{0,1\}}^B$ , where tree  $t \in T_{\{0,1\}}^B$  is treated as a characteristic function of a subset of  $B^*$ .

**Fact 1.1.21** *Set  $T^B$  is a closed subset of  $T_{\{0,1\}}^B$ .*

---

<sup>4</sup>See e.g. [Kec95, Theorem 7.4].

<sup>5</sup>See e.g. [Kec95, Theorem 7.7].

**Proof:**

An element  $t \in T_{\{0,1\}}^B$  does not represent an unlabeled tree, i.e. a prefix-closed subset of  $B^*$ , if there is a node  $v$  such that  $t(v) = 1$  and there is a prefix  $w$  of  $v$  such that  $t(w) = 0$ . For any fixed  $v$  and  $w$  the set of such trees is open. The complement of  $T^B$  is the union of such open sets over all possible  $v$ 's and  $w$ 's, therefore, it is also open.  $\square$

Since each closed subspace of a Polish space is Polish (see e.g. [Kec95, Proposition 3.3.ii]), we get:

**Fact 1.1.22** *Space  $T^B$  is Polish.*

Since each closed subset of a compact space is compact (see e.g. [Kec95, Proposition 4.1.ii]), by Fact 1.1.15, we get:

**Fact 1.1.23** *Space  $T^B$  is compact.*

Thanks to the characterization of the Cantor space given in Theorem 1.1.17, it suffices to show the following to observe that  $T^B$  is also the Cantor space.

**Fact 1.1.24** *If  $B$  is infinite then  $T^B$  is perfect.*

**Proof:**

It suffices to show that for each  $t \in T^B$ , there is a sequence of trees  $\{t_n\} \subseteq T^B \setminus \{t\}$  that converges to  $t$ .

Take any  $t \in T^B$ . We consider two cases. First assume that  $t$  has an infinite branch. Then, as a needed sequence, we can take the sequence of trees that instead of this branch have longer and longer its finite prefixes.

Now assume that  $t$  does not have an infinite branch. Then the following can be taken as a needed sequence:  $t_n$  is a tree  $t$  to which branch  $n0^\omega$  was added.

Convergence of sequences in both cases is easy to verify.  $\square$

**Corollary 1.1.25** *If  $|B| = \omega$  then  $T^B$  is homeomorphic to the Cantor space.*

**Additional Notations and Terminology**

Let  $L \subseteq A^*$  and  $M \subseteq A^{\leq \omega}$ . The left quotient  $L^{-1}M$  of language  $M$  with respect to language  $L$  is defined as:

$$L^{-1}M = \{v : \exists u \in L \ uv \in M\}$$

We write  $v^{-1}w$  to designate a suffix of  $w$ , precisely the unique element of  $\{v\}^{-1}\{w\}$ , if we know that  $v \preceq w$ .

Let  $L, M \subseteq A^*$ . The right quotient  $LM^{-1}$  of language  $L$  with respect to language  $M$  is defined as:

$$LM^{-1} = \{v : \exists_{u \in M} vu \in L\}$$

We write  $wv^{-1}$  to designate a prefix of  $w$ , precisely the unique element of  $\{w\}\{v\}^{-1}$ , if we know that  $v$  is a suffix of  $w$ .

We use graph-theoretic terminology for trees. Therefore, the elements of the domain of a tree are called **nodes**; we consider paths connecting nodes (typically the root, i.e. node  $\varepsilon$ , with some other node). Node  $w$  is a **successor** of node  $v$  in a  $B$ -branching tree  $t$  if  $v, w \in \text{dom}(t)$  and  $w = vb$  for  $b \in B$ ;  $v$  is then a **predecessor** (or a **parent**) of  $w$ . Node  $w$  is a **descendant** of node  $v$  in  $t$  if  $v, w \in \text{dom}(t)$  and  $w = vu$  for some  $u \in B^*$ ;  $v$  is then an **ancestor** of  $w$ . If a node does not have any successors it is called a **leaf**, otherwise it is called an **inner node**.

For a tree  $t$  and a node  $v \in \text{dom}(t)$ , **subtree** of  $t$  rooted in  $v$ , denoted  $t_v$ , is a tree such that  $\text{dom}(t_v) = \{w : vw \in \text{dom}(t)\}$  and, if  $t$  is labeled, for each  $w \in \text{dom}(t_v)$ ,  $t_v(w) = t(vw)$ .

## 1.2 Topological Complexity Classes

Analogously to the computational complexity theory, the topological complexity theory uses the notions of **reduction** and **completeness**. Let  $X, Y$  be two zero-dimensional Polish spaces and let  $K \subseteq X$  and  $L \subseteq Y$ . In general, a reduction of  $K$  to  $L$  is a function  $f : X \rightarrow Y$  such that  $K = f^{-1}(L)$ . We say that  $K$  is **Wadge reducible** to  $L$ , in symbols  $K \leq_w L$ , if there is a continuous reduction of  $K$  to  $L$ .

We call a class  $\mathbf{C}$  of subsets of zero-dimensional Polish spaces a **topological complexity class** if the class is downward closed under Wadge reductions, i.e. if for each  $L \in \mathbf{C}$  and  $K \leq_w L$ ,  $K \in \mathbf{C}$ . The notion corresponds to a pointclass from Srivastava's book [Sri98]. A set  $L$  is called **C-hard** if  $K \leq_w L$  for each  $K \in \mathbf{C}$ . We say that  $L$  is **C-complete** if additionally  $L \in \mathbf{C}$ .

The following fact presents a standard way of using the above notions.

**Fact 1.2.1** *If  $\mathbf{C} \subsetneq \mathbf{D}$  are two complexity classes and  $L$  is **D-hard**, then  $L \notin \mathbf{C}$ .*

### 1.2.1 Borel and Projective Hierarchy

Standard examples of topological complexity classes come from the Borel and the projective hierarchies, that were introduced by descriptive set theory.

Definitions follow.

Let us fix a Polish space  $X$ . The **Borel hierarchy** consists of classes  $\Sigma_\alpha^0$ ,  $\Pi_\alpha^0$ , for  $\alpha < \omega_1$ . The classes, restricted to subsets of space  $X$ , are defined inductively:

- $\Sigma_1^0(X)$  denotes the class of open subsets of  $X$ ,
- $\Pi_1^0(X)$  denotes the class of closed subsets of  $X$ ,

for a countable ordinal  $\alpha$ :

- $\Sigma_\alpha^0(X)$  is the class of countable unions of sets from  $\bigcup_{\beta < \alpha} \Pi_\beta^0(X)$ ,
- $\Pi_\alpha^0(X)$  is the class of countable intersections of sets from  $\bigcup_{\beta < \alpha} \Sigma_\beta^0(X)$ .

Note that for each  $\alpha$  the class  $\Sigma_\alpha^0(X)$  consists exactly of the complements of the languages from  $\Pi_\alpha^0(X)$ . The classes constitute a hierarchy—each class is included in all classes with greater subindex (see Figure 1.1).

One also considers two kinds of intermediate classes.

$$\begin{aligned}\Delta_\alpha^0(X) &= \Sigma_\alpha^0(X) \cap \Pi_\alpha^0(X) \\ \mathcal{BC}_\alpha^0(X) &= BC(\Sigma_\alpha^0(X)) = BC(\Pi_\alpha^0(X))\end{aligned}$$

Classes  $\Sigma_\alpha^0(X)$  and  $\Pi_\alpha^0(X)$  are closed under finite unions and intersections. Therefore, classes  $\Delta_\alpha^0(X)$  are algebras, hence  $\mathcal{BC}_\alpha^0(X) \subseteq \Delta_{\alpha+1}^0(X)$ .

The hierarchy is strict. Namely:

**Theorem 1.2.2** (see e.g. [Sri98, Corollary 3.6.8]) *If  $X$  is an uncountable Polish space, then for  $\alpha \geq 1$ ,  $\Sigma_\alpha^0(X) \setminus \Delta_\alpha^0(X) \neq \emptyset$  and  $\Pi_\alpha^0(X) \setminus \Delta_\alpha^0(X) \neq \emptyset$ .*

The class of **Borel sets** is defined as

$$\text{Bor}(X) := \sigma(\Sigma_1^0(X)) = \bigcup_{\alpha < \omega_1} \Sigma_\alpha^0(X) = \bigcup_{\alpha < \omega_1} \Pi_\alpha^0(X)$$

We use notation  $\Sigma_\alpha^0$ ,  $\Pi_\alpha^0$ ,  $\text{Bor}$ , etc. for topological complexity classes corresponding to appropriate classes of subsets of Polish spaces. Therefore,  $\Sigma_\alpha^0$  aggregates  $\Sigma_\alpha^0(X)$  for all Polish spaces  $X$ , etc. We note that all classes of the hierarchy are closed under continuous preimages (see [Sri98, Exercise 3.6.4]), hence are topological complexity classes.

Proofs and details about the Borel hierarchy can be found e.g. in Srivastava's handbook [Sri98, Chapter 3.6]. The hierarchy is presented on Figure 1.1.

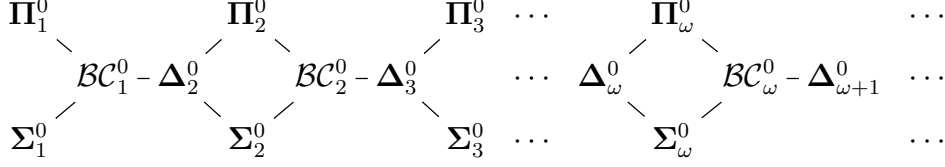


Figure 1.1: The Borel hierarchy ( $\omega_1$  levels).

The class of Borel sets is not closed under continuous images. In particular, a Borel subset of a product space when projected onto one of the coordinate spaces can no longer be Borel. We define the following class of projections of Borel sets:

$$\Sigma_1^1(X) = \{P \subseteq X : \exists B \in \text{Bor}(\mathbb{N}^\omega \times X) \ P = \pi_2(B)\},$$

where  $\pi_2$  is the projection on the second coordinate, and call each set in  $\Sigma_1^1$  **analytic**. The superscript 1 in  $\Sigma_1^1$  means that the class is a part of the projective hierarchy. The rest of the **projective hierarchy** is defined as follows. For  $i < \omega$ :

- $\Pi_i^1(X)$  consists of the complements of the sets from  $\Sigma_i^1(X)$ ,
- $\Sigma_{i+1}^1(X)$  consists of the projections of the sets from  $\Pi_i^1(X)$ .

The sets from class  $\Pi_1^1$  are called **co-analytic**.

The two intermediate classes considered for the projective hierarchy are:

$$\begin{aligned} \Delta_i^1(X) &= \Sigma_i^1(X) \cap \Pi_i^1(X) \\ \sigma(\Sigma_i^1(X)) &= \sigma(\Sigma_i^1(X)) = \sigma(\Pi_i^1(X)) \end{aligned}$$

Classes  $\Sigma_i^1(X)$  and  $\Pi_i^1(X)$  are closed under countable unions and intersections (see [Sri98, Proposition 4.1.7]). Therefore,  $\Delta_i^1(X)$  is a sigma-algebra and  $\sigma(\Sigma_i^1(X)) \subseteq \Delta_i^1(X)$ .

The projective hierarchy is also strict.

**Theorem 1.2.3** (see e.g. [Sri98, Theorem 4.1.11]) *If  $X$  is an uncountable Polish space, then for  $i < \omega$ ,  $\Sigma_i^1(X) \setminus \Delta_i^1(X) \neq \emptyset$  and  $\Pi_i^1(X) \setminus \Delta_i^1(X) \neq \emptyset$ .*

The remarkable result by Souslin ties the two hierarchies together:

**Theorem 1.2.4** ([Sou17]) *If  $X$  is a Polish space, then  $\Delta_1^1(X) = \text{Bor}(X)$ .*

By Theorem 1.2.3 and Theorem 1.2.4 we get:

**Corollary 1.2.5** *In every uncountable Polish space there is an analytic set that is not Borel and there is a co-analytic set that is not Borel.*

Since all the classes of the projective hierarchy are closed under continuous preimages, they are topological complexity classes.

A set  $P \subseteq X$  is called **projective** if  $P \in \Sigma_i^1$  for some  $i < \omega$ .

The Borel hierarchy together with the projective hierarchy constitute the so-called **boldface hierarchy**, see the diagram on Figure 1.2.

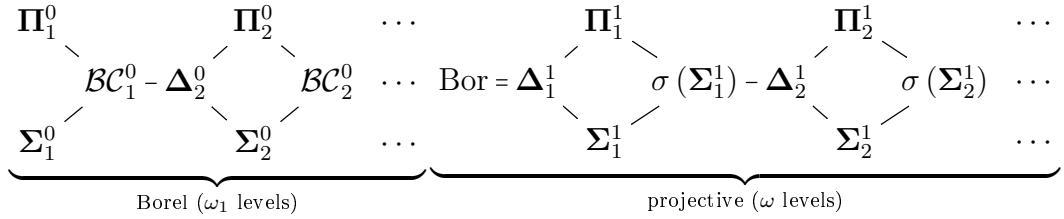


Figure 1.2: The boldface hierarchy.

### Examples of Complete Sets

**Theorem 1.2.6** (see e.g. [Kec95, Theorem 22.10]) *Let  $X$  be a zero-dimensional Polish space. Set  $A \subseteq X$  is  $\Sigma_\alpha^0$ -complete if and only if  $A \in \Sigma_\alpha^0 \setminus \Pi_\alpha^0$ .*

We use the following abbreviated notation:

- $\exists_x^\infty \varphi(x)$  — there exists infinitely many positions  $x$  for which  $\varphi$  holds
- $\forall_x^\infty \varphi(x)$  — for all  $x$  except finitely many,  $\varphi$  holds

**Fact 1.2.7** ([Kec95, Page 179])

1. Language  $Q_1 := \{w \in \{0, 1\}^\omega : \exists_n w_n = 1\}$  is  $\Sigma_1^0$ -complete.
2. Language  $N_1 := \{w \in \{0, 1\}^\omega : \forall_n w_n = 0\}$  is  $\Pi_1^0$ -complete.
3. Language  $N_2 := \{w \in \{0, 1\}^\omega : \exists_n^\infty w_n = 1\}$  is  $\Pi_2^0$ -complete.
4. Language  $Q_2 := \{w \in \{0, 1\}^\omega : \forall_n^\infty w_n = 0\}$  is  $\Sigma_2^0$ -complete.

Before we proceed with a proof, we recall two more facts from Descriptive Set Theory. We use the following notation for the union of the family of languages:

$$\bigsqcup_{i \in I} L_i$$

to indicate that the sets in the family are pairwise disjoint.

**Lemma 1.2.8** (see e.g. [Sri98, Proposition 3.6.3]) *Let  $X$  be a zero-dimensional Polish space, and  $\alpha > 2$ . Each set  $L \in \Sigma_\alpha^0(X)$  is a countable disjoint union  $\bigsqcup_{n < \omega} L_n$  of sets from  $\bigcup_{\beta < \alpha} \Pi_\beta^0(X)$ .*

We use the following notation for the intersection of a decreasing (not necessarily strictly) sequence of sets:

$$\bigcap_{n < \omega}^{\rightarrow} L_n$$

Again, semantics of this operator is identical as of ordinary intersection. It is only used to indicate that for each  $n < m$ ,  $L_m \subseteq L_n$ .

**Lemma 1.2.9** *Let  $X$  be a Polish space, and  $\alpha > 1$ . Each set  $L \in \Pi_\alpha^0(X)$  is an intersection  $\bigcap_{n < \omega}^{\rightarrow} L_n$  of a decreasing  $\omega$ -sequence of sets from  $\bigcup_{\beta < \alpha} \Sigma_\beta^0(X)$ .*

**Proof:**

By the definition  $L = \bigcap_{n < \omega} M_n$ , where  $M_n \in \bigcup_{\beta < \alpha} \Sigma_\beta^0(X)$  for each  $n$ . It is enough to take  $L_n = \bigcap_{k \leq n} M_k$ .  $\square$

**Proof (of Fact 1.2.7):**

(1) Note that:

$$Q_1 = \bigcup_{n < \omega} \underbrace{\{w : w_n = 1\}}_{D_n}$$

Since for a fixed  $n < \omega$ ,  $D_n$  is a clopen set,  $Q_1 \in \Sigma_1^0$ , as a countable union of clopen sets.

To prove hardness, we note that, since  $\{0, 1\}^\omega$  is a zero-dimensional uncountable Polish space, there is a  $\Sigma_1^0$ -complete subset  $U$  of  $\{0, 1\}^\omega$  (see Theorem 1.2.6, Theorem 1.2.2, Corollary 1.1.14, Corollary 1.1.12). By the definition of the topology on words:

$$U = \bigcup_{w \in W} w\{0, 1\}^\omega, \text{ for some } W \subseteq \{0, 1\}^*.$$

Hence the following continuous function  $f : \{0, 1\}^\omega \rightarrow \{0, 1\}^\omega$ :

$$(f(x))_n = \begin{cases} 1 & \text{if } x \upharpoonright_n \in W \\ 0 & \text{otherwise} \end{cases}$$

is a reduction of  $U$  to  $Q_1$ , what concludes the proof of  $\Sigma_1^0$ -hardness of  $Q_1$ .

(2) It is enough to note that  $N_1 = \overline{Q_1}$ .

(3) For  $\omega$ -words formula  $\exists_n^\infty \varphi(n)$  can be unravelled to:

$$\exists_n^\infty \varphi(n) = \forall_m \exists_{n>m} \varphi(n)$$

Therefore:

$$N_2 = \bigcap_m \bigcup_{n>m} D_n \in \Pi_2^0(\{0, 1\}^\omega)$$

To prove hardness, as in point (1), we note that there is a  $\Pi_2^0$ -complete set  $F \subseteq \{0, 1\}^\omega$ . By, Lemma 1.2.9:

$$F = \bigcap_{m<\omega} \overrightarrow{U_m},$$

where  $U_m \in \Sigma_1^0$  for each  $m$ . Now, let:

$$U_m = \bigcup_{n<\omega} w_{m,n} \{0, 1\}^\omega,$$

where  $w_{m,n} \in \{0, 1\}^*$ , for each  $m$  and each  $n$ . Note that we can assume that<sup>6</sup>:

$$\forall_m \neg \exists_{n_1, n_2} w_{m, n_1} \preceq w_{m, n_2} \quad (1.6)$$

Let us fix a bijection  $\iota : \omega \times \omega \rightarrow \omega$ , and define a function  $f : \{0, 1\}^\omega \rightarrow \{0, 1\}^\omega$ , by:

$$(f(x))_{\iota(m,n)} = \begin{cases} 1 & \text{if } w_{m,n} \preceq x \\ 0 & \text{otherwise} \end{cases}$$

Since position  $\iota(m, n)$  of  $f(x)$  depends only on the prefix of  $x$  of length  $\max\{|w_{m,k}| : k \leq n\}$ , the function is continuous. The following proves that  $f$  reduces  $F$  to  $N_2$ .

$$\begin{aligned} f(x) \in N_2 & \iff \exists_k^\infty (f(x))_k = 1 \\ & \iff \exists_{(m,n)}^\infty w_{m,n} \preceq x && \text{by the definition of } f \\ & \iff \exists_m^\infty \exists_n w_{m,n} \preceq x && \text{by (1.6)} \\ & \iff \exists_m^\infty x \in U_m \\ & \iff \forall_m x \in U_m && \text{because } \{U_m\}_{m<\omega} \text{ is decreasing} \\ & \iff x \in F \end{aligned}$$

---

<sup>6</sup>Note that Lemma 1.2.8 does not imply (1.6), but they both follow the same idea.



Hence,  $N_2$  is  $\Pi_2^0$ -hard.

(4) It is enough to note that  $Q_2 = \overline{N_2}$ .

□

### 1.2.2 Wadge Hierarchy

The finest topological complexity measure is given by the Wadge hierarchy. It is the finest because the classes in this hierarchy (called **Wadge degrees**) correspond exactly to the notion of reducibility.

The hierarchy is defined by **Wadge quasiorder**. Let  $X$  and  $Y$  be Polish spaces. **Wadge equivalence** relation is defined for  $A \subseteq X$ ,  $B \subseteq Y$  as:  $A \equiv_w B$  if  $A \leq_w B$  and  $B \leq_w A$ . Equivalence classes of the relation, called **Wadge degrees**, constitute a hierarchy ordered by Wadge quasiorder. The shape and characteristics of this hierarchy are described e.g. by Kechris [Kec95, Section 21.E].

## 1.3 Automata, Formulas and Definability

We give an abstract definition of an automaton. Although types of automata significantly differ in the shape of runs and other technical details, in this chapter we want to introduce as much of notions as it is necessary to be able to link automata theory to topological complexity. In particular we say what it means for an automaton to define a language. The detailed definitions of particular types of automata of interest will be given in the following chapters.

Each type of automata is compatible with a particular class of structures, e.g. infinite words over finite alphabets, or infinite binary trees over finite alphabets, or other graph-like structures. Let  $\mathcal{A}$  be an automaton of a given type  $\mathcal{T}$ . Each structure compatible with  $\mathcal{A}$  can be **accepted** by  $\mathcal{A}$  or not. In the latter case we say that the structure is **rejected** by  $\mathcal{A}$ . Language **recognized** by  $\mathcal{A}$  is the subset of space of all structures compatible with  $\mathcal{A}$  that contains structures accepted by  $\mathcal{A}$ . For automata “recognizing” is the way of **defining** a language. The class of languages recognized by automata of a given type is an object of consideration of this thesis.

Using logical formulas to define sets is a common activity in the field of mathematics. The definability notion here comes from model theory. In our context, a logic is a set of connectives and quantifiers in use, together with their associated meaning. Let us fix a logic  $\mathcal{L}$  and a signature  $S$  (set of relational symbols with their arities). Let  $\varphi$  be a formula of logic  $\mathcal{L}$  over

signature  $S$ . In language theory, in order to be able to say what language  $\varphi$  defines, we need to fix a kind of structures we are interested in. From the model theoretical point of view it is usually equivalent to putting restrictions not only on the domain of a model, but also on the interpretation of relations from  $S$  in the model. For example if we want to consider languages of infinite binary trees over alphabet  $\{a, b\}$ , the signature could contain two binary relations  $\mathbf{l}$  and  $\mathbf{r}$ , and two unary relations  $\mathbf{a}$  and  $\mathbf{b}$ , the domain would be fixed as  $\{l, r\}^*$ , and the interpretation of the binary relations would be fixed as left and right successor relation, while interpretation of the unary relations would reflect the label of a given node, i.e. for each element of the domain exactly one of the predicates  $\mathbf{a}$ ,  $\mathbf{b}$  would hold. If, then,  $\mathbf{X}$  is the class of all structures (models) of interest,  $\varphi$  **defines** the language of all structures from class  $\mathbf{X}$  that are models of  $\varphi$ . In metamathematical notation:

$$L(\varphi) = \{L \in \mathbf{X} : L \models \varphi\}$$

We use word **formalism** as a common name for a type of objects that can be used to define languages. In particular, certain type of automata is a formalism, as is a logic. The class of languages that can be defined using a given formalism is often referred to as the **expressive power** of the formalism.

## 1.4 Topological Complexity of Classes of Languages

The questions that we concentrate on in this thesis is of the form:

Given a class of languages  $\mathbf{X}$ .  
What is the topological complexity of  $\mathbf{X}$ ?

Here comes an explanation how to read this question.

Our context for this question is always the definability theory (see e.g. [Add04]). Therefore,  $\mathbf{X}$  is always a class of languages definable by automata of some specific type or by formulas of some specific logic (language theoretic class). Standard examples of language theoretic classes would be: the class of languages of words of length  $\omega$  recognized by finite parity automata or the class of languages of infinite binary trees defined by formulas of MSO logic with two successor relations and predicates for labels.

**Definition 1.4.1** *The **topological complexity of class  $\mathbf{X}$**  is the downward closure of  $\mathbf{X}$  under Wadge reduction, i.e. the least topological complexity class  $\mathbf{C}$  such that  $\mathbf{X} \subseteq \mathbf{C}$ .*

*The **topological complexity of a formalism** is defined as the topological complexity of the class of all languages defined in this formalism.*

Note that, under the above definition, if  $\mathbf{C}$  is such a topological complexity class that  $\mathbf{X} \subseteq \mathbf{C}$ , then  $\mathbf{C}$  is an **upper topological complexity bound** of  $\mathbf{X}$ . If  $\mathbf{C}$  is such a topological complexity class that for each set  $A$  from  $\mathbf{C}$  there is such a language  $A' \in \mathbf{X}$  that  $A \leq_w A'$ , then  $\mathbf{C}$  is a **lower topological complexity bound** of  $\mathbf{X}$ .

A lower topological complexity bound is usually determined using complete sets. Namely, if topological complexity class  $\mathbf{C}$  admits complete sets, then it can be proven that  $\mathbf{C}$  is a lower topological complexity bound of class  $\mathbf{X}$  by showing  $\mathbf{C}$ -complete set in class  $\mathbf{X}$ .

## 1.5 Applications

In this section we enumerate some basic applications of topological complexity methods in formal language theory.

### 1.5.1 Separation of Classes

First we show how topological complexity can be used to prove undefinability in a given formalism. Let  $\mathcal{F}$  be a formalism, and assume that it is known that its upper topological complexity bound is  $\mathbf{C}$ . By definition, any language that is not in the topological complexity class  $\mathbf{C}$  is not definable in formalism  $\mathcal{F}$ . Since descriptive set theory is a well developed mathematical discipline the simple observation above gives a powerful tool to prove that a certain language is not definable in a given formalism – for some languages calculating their topological hardness is easier than providing a direct proof of undefinability.

The method was used by Mikołaj Bojańczyk to prove that nondeterministic max automata recognize strictly more languages than deterministic ones (see [Boj11, Theorem 24]). We will come back to this example in Section 1.5.3 below. This result has its consequences in separating weak and full second order logic with the unbounding quantifier, which will be discussed in Section 2.3.1.

### 1.5.2 Logics and Topological Complexity

Although, it was already implicitly used above, we want to mention explicitly the relation between construction of logics and their topological complexity. It is hard to say anything generic about the impact of predicates or relations

used in a logic on its topological complexity. On the other hand, there is a close and usually generic relationship between the quantifiers used and the topological complexity of a logic.

Let us think of trees or words as structures. Let us first consider first order quantifiers. We use a convention that variables in scope of such quantifiers are denoted with lowercase letters. If  $L(\varphi)$  denotes the language defined by a formula  $\varphi$ , and  $\varphi[\alpha]$  denotes formula  $\varphi$  with substitution  $\alpha$  applied, then:

$$L(\exists_x \varphi(x)) = \bigcup_v L(\varphi[x \mapsto v]) \quad (1.7)$$

$$L(\forall_x \varphi(x)) = \bigcap_v L(\varphi[x \mapsto v]) \quad (1.8)$$

Therefore, if structures have countable domains, like trees with countable branching or words of countable length, then:

$$\forall_v (L(\varphi[x \mapsto v]) \in \Sigma_\alpha^0) \implies L(\exists_x \varphi(x)) \in \Sigma_\alpha^0 \quad (1.9)$$

$$\forall_v (L(\varphi[x \mapsto v]) \in \Pi_\alpha^0) \implies L(\forall_x \varphi(x)) \in \Pi_\alpha^0 \quad (1.10)$$

and

$$\forall_v (L(\varphi[x \mapsto v]) \in \Sigma_i^1) \implies L(\exists_x \varphi(x)), L(\forall_x \varphi(x)) \in \Sigma_i^1 \quad (1.11)$$

$$\forall_v (L(\varphi[x \mapsto v]) \in \Pi_i^1) \implies L(\exists_x \varphi(x)), L(\forall_x \varphi(x)) \in \Pi_i^1 \quad (1.12)$$

If we restrict structures to  $\omega$ -words, we have:

$$\exists_n^\infty \varphi(n) = \forall_m \exists_{n>m} \varphi(n) \quad \text{and} \quad \forall_n^\infty \varphi(n) = \exists_m \forall_{n>m} \varphi(n)$$

Therefore, by (1.9) and (1.10), for words we immediately get<sup>7</sup>:

$$\forall_v (L(\varphi[x \mapsto v]) \in \Sigma_\alpha^0) \implies L(\exists_x^\infty \varphi(x)) \in \Pi_{\alpha+1}^0 \quad (1.13)$$

$$\forall_v (L(\varphi[x \mapsto v]) \in \Pi_\alpha^0) \implies L(\forall_x^\infty \varphi(x)) \in \Sigma_{\alpha+1}^0 \quad (1.14)$$

Let us now look at second order quantifiers, i.e. quantifiers ranging over subsets of the domain of a structure (with variables denoted with uppercase letters). If  $D$  is the domain of structures of interest, then:

$$L(\exists_X \varphi(X)) = \bigcup_{S \subseteq D} L(\varphi[X \mapsto S]) \quad (1.15)$$

Note that if  $D$  is infinite then the above is not a countable union. Therefore, in order to estimate the topological complexity of the left hand side language,

---

<sup>7</sup>The first of the equations was already used in the proof of Fact 1.2.7.(3).

we look at set variables as at bit vectors (i.e. characteristic functions). Let  $\mathbb{1}_X \in \{0, 1\}^D$  denotes the characteristic function of a subset  $X \subseteq D$ , and observe:

$$L(\exists_X \varphi(X)) = \pi_1(\{(t, \mathbb{1}_S) : t \in L(\varphi[X \mapsto S])\}) \quad (1.16)$$

Using this equation we obtain:

$$\{(t, \mathbb{1}_S) : t \in L(\varphi[X \mapsto S])\} \in \Sigma_i^1 \implies L(\exists_X \varphi(X)) \in \Sigma_i^1 \quad (1.17)$$

By the duality of quantifiers  $\forall$  and  $\exists$  and of classes  $\Sigma_i^1$  and  $\Pi_i^1$ , we immediately get:

$$\{(t, \mathbb{1}_S) : t \in L(\varphi[X \mapsto S])\} \in \Pi_i^1 \implies L(\forall_X \varphi(X)) \in \Pi_i^1 \quad (1.18)$$

It is also valuable to mention weak second order quantifiers, i.e. the ones that range over finite sets (denoted  $\exists^{fin}$  and  $\forall^{fin}$ ). Note that in this case the following union and intersection range over countable sets (in case of a countable domain).

$$L(\exists_X^{fin} \varphi(X)) = \bigcup_{\substack{S \subseteq D \\ S \text{ finite}}} L(\varphi[X \mapsto S]) \quad (1.19)$$

$$L(\forall_X^{fin} \varphi(X)) = \bigcap_{\substack{S \subseteq D \\ S \text{ finite}}} L(\varphi[X \mapsto S]) \quad (1.20)$$

Therefore, if structures have countable domains then:

$$\forall_S^{fin} (L(\varphi[X \mapsto S]) \in \Sigma_\alpha^0) \implies L(\exists_X^{fin} \varphi(X)) \in \Sigma_\alpha^0 \quad (1.21)$$

$$\forall_S^{fin} (L(\varphi[X \mapsto S]) \in \Pi_\alpha^0) \implies L(\forall_X^{fin} \varphi(X)) \in \Pi_\alpha^0 \quad (1.22)$$

and

$$\forall_S^{fin} (L(\varphi[X \mapsto S]) \in \Sigma_i^1) \implies L(\exists_X^{fin} \varphi(X)), L(\forall_X^{fin} \varphi(X)) \in \Sigma_i^1 \quad (1.23)$$

$$\forall_S^{fin} (L(\varphi[X \mapsto S]) \in \Pi_i^1) \implies L(\exists_X^{fin} \varphi(X)), L(\forall_X^{fin} \varphi(X)) \in \Pi_i^1 \quad (1.24)$$

### 1.5.3 Automata Models and Their Limitations

In this section, still talking about them on an abstract level, we classify automata classes with respect to the way a run is constructed, where a run is an intermediate notion that allows an automaton to decide acceptance of an input structure. We will see that this classification is very well aligned with the topological complexity of the classes of languages recognized by respective automata.

## Deterministic Automata

A complete deterministic automaton has exactly one run on each input. For all types of automata considered in this thesis (and for many others), the run is a labeling of an input structure with the states of the automaton, or more generally its configurations. Moreover, the run is constructed step by step, where the next step is fully determined by the so far constructed part of the run and by the so far seen input structure. Depending on the type of automaton there are other restrictions on the run construction for a deterministic automaton, but the ones enumerated above allow us to conclude that the construction of a run is the application of a continuous function that maps a (labeled) input structure to the structure of the same shape but with labels from possibly different alphabet. To give an intuition why the function is continuous we note that in the context of infinite words, each label of the run depends only on a finite prefix of an input word.

The **acceptance condition** of an automaton is a subset of the space of all possible runs. The automaton accepts an input if the unique run on the input is in this subset (we usually say that the run “satisfies the acceptance condition”). Therefore, the language recognized by the automaton is a continuous preimage of the acceptance condition. Since topological complexity classes are closed under continuous preimages, we get:

**Remark 1.5.1** *The topological complexity of the language recognized by a complete deterministic automaton is bounded from above by the topological complexity of the acceptance condition of the automaton.*

As a result, we get:

**Remark 1.5.2** *If  $\mathbf{A}$  is a class of complete deterministic automata and there is such a topological complexity class  $\mathbf{C}$  that each automaton in  $\mathbf{A}$  has the acceptance condition in  $\mathbf{C}$  (i.e. the acceptance condition has a bounded topological complexity) then  $\mathbf{C}$  is an upper topological complexity bound of  $\mathbf{A}$ .*

A deterministic automaton, in contrast to a complete deterministic automaton, has at most (not necessarily “exactly”) one run on each input. This is because a run of such an automaton may lock in some point. As a result, a mapping of input structures to runs, defined by such an automaton, is only a partial function. However, since in all contexts considered in this thesis each deterministic automaton can be easily extended to a complete one without a change in the recognized language, the expressive power of both classes of automata is the same. Therefore, the topological complexity bound mentioned in Remark 1.5.2 can be applied to (not necessarily complete) deterministic automata.

The method was used by Bojańczyk [Boj11] to show the separation of deterministic and nondeterministic max automata mentioned above. Namely, it was shown that the acceptance condition of a max automaton is always in  $\mathcal{BC}_2^0$ . Since a natural example of a language recognized by nondeterministic max automata is, in essence, the set  $D_3$  mentioned by Kechris [Kec95, Exercise 32.2] as  $\Pi_3^0$ -complete, it cannot be recognized by any deterministic max automaton.

## Nondeterministic Automata

A run of a nondeterministic automaton is, as in the case of a deterministic one, a labeling of an input structure with the alphabet of configurations of the automaton. The difference here is that there can be many possible runs on one input structure. The automaton accepts an input if there is a run that satisfies the acceptance condition (an **accepting run**). Let  $\mathcal{X}$  be the space of all structures compatible with the automaton,  $\mathcal{R}$  the space of all runs of the automaton, and  $\text{Acc} \subseteq \mathcal{R}$  the set of all accepting runs of the automaton. The language accepted by the automaton is the following projection:

$$\pi_1((\mathcal{X} \times \text{Acc}) \cap \{(s, \rho) \in \mathcal{X} \times \mathcal{R} : \rho \text{ is a run on } s\}) \quad (1.25)$$

The first factor of the intersection in the above formula has the same complexity as the acceptance condition. The second is usually a closed set—consistency of a run with a structure and the adjacent parts of the run needs to be verified locally in all positions of the structure. As a result we get:

**Remark 1.5.3** *The language recognized by a nondeterministic automaton belongs to the smallest  $\Sigma_n^1$  class that contains the acceptance condition.*

In particular, if the acceptance condition is Borel then we get  $\Sigma_1^1$  upper topological complexity bound from the remark.

The upper topological complexity bound that we get from the above remark for a class of nondeterministic automata is usually not satisfiable, therefore, we often need to find some other arguments. Examples can be found in Sections 2.5.1 and 2.5.2. However, for example, for nondeterministic parity automata on infinite trees the method gives quite a good and commonly used upper bound (see Section 3.2).

## Alternating Automata

For an alternating automaton, the acceptance of an input structure is defined in terms of a game played by an existential player  $\exists$  and a universal player  $\forall$ . The language recognized by the automaton consists of those input structures,

for which  $\exists$  has a winning strategy. In an abstract setting, each player has a set of strategies,  $\mathcal{S}_\exists$  and  $\mathcal{S}_\forall$ , respectively, depending only on the automaton. The sets of strategies, the set  $\mathcal{X}$  of all input structures compatible with the automaton, as well as the set  $\mathcal{P}$  of plays produced by players playing their strategies, are topological spaces. The game is defined by two relations and a function:

$$\begin{aligned}\text{Correct}_\exists &\subseteq \mathcal{X} \times \mathcal{S}_\exists \\ \text{Acc} &\subseteq \mathcal{P} \\ \text{Play} &: \mathcal{S}_\exists \times \mathcal{S}_\forall \rightarrow \mathcal{P}\end{aligned}$$

Here,  $\text{Correct}_\exists(x, s_\exists)$  expresses that the strategy  $s_\exists$  of  $\exists$  is compatible with an input structure  $x$ . This relation is usually defined in terms of local properties and has low, in particular Borel, topological complexity. Whereas  $\text{Acc}$  is the acceptance condition of an automaton, i.e. the set of plays won by  $\exists$ . Finally,  $\text{Play}$  is the continuous function that maps a pair of strategies to the unique play they determine.

Now the set of input structures accepted by an automaton can be presented by:

$$\underbrace{\underbrace{\{x \in \mathcal{X} : \exists_{s_\exists \in \mathcal{S}_\exists} (\text{Correct}_\exists(x, s_\exists) \wedge \underbrace{\forall_{s_\forall \in \mathcal{S}_\forall} (s_\exists, s_\forall) \in \text{Play}^{-1}(\text{Acc}))\}_{\substack{\text{as complex as Acc} \\ \text{smallest } \Pi_i^1 \text{ containing Acc}}}\}_{\Pi_i^1}}_{\Sigma_{i+1}^1}}$$

Therefore:

**Remark 1.5.4** *The topological complexity of the language recognized by an alternating automaton is bounded by the smallest  $\Sigma_{n+1}^1$  class such that  $\Pi_n^1$  contains the acceptance condition.*

In particular, if the acceptance condition is Borel then we get  $\Sigma_2^1$  upper topological complexity bound from the remark.

### Unambiguous Automata

An unambiguous automaton is a nondeterministic automaton that has at most one accepting run on each input. It might seem that the complexity of languages recognized by such automata is similar to the complexity of languages recognized by deterministic automata. It was proven by the author



of this thesis [Hum12] and by his coauthors [DFH15] that the complexities significantly differ in the context of infinite trees. The result of the first of the mentioned papers [Hum12] is extended in Chapter 3.

Let  $A \subseteq X \times Y$  be a set in a product space. A set  $U \subseteq A$  is a **uniformization** of  $A$  if for each  $x \in X$ ,  $\exists_y(x, y) \in A \iff \exists!_y(x, y) \in U$ , where  $\exists!$  stands for “there exists exactly one”. Let us identify a nondeterministic automaton  $\mathcal{A}$  with the set of pairs  $(x, \rho)$ , where  $x$  is a structure and  $\rho$  is an accepting run of  $\mathcal{A}$  on  $x$ , as used in formula (1.25). Under this identification there is some analogy between an unambiguous automaton recognizing the same language as a nondeterministic automaton, and a uniformization of a set. The analogy is not exact, though, because the unambiguous automaton is not necessarily a subset of the nondeterministic one in this setting. This difference makes the analogy hard to use in practice. The author is not aware of any generic way of using unambiguity to get an upper topological complexity bound, nor any such way was invented in this thesis. Chapter 3 describes only some lower topological complexity bound for languages defined by unambiguous automata.

## Nested Automata

Sometimes, in order to obtain decidability of some logic, models of nested automata are used. For example Mikołaj Bojańczyk and Szymon Toruńczyk [BT12] show a model of nested deterministic automata that has the same expressive power as Weak Monadic Second order Logic with the unbounding quantifier (WMSO+U) interpreted on infinite trees.

A run of a nested automaton is constructed in such a way that on every stage of its construction it can execute a run of a subautomaton on a substructure of the input structure. A result (acceptance or rejection) of this inner run can be used to determine an extension (a next step) of a run.

In general, nested automata can be simulated by alternating ones. Indeed, a possibility to execute an inner run, can be replaced by existential player declaring a result of an inner run, and universal player choosing whether the play will continue with this declared value, or will the players switch to play the inner game instead. If the declared value of an inner run is “reject”, then in the inner game the roles of the players swap.

The author is not aware of any case where the topological complexity bound for nested automata calculated using alternating automata is tight. For example, in the mentioned article by Bojańczyk and Toruńczyk [BT12] the construction of logic was used to obtain much better (Borel) bound. However, the method presented in this section gives some theoretical upper bound and we will use it in Section 2.4.

### 1.5.4 Undecidability

In this section we mention an important contribution of the topological complexity methods to the undecidability result.

It was shown by Michał Skrzypczak and the author of this thesis [HS12] that for each level of the projective hierarchy there exists a language hard for this level and definable in  $\text{MSO}+\text{U}$  logic (see Section 2.4). This result was used by Bojańczyk, Gogacz, Michlewski and Skrzypczak [BGMS14] in a forcing-based proof that undecidability of  $\text{MSO}+\text{U}$  logic is relatively consistent with ZFC axioms. We note that a year later a non-topological proof of undecidability of the logic was given by Bojańczyk, Parys and Toruńczyk [BPT16]. However, the topological results have given a direction towards a solution of a problem that was open for a decade<sup>8</sup>.

---

<sup>8</sup>The bounding quantifier was introduced by Bojańczyk in 2004 [Boj04].

# Chapter 2

## Infinite Words – Beyond Regularity

As we will see in Section 2.2, the languages of  $\omega$ -words that are called regular, are computationally and topologically not too complex. This fact encourages us to seek extensions of the class that would maintain at least some of their tractability. In this chapter we study how complex, from the topological point of view, are some of the extensions.

The chapter essentially presents results of a paper coauthored by the author of this thesis [HS12]. The presentation is preceded by a short introduction motivating the study.

### 2.1 Regular Languages of Infinite Words

There are several equivalent ways to say what it means for a language of  $\omega$ -words to be regular. In this thesis we choose the definition through an appropriate model of automata.

**Definition 2.1.1** *A (nondeterministic) parity word automaton is a tuple  $\mathcal{A} = \langle A, Q, \delta, I, p \rangle$ , where:*

- *$A$  is a finite alphabet,*
- *$Q$  is a finite set of states,*
- *$\delta \subseteq Q \times A \times Q$  is a transition relation,*
- *$I \subseteq Q$  is a set of initial states,*
- *$p : Q \rightarrow \mathbb{N}$  is a priority function.*

A **run** of automaton  $\mathcal{A}$  on an input word  $w \in A^\omega$  is a word  $\rho \in Q^\omega$  such that  $\rho_0 \in I$  and for all  $n < \omega$ ,  $(\rho_n, w_n, \rho_{n+1}) \in \delta$ . Function  $p$ , that assigns a **priority** to each state, is used to decide acceptance. Let us consider a sequence  $p(\rho) := p(\rho_0)p(\rho_1)p(\rho_2)\dots$ . Run  $\rho$  is **accepting** if the greatest priority occurring infinitely often in  $p(\rho)$  is even. This condition on the sequences of natural numbers is called **the parity condition**. A word is **accepted** by an automaton if there is an accepting run of the automaton on the word. The language **recognized** by automaton  $\mathcal{A}$ , denoted  $L(\mathcal{A})$ , consists of all words accepted by  $\mathcal{A}$ .

A language is called  **$\omega$ -regular** (or simply **regular**) if there is a parity automaton recognizing it.

There exist several interesting subclasses of the class of parity automata.

**Definition 2.1.2** A **Büchi** automaton (or automaton with Büchi acceptance condition) is a parity automaton  $\mathcal{A} = \langle A, Q, \delta, I, p \rangle$  such that  $p(Q) \subseteq \{1, 2\}$ . States of priority 2 are called **accepting states** in a Büchi automaton, and Büchi acceptance condition can be restated as “an accepting state occurs infinitely many times in a run”.

**Fact 2.1.3** (see e.g. [Far01, Theorem 1.19]) Each  $\omega$ -regular language can be recognized by a nondeterministic Büchi automaton.

**Definition 2.1.4** Parity word automaton  $\mathcal{A} = \langle A, Q, \delta, I, p \rangle$  is called **deterministic** if  $|I| = 1$  and  $\delta$  is a function  $\delta : (Q \times A) \rightarrow Q$ .

Note that, as it was noted in Section 1.5.3, a deterministic parity automaton has exactly one run on each input.

We say that two automata are **equivalent** if they recognize the same language.

**Theorem 2.1.5** (McNaughton [McN66]) For every nondeterministic parity word automaton there is an equivalent deterministic parity word automaton.

In the original proof deterministic automaton with Muller acceptance condition were used<sup>1</sup>. While Muller condition is a priori more general, it can be proven that deterministic parity automata have the same expressive

---

<sup>1</sup>In his paper McNaughton uses term “regular  $\omega$ -event” instead of  $\omega$ -regular language and the term is defined using  $\omega$ -regular expressions. The proof of the equivalence of the notions can be found e.g. in Farwer’s handbook article [Far01, Theorem 1.5]. Languages recognized by Muller automata are, in turn, called “finite-state  $\omega$ -events” in the paper.

power as deterministic Muller automata. A proof using latest appearance record (LAR) construction is presented e.g. by Farwer [Far01, Section 1.4.2].

Contemporary textbooks usually present the proof of McNaughton's Theorem given by Safra [Saf88]. The proof is a refinement of the powerset construction used in determinization of finite automata on finite words, keeping track of particular runs. See handbook articles by Roggenbach [Rog01] or Thomas [Tho97] for a comprehensive presentation of Safra's construction.

The original use of finite automata on infinite words, namely Büchi automata, was to prove decidability of Monadic Second Order (MSO) logic on  $\omega$  with the successor relation (called S1S theory) [Büc62]. As a by-product Büchi has shown that MSO logic over infinite words defines exactly regular languages. Let us introduce the logic and state the equivalence.

With a word  $u \in A^\omega$  over an alphabet  $A = \{a_0, a_1, \dots, a_n\}$ , we associate the relational structure, called **word model**:

$$\underline{u} = (\text{dom}(u), S^\omega, \leq^\omega, \mathbf{a}_0^u, \mathbf{a}_1^u, \dots, \mathbf{a}_n^u),$$

where the domain  $\text{dom}(u) = \omega$  can be identified with the set of positions in the word,  $S^\omega$  is the successor relation, i.e.  $(i, i+1) \in S^\omega$  for  $i < \omega$ ,  $\leq^\omega$  is the natural order on word positions, and  $\mathbf{a}_0^u, \mathbf{a}_1^u, \dots, \mathbf{a}_n^u$  are letter predicates, i.e.  $\mathbf{a}_i^u(k)$  holds if and only if  $u_k = a_i$ . Note that, as mentioned before, for each  $k \in \text{dom}(u)$ ,  $k$  belongs to exactly one of the sets  $\mathbf{a}_0^u, \mathbf{a}_1^u, \dots, \mathbf{a}_n^u$ .

Monadic Second Order logic (MSO for short) on  $\omega$ -words over alphabet  $A$  uses:

- first-order (individual) variables  $x, y, z, \dots$  ranging over positions in word models,
- monadic second-order (set) variables  $X, Y, Z, \dots$  ranging over sets of positions in word models,
- $\forall, \exists$  quantifiers that can bind both, first- and second-order variables,
- atomic formulas (in the following,  $x, y$  are individual variables, and  $X$  is a set variable):  $x = y, x \in X, S(x, y), x \leq y, \mathbf{a}(x)$  for  $a \in A$ , where the last three are interpreted as the respective relations in word models as presented above,
- logical connectives:  $\wedge, \vee, \neg, \implies, \iff$ .

Let us note by  $\text{Sig}_A^\omega$  the signature of this logic on  $\omega$ -words over alphabet  $A$ . Namely, for  $A = \{a_0, a_1, \dots, a_n\}$ ,  $\text{Sig}_A^\omega = \{S, \leq, \mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_n\}$ . Definability notions are defined as presented in Section 1.3. Therefore, for na MSO

formula  $\varphi$  over signature  $\text{Sig}_A^\omega$ ,  $L(\varphi)$  is the language of such words  $u$  over  $A$  that  $\varphi$  is satisfied in  $\underline{u}$ .

From the expressive power perspective and in the context of Monadic Second Order logic, only one of the relations  $S^\omega$ ,  $\leq^\omega$  suffices, since they are mutually expressible by each other in the logic (see [Tho97, Section 2.3] for details). We assume the signature includes both for further convenience. We also note that if  $\omega$  is fixed as the domain, then 0 is clearly definable using  $S^\omega$  (even in first order logic). Similarly, each natural number is definable using  $S^\omega$ . Therefore, although we do not put functional symbols explicitly into the signature, we will use natural numbers as constants in the logic.

**Theorem 2.1.6 (Büchi [Büc62])** *Languages of  $\omega$ -words defined by MSO are exactly  $\omega$ -regular languages.*

Although the original proof of Büchi have not used determinization of automata, the theorem may be seen as a consequence of McNaughton's Theorem. To prove that a model of automata captures a logic that uses  $\neg$  connective, we need to be able to construct an automaton for the complement of the language recognized by a given automaton. Models of deterministic automata are convenient for this purpose. Essentially, to complement a deterministic automaton we only need to modify its acceptance parameter. In the case of the parity condition, shifting values of the priority function by 1 suffices.

Another feature of a logic to be captured is quantification. Büchi (following what he did for finite words [Büc60]) does it using nondeterminism. Consecutive positions of quantified set are guessed by nondeterministic automaton.

We summarize the above two observations in the following note:

**Remark 2.1.7** *In order to capture a logic using negation and existential quantification, it is convenient to have a model of nondeterministic automata that admits determinization.*

**Definition 2.1.8** *Weak Monadic Second Order logic (WMSO for short) uses the same syntax as MSO, but the second order quantification is restricted to finite sets.*

Since in MSO on  $\omega$ -words finiteness is definable using order, it is not hard to see that each language definable in WMSO can also be defined in MSO. The converse also holds for  $\omega$ -words and the proof can use Büchi's Theorem and determinization.

**Theorem 2.1.9** (see e.g. [Tho97, Corollary 5.2]) *Logics MSO and WMSO are equally expressive on  $\omega$ -words.*

For a presentation of automata on  $\omega$ -words and their relations to logics see a survey article by Wolfgang Thomas [Tho97] or a handbook [GTW02].

## 2.2 Complexity of Regular Languages of Infinite Words

Since, thanks to Theorem 2.1.5, the class of  $\omega$ -regular languages can be characterized using deterministic automata, we can use Remark 1.5.2 to estimate the topological complexity of the class. We only need to calculate topological complexity of the acceptance condition.

Let  $Q$  be the set of states of an automaton, and let  $p$  be its priority function.

Let us consider the following subset of  $Q^\omega$ :

$$O_k := \{\rho \in Q^\omega : \exists_n^\infty p(\rho_n) = k\}$$

Note that  $O_k = f_k^{-1}(N_2)$  for  $f_k : Q^\omega \rightarrow \{0, 1\}^\omega$  defined by:

$$(f_k(\rho))_n = \begin{cases} 1 & \text{if } p(\rho_n) = k \\ 0 & \text{otherwise} \end{cases},$$

and  $N_2$  defined as in Fact 1.2.7. Therefore,  $O_k$  is in class  $\mathbf{\Pi}_2^0$ .

Since  $Q$  is finite, there is such even number  $n$  that for each  $q \in Q$ ,  $p(q) \leq n$ . In such case the parity condition can be expressed as the following subset of  $Q^\omega$ :

$$Acc_n = ((\dots (((O_0 \setminus O_1) \cup O_2) \setminus O_3) \cup O_4) \dots) \setminus O_{n-1}) \cup O_n$$

Therefore, for each parity word automaton, the acceptance condition is a finite boolean combination of  $\mathbf{\Pi}_2^0$  sets. Using Remark 1.5.2 we conclude:

**Theorem 2.2.1 ([BL69, Corollary 1])** *Each  $\omega$ -regular language is in  $\mathcal{BC}_2^0$  topological complexity class.*

Let us add several notes suggesting that  $\omega$ -regular languages are relatively simple from the computational complexity point of view<sup>2</sup>. The most popular decision problem considered for automata is emptiness.

---

<sup>2</sup>We do not recall computational complexity notions here, because the computational complexity references are only side notes for the thesis.

**Emptiness Problem:**

Given an automaton,  
decide whether the language recognized by the automaton is empty.

**Fact 2.2.2** (see e.g. [PP04, Propositions 10.12, 10.10])

1. *The emptiness problem for Büchi automata is decidable in linear time with respect to the number of states. It is NL-complete.*
2. *Let  $\mathcal{A}$  be a Büchi automaton, and let  $n$  be a number of its states. There is a Büchi automaton with  $n^{\mathcal{O}(n)}$  states recognizing the complement of language  $L(\mathcal{A})$ .*

Before we enumerate conclusions of the fact, we recall two more famous decision problems.

**Universality Problem:**

Given an automaton,  
decide whether the automaton accepts each input.

**Language Inclusion Problem:**

Given automata  $\mathcal{A}$  and  $\mathcal{B}$ ,  
decide whether  $L(\mathcal{A}) \subseteq L(\mathcal{B})$ .

The following can be viewed as a consequence of Fact 2.2.2.

**Fact 2.2.3** (see e.g. [PP04, Proposition 10.13]) *The universality problem for Büchi automata is decidable in exponential time. It is PSPACE-complete.*

The following mainly relies on the fact that verifying inclusion is equivalent to verifying emptiness of the language  $L(\mathcal{A}) \cap \overline{L(\mathcal{B})}$ .

**Fact 2.2.4** ([KV98, Theorem 1]) *The language inclusion problem for Büchi automata is decidable in exponential time.*

Since a conversion of a parity automaton into a Büchi automaton causes polynomial increase in the number of states (see [PP04, Proposition 10.6]<sup>3</sup>), we immediately get the following.

**Corollary 2.2.5**


---

<sup>3</sup>In fact, a conversion of a Rabin automaton into a Büchi automaton is shown there, but a parity automaton is just a special case of a Rabin automaton.



1. *The emptiness problem is decidable for parity word automata in polynomial time with respect to the number of states.*
2. *The universality problem for parity word automata is decidable in exponential time.*
3. *The language inclusion problem for parity word automata is decidable in exponential time.*

Also logics are studied from the computational complexity perspective. When talking about **decidability of a logic** we mean the following decision problem.

**Satisfiability Problem for logic  $\mathcal{L}$  over words:**

Given a formula  $\varphi$  of  $\mathcal{L}$ ,  
decide whether there is a word model satisfying this formula.

Thanks to Büchi’s result cited as Theorem 2.1.6, the satisfiability problem for MSO on  $\omega$ -words can be reduced to emptiness problem for automata, therefore, is decidable. The decision procedure coming from this construction has nonelementary time complexity – alternating between existential and universal quantification requires determinisation of an automaton which may cause exponential blowup in the number of states – however, the situation is not worse than in the case of finite words.

The decidability of MSO, linear complexity of emptiness problem for Büchi automata, and Borel topological complexity of  $\omega$ -regular languages are what we have referred to as “computationally and topologically not too complex” in the introduction to this chapter. In the next section we present a logic that was considered as a potential decidable extension of MSO. Unfortunately, as we know now, the logic is not decidable. We show a topological complexity result for the logic that have contributed to an undecidability proof. In the following sections we consider, from the topological point of view, automata models that extend parity automata<sup>4</sup>, have decidable emptiness, and implement a similar idea that the extended logic.

## 2.3 Unbounding Quantifier

Mikołaj Bojańczyk has proposed an extension of the class of  $\omega$ -regular languages which is able to express some asymptotic properties of words. The

---

<sup>4</sup>The  $\omega$ B-automata and  $\omega$ S-automata are not exactly syntactic extensions of parity automata, but it will be clear that the encoding of a parity automaton as a  $\omega$ B-automaton or a  $\omega$ S-automaton is straightforward.

extension was first introduced to tree languages [Boj04], and then studied both for  $\omega$ -words (see e.g. [BC06, Boj11, Boj10]) and trees (see e.g. [BT12]).

The extension was designed to be able to define properties as captured by the following  $\omega$ -word languages:

$$L_B = \{a^{n_0}ba^{n_1}ba^{n_2}b\ldots \mid \limsup n_i < \infty\} \quad (2.1)$$

$$L_S = \{a^{n_0}ba^{n_1}ba^{n_2}b\ldots \mid \liminf n_i = \infty\} \quad (2.2)$$

First observe that these languages are not regular.

**Fact 2.3.1 ([BC06])** *Language  $L_B \subseteq \{a, b\}^\omega$  is not  $\omega$ -regular.*

**Proof:**

Assume, towards a contradiction, that there is a Büchi automaton  $\mathcal{A} = \langle \{a, b\}, Q, \delta, I, p \rangle$  that recognizes language  $L_B$ . Consider a word:

$$w = a^{|Q|+1}ba^{|Q|+1}b\ldots$$

Since  $w \in L_B$ , there is an accepting run  $\rho$  of automaton  $\mathcal{A}$  on  $w$ . Since each block of consecutive letters  $a$  surrounded by  $b$ 's has length greater than the number of states of  $\mathcal{A}$ , there is a loop (i.e. some state occurs at least twice) in  $\rho$  inside each of those blocks. We now use the fact that each such loop can be repeated arbitrary (positive) number of times without affecting acceptance. The process is often used in automata theory and is called **pumping**. Let for  $i$ 'th block of  $a$ 's,  $s_i$  be the position of the first and  $e_i$  of the next occurrence of the same state inside the block. Now consider the following word:

$$w' = a^{|Q|+1+1(e_1-s_1)}ba^{|Q|+1+2(e_2-s_2)}ba^{|Q|+1+3(e_3-s_3)}b\ldots$$

and the following run on it:

$$\rho' = \rho_{1,s_1-1}(\rho_{s_1,e_1})^2\rho_{e_1+1,s_2-1}(\rho_{s_2,e_2})^3\rho_{e_2+1,s_3-1}(\rho_{s_3,e_3})^4\rho_{e_3+1,s_4-1}\ldots,$$

where  $\rho_{i,j}$  denotes sequence of consecutive elements of sequence  $\rho$  from position  $i$  to  $j$  inclusive.

Since  $(e_i - s_i)$  is positive for each  $i \geq 1$ , the sequence of lengths of blocks of  $a$ 's in  $w'$  tends to infinity, therefore,  $w' \notin L_B$ . On the other hand  $\rho'$  has exactly the same set of states that occur infinitely many times that run  $\rho$ , hence  $\rho'$  is accepting. This is a contradiction with the assumption that  $\mathcal{A}$  recognizes  $L_B$ .  $\square$

**Fact 2.3.2 ([BC06])** *Language  $L_S \subseteq \{a, b\}^\omega$  is not  $\omega$ -regular.*

To prove this fact we use a topological argument. The following fact is given as an exercise in Kechris's book. We prove it for completeness.

**Fact 2.3.3** (see e.g. [Kec95, Exercise 23.2])

Set  $C_3 = \{x \in \mathbb{N}^\omega : \lim_n x_n = \infty\}$  is  $\Pi_3^0$ -hard<sup>5</sup>.

**Proof:**

Let  $L \in \Pi_3^0(X)$ , for a zero-dimensional Polish space  $X$ . By the definition,  $L = \bigcap_{n < \omega} L_n$ , for some sequence  $\{L_n\}_{n < \omega}$  of  $\Sigma_2^0$  sets. By Fact 1.2.7, Point 4, there exists a sequence  $\{f_n\}_{n < \omega}$  of continuous functions  $f_n : X \rightarrow \mathcal{C}$ , where  $f_n$  reduces  $L_n$  to  $Q_2$ . Let us now fix a bijection  $\iota : \omega \times \omega \rightarrow \omega$ . The function  $f : X \rightarrow \mathbb{N}^\omega$  defined by:

$$(f(x))_{\iota(n,m)} = \begin{cases} n & \text{if } (f_n(x))_m = 1 \\ \iota(n,m) & \text{otherwise} \end{cases}$$

is continuous and reduces  $L$  to  $C_3$ . Indeed, by the definition of set  $C_3$ ,  $f(x) \notin C_3$  if and only if some value occurs infinitely many times in sequence  $f(x)$ . By the definition of  $f$ , this happens if and only if for some  $n$  there is infinitely many 1's in sequence  $f_n(x)$ , what, by the assumption on  $f_n$ , holds if and only if  $x \notin L_n$ .  $\square$

Let us introduce continuous embedding  $g : \mathbb{N}^\omega \rightarrow \{a,b\}^\omega$  of the Baire space into the Cantor space  $\{a,b\}^\omega$ , defined by:

$$g(x) = a^{x_0} b a^{x_1} b a^{x_2} \dots \quad (2.3)$$

**Lemma 2.3.4** *Language  $C_3$  is Wadge-reducible to  $L_S$ . As a result,  $L_S$  is  $\Pi_3^0$ -hard.*

**Proof:**

Function  $g$  reduces  $C_3$  to  $L_S$ .  $\square$

**Proof (of Fact 2.3.2):**

By Theorem 2.2.1 and Lemma 2.3.4, language  $L_S$  cannot be  $\omega$ -regular.

$\square$

We note that a similar topological argument could not be used for the proof of Fact 2.3.1. This is because, it can be proven that  $L_B \in \Sigma_2^0$ .

As we can see, being able to define languages  $L_S$  and  $L_B$  requires an extension of  $\omega$ -regularity. On the logical side, the extension was implemented

---

<sup>5</sup>In fact,  $C_3$  is  $\Pi_3^0$ -complete, but we do not need it here. Additionally the fact that  $C_3 \in \Pi_3^0$  will follow from Lemma 2.3.4, Example 2.5.6 and Corollary 2.5.12.

by introduction of an additional set quantifier  $\mathbf{U}$ , called the **unbounding quantifier**<sup>6</sup>. The quantifier is defined so that the formula  $\mathbf{U}_X\varphi(X)$  is equivalent to writing:

*“ $\varphi(X)$  is satisfied by finite sets  $X$  of arbitrarily large cardinality”*

More formally:

$$\mathbf{U}_X\varphi(X) := \bigwedge_{n < \omega} \exists_X (\varphi(X) \wedge n < |X| < \omega) \quad (2.4)$$

Towards a usage example, consider the following formula with one free set variable:

$$\text{Block}_a(X) := \forall_{x \in X} \mathbf{a}(x) \wedge \forall_{x < y < z} (x \in X \wedge z \in X) \implies y \in X$$

The formula says that  $X$  is a set of consecutive positions labeled with  $a$ . Now language  $L_B$  can be expressed by the formula:

$$\neg \mathbf{U}_X \text{Block}_a(X) \quad (2.5)$$

Adding this formula defining infinity:

$$\text{Inf}(X) := \forall_x \exists_{y > x} y \in X,$$

and the following formula defining maximality of blocks of consecutive letters  $a$  in a set:

$$\text{Max}_a(X) := \forall_x \forall_y \left( \begin{array}{l} (x \in X \wedge S(x, y) \wedge \mathbf{a}(y)) \implies y \in X \\ \wedge \\ (x \in X \wedge S(y, x) \wedge \mathbf{a}(y)) \implies y \in X \end{array} \right), \quad (2.6)$$

we can define language  $L_S$  using formula:

$$\neg \exists_X (\text{Inf}(X) \wedge \forall_{x \in X} \mathbf{a}(x) \wedge \text{Max}_a(X) \wedge \neg \mathbf{U}_B (\text{Block}_a(B) \wedge \forall_{x \in B} x \in X)) \quad (2.7)$$

Recall, that by Theorem 2.1.9, there are two logics that characterize  $\omega$ -regular languages, namely MSO and WMSO. Both of them can be extended by a use of quantifier  $\mathbf{U}$ .

**Definition 2.3.5** *Monadic Second Order logic with Unbounding Quantifier* (MSO+U for short) is MSO logic where additionally quantifier  $\mathbf{U}$  is used.

---

<sup>6</sup>Originally **bounding quantifier** was introduced [Boj04], but in later works its dual, **unbounding quantifier**, was considered.

**Definition 2.3.6** *Weak Monadic Second Order logic with Unbounding Quantifier* (WMSO+U for short) is WMSO logic where additionally quantifier  $U$  is used.

Note that formula (2.5) defining  $L_B$  does not use set quantification except from quantifier  $U$ , so, in particular, it is a formula of WMSO+U. On the other hand, language  $L_S$  is defined by MSO+U formula 2.7 and, as we will see in Section 2.3.1, it cannot be defined in WMSO+U. Adding to the above observation about  $L_B$ , the fact that this language is not definable in MSO (see Fact 2.3.1), we get that extensions of both the logics are strict:

**Fact 2.3.7** *On  $\omega$ -words:*

1. *expressive power of MSO+U is strictly greater than the expressive power of MSO,*
2. *expressive power of WMSO+U is strictly greater than the expressive power of WMSO (and of MSO).*

In the following sections we discuss how big, from the topological viewpoint, are the extensions. Section 2.3.1 presents a result of Mikołaj Bojańczyk and serves only as a context for the thesis.

### 2.3.1 Topological Complexity of WMSO+U

Mikołaj Bojańczyk [Boj11] has proven decidability of WMSO+U logic on  $\omega$ -words by providing a model of automata, called **max automata**, with the same expressive power. Effective translations between formulas and automata were also given in his paper. What is important from our perspective, automata of this model are deterministic. As it was already mentioned in Section 1.5.3, the acceptance condition of a max automaton is always in  $\mathcal{BC}_2^0$ . Therefore, using Remark 1.5.2, all languages defined in WMSO+U logic are in  $\mathcal{BC}_2^0$ .

Additionally, Cabessa, Duparc, Facchini and Murlak [CDFM09] have shown that languages defined by WMSO+U occupy exactly the same levels of the Wadge hierarchy as  $\omega$ -regular languages.

Note that, since language  $L_S$  is definable using an MSO+U formula (2.7) and is  $\Pi_3^0$ -hard, we get:

**Corollary 2.3.8** *MSO+U has greater expressive power than WMSO+U on  $\omega$ -words.*

To emphasize the contrast between WMSO+U and the full MSO+U, we mention that WMSO+U is decidable also on infinite trees [BT12].

## 2.4 Topological Complexity of MSO+U

In this section we inductively construct a sequence of languages  $\{H_i\}_{i < \omega}$ . We show that for each  $i > 0$  the language  $H_i$  is MSO+U definable and  $\Sigma_i^1$ -hard. Therefore, we prove the following theorem.

**Theorem 2.4.1** *For every  $i > 0$  there exists an MSO+U formula  $\varphi_i$  such that the language  $L(\varphi_i)$  is  $\Sigma_i^1$ -hard.*

In our construction we use a sequence  $\text{IF}^i$  of languages of “multi-branching” trees (i.e. trees on  $\mathbb{N}^i$ ). First, we prove that for each  $i$  the language  $\text{IF}^i$  is  $\Sigma_i^1$ -hard. Then we inductively show that the languages can be reduced to  $\omega$ -word languages  $H_i$  definable in MSO+U. We use a function  $r_{i-1}$  reducing  $\text{IF}^{i-1}$  to  $H_{i-1}$  to construct a reduction  $c_i$  of  $\text{IF}^i$  to the language  $\text{EPath}(\overline{H_{i-1}})$  of labeled trees on  $\mathbb{N}$  that have a branch labeled with a word  $w \notin H_{i-1}$ . Then we again code such labeled trees in  $\omega$ -words. The details follow.

Let us fix a finite alphabet  $B_0 := \{a, |_0, b\}$  and define inductively  $B_i := B_{i-1} \cup \{[_{i-1}, |_i, ]_{i-1}\}$ , for  $i > 0$ .

Recall that  $T^X$  is the set of all unlabeled  $X$ -branching trees,  $T_B^X$  is the set of all full  $X$ -branching  $B$ -labeled trees. By  $B^+$  we denote the set of all nonempty words over  $B$ .

The inductive construction begins from step  $i = 1$  and in each step the picture looks as follows:

$$\begin{array}{ccccc} T^{\mathbb{N}^i} & \xrightarrow{c_i} & T_{B_{i-1}^+}^{\mathbb{N}} & \xrightarrow{d_i} & (B_i^+)^{\omega} \\ \cup & & \cup & & \cup \\ \text{IF}^i & & \text{EPath}(\overline{H_{i-1}}) & & H_i \end{array}$$

The construction ensures that:

$$d_i^{-1}(H_i) = \text{EPath}(\overline{H_{i-1}}), \quad c_i^{-1}(\text{EPath}(\overline{H_{i-1}})) = \text{IF}^i, \quad r_i = d_i \circ c_i.$$

The formal definitions and proofs of properties of the elements of the above diagram are the subject of following sections.

### Trees

Fix an order  $\sqsubseteq$  of type  $\omega$  on  $\mathbb{N}^*$ , such that  $\mathbb{N}^* = \{v_0, v_1, \dots\}$  and that for all  $n < \omega$  we have  $|v_n| \leq n$ . For example any order consistent with the prefix order on  $\mathbb{N}^*$  has this property.

**Definition 2.4.2** Consider  $i > 0$ , a tree  $t \in T^{\mathbb{N}^{i+1}}$  and a word  $w \in \mathbb{N}^{\leq \omega}$ . We define the **section**  $t|_w \in T^{\mathbb{N}^i}$  of the tree  $t$  as follows

$$t|_w = \{w' \in (\mathbb{N}^i)^* : |w'| \leq |w| \wedge (w|_{|w'|} \times w') \in t\},$$

where

$$(w_0, w_1, w_2, \dots) \times (w'_0, w'_1, w'_2, \dots) = (w_0 \cdot w'_0, w_1 \cdot w'_1, w_2 \cdot w'_2, \dots).$$

The dots in the above definition can stand for a finite or an infinite sequence.

Figure 2.1 presents an example tree  $t$  on  $\mathbb{N}^2$ , i.e.  $t \in T^{\mathbb{N}^2}$ , together with example sections of this tree along two words of length 1.

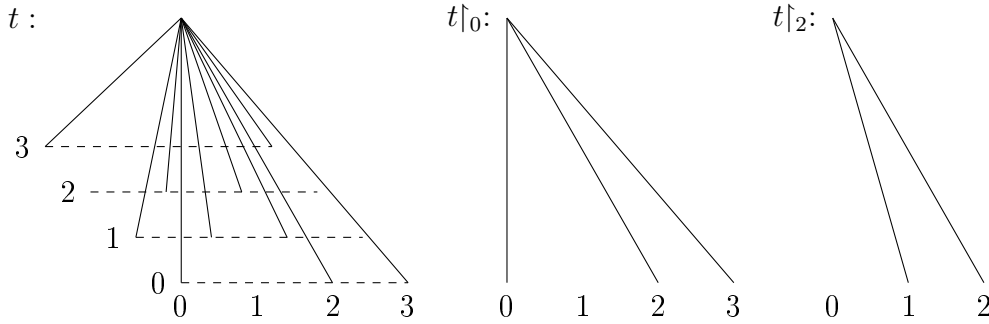


Figure 2.1: An example tree on  $\mathbb{N}^2$ , and its sections.

Observe that if  $w$  is a finite word,  $t|_w$  is a finite-depth tree—its depth is bounded by  $|w|$ .

**Definition 2.4.3** We define inductively  $\text{IF}^i \subseteq T^{\mathbb{N}^i}$ .

Let  $\text{IF}^1$  be the set of all trees  $t \in T^{\mathbb{N}^1}$  that contain an infinite branch.

Take  $i > 0$ . Let  $\text{IF}^{i+1}$  be a set of all trees  $t \in T^{\mathbb{N}^{i+1}}$  such that there exists an infinite word  $\alpha \in \mathbb{N}^\omega$  such that

$$t|_\alpha \notin \text{IF}^i.$$

The following is proven e.g. in the textbook by Guzikki and Zbierski [GZ78, Theorem 5.1], but we prove it here for the sake of completeness.

**Fact 2.4.4** For each  $i \geq 1$  the set  $\text{IF}^i$  is a  $\Sigma_i^1$ -complete subset of  $T^{\mathbb{N}^i}$ .

**Proof:**

**(Upper bound)** We proceed by induction.  $\text{IF}^1$  is a set of Ill-Founded trees (usually denoted by  $IF$ ), that is well known to be  $\Sigma_1^1$ -complete (see e.g. [Kec95, Theorem 27.1]). For the inductive step, assume that  $\text{IF}^i \in \Sigma_i^1$ . Then the set

$$P_i = \left\{ (\alpha, t) \in \mathbb{N}^\omega \times T^{\mathbb{N}^{i+1}} : t|_\alpha \notin \text{IF}^i \right\} \in \Pi_i^1,$$

because function  $p : (\alpha, t) \mapsto t|_\alpha$  is continuous, and  $P_i$  is an inverse image of  $\overline{\text{IF}^i}$  with respect to  $p$ . Note that  $\text{IF}^{i+1}$  is a projection of  $P_i$ , so it is in  $\Sigma_{i+1}^1$ .

**(Hardness)** It is enough to show that each  $\Sigma_i^1$  set in  $\mathbb{N}^\omega$  is continuously reducible to  $\text{IF}^i$ .

As we know (see e.g. [Sri98, Proposition 4.1.1.(iv)]), each analytic ( $\Sigma_1^1$ ) set in a space  $X$  is a projection of a closed set in  $\mathbb{N}^\omega \times X$ . Recall that, by definition, each  $\Sigma_{i+1}^1$  set is a projection of some  $\Pi_i^1$  set. Therefore, each  $\Sigma_i^1$  set in  $\mathbb{N}^\omega$  is of the form<sup>7</sup>:

$$S = \{x : \exists_{x_1 \in \mathbb{N}^\omega} \neg \exists_{x_2 \in \mathbb{N}^\omega} \neg \exists_{x_3 \in \mathbb{N}^\omega} \dots \neg \exists_{x_i \in \mathbb{N}^\omega} (x_1, x_2, \dots, x_i, x) \in F_S\},$$

for some closed set  $F_S \in (\mathbb{N}^\omega)^{i+1}$ . Since every second  $\exists$  quantifier is under even number of negations, the above formula unravels to:

$$\begin{aligned} \exists_{x_1} \forall_{x_2} \exists_{x_3} \dots \exists_{x_i} (x_1, x_2, \dots, x_i, x) \in F_S & \quad \text{if } i \text{ is odd, and to:} \\ \exists_{x_1} \forall_{x_2} \exists_{x_3} \dots \forall_{x_i} (x_1, x_2, \dots, x_i, x) \notin F_S & \quad \text{if } i \text{ is even.} \end{aligned}$$

The set  $F_S$  can be seen as a set in the space  $(\mathbb{N}^{i+1})^\omega$ , by simple transposition. This space is obviously homeomorphic to the Baire space  $\mathbb{N}^\omega$ . Each closed set in the Baire space can be expressed as the set of branches of some tree (see e.g. [Kec95, Proposition 2.4]). So there is  $t_S \in T^{\mathbb{N}^{i+1}}$  such that:

$$F_S = \{x \in (\mathbb{N}^{i+1})^\omega : \forall_{n < \omega} x|_n \in t_S\} \quad (2.8)$$

To simplify the notation, for a tree  $t$  on set  $X$ , by  $[t] \subseteq X^\omega$  we denote the set of infinite branches of  $t$ . Using this notation, the above equation can be formulated as

$$F_S = [t_S].$$

We will use the tree  $t_S$  to define the needed reduction. Let  $f : \mathbb{N}^\omega \rightarrow T^{\mathbb{N}^i}$  be defined as follows:

$$f(x) = \left\{ v \in (\mathbb{N}^i)^k : (v \times x|_k) \in t_S, k < \omega \right\}$$

---

<sup>7</sup>Formally, for  $i = 1$  the formula takes the form  $S = \{x : \exists_{x_1 \in \mathbb{N}^\omega} (x_1, x) \in F_S\}$ .



To determine whether a node at some level  $k$  belongs to  $f(x)$  we only need to know the first  $k$  numbers in the sequence  $x$ , so the function is continuous. To prove that this is a reduction of  $S$  to  $\text{IF}^i$  we need:

$$f(x) \in \text{IF}^i \iff x \in S \quad (2.9)$$

Now we will take a closer look at the sets  $\text{IF}^i$ . Observe that:

$$\begin{aligned} \text{IF}^i &= \{t : \exists_{x_1} \forall_{x_2} \exists_{x_3} \dots \exists_{x_i} (x_1 \times x_2 \times \dots \times x_i) \in [t]\} && \text{if } i \text{ is odd, and:} \\ \text{IF}^i &= \{t : \exists_{x_1} \forall_{x_2} \exists_{x_3} \dots \forall_{x_i} (x_1 \times x_2 \times \dots \times x_i) \notin [t]\} && \text{if } i \text{ is even.} \end{aligned}$$

So the quantifier structure is the same as in case of the above representation of  $S$ . Therefore, to obtain (2.9), it **suffices** to show that for any fixed  $x_1, x_2, \dots, x_i$ :

$$(x_1 \times x_2 \times \dots \times x_i) \in [f(x)] \iff (x_1, x_2, \dots, x_i, x) \in F_S.$$

By (2.8) it is equivalent to:

$$(x_1 \times x_2 \times \dots \times x_i) \in [f(x)] \iff (x_1 \times x_2 \times \dots \times x_i \times x) \in [t_S].$$

But the latter follows immediately from the definition of  $f$ .  $\square$

### Functions $c_i, d_i$

In this section we define functions  $c_i, d_i$ . The idea is that both these functions are continuous and  $1-1$ . Their task is to present a tree  $t \in T^{\mathbb{N}^i}$  as an infinite word in such a way that an  $\text{MSO}+\text{U}$  formula can determine whether  $t \in \text{IF}^i$  or not.

Recall our inductively defined alphabets  $B_0 = \{a, |_0, b\}$ ,  $B_i = B_{i-1} \cup \{[_{i-1}, |_i, ]_{i-1}\}$  and define:

**Definition 2.4.5** For a node  $u = (u_1, u_2, \dots, u_m) \in \mathbb{N}^*$  of a tree, we call the word  $a^{u_1}ba^{u_2}b \dots ba^{u_m}b$  the **address** of  $u$  in the tree.

Let an  **$i$ -block** be a word of the form  $[_i w |_i w']_i$  where  $w \in \{a, b\}^*$  and  $w' \in (B_i \setminus \{|_i\})^+$ . We call the word  $w$  the **address** of this  $i$ -block (since it will be interpreted as an address of a node in a tree) and the word  $w'$  the **body** of this  $i$ -block.

### Functions $d_i$

Take any  $i > 0$ . We encode a tree  $t \in T^{\mathbb{N}}_{B_{i-1}^+}$  into a word  $d_i(t) \in (B_i^+)^{\omega}$  in the following way. Take a tree  $t \in T^{\mathbb{N}}_{B_{i-1}^+}$  and a node  $v_n \in \mathbb{N}^*$ , i.e the

$n$ 'th node with respect to the order  $\sqsubseteq$ . Let  $v_n = (u_1, u_2, \dots, u_m)$  and let  $w_0, w_1, \dots, w_m \in B_{i-1}^+$  be the list of labels of  $t$  on the path from the root to  $v_n$ . Then:

$$d_i(t)_n := a^{u_1} b a^{u_2} b \dots b a^{u_m} b |_i [_{i-1} w_0]_{i-1} \cdot [_{i-1} w_1]_{i-1} \cdot \dots \cdot [_{i-1} w_m]_{i-1} \in B_i^+$$

Intuitively,  $d_i(t)_n$  encodes the node  $v_n$  in  $t$ . Such an encoding consists of two parts: the part before  $|_i$  is the address of  $v_n$  in the tree, while the part after  $|_i$  is intended to store labels in  $t$  on the path from the root to  $v_n$  as  $(i-1)$ -blocks. The fact that we store not only the label but also the address of the given node in this coding will be crucial for the following parts of the construction.

**Functions  $r_i, c_i$**

Now we can inductively define functions  $c_i : T^{\mathbb{N}^i} \rightarrow T_{B_{i-1}^+}^{\mathbb{N}}$  and  $r_i = d_i \circ c_i$ .

Take a tree  $t \in T^{\mathbb{N}^1}$  and a node  $v = (u_1, u_2, \dots, u_m) \in \mathbb{N}^*$ . Define  $c_1(t) \in T_{B_0^+}^{\mathbb{N}}$  by:

$$c_1(t)(v) = \begin{cases} a^{u_1} b a^{u_2} b \dots b a^{u_m} b |_0 a & \text{if } v \in t \\ a^{u_1} b a^{u_2} b \dots b a^{u_m} b |_0 b & \text{if } v \notin t \end{cases}$$

For  $i > 1$  take a tree  $t \in T^{\mathbb{N}^i}$  and a node  $v \in \mathbb{N}^*$ . Let:

$$c_i(t)(v) = (r_{i-1}(t \upharpoonright_v))_{|v|} \in B_{i-1}^+$$

**Lemma 2.4.6** *For  $i < \omega$ , functions  $c_i, d_i, r_i$  defined above are continuous.*

**Proof:**

For  $d_i$  it holds by the definition— $n$ 'th element of the resulting sequence depends on finitely many nodes of an input tree.

The continuity of  $c_i$  can be proved by induction together with the continuity of  $r_i$ , since they cyclically depend on each other. Function  $c_1$  is clearly continuous. Function  $r_i$  is continuous as a composition of continuous functions, likewise, for  $i > 1$ ,  $c_i$  at each coordinate  $v$  is a composition of continuous functions:  $- \upharpoonright_v, r_{i-1}, -|_v$ .  $\square$

The following lemma states that functions  $r_i$  are in some sense **sequential**.

**Lemma 2.4.7** *Let  $i > 0$  and  $m < \omega$ . If  $t_1, t_2 \in T^{\mathbb{N}^i}$  agree on all  $v \in (\mathbb{N}^i)^*$  such that  $|v| \leq m$  then:*

$$r_i(t_1)_m = r_i(t_2)_m$$

**Proof:**

Recall that  $r_i(t) = d_i(c_i(t))$ . First observe that for a given tree  $t' \in T_{B_{i-1}^+}^{\mathbb{N}}$ , by the definition of  $d_i$ , the value  $d_i(t')_m$  depends only on  $v_m$  and the labels of  $t'$  on the path from the root to  $v_m$ .

Now use an induction on  $i$  and consider labels of  $c_i(t_1)$  and  $c_i(t_2)$  on the path from the root to  $v_m$ . For  $i = 1$  they depend only on  $t_1, t_2$  up to the depth of  $|v_m|$ , and  $|v_m| \leq m$ , thanks to our assumption about the order  $\sqsubseteq$ .

Take  $i > 1$  and a node  $v \preccurlyeq v_m$ . By the definition,  $c_i(t)(v) = r_{i-1}(t|_v)_{|v|}$ . Therefore, by the inductive assumption, this value depends only on  $t$  at the depth of at most  $|v| \leq |v_m| \leq m$ .  $\square$

From the above lemma we conclude that the labels on each branch  $\alpha \in \mathbb{N}^\omega$  in  $c_i(t)$  code the section  $t|_\alpha$ . Formally:

**Lemma 2.4.8** *For  $i > 1$ , a given tree  $t \in T^{\mathbb{N}^i}$  and an infinite branch  $\alpha \in \mathbb{N}^\omega$  we have:*

$$c_i(t)(\alpha) = r_{i-1}(t|_\alpha) \in (B_{i-1}^+)^{\omega}$$

**Proof:**

Take any  $m < \omega$  and consider  $v = \alpha|_m \in \mathbb{N}^m$ . By the definition:

$$(c_i(t)(\alpha))_m = c_i(t)(\alpha|_m) = (r_{i-1}(t|_v))_m$$

Since  $t|_v$  and  $t|_\alpha$  agree on all nodes up to the depth  $m$ , by Lemma 2.4.7, we have:

$$(r_{i-1}(t|_\alpha))_m = (r_{i-1}(t|_v))_m = c_i(t)(\alpha)_m$$

$\square$

**Languages  $H_i$  and Formulas**

In this section we define MSO+U formulas  $\varphi_i$ . The  $i$ 'th formula  $\varphi_i$  expresses properties of infinite words over  $B_{i+1}$ .

All the above work is done in the spaces  $(B_i^+)^{\omega}$ . Since we want to build MSO+U formulas over finite signatures, we need to work with finite alphabets. To achieve this we will use one additional encoding which is simply a kind of concatenation.

For  $i \geq 0$ , consider functions  $j_i : (B_i^+)^{\omega} \rightarrow (B_{i+1})^{\omega}$  defined as follows.

$$j_i(w_0, w_1, \dots) := [{}_i w_0]_i \cdot [{}_i w_1]_i \cdot \dots \quad (2.10)$$

Of course functions  $j_i$  defined above are continuous and  $1-1$ , where the latter comes from the fact that  $[{}_i, ]_i \notin B_i$ .

Recall that the address of an  $i$ -block is supposed to represent a node of a tree (see Definition 2.4.5). We say that such an  $i$ -block (or its address) **corresponds** to this node.

We will call a set  $A$  of addresses of nodes:

**deep** if the number of letters  $b$  in elements of  $A$  is unbounded,

**thin** if for any set  $P$  of prefixes of elements of  $A$  such that the number of letters  $b$  in elements of  $P$  is bounded, the lengths of sequences  $a^*$  in elements of  $P$  are bounded (see Figure 2.2).

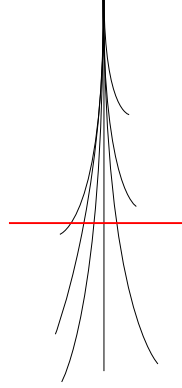


Figure 2.2: An illustration of the thin property—any section of finite depth contains only finitely many prefixes of branches in  $A$ .

The following remark provides a way of using the above properties.

**Remark 2.4.9** *A tree  $t \subseteq \mathbb{N}^*$  has an infinite branch if and only if there is a thin and deep set  $A$  of addresses of nodes in  $t$ .*

**Proof:**

First assume that  $t$  has an infinite branch  $\alpha \in \mathbb{N}^\omega$ . Take as  $A$  the set of addresses of nodes in  $\{\alpha|_n : n < \omega\}$ . Of course such  $A$  is deep. We show that  $A$  is thin. Consider any set  $P$  of prefixes of addresses in  $A$ , such that the number of letters  $b$  in elements of  $P$  is bounded by some number  $k \in \mathbb{N}$ . Since in each element of  $A$  the sequence  $a^*$  before the  $n$ 'th letter  $b$  has length  $\alpha_{n-1}$ , lengths of sequences  $a^*$  in  $P$  are bounded by  $\max_{n \leq k} \alpha_n$ .

Now take a thin and deep set  $A$  of addresses of nodes of  $t$ . We identify elements of  $A$  with those nodes, i.e.  $A \subseteq t$ . Consider as  $T$  the closure of  $A$  under prefixes, i.e.:

$$T := \{v \in \mathbb{N}^* : \exists_{v' \in A} v \preceq v'\}$$

Then  $T$  is an infinite tree, because  $A$  is deep. Additionally, at each level  $k < \omega$ , there are only finitely many nodes in  $T \cap \mathbb{N}^k$ , by thinness of  $A$ . So  $T$  is a finitely branching tree. Therefore, by König's Lemma,  $T$  contains an infinite branch  $\alpha$ . But  $T \subseteq t$ , so  $\alpha$  is also an infinite branch of  $t$ .  $\square$

Observe that both properties **deepness** and **thinness** of a set of addresses of a sequence of  $i$ -blocks can be expressed in MSO+U. It is because in those definitions we only use regular properties and properties like **the number of letters  $b$  is unbounded** or **the length of sequences  $a^*$  is bounded**.

We now define a series of MSO+U formulas  $\varphi_i$ . It is easy to see that we can express in MSO that a given word  $\alpha \in (B_{i+1})^\omega$  is of the form  $b_0 \cdot b_1 \cdot \dots$  such that each  $b_n$  is an  $i$ -block. We implicitly assume that all formulas  $\varphi_i$  express it.

Let  $\varphi_0$  additionally express that a given word is not of the form:

$$([_0 (a^*b)^* \mid_0 a]_0)^\omega$$

For  $i > 0$ , let  $\varphi_i$  express the following property:

There exists a set  $G$  containing only whole  $i$ -blocks such that:

1. the set of addresses of the  $i$ -blocks of  $G$  is deep,
2. the set of addresses of the  $i$ -blocks of  $G$  is thin,
3. the body of each  $i$ -block of  $G$  is a concatenation of  $(i-1)$ -blocks,
4. the bodies of the  $i$ -blocks of  $G$ , when concatenated (in the order as they occur in a word<sup>8</sup>), form an infinite word that satisfies  $\neg\varphi_{i-1}$ .

Take  $i \geq 0$ . Since  $L(\varphi_i) \subseteq (B_{i+1})^\omega$ , we can define:

$$H_i := j_i^{-1}(L(\varphi_i)) \subseteq (B_i^+)^\omega$$

Therefore, languages  $H_i$  defined above are (up to encoding  $j$ ) MSO+U definable.

We will use one important property of languages  $H_i$ .

**Definition 2.4.10** *A language  $L \subseteq X^\omega$  is monotone if for any  $\alpha, \beta \in X^\omega$ :*

$$\{\alpha_n : n < \omega\} \subseteq \{\beta_n : n < \omega\} \implies (\alpha \in L \implies \beta \in L)$$

---

<sup>8</sup>It will occur further that the order does not matter, but we fix the order to make the definition unambiguous.

Note, that belonging to a monotone language depends only on the set of letters occurring in a word, namely:

**Remark 2.4.11** *If  $L \subseteq X^\omega$  is a monotone language, then for any  $\alpha, \beta \in X^\omega$ :*

$$\{\alpha_n : n < \omega\} = \{\beta_n : n < \omega\} \implies (\alpha \in L \Leftrightarrow \beta \in L)$$

**Lemma 2.4.12** *Languages  $H_i \subseteq (B_i^+)^{\omega}$  are monotone.*

**Proof:**

The proof is by an induction on  $i$ .

For  $i = 0$  the claim is obvious.

For  $i > 0$ , formula  $\varphi_i$  expresses that there exists a set of  $i$ -blocks such that points (1)–(4) are satisfied. The order of the  $i$ -blocks does not affect points (1), (2), (3). We show that the order in which we concatenate bodies of  $i$ -blocks in point (4) does not matter. Observe, that the bodies are, by point (3), concatenations of  $(i-1)$ -blocks, so the concatenation of the bodies is also a concatenation of  $(i-1)$ -blocks. Now, by the inductive assumption, whether this concatenated word satisfies  $\varphi_{i-1}$ , depends only on the set of  $(i-1)$ -blocks it contains. As a result, indeed,  $\varphi_i$  depends only on the set of  $i$ -blocks in a word, and, since the outermost symbol in the formula is existential quantification, the dependency is monotonic.  $\square$

## Reductions

In this section we show that  $r_i$  is a reduction of  $\text{IF}^i$  to  $H_i$ . We do it in two steps.

**Definition 2.4.13** *For  $L \subseteq X^\omega$  let  $\text{EPath}(L) \subseteq T_X^{\mathbb{N}}$  be a set of such trees  $t$  that there exists an infinite word  $\alpha \in \mathbb{N}^\omega$  such that  $t(\alpha) \in L$ .*

*In other words  $\text{EPath}(L)$  is the set of trees that contain an infinite branch such that labels on this branch form a word in  $L$ .*

**Lemma 2.4.14** *For  $i > 0$ , function  $d_i : T_{B_{i-1}^+}^{\mathbb{N}} \rightarrow (B_i^+)^{\omega}$  is a reduction of  $\text{EPath}(\overline{H_{i-1}})$  to  $H_i$ .*

**Proof:**

We have to prove that for any  $t \in T_{B_i^+}^{\mathbb{N}}$

$$t \in \text{EPath}(\overline{H_{i-1}}) \iff d_i(t) \in H_i.$$

First assume that  $t \in \text{EPath}(\overline{H_{i-1}})$ . Let  $\alpha \in \mathbb{N}^\omega$  be a branch such that  $t(\alpha) \notin H_{i-1}$ . Let  $w = j_i(d_i(t)) \in (B_{i+1})^\omega$ . We show that  $w \models \varphi_i$ . Take as

$G$  the set containing  $i$ -blocks corresponding to nodes of  $\alpha$ . Then the set of addresses of  $i$ -blocks of  $G$  is obviously thin and deep (one node at each level of the tree). By the definition of  $d_i$ , the bodies of  $i$ -blocks are concatenations of  $(i-1)$ -blocks. Additionally, the set of  $(i-1)$ -blocks occurring in the bodies of  $i$ -blocks of  $G$  is exactly the set

$$\{[_{i-1} \cdot (t(\alpha))_n \cdot ]_{i-1} : n < \omega\}.$$

Language  $H_{i-1}$  is monotone, so, by Remark 2.4.11, since  $t(\alpha) \notin H_{i-1}$ , the set  $G$  satisfies point 4 in the definition of  $\varphi_i$ .

The other direction is a little more tricky. Assume that  $j_i(d_i(t)) \models \varphi_i$ . Let  $G$  be as in the definition of  $\varphi_i$ . Then the set of addresses of  $i$ -blocks of  $G$  is deep and thin. Let  $B \subseteq \mathbb{N}^*$  be the set of nodes corresponding to these addresses and let  $T$  be the closure of  $B$  under prefixes, i.e.:

$$T = \{v \in \mathbb{N}^* : \exists v' \in B \ v \preceq v'\}.$$

As in Remark 2.4.9, there exists an infinite branch  $\alpha \in \mathbb{N}^\omega$  of  $T$ . Observe that the set

$$\{[_{i-1} \cdot (t(\alpha))_n \cdot ]_{i-1} : n < \omega\}$$

is contained in the set of  $(i-1)$ -blocks in bodies of  $i$ -blocks in  $G$ . Because of the monotonicity of  $H_{i-1}$  and point 4 in the definition of  $\varphi_i$ ,  $t(\alpha) \notin H_{i-1}$ .  $\square$

**Lemma 2.4.15** *For  $i > 0$ , function  $c_i$  is a reduction of  $\text{IF}^i$  to  $\text{EPath}(\overline{H_{i-1}})$ .*

**Proof:**

We prove it by an induction on  $i$ .

Take  $i = 1$ . A tree  $t \in T^{\mathbb{N}^1}$  contains an infinite branch if and only if  $c_1(t)$  contains a branch labeled by words of the form  $(a^*b)^*|_0a$ , what happens if and only if  $c_1(t) \in \text{EPath}(\overline{H_0})$ .

Let  $i > 1$ . Take a tree  $t \in T^{\mathbb{N}^i}$ . The following conditions are equivalent:

$$\begin{array}{ll} t \in \text{IF}^i & \\ \exists \alpha \in \mathbb{N}^\omega \quad t|_\alpha \notin \text{IF}^{i-1} & \text{by the definition of } \text{IF}^i \\ \exists \alpha \in \mathbb{N}^\omega \quad c_{i-1}(t|_\alpha) \notin \text{EPath}(\overline{H_{i-2}}) & \text{by the inductive assumption} \\ \exists \alpha \in \mathbb{N}^\omega \quad r_{i-1}(t|_\alpha) \notin H_{i-1} & \text{by Lemma 2.4.14} \\ \exists \alpha \in \mathbb{N}^\omega \quad c_i(t)(\alpha) \notin H_{i-1} & \text{by Lemma 2.4.8} \\ c_i(t) \in \text{EPath}(\overline{H_{i-1}}) & \text{by the definition of } \text{EPath}(L). \end{array}$$

$\square$

We are now ready to prove the theorem.

**Proof (of Theorem 2.4.1):**

Take  $i < \omega$  and  $\varphi_i$  as defined above. Functions  $c_i, d_i, j_i$  are continuous by Lemma 2.4.6 and the definition of  $j_i$ . Moreover, using the definition of  $H_i$  and Lemmas 2.4.15, 2.4.14, their composition reduces  $\text{IF}^i$  to  $L(\varphi_i)$ . Thanks to Fact 2.4.4, the set  $\text{IF}^i$  is  $\Sigma_i^1$ -hard.  $\square$

**Upper Complexity Bound**

Theorem 2.4.1 gives a lower topological complexity bound for MSO+U logic. The corresponding upper bound relies on an easy observation.

**Proposition 2.4.16** *For every MSO+U formula  $\varphi$  over  $\omega$ -words or infinite trees, the language  $L(\varphi)$  is in  $\Sigma_i^1$  for some  $i < \omega$ .*

**Proof:**

Quantifiers  $\exists, \forall$  lift topological complexity by at most one level of the projective hierarchy, as expressed by formulas (1.17) and (1.18). Equation (2.4) shows that quantifier U can be interpreted as a countable intersection of countable unions ranging over finite sets. Therefore, for a given MSO+U formula  $\varphi$  we can inductively show that  $L(\varphi) \in \Sigma_{|\varphi|}^0$ , no matter whether  $\omega$ -word or infinite tree languages are concerned.  $\square$

As a result we get the exact topological complexity of MSO+U logic. Historically, we have started from considering  $\omega$ -word languages, hoping that the complexity of the logic would not be too high. Finally, it turned out to be so high that even changing the domain to trees is not able to increase it.

**Theorem 2.4.17** *The topological complexity of MSO+U logic over  $\omega$ -words or infinite trees is the class of all projective sets.*

**Proof:**

Upper bound is obtained by Proposition 2.4.16.

Theorem 2.4.1 gives the lower bound for  $\omega$ -words. For trees, we, for example, may observe that being the left-most branch is expressed in MSO and that each language  $H_i$  can be continuously reduced to the language of all trees that have a word from  $H_i$  labeling the left-most branch.  $\square$

As a side note of this chapter, we observe that the theorem is also true for languages of countable graphs, digraphs, grids, and many other countable structures that do not have non-projective predicates and that we can encode  $\omega$ -words in (in an MSO definable way).



## Consequences

The first level of the construction used to prove Theorem 2.4.1 was shown in a paper by Skrzypczak, Toruńczyk and the author of this thesis [HMT10]. Theorem 1 of the paper states that there is a  $\Sigma_1^1$ -complete language definable in MSO+U. Since in MSO+U we can use negation, we immediately got from that result the  $\Pi_1^1$ -complete language definable in MSO+U. From the discussion of the topological complexity of automata (see Remark 1.5.3), we get that there is no model of nondeterministic automata with Borel acceptance condition that captures MSO+U logic (this was noted as Corollary 1 in the paper).

Theorem 2.4.1, together with remarks in Section 1.5.3, leads to much stronger conclusion (first observed in the paper being the main source of this chapter [HS12, Theorem 5.2]):

**Corollary 2.4.18** *There is no model of deterministic, nondeterministic, alternating, or nested automata with an accepting condition on a fixed level of the projective hierarchy that can capture the whole expressive power of MSO+U on  $\omega$ -words.*

The above summarizes the automata theoretic consequences of the result about MSO+U presented in this thesis. The undecidability theorem [BPT16] implies that there is no automata model capturing MSO+U, that has decidable emptiness problem.

## 2.5 $\omega$ BS-automata

Prior to the results of the article coauthored by the author of this thesis [HMT10], several models of automata capturing some interesting fragments of MSO+U were considered. In this section we show the exact topological complexity of  $\omega$ BS-,  $\omega$ B- and  $\omega$ S-automata introduced by Bojańczyk and Colcombet [BC06]. As we will see, the topological complexity gap between MSO+U logic and these models of automata is huge. However, the automata capture the motivating example languages  $L_B$  and  $L_S$ , and, what is particularly important, have decidable emptiness problem (see [BC06]). To the best author's knowledge, the class of languages recognized by  $\omega$ BS-automata is still the largest known subclass of MSO+U definable languages with decidable emptiness problem<sup>9</sup>. In Section 2.5.3 we add a discussion about a

---

<sup>9</sup>Note, for example, that all min- and max-automata considered by Bojańczyk and Toruńczyk [BT09, Boj11] can be simulated by nondeterministic  $\omega$ BS-automata (see [BT09, Theorem 3]).

possible alternating variant of the automata. All the topological complexity results of this section were originally presented in the mentioned article [HMT10].

First, we present  $\omega$ BS-automata in an equivalent way to those used by Bojańczyk and Colcombet [BC06, Boj10]. The automata use counters. Note that, according to the definition below, the counters can only be updated and cannot be read during the run. They are used by the acceptance condition.

**Definition 2.5.1** *A (nondeterministic)  $\omega$ BS-automaton is a tuple  $\mathcal{A} = \langle A, Q, \Gamma, \delta, I, Acc \rangle$ , where:*

- *A is a finite alphabet,*
- *Q is a finite set of states,*
- *$\Gamma$  is a finite set of counters,*
- *$\delta \subseteq Q \times A \times Q \times (\{res, inc\} \times \Gamma)^*$  is a finite transition relation, where *inc* stands for the increment and *res* for the reset of a given counter,*
- *$I \subseteq Q$  is a set of initial states,*
- *Acc is an acceptance condition.*

A **run** of the automaton  $\mathcal{A}$  on a word  $w \in A^\omega$  is a sequence of transitions from  $\delta$  of the form:

$$(q_0, w_0, q_1, \gamma_0), (q_1, w_1, q_2, \gamma_1), (q_2, w_2, q_3, \gamma_2), \dots,$$

where  $q_0 \in I$ .

The value of each counter  $c \in \Gamma$  is initially 0 and is incremented or reset to 0 according to the consecutive transitions in a run. For  $c \in \Gamma$  and a run  $\rho$  we define a sequence  $val_\rho(c)$ , where  $val_\rho(c)_i$  is the value of counter  $c$  right before its  $i$ 'th reset in the run  $\rho$ . Note that if a counter  $c$  is reset only finitely many times then the sequence  $val_\rho(c)$  is finite.

The acceptance condition  $Acc$  is a boolean combination of constraints that can be of one of the forms:

$$\limsup_i val_\rho(c)_i < \infty \qquad \liminf_i val_\rho(c)_i = \infty,$$

for  $c \in \Gamma$ . The first constraint is called the **B-condition** (bounded), the second—the **S-condition** (strongly unbounded). In order for  $\liminf$  and  $\limsup$  to make sense, the constraints implicitly require the corresponding sequences to be infinite.

We use the notation  $B(c)$  for the B-condition and  $S(c)$  for the S-condition imposed on a counter  $c$ .

**Fact 2.5.2** *For each nondeterministic  $\omega BS$ -automaton, there is an equivalent nondeterministic  $\omega BS$ -automaton that uses a **positive** boolean combination of  $S$ -conditions and  $B$ -conditions as the acceptance condition.*

**Proof:**

By De Morgan laws, without loss of generality, we can assume that the acceptance condition of a given automaton  $\mathcal{A}$  is a positive boolean combination of literals of the form:

$$B(c), \quad \neg B(c), \quad S(c), \quad \neg S(c),$$

where  $c$  is a counter of the automaton.

We show how to replace literal  $\neg B(c)$  with  $S(c')$  for a new counter  $c'$ . We modify automaton  $\mathcal{A}$  by introduction of counter  $c'$  and extension of transition relation by the following operations on the new counter. After each reset of counter  $c$  and at the beginning of a run, automaton can nondeterministically decide to apply an increment to counter  $c'$  with each increment of  $c$  until the next reset, or not to apply any of those increments. If the decision was to apply the increments, then also the reset is applied to  $c'$ . Additionally in any moment when the automaton decides not to reset counter  $c$  any more, it can enter a mode in which an increment or a reset of counter  $c'$  is possible on every transition. After entering this mode the automaton stays in it until the end of the run, and it cannot reset counter  $c$  any more. No other operations on  $c'$  are performed.

Note that for any run  $\rho$  of the modified automaton there is a run of  $\mathcal{A}$  that has the same sequence of operations on counter  $c$  and other counters of  $\mathcal{A}$  as  $\rho$ . Additionally, any run of  $\mathcal{A}$  can be extended to a run of the modified automaton. Note that under the above construction, if  $val_\rho(c)$  is infinite, then  $val_\rho(c')$  can be any subsequence but only a subsequence of  $val_\rho(c)$  for a run  $\rho$  in the extended automaton. If  $val_\rho(c)$  is finite, then starting from some moment there are no resets and the automaton may use free increments and resets of counter  $c'$ . In particular, it can make condition  $S(c)$  satisfied. Since counter  $c$  does not satisfy condition  $B(c)$  if and only if sequence  $val_\rho(c)$  is finite or has a subsequence tending to infinity, literal  $\neg B(c)$  can be replaced with  $S(c')$  in the extended automaton.

The proof that literal  $\neg S(c)$  can be replaced with  $B(c)$  is exactly analogous.  $\square$

**Definition 2.5.3** *If the acceptance condition of an automaton is a positive boolean combination of B-conditions, the automaton is called an  $\omega\mathbf{B}$ -automaton. We analogously define  $\omega\mathbf{S}$ -automata.*

*Languages recognized by  $\omega\mathbf{BS}$ -automata (respectively  $\omega\mathbf{B}$ -automata,  $\omega\mathbf{S}$ -automata) are called  $\omega\mathbf{BS}$ -regular (respectively  $\omega\mathbf{B}$ -regular,  $\omega\mathbf{S}$ -regular).*

An important result of Bojańczyk and Colcombet [BC06] states:

**Theorem 2.5.4 ([BC06, Theorem 4.1])** *The complement of an  $\omega\mathbf{B}$ -regular language is an  $\omega\mathbf{S}$ -regular language, and vice versa.*

The result is much more involved than the partial duality of the B-condition and the S-condition used in the proof of Fact 2.5.2, because, by the straightforward transformation, while negating a nondeterministic automaton we obtain a co-nondeterministic (universal) one, not a nondeterministic one.

Both the classes are extensions of the class of  $\omega$ -regular languages, since the Büchi condition can be simulated by either a B-condition or an S-condition. The simulation relies on the fact that both, B- and S-condition, require infinite number of resets. We get:

**Remark 2.5.5 ([BC06])** *Each  $\omega$ -regular language is both  $\omega\mathbf{B}$ -regular and  $\omega\mathbf{S}$ -regular.*

**Example 2.5.6** *Language  $L_S$  can be recognized by an  $\omega\mathbf{S}$ -automaton. The automaton has one state and uses one counter that is increased when reading letter  $a$  and is reset after each  $b$ . The acceptance condition is simply an S-condition on the only counter.*

### 2.5.1 Complexity of $\omega\mathbf{B}$ - and $\omega\mathbf{S}$ -regular Languages

As it is argued by Bojańczyk and Colcombet [BC06],  $\omega\mathbf{B}$ -,  $\omega\mathbf{S}$ -, and  $\omega\mathbf{BS}$ -automata cannot be determinized. For  $\omega\mathbf{B}$ -automata it can be proven using topological argument.

**Definition 2.5.7** *An  $\omega\mathbf{B}$ -,  $\omega\mathbf{S}$ -, or  $\omega\mathbf{BS}$ -automaton is deterministic, if it has exactly one initial state, and transition relation is a function  $\delta : Q \times A \rightarrow Q \times (\{res, inc\} \times \Gamma)^*$ .*

**Lemma 2.5.8** *Each language recognized by a deterministic  $\omega\mathbf{B}$ -automaton is in class  $\mathcal{BC}_2^0$ .*

**Proof:**

Recall that the acceptance condition of an  $\omega B$ -automaton is a positive boolean combination of B-condition and that B-condition is a conjunction of:

1. the lengths of sequences of increments of counter  $c$  separated by resets are commonly bounded, and
2. counter  $c$  is reset infinitely many times.

Condition (1) can be unraveled as the following countable disjunction:

$$\bigvee_{n < \omega} \underbrace{\text{there is no sequence of } n \text{ increments of } c \text{ without a reset in between}}_{L_n}$$

Observe that for a fixed  $n$ ,  $L_n$  is closed. Therefore, condition (1) describes a  $\Sigma_2^0$  set of runs. Condition (2) is  $\Pi_2^0$ , since it can be projected to set  $N_2$  from Fact 1.2.7. Therefore, the acceptance condition of a  $\omega B$ -automaton is in class  $\mathcal{BC}_2^0$ . By Remark 1.5.2, all languages recognized by deterministic  $\omega B$ -automata are in class  $\mathcal{BC}_2^0$ .  $\square$

**Corollary 2.5.9** *Deterministic  $\omega B$ -automata have less expressive power than the nondeterministic ones.*

**Proof:**

Note that, by the complementation result (Theorem 2.5.4) and by Example 2.5.6, the complement of language  $L_S$  is recognized by a nondeterministic  $\omega B$ -automaton. By Lemma 2.3.4, language  $\overline{L_S}$  is  $\Sigma_3^0$ -hard, hence cannot be recognized by any deterministic  $\omega B$ -automaton.  $\square$

The remainder of the section is dedicated to the proof of the following.

**Theorem 2.5.10** *The topological complexity of the class of  $\omega B$ -regular languages is  $\Sigma_3^0$ . The topological complexity of the class of  $\omega S$ -regular languages is  $\Pi_3^0$ .*

**Lemma 2.5.11** *Each  $\omega B$ -regular language is in  $\Sigma_3^0$ .*

Corollary 2.5.9 implies that we cannot use the common routine of estimating topological complexity using the complexity of the acceptance condition of a deterministic automaton. On the other hand, straightforward inference of the topological complexity using nondeterministic automata would give

non-Borel complexity (see Remark 1.5.3). Recall that this is because non-determinism corresponds to second order quantification (or projection) “exists a run”. The trick of the following proof is to push the quantification inside the formula where we can, in a way, get rid of it using determinization of parity automata.

**Proof:**

Fix an  $\omega$ B-automaton  $\mathcal{A}$  recognizing a language  $L$ , and let us first assume that its accepting condition is a conjunction of B-conditions, i.e. is of the form

$$\bigwedge_{c \in \Gamma_B} B(c),$$

for a set of counters  $\Gamma_B$ .

Since there are finitely many counters, each of the considered counters is bounded if and only if there is a common bound  $k$  for all of them. Therefore  $L$  can be defined as:

$$\begin{aligned} L &= \{w : \exists_\rho \bigwedge_{c \in \Gamma_B} \text{val}_\rho(c) \text{ is infinite but bounded}\} \\ &= \bigcup_{k < \omega} \underbrace{\{w : \exists_\rho \bigwedge_{c \in \Gamma_B} \text{val}_\rho(c) \text{ is infinite and bounded by } k\}}_{L_k}, \end{aligned}$$

where the quantification is over the set of all runs of  $\mathcal{A}$  on  $w$ .

It is easy to see that for a fixed  $k$ ,  $L_k$  can be recognized by a non-deterministic Büchi automaton. We simply store counter values in states and do not allow them to be incremented above  $k$ . The acceptance condition requires each of the counters  $c \in \Gamma_B$  to be reset infinitely often. Hence  $L_k$  is  $\omega$ -regular. Since, by Theorem 2.2.1, each  $\omega$ -regular language is in  $\mathcal{BC}_2^0$  and  $L$  is a countable union of such sets,  $L \in \Sigma_3^0$ .

In the general form, the acceptance condition of an  $\omega$ B-automaton is a positive boolean combination of B-conditions. We can write such a condition in disjunctive normal form (DNF).

$$\bigvee_{i=1}^n \bigwedge_{c \in \Gamma_i} B(c),$$

for some  $n < \omega$  and some sets of counters  $\Gamma_i$ . The language accepted by this automaton is:

$$\begin{aligned} L &= \{w : \exists_\rho \bigvee_{i=1}^n \bigwedge_{c \in \Gamma_i} \text{val}_\rho(c) \text{ is infinite but bounded}\} \\ &= \{w : \bigvee_{i=1}^n \exists_\rho \bigwedge_{c \in \Gamma_i} \text{val}_\rho(c) \text{ is infinite but bounded}\} \\ &= \bigcup_{i=1}^n \underbrace{\{w : \exists_\rho \bigwedge_{c \in \Gamma_i} \text{val}_\rho(c) \text{ is infinite but bounded}\}}_{\in \Sigma_3^0} \end{aligned}$$

Hence  $L \in \Sigma_3^0$ . □

Thanks to Theorem 2.5.4, we have:

**Corollary 2.5.12** *Each  $\omega S$ -regular language is in  $\Pi_3^0$ .*

**Proof (of Theorem 2.5.10):**

The upper complexity bound is given by Lemma 2.5.11 and Corollary 2.5.12.

Language  $L_S$  provides the lower complexity bound for  $\omega S$ -regular languages (see Example 2.5.6 and Lemma 2.3.4), while its complement—for  $\omega B$ -regular languages. □

## 2.5.2 Complexity of $\omega BS$ -regular Languages

In this section we show that the reasoning presented in the previous section can be lifted to the case of automata that can use both S- and B-conditions.

**Theorem 2.5.13** *The topological complexity of the class of  $\omega BS$ -regular languages is  $\Sigma_4^0$ .*

The proof of the following is by Szymon Toruńczyk.

**Lemma 2.5.14** *Each  $\omega BS$ -regular language is in  $\Sigma_4^0$ .*

**Proof:**

The proof, on one hand, will use the result of Corollary 2.5.12 and, on the other hand, will repeat a reasoning similar to the one from the proof of Lemma 2.5.11.

Let us fix an  $\omega BS$ -regular language  $L$  and an automaton  $\mathcal{A}$  recognizing it. First assume that an acceptance condition of  $\mathcal{A}$  is of the form:

$$\bigwedge_{c \in \Gamma_B} B(c) \quad \wedge \quad \bigwedge_{c \in \Gamma_S} S(c)$$

Then language  $L$  satisfies:

$$L = \bigcup_{k < \omega} \underbrace{\left\{ w : \exists_\rho \left( \bigwedge_{c \in \Gamma_B} \text{val}_\rho(c) \text{ is infinite and bounded by } k \right) \wedge \bigwedge_{c \in \Gamma_S} \text{val}_\rho(c) \text{ converges to } \infty \right\}}_{L_k}$$

Note that each language  $L_k$  is  $\omega S$ -regular, hence, by Corollary 2.5.12, it is in  $\Pi_3^0$ . Therefore  $L$ , as a countable union of such languages, is in  $\Sigma_4^0$ .

A general acceptance condition can be written in disjunctive normal form (DNF), where literals are of the form  $B(c)$  or  $S(c)$  (see Fact 2.5.2). Again, the language accepted by such an automaton is a union of languages corresponding to each disjunct, so it is in  $\Sigma_4^0$ .  $\square$

Now we need to show that the bound is tight. For that we consider the same language, that was used in the paper by Bojańczyk and Colcombet [BC06, Corollary 2.8] to show that the class of  $\omega$ BS-regular languages is not closed under complements. Let

$$S = \left\{ x \in \mathbb{N}^\omega : \begin{array}{l} \text{the sequence } x \text{ can be partitioned into} \\ \text{a (possibly finite) bounded subsequence and} \\ \text{a subsequence that is empty or tends to } \infty \end{array} \right\}$$

For  $g : \mathbb{N}^\omega \rightarrow \{a, b\}^\omega$  as defined by equation 2.3,  $g(S)$  is exactly the language used in the mentioned paper [BC06, Corollary 2.8]. The same language is given by Thomas and Lescow [TL93, page 595] as an example of  $\Sigma_4^0$ -complete set.

First, observe that:

$$\overline{S} = \{x \in \mathbb{N}^\omega : \exists_{k \in \mathbb{N}}^\infty \exists_{j < \omega}^\infty x_j = k\}$$

**Fact 2.5.15** *Language  $\overline{S}$  is  $\Pi_4^0$ -complete.*

**Proof:**

First we prove  $\Pi_4^0$ -hardness. Let  $L \in \Pi_4^0(X)$  for a zero-dimensional Polish space  $X$ . By Lemma 1.2.9 and Lemma 1.2.8:

$$L = \bigcap_{m < \omega} \bigcup_{n < \omega} L_{m,n}$$

for  $L_{m,n} \in \Pi_2^0(X)$  for each  $m$  and  $n$ . Since, by Fact 1.2.7,  $N_2$  is  $\Pi_2^0$ -complete, for each  $m$  and each  $n$  there is a continuous reduction  $f_{m,n} : X \rightarrow \{0, 1\}^\omega$  of  $L_{m,n}$  to  $N_2$ .

Let us fix a bijection  $\iota : \omega \times \omega \rightarrow \omega$  and define a continuous function  $f : X \rightarrow \mathbb{N}^\omega$  by:

$$(f(x))_{\iota(m,n,l)} = \begin{cases} \iota(m,n) & \text{if } (f_{m,n}(x))_l = 1 \\ 0 & \text{otherwise} \end{cases}$$



Let us now argue that  $f$  reduces  $L$  to  $\overline{S}$ .

$$\begin{aligned}
f(x) \in \overline{S} & \\
\iff \exists_k^\infty \exists_j^\infty (f(x))_j = k & \quad \text{by the definition of } f \\
\iff \exists_k^\infty \exists_l^\infty (f(x))_{\iota(k,l)} = k & \quad k \text{ can only occur at positions} \\
& \quad \text{of the form } \iota(k, \_) \text{ in } f(x) \\
\iff \exists_{(m,n)}^\infty \exists_l^\infty (f(x))_{\iota(\iota(m,n),l)} = \iota(m,n) & \\
\iff \exists_{(m,n)}^\infty \exists_l^\infty (f_{m,n}(x))_l = 1 & \quad \text{by the definition of } f \\
\iff \exists_{(m,n)}^\infty f_{m,n}(x) \in N_2 & \quad \text{by the definition of } N_2 \\
\iff \exists_{(m,n)}^\infty x \in L_{m,n} & \quad \text{because } f_{m,n} \text{ is a reduction} \\
\iff \exists_m^\infty x \in \bigsqcup_{n < \omega} L_{m,n} & \quad \text{because for each } m \text{ there is} \\
& \quad \text{at most one } n \text{ s.t. } x \in L_{m,n} \\
\iff x \in \bigcap_{m < \omega} \bigsqcup_{n < \omega} L_{m,n} & \quad \text{because } \{\bigsqcup_{n < \omega} L_{m,n}\}_{m < \omega} \text{ is} \\
& \quad \text{decreasing} \\
\iff x \in L &
\end{aligned}$$

This concludes the proof of hardness.

We prove that  $\overline{S} \in \mathbf{\Pi}_4^0$ . Let us observe that:

$$\overline{S} = \{x \in \mathbb{N}^\omega : \exists_{k \in \mathbb{N}}^\infty x \in h_k^{-1}(N_2)\},$$

for  $h_k : \mathbb{N}^\omega \rightarrow \{0, 1\}^\omega$  defined by:

$$(h_k(x))_i = \begin{cases} 1 & \text{if } x_i = k \\ 0 & \text{otherwise} \end{cases}$$

Therefore, by equation (1.13),  $\overline{S} \in \mathbf{\Pi}_4^0$ . □

**Corollary 2.5.16** *Language  $S$  is  $\Sigma_4^0$ -complete.*

For function  $g$  as defined by equation (2.3), we have:

**Corollary 2.5.17** *Language  $g(S)$  is  $\Sigma_4^0$ -hard.*

**Proof:**

Function  $g$  is an injection, so  $g^{-1}(g(S)) = S$  and  $g$  reduces  $S$  to  $g(S)$ . Function  $g$  is also continuous, hence, language  $g(S)$  is  $\Sigma_4^0$ -hard. □

Now it suffices to note that:

**Fact 2.5.18** ([BC06]) *Language  $g(S)$  is  $\omega$ BS-regular.*

Originally the fact is proven by defining the language using an appropriate  $\omega$ BS-regular expression<sup>10</sup>, however it is straightforward to define a nondeterministic automaton recognizing this language.

**Proof (of Fact 2.5.18, sketch):**

We construct a nondeterministic  $\omega$ BS-automaton recognizing the language. The automaton uses two counters and before each block of  $a$ 's nondeterministically decides which one of the counters will be increased while reading the block. The acceptance condition imposes S-condition on one counter and B-condition on the other one.  $\square$

**Proof (of Theorem 2.5.13):**

The upper topological complexity bound is given by Lemma 2.5.14, while the lower bound comes from Fact 2.5.18 and Corollary 2.5.17.  $\square$

### 2.5.3 Alternating $\omega$ BS-automata

On the way towards finding a model of automata for MSO+U logic, alternating  $\omega$ BS-automata were considered (see [HMT10]). As it was later discovered (see Corollary 2.4.18), the model is not able to capture the whole logic. On the other hand, to the author's best knowledge, it is not known if each language recognized by an alternating  $\omega$ BS-automaton is definable in MSO+U logic. However, since the class seems to be a natural extension of the class of nondeterministic  $\omega$ BS-automata, we give here a lower bound for the topological complexity of alternating  $\omega$ BS-automata. The bound is  $\omega$  levels of the Borel hierarchy, which can be compared to 4 levels inhabited in case of nondeterministic automata.

**Alternating  $\omega$ BS-automata** are defined similarly as nondeterministic  $\omega$ BS-automata. The main difference is that the state space  $Q$  is partitioned into  $Q_{\forall}$  (universal states) and  $Q_{\exists}$  (existential states). We use game semantics for such automata. For a given alternating automaton  $\mathcal{A}$  and a word  $w \in A^{\omega}$  we define a two-player game. A play in this game starts in the initial state<sup>11</sup> of the automaton and in the first position of the word, and proceeds by

---

<sup>10</sup>In their paper [BC06], Bojańczyk and Colcombet introduce  $\omega$ BS-regular expressions as an extension of standard regular expressions by introduction of bounded and strongly unbounded exponents. They are proven to have the same expressive power as  $\omega$ BS-automata. We refer the reader to the paper [BC06] for details.

<sup>11</sup>To provide a convenient way of complementing alternating automata it is important that there is only one initial state. However, since nondeterministic  $\omega$ BS-automata with one initial state have the same expressive power as automata with a set of initial states, the alternating automata are still a proper extension of the nondeterministic ones.

applying transitions of the automaton on word  $w$  consistent with the current state and the letter in the current position in the word. Player  $\forall$  (respectively  $\exists$ ) chooses transitions when the automaton is in a state from  $Q_\forall$  (respectively  $Q_\exists$ ). Additionally, in such automata, the transition relation can contain  $\varepsilon$ -transitions. An  $\varepsilon$ -transition consists only of a state change; the letter is not read (position in the word does not change) nor are the counters updated by such a transition. The result of the play is an infinite sequence of transitions consistent with the transition relation and consecutive letters of the word. The play is winning for  $\exists$  if the sequence of counter operations induced by the transitions satisfies the acceptance BS-condition of  $\mathcal{A}$ , i.e. a boolean combination of B- and S-conditions. Word  $w$  is accepted by the automaton if and only if Player  $\exists$  has a winning strategy in the above game.

### Languages Complete for the Classes $\Pi_{2n}^0$

We now present examples of languages of infinite words complete for the Borel classes  $\Pi_{2n}^0$ , which are recognized by alternating  $\omega$ BS-automata. The languages essentially generalize set  $\overline{S}$  from Section 2.5.2.

First, we work with the spaces of sequences of vectors of numbers  $\mathcal{N}_n = (\mathbb{N}^n)^\omega$ . An easy embedding, described below, will transfer the results into the space of infinite words over finite alphabet. For  $n = 0$ , the above definition gives the space consisting of the unique infinite  $\omega$ -sequence of empty vectors,  $\mathcal{N}_0 = \{(\varepsilon)^\omega\}$ .

Let us fix an alphabet  $A = \{a, b, c\}$ . We encode a sequence of vectors in space  $A^\omega$ . Each vector  $(z_n, z_{n-1}, \dots, z_1)$  is mapped to the word  $ca^{z_n}ba^{z_{n-1}}b \dots ba^{z_1}$ , and the codes of consecutive vectors are concatenated. We denote the embedding defined this way  $W_n: \mathcal{N}_n \rightarrow A^\omega$ .

We use the following notations to easily operate on sequences of vectors.

- For  $n > 0$ ,  $\eta \in (\mathbb{N}^n)^{\leq \omega}$  and  $S \subseteq \mathbb{N}$ , let  $\eta \upharpoonright_{\in S}$  be the subsequence of  $\eta$  consisting of those vectors that have a value from  $S$  at the first coordinate. If  $S = \{m\}$  for some  $m \in \mathbb{N}$ , we simply write  $\eta \upharpoonright_{=m}$  instead of  $\eta \upharpoonright_{\in \{m\}}$ .
- For  $n > 0$ , let  $\pi_1: (\mathbb{N}^n)^{\leq \omega} \rightarrow (\mathbb{N}^{n-1})^{\leq \omega}$  be the projection that cuts off the first coordinate from each vector in a given sequence.

The following sequence of languages is, up to an encoding, presented as an example by Thomas and Lescow [TL93, pages 595–596].

**Definition 2.5.19** *For  $n > 0$  we define:*

$$L_n := \{\eta \in \mathcal{N}_n : \exists_{m_n \in \mathbb{N}} \exists_{m_{n-1} \in \mathbb{N}} \dots \exists_{m_1 \in \mathbb{N}} \exists_{x < \omega} \eta_x = (m_n, m_{n-1}, \dots, m_1)\}$$

Note that the order of quantifiers is important, because the quantifiers  $\exists_x^\infty \exists_y^\infty$  do not commute, in general. Additionally note that  $L_0 = \{\varepsilon\}^\omega = \mathcal{N}_0$ .

Let us observe that  $L_1 = \overline{S}$ , for  $S$  defined as in Section 2.5.2.

The following remark describes languages  $L_n$  in an inductive fashion.

**Remark 2.5.20** For  $n > 0$ , a sequence  $\eta \in \mathcal{N}_n$  belongs to  $L_n$  if and only if there exist infinitely many  $m \in \mathbb{N}$  such that  $\eta \upharpoonright_{=m}$  is an infinite sequence and  $\pi_{\bar{1}}(\eta \upharpoonright_{=m}) \in L_{n-1}$ .

Note the following properties of languages  $L_n$ .

**Proposition 2.5.21**

- **monotonicity:** If  $\eta \in \mathcal{N}_n$  and  $\nu$  is a subsequence of  $\eta$ , then  $\nu \in L_n \implies \eta \in L_n$ ,
- **prefix independence:** For  $\eta \in \mathcal{N}_n$  and  $\nu \in (\mathbb{N}^n)^*$ ,  $\eta \in L_n \iff \nu\eta \in L_n$ .
- **pigeonhole property:** Let  $\nu_1, \nu_2, \dots, \nu_k$  be a partition of a sequence  $\eta \in L_n$  into subsequences, then for some  $j \in \{1, 2, \dots, k\}$ ,  $\nu_j \in L_n$ .

**Proof:**

Monotonicity and prefix independence come straight from the corresponding properties of  $\exists^\infty$  quantifier, namely the innermost occurrence of this quantifier in the formula defining  $L_n$ .

We prove pigeonhole property by an induction on  $n$  using the characterization of languages  $L_n$  given by Remark 2.5.20.

Let  $n = 0$ . In a partition of  $\eta \in L_0$  into finitely many subsequences, one of the subsequences is infinite. Each infinite sequence in  $\mathcal{N}_0$  is in  $L_0$ , what concludes the proof of the induction basis.

For the inductive step take  $n > 0$  and assume that pigeonhole property holds for  $L_{n-1}$ . Now take  $\eta \in L_n$  and let  $\nu_1, \nu_2, \dots, \nu_k$  be a partition of  $\eta$  into subsequences. By Remark 2.5.20, there is an infinite set of numbers  $\{m_0, m_1, m_2, \dots\} \subseteq \mathbb{N}$  such that for each  $i < \omega$ ,  $\eta \upharpoonright_{=m_i}$  is an infinite sequence and  $\pi_{\bar{1}}(\eta \upharpoonright_{=m_i}) \in L_{n-1}$ . Note that for each  $i$ ,  $\pi_{\bar{1}}(\nu_1 \upharpoonright_{=m_i}), \pi_{\bar{1}}(\nu_2 \upharpoonright_{=m_i}), \dots, \pi_{\bar{1}}(\nu_k \upharpoonright_{=m_i})$  is a partition of  $\pi_{\bar{1}}(\eta \upharpoonright_{=m_i})$ . Therefore, by the inductive assumption, there is such  $j_i \in \{1, 2, \dots, k\}$  that  $\pi_{\bar{1}}(\nu_{j_i} \upharpoonright_{=m_i}) \in L_{n-1}$ . Since each  $j_i \leq k$ , there is such  $p \leq k$  that  $I_p := \{i : j_i = p\}$  is an infinite set. Therefore,  $\{m_i\}_{i \in I_p}$  is an infinite sequence of numbers, such that for each  $i \in I_p$ ,  $\nu_p \upharpoonright_{=m_i}$  is an infinite sequence and  $\pi_{\bar{1}}(\nu_p \upharpoonright_{=m_i}) \in L_{n-1}$ . Hence,  $\nu_p \in L_n$ , what concludes the inductive step proof.  $\square$

Let us now transform characterization given by Remark 2.5.20 to obtain characterization in logical terms, that will serve as a guideline in the construction of alternating automata recognizing the languages.

For  $n > 0$ , for a sequence  $\eta \in \mathcal{N}_n$  and for a set  $X$  of positions in  $\eta$ , let  $\text{Bnd}_n(\eta, X)$  express that the values of the first coordinate of vectors in  $\eta$  at positions from  $X$  are bounded.

**Proposition 2.5.22** *For  $n > 0$ , a sequence  $\eta \in \mathcal{N}_n$  belongs to  $L_n$  if and only if:*

$$\forall_X [\text{Bnd}_n(\eta, X) \implies \exists_Y [\text{Bnd}_n(\eta, Y) \wedge (X \cap Y = \emptyset) \wedge (\pi_{\bar{1}}(\{\eta_i\}_{i \in Y}) \in L_{n-1})]] \quad (2.11)$$

*In the above,  $\{\eta_i\}_{i \in Y}$  denotes the subsequence of  $\eta$  consisting of the elements whose positions belong to  $Y$ .*

**Proof:**

Let us fix  $n > 0$ .

( $\Rightarrow$ ) Let  $\eta \in L_n$ . Take any set  $X$  of positions of  $\eta$  such that values of the first coordinate of vectors at those positions are commonly bounded. We need to find a set  $Y$  as required by (2.11). Let  $m_X$  be the greatest value occurring at the first coordinate at positions from  $X$ . By Remark 2.5.20, there are infinitely many numbers  $m$  such that  $\pi_{\bar{1}}(\eta \upharpoonright_{=m}) \in L_{n-1}$ . Let us then take  $m_Y > m_X$  such that  $\pi_{\bar{1}}(\eta \upharpoonright_{=m_Y}) \in L_{n-1}$ . Now take as  $Y$  the set of all positions at which  $\eta$  has value  $m_Y$  at the first coordinate. Such  $Y$  is clearly disjoint with  $X$ ;  $\text{Bnd}_n(\eta, Y)$  clearly holds; and  $\pi_{\bar{1}}(\{\eta_i\}_{i \in Y}) = \pi_{\bar{1}}(\eta \upharpoonright_{=m_Y}) \in L_{n-1}$ . Hence (2.11) holds for  $\eta$ .

( $\Leftarrow$ ) Take any  $\eta \in \mathcal{N}_n$  that satisfies (2.11). We prove that there are arbitrary large  $m$  such that  $\pi_{\bar{1}}(\eta \upharpoonright_{=m}) \in L_{n-1}$ . Let  $m_0 \in \mathbb{N}$  and let  $X$  be the set of all positions at which  $\eta$  has a value less than  $m_0$  at the first coordinate. Let us take  $Y$  such that (2.11) holds, in particular  $\pi_{\bar{1}}(\{\eta_i\}_{i \in Y}) \in L_{n-1}$ . Let  $m_Y$  be the greatest value occurring as the first coordinate at positions from  $Y$ . Observe that  $\pi_{\bar{1}}((\{\eta_i\}_{i \in Y}) \upharpoonright_{=m_0})$ ,  $\pi_{\bar{1}}((\{\eta_i\}_{i \in Y}) \upharpoonright_{=(m_0+1)})$ ,  $\dots$ ,  $\pi_{\bar{1}}((\{\eta_i\}_{i \in Y}) \upharpoonright_{=m_Y})$  is a partition of  $\pi_{\bar{1}}(\{\eta_i\}_{i \in Y})$ . By pigeonhole property, there is such  $m_0 \leq m \leq m_Y$  that  $\pi_{\bar{1}}((\{\eta_i\}_{i \in Y}) \upharpoonright_{=m}) \in L_{n-1}$ . Now, by monotonicity,  $\pi_{\bar{1}}(\eta \upharpoonright_{=m}) \in L_{n-1}$ . We have shown that for each  $m_0$  there is such  $m \geq m_0$  that  $\pi_{\bar{1}}(\eta \upharpoonright_{=m}) \in L_{n-1}$ . Therefore, there is infinitely many  $m$  such that  $\pi_{\bar{1}}(\eta \upharpoonright_{=m}) \in L_{n-1}$ . Hence, by Remark 2.5.20,  $\eta \in L_n$ .  $\square$

We sketch a proof of the following as a side note of the section:

**Proposition 2.5.23** *Languages  $W_n(L_n)$  are MSO+U definable.*

**Proof (sketch):**

The idea is to formalize in MSO+U the property given by (2.11). The first step to achieve it is to work with  $W_n$ -encodings of sequences of vectors. For such encoded sequences, using i.a. techniques as shown in formula (2.6), we can express in MSO properties like “being a maximal block of consecutive  $a$ ’s that correspond to the  $k$ -th coordinate of one of the vectors in a sequence”. One can observe that, by similar means that are used in equation 2.5, property  $\text{Bnd}_n$  can also be defined in MSO+U in this context.  $\square$

**Topological Complexity**

Since the topological complexity of languages  $W_n(L_n)$  is mentioned but not proven in the paper by Thomas and Lescow [TL93, pages 595–596], we prove the following fact, for completeness.

**Proposition 2.5.24** *For every  $n > 0$ , language  $L_n$  is  $\Pi_{2n+2}^0$ -complete.*

**Proof:**

The proof is inductive. For  $n = 1$  this is a consequence of Fact 2.5.15. For  $n > 1$ , by equation (1.13) and by the inductive assumption, language  $L_n$  is in class  $\Pi_{2n+2}^0$ .

Let us take  $n > 1$ , and any  $M \in \Pi_{2n+2}^0(X)$ , for a zero-dimensional Polish space  $X$ . We construct a continuous reduction of  $M$  to  $L_n$ .

By Lemma 1.2.9, there is a decreasing sequence  $\{M_i\}_{i < \omega}$  of sets from  $\Sigma_{2n+1}^0$  such that  $M = \bigcap_i M_i$ . By Lemma 1.2.8, there is a sequence  $\{M_k^{(i)}\}_{k < \omega}$  of sets from  $\Pi_{2n}^0$  that are (for fixed  $i$ ) pairwise disjoint, and:

$$\bigsqcup_k M_k^{(i)} = M_i$$

By the inductive assumption, language  $L_{n-1}$  is  $\Pi_{2n}^0$ -hard, so there are continuous reductions  $R_k^{(i)}: X \rightarrow \mathcal{N}_{n-1}$  of sets  $M_k^{(i)}$  to  $L_{n-1}$ .

Let us fix a bijection  $\iota: \omega^2 \rightarrow \omega$ , and define function  $R: X \rightarrow \mathcal{N}_n$  by:

$$(R(x))_{\iota(i,k),m} := \left( \iota(i,k), \left( R_k^{(i)}(x) \right)_m \right) \in \mathbb{N}^n, \quad (2.12)$$

where the first element in braces is a number and the second is an  $(n-1)$ -vector of numbers, so they form an  $n$ -vector.

Since functions  $R_k^{(i)}$  are continuous,  $R$  is also continuous. Now, it is enough to show that  $x \in M \Leftrightarrow R(x) \in L_n$ .

$$\begin{aligned}
& R(x) \in L_n \\
& \iff \exists_m^\infty \pi_1 (R(x) \upharpoonright_{=m}) \in L_{n-1} && \text{by Remark 2.5.20} \\
& \iff \exists_{(i,k)}^\infty \pi_1 (R(x) \upharpoonright_{=\iota(i,k)}) \in L_{n-1} && \text{by (2.12)} \\
& \iff \exists_{(i,k)}^\infty R_k^{(i)}(x) \in L_{n-1} && \text{by (2.12)} \\
& \iff \exists_{(i,k)}^\infty x \in M_k^{(i)} && \text{because } R_k^{(i)} \text{ is a reduction} \\
& && \text{of } M_k^{(i)} \text{ to } L_{n-1} \\
& \iff \exists_i^\infty \exists_k x \in M_k^{(i)} && \text{because for a fixed } i \text{ there is} \\
& && \text{at most one } k \text{ s.t. } x \in M_k^{(i)} \\
& \iff \exists_i^\infty x \in M_i \\
& \iff x \in \bigcap_i M_i = M
\end{aligned}$$

□

## Automata Construction

**Theorem 2.5.25** *For each  $n < \omega$ , there is an alternating  $\omega$ BS-automaton recognizing a  $\Pi_{2n+2}^0$ -hard language.*

In order to prove this theorem we provide yet another characterization of languages  $L_n$ . This time in terms of games.

For  $n < \omega$  and  $\eta \in \mathcal{N}_n$ , let  $G_n(\eta)$  be the following game. The game is played by two players, the existential one  $\exists$  and the universal one  $\forall$ . During a play in the game players process consecutive vectors of sequence  $\eta$ . In each vector  $\eta_k = (z_n, z_{n-1}, \dots, z_1)$  they consider consecutive components  $z_n, z_{n-1}, \dots, z_1$ . A move of each player can be to select a given component or not to select it, according to the following rules. If a component currently considered is the first component of some vector, Player  $\forall$  decides to select it or not. If  $\forall$  has not selected the component,  $\exists$  decides to select it or not, otherwise she cannot select the component. If the component is not the first one in a vector, Player  $\forall$  can select it only if all preceding components of this vector were selected by  $\exists$ . Player  $\exists$  can select any given component only if she has selected all preceding components of the same vector and if Player  $\forall$  has decided not to select the considered component. As soon as players make their decisions, the play progresses to the next component, or the first component of the next vector if the considered component is the last one in a vector.

The winning condition is stated using sequences  $a^{(n)}, e^{(n)}, a^{(n-1)}, e^{(n-1)}, \dots, a^{(1)}, e^{(1)}$ , where  $a^{(n)}$  is the sequence of the first components selected by Player  $\forall$ ,  $e^{(n)}$  is the sequence of the first components selected by Player

$\exists$ , and, in general,  $a^{(n-k+1)}$  (respectively  $e^{(n-k+1)}$ ) is the sequence of  $k$ th components selected by Player  $\forall$  (respectively Player  $\exists$ ). Player  $\exists$  wins if and only if for each  $0 < k \leq n$ :

$$\begin{aligned} &\text{sequence } e^{(k)} \text{ is infinite and bounded, or} \\ &\text{there exists such } m \geq k \text{ that sequence } a^{(m)} \text{ is unbounded.} \end{aligned} \quad (2.13)$$

Otherwise Player  $\forall$  wins.

**Lemma 2.5.26** *For  $n < \omega$  and  $\eta \in \mathcal{N}_n$ ,  $\eta \in L_n$  if and only if Player  $\exists$  has a winning strategy in game  $G_n(\eta)$ .*

**Proof:**

( $\Rightarrow$ ) We show a winning strategy  $\sigma_n(\eta)$  for the existential player in  $G_n(\eta)$  for each  $\eta \in L_n$ . The strategy is constructed inductively. For  $n = 0$ , there is only one sequence in  $L_n$ , namely the sequence of empty vectors  $(\varepsilon)^\omega$ . There is only one play in the game on this sequence and players do not need to make any decisions during this play, because there are no components of vectors. Therefore, there is only one strategy of  $\exists$  in this case. We take the unique strategy as  $\sigma_0((\varepsilon)^\omega)$ .

In the inductive step, we assume that for each  $\theta \in L_{n-1}$  strategy  $\sigma_{n-1}(\theta)$  is constructed and we construct strategy  $\sigma_n(\eta)$  for a given  $\eta \in L_n$ . The strategy uses a register  $m_n^\forall$ , which, in each moment, stores the greatest value of the first component selected by Player  $\forall$  so far (initially value  $-1$  is stored). Recall that for  $n > 0$ ,  $\eta \in L_n$  if and only if there exist infinitely many  $m \in \mathbb{N}$  such that

$$\eta \upharpoonright_{=m} \text{ is infinite and } \pi_1(\eta \upharpoonright_{=m}) \in L_{n-1} \quad (2.14)$$

Observe, that register  $m_n^\forall$  increases its value when Player  $\forall$  selects the first component of a vector and the value of this component is greater than the value stored in the register so far. Let a play be at its beginning or in the moment when register  $m_n^\forall$  has just increased. Let  $i$  be the number of currently processed vector. Let  $m_n^\exists$  be the least  $m$  greater than  $m_n^\forall$ , for which condition (2.14) holds. Let  $\eta_{i:}$  be the suffix of  $\eta$  starting at position  $i$ . By prefix independence of  $L_{n-1}$ ,  $\pi_1(\eta_{i:} \upharpoonright_{=m_n^\exists}) \in L_{n-1}$ . Starting from this point (until the next increase of the register) strategy  $\sigma_n(\eta)$  selects the first component of each vector if and only if its value is  $m_n^\exists$  (note that in such a case Player  $\forall$  has not selected it because otherwise  $m_n^\forall \geq m_n^\exists$ , what is impossible by the definition of  $m_n^\exists$ ). On the remaining components of each vector, the strategy follows strategy  $\sigma_{n-1}(\pi_1(\eta_{i:} \upharpoonright_{=m_n^\exists}))$ .

The proof that the strategies are winning is inductive.

Induction basis: It is enough to note that the unique play on  $\eta \in L_0$  is winning for  $\exists$ .



Inductive step: Let  $n > 0$ . Note that if  $m_n^\forall$  increases infinitely many times during a play, then sequence  $a^{(n)}$  from the definition of the game is unbounded and Player  $\exists$  wins the play. Otherwise, the register stabilizes, so there exists the greatest value of the first components selected by Player  $\forall$  during the whole play. Let us denote this value by  $\widehat{m}_n^\forall$ . Let  $\widehat{m}_n^\exists$  be the least  $m$  greater than  $\widehat{m}_n^\forall$ , for which condition (2.14) holds. There exists such  $i$  that Player  $\exists$ , playing according to strategy  $\sigma_n(\eta)$ , starting from vector  $i$  till the end of the play have selected exactly those first components that have value  $\widehat{m}_n^\exists$  and played according to  $\sigma_{n-1} \left( \pi_1 \left( \eta_i \upharpoonright_{=\widehat{m}_n^\exists} \right) \right)$  on the remaining components. Note that in such a play  $e^{(n)}$  is clearly infinite and bounded, and that winning condition (2.13) holds for the suffix of a play on  $\eta_i$  by the inductive assumption. By prefix independence of (2.13), winning condition also holds for the whole play.

( $\Leftarrow$ ) Now we show a winning strategy  $\sigma_n(\eta)$  for Player  $\forall$  in  $G_n(\eta)$  for each  $\eta \in \mathcal{N}_n \setminus L_n$ . Since there is no  $\eta \in \mathcal{N}_0 \setminus L_0$ , no strategy needs to be given for  $n = 0$ .

Note that, for  $n > 0$ , if  $\eta \in (\mathbb{N}^k)^{\leq \omega} \setminus L_n$ , then there exists natural number  $M$  such that for all  $m \geq M$ :

$$\eta \upharpoonright_{=m} \text{ is finite} \quad \text{or} \quad \pi_1(\eta \upharpoonright_{=m}) \notin L_{n-1} \quad (2.15)$$

Let  $M_n^\forall(\eta)$  be the least such  $M$ . In particular,  $M_n^\forall(\varepsilon) = 0$ .

The strategy uses two types of registers to make decisions: registers  $m_n^\forall, m_{n-1}^\forall, \dots, m_1^\forall$ , and registers  $m_n^\exists, m_{n-1}^\exists, \dots, m_1^\exists$ . Registers  $m_n^\forall$  and  $m_n^\exists$  are used to determine behavior on the first components of vectors, registers  $m_{n-1}^\forall, m_{n-1}^\exists$  are used to determine behavior on the second components, and so on. For each  $k$ , register  $m_k^\exists$ , in each moment, stores the largest value of the  $(n - k + 1)$ st component selected by Player  $\exists$  (value  $-1$  is stored initially). Register  $m_n^\forall$  initially stores value  $M_n^\forall(\eta)$ . For  $k < n$ , register  $m_k^\forall$  initially stores value  $0$ . The value stored by register  $m_n^\forall$  never changes. Each time when register  $m_k^\exists$  increases (Player  $\exists$  selects  $(n - k + 1)$ th component with value greater than the value stored in the register so far), the values of all registers  $m_l^\forall$  for  $l < k$  are recalculated. Now we describe how the recalculation procedure looks like.

For each  $0 < k \leq n$ , we define sequence  $\eta^{(k)} \in (\mathbb{N}^k)^{\leq \omega}$  as follows.

$$\begin{aligned} \eta^{(n)} &:= \eta \\ \eta^{(k)} &:= \pi_1 \left( \eta^{(k+1)} \upharpoonright_{\in \{m_{k+1}^\forall, m_{k+1}^\forall+1, \dots, m_{k+1}^\exists\}} \right) \text{ for } k < n \end{aligned}$$

We maintain the invariant that for each  $0 < k \leq n$  and for each  $m \geq m_k^\forall$ :

$$\eta^{(k)} \upharpoonright_{=m} \text{ is finite} \quad \text{or} \quad \pi_1(\eta^{(k)} \upharpoonright_{=m}) \notin L_{k-1} \quad (2.16)$$

For  $k = n$  the invariant is initially satisfied, by (2.15). Initially for each  $k$ ,  $m_k^\forall > m_k^\exists$ , therefore, for  $k < n$ ,  $\eta^{(k)} = \varepsilon$  and the invariant is satisfied.

Let a play be in the moment when register  $m_k^\exists$  has just increased. Let  $i$  be the number of currently processed vector. By invariant (2.16), for every  $m \in \{m_k^\forall, m_k^\forall + 1, \dots, m_k^\exists\}$ ,  $\eta^{(k)} \upharpoonright_{=m}$  is finite or  $\pi_1(\eta^{(k)} \upharpoonright_{=m}) \notin L_{k-1}$ . Therefore, by the pigeonhole property of  $L_{k-1}$ :

$$\eta^{(k)} \upharpoonright_{\in \{m_k^\forall, m_k^\forall + 1, \dots, m_k^\exists\}} \text{ is finite} \quad \text{or} \quad \eta^{(k-1)} = \pi_1 \left( \eta^{(k)} \upharpoonright_{\in \{m_k^\forall, m_k^\forall + 1, \dots, m_k^\exists\}} \right) \notin L_{k-1}$$

Therefore, we can set  $m_{k-1}^\forall := M_{k-1}^\forall(\eta^{(k-1)})$ . By the definition of  $M_{k-1}^\forall$ , the invariant for  $k-1$  is satisfied. The change of  $m_{k-1}^\forall$  causes the change of  $\eta^{(k-2)}$ , by the definition. Therefore, in order to maintain the invariant, we set  $m_{k-2}^\forall := M_{k-2}^\forall(\eta^{(k-2)})$ . Further, we set  $m_{k-3}^\forall, m_{k-4}^\forall, \dots, m_1^\forall$  analogously.

Strategy  $\sigma_n(\eta)$  tells Player  $\forall$  to select all the first components with value less than  $m_n^\forall$ . If Player  $\exists$  selects the first component of a vector, the strategy tells Player  $\forall$  to select the second component of the vector if and only if its value is less than  $m_{n-1}^\forall$ . In general, if  $k$  first components of a given vector are selected by Player  $\exists$ , then Player  $\forall$ , playing according to the strategy, selects  $(k+1)$ st component if and only if its value is less than  $m_{n-k}^\forall$ .

Now we prove that the strategy is winning.

Let  $\eta \in \mathcal{N}_n \setminus L_n$ . Note that there is no  $\eta \in \mathcal{N}_0 \setminus L_0$ , so we may assume that  $n > 0$ . While following strategy  $\sigma_n(\eta)$ , Player  $\forall$  selects the first coordinates with values less than  $m_n^\forall$ , which does not change during a play, so sequence  $a^{(n)}$  is bounded. If sequence  $e^{(n)}$  is unbounded or finite then Player  $\forall$  wins. Otherwise the value of register  $m_n^\exists$  increases finitely many times. Let  $i_n$  be the number of vector on which  $m_n^\exists$  increases for the last time, and let  $\widehat{m}_n^\exists$  be the maximal value of  $m_n^\exists$ . Note that, according to the rules of the game and by the construction of register  $m_n^\exists$ , starting from  $i_n$ th vector, Player  $\exists$  selects only the first components of vectors with values between  $m_n^\forall$  and  $\widehat{m}_n^\exists$  (potentially selects not all of them). Starting from  $i_n$ th vector,  $m_{n-1}^\forall = M_{n-1}^\forall(\eta^{(n-1)})$ , and Player  $\forall$  selects second components with value less than  $m_{n-1}^\forall$ . Therefore sequence  $a^{(n-1)}$  is bounded. If sequence  $e^{(n-1)}$  is unbounded or finite then Player  $\forall$  wins. Otherwise,  $m_{n-1}^\exists$  stabilizes at some vector and, repeating the above reasoning, we get that  $m_{n-2}^\forall$  also stabilizes at that point. We repeat the reasoning for the remaining coordinates and obtain that either Player  $\forall$  wins because Player  $\exists$  makes one of the registers  $m_n^\exists, m_{n-1}^\exists, \dots, m_1^\exists$  to increase infinitely many times, or all the registers  $m_n^\exists, m_{n-1}^\exists, \dots, m_1^\exists$  and  $m_n^\forall, m_{n-1}^\forall, \dots, m_1^\forall$  stabilize at some point with values that we denote  $\widehat{m}_k^\exists$  and  $\widehat{m}_k^\forall$ , respectively. In the latter case, also all sequences  $\eta^{(k)}$  do not change from some point. By invariant (2.16), for each  $\widehat{m}_1^\forall \leq m \leq \widehat{m}_1^\exists$ ,  $\eta^{(1)} \upharpoonright_{=m}$

is finite or  $\pi_{\exists}(\eta^{(1)} \upharpoonright_{=m}) \notin L_0$ . Since, by the definition of  $L_0$ , if  $\eta^{(1)} \upharpoonright_{=m}$  is infinite then  $\pi_{\exists}(\eta^{(1)} \upharpoonright_{=m}) \in L_0$ , we get that  $\eta^{(1)} \upharpoonright_{=m}$  is finite in this case. Hence also  $\eta^{(1)} \upharpoonright_{\in\{\widehat{m_1^{\forall}}, \widehat{m_1^{\forall}+1}, \dots, \widehat{m_1^{\exists}}\}}$  is finite. Now observe that Player  $\exists$  selects, from some moment, only those last components of vectors, that belong to  $\eta^{(1)} \upharpoonright_{\in\{\widehat{m_1^{\forall}}, \widehat{m_1^{\forall}+1}, \dots, \widehat{m_1^{\exists}}\}}$ . Therefore, finiteness of sequence  $\eta^{(1)} \upharpoonright_{\in\{\widehat{m_1^{\forall}}, \widehat{m_1^{\forall}+1}, \dots, \widehat{m_1^{\exists}}\}}$  implies finiteness of sequence  $e^{(1)}$  and Player  $\forall$  wins the play.

We conclude that all plays played by Player  $\forall$  according to the strategy are won by  $\forall$ .  $\square$

**Proof (of Theorem 2.5.25):**

It is possible to construct an alternating  $\omega$ BS-automaton recognizing exactly the language  $W_n(L_n)$ , for any given  $n$ . However, to avoid technical inconveniences, we construct an automaton  $\mathcal{A}_n$  for which we only require that it accepts a word  $W_n(\eta)$  if and only if  $\eta \in L_n$ . The latter is sufficient for the proof of hardness. The automaton implements game  $G_n(\eta)$ .

The sequences  $a^{(n)}, e^{(n)}, a^{(n-1)}, e^{(n-1)}, \dots, a^{(1)}, e^{(1)}$  are realized by counters. Selecting a given component by a given player in the game is done by incrementing the corresponding counter on each letter of the block of  $a$ s corresponding to the component and resetting it after this block (when letter  $b$  is read). Additionally all  $a^{(i)}$  counters are reset on each letter  $b$ .

The acceptance condition follows exactly the winning condition (2.13) and uses B-conditions. Recall that B-condition implicitly requires infinite number of resets. This is why we reset counters  $a^{(i)}$  on each letter  $b$ , regardless whether Player  $\forall$  have selected the preceding block of  $a$ s or not. This allows Player  $\forall$  to select a finite set without losing a play.  $\square$

## 2.6 Remarks and Open Questions

### Automata

As far as the author knows, the following was never observed before the paper coauthored by him [HMT10].

**Corollary 2.6.1 (of Theorem 2.5.25)** *Alternating  $\omega$ BS-automata are more expressive than boolean combinations of nondeterministic  $\omega$ BS-automata.*

While the topological complexity of  $\omega$ B-,  $\omega$ S-, and  $\omega$ BS-regular languages is solved by Theorem 2.5.10 and Theorem 2.5.13, there is one question on the automata side that we leave open. There is a huge gap between the

upper and the lower bound for the complexity of alternating  $\omega$ BS-automata that we provide. On one hand we know that they inhabit at least all finite levels of the Borel hierarchy. On the other hand, by Remark 1.5.4, each language recognized by such an automaton is in  $\Sigma_2^1$ . The gap is significant, however, the importance of the model has decreased, as we know that it is not sufficient to cover the whole expressive power of MSO+U logic.

To the best authors knowledge, it is open whether emptiness problem is decidable for alternating  $\omega$ BS-automata. If it occurs to be undecidable, the topological complexity of this model may occur even less interesting.

Alternating  $\omega$ BS-automata may regain the interest of researchers if it is proven that, as it is widely believed, they recognize a subclass of MSO+U-definable languages.

## MSO+U

While in this thesis we concentrate on calculating the topological complexity of a class of languages, what was done for MSO+U in Theorem 2.4.17, another question that could be asked from the descriptive set theoretical perspective is what levels of the Borel hierarchy are inhabited by sets defined by the logic (see Figure 1).

In general, each language theoretic class is doomed to have a gap property similar to the one observed for deterministic tree languages in paper by Damian Niwiński and Igor Walukiewicz [NW03]. To observe this, first note that since all sets in a language theoretic class are described by finitely represented objects, like formulas or automata, the class is countable. On the other hand, there are  $\omega_1$  levels of the Borel hierarchy. Moreover, since the cofinality of  $\omega_1$  is  $\omega_1$ , for each language theoretic class there is a countable upper bound on the level of the Borel hierarchy inhabited by sets of the class. For each given class the question can be asked what is the lowest such bound.

Proposition 2.5.23 gives a partial answer to this question, by providing languages definable in MSO+U on each of the first  $\omega$  levels of the Borel hierarchy. A single step behind  $\omega$  is given by Skrzypczak in his paper with the author of this thesis (see [HS12, Example 6.2]). However, according to the author's best knowledge, no countable upper bound for the occupied levels is known.

## Chapter 3

# Infinite Trees – Within Regularity

The notion of regularity extends to the domain of trees. A natural modification of the model of parity automata and MSO logic with signature suitable for tree-models, both lead to the same class of languages called **regular languages of infinite trees** (or shorter: **regular tree languages**). Compared to regular languages of  $\omega$ -words, the class is much more complex. Topologically, it contains non-Borel languages, and even languages beyond the first level of the projective hierarchy (see Section 3.3). There are several widely studied and well motivated decidability questions that are still unanswered for this class. A notable example is the problem of calculating Rabin-Mostowski index of a language, as defined in Section 3.4. The situation motivates analysis of subclasses of the class of regular tree languages, for which some of the complexities would decrease and some of the questions would be easier to answer. Some of the subclasses come from restrictions of the automata model in consideration—and those are the ones that we concentrate on in this thesis.

In this chapter we give a lower bound for the topological complexity of the class of languages recognized by unambiguous parity tree automata. We show, in particular, that it reaches beyond the sigma-algebra generated by  $\Sigma_1^1$  and  $\Pi_1^1$  sets. Recall that languages recognized by deterministic automata are all in  $\Pi_1^1$ . The research was initiated in the paper by the author of this thesis [Hum12], but the majority of the results presented are not published yet. The results are preceded by a short introduction to regular tree languages including the topological complexity of the class and some of its subclasses. We also introduce Rabin-Mostowski index hierarchy as it is a complexity measure that we compare to the topological complexity.

### 3.1 Parametrized Tree Languages

Although we are mainly interested in full binary trees, some constructions use trees with leaves. Therefore, we introduce the notation  $T_{A,X}$  for labeled trees, in which all inner nodes are of arity 2 (have two successors) and have labels from  $A$ , and all leaves are labeled with letters from  $X$  (sets  $X$  and  $A$  do not need to be disjoint). Trees from  $T_{A,X}$  are sometimes called **contexts**, languages of such trees are called **parametrized languages**, and letters from  $X$  are **parameters** (or **variables**). Leaves of contexts are often called **holes**, since they are designed to be filled with trees or other contexts.

Now we define **language substitution**, that is one of possible extensions of concatenation of word languages to trees.

Let  $L \subseteq T_{A,X}$  be a parametrized language and, for  $B, Y$  being arbitrary alphabets, let  $\alpha : X \multimap \mathbb{P}(T_{B,Y})$  be a partial function. We define  $L[\alpha]$  to be the language of all trees that come from substituting each occurrence of any variable  $x \in \text{dom}(\alpha)$  with some tree from  $\alpha(x)$  in some tree from  $L$ . More formally:

$$t \in L[\alpha] \iff \begin{array}{l} \exists t' \in L \forall v \in \{l,r\}^* \\ \text{if } v \text{ is an inner node in } t' \text{ then } t(v) = t'(v) \\ \text{if } v \text{ is a leaf in } t' \text{ then} \\ \quad \text{if } t'(v) \in \text{dom}(\alpha) \text{ then } t_v \in \alpha(t'(v)) \\ \quad \text{otherwise } v \text{ is a leaf of } t \text{ and } t(v) = t'(v). \end{array}$$

Note that if  $\text{dom}(\alpha) = X$  and  $Y = \emptyset$  then  $L[\alpha] \subseteq T_{A \cup B}$  is a language of full binary trees.

We sometimes write  $L(x_1, x_2, \dots, x_k)$  for  $L \subseteq T_{A, \{x_1, x_2, \dots, x_k\}}$  to indicate the set of all parameters. For  $\alpha = \{x_1 \mapsto L_1, x_2 \mapsto L_2, \dots, x_k \mapsto L_k\}$ , we sometimes write

$$L[x_1 \mapsto L_1, x_2 \mapsto L_2, \dots, x_k \mapsto L_k], \quad \text{or even} \quad L(L_1, L_2, \dots, L_k)$$

instead of  $L[\alpha]$ .

Note that tree substitution is monotonic, in the following sense:

**Remark 3.1.1** *If  $\text{dom}(\alpha) = \text{dom}(\beta) =: Z$ ,  $\forall x \in Z \alpha(x) \subseteq \beta(x)$ , and  $L \subseteq L'$ , then  $L[\alpha] \subseteq L'[\beta]$ .*

### 3.2 Parity Tree Automata

Parity tree automata constitute one of the standard models of automata for regular languages of infinite binary trees. Let us start with a more general model.

**Definition 3.2.1 (Parametrized Automaton)** A *parametrized parity tree automaton* (or simply *parametrized automaton*) is a tuple  $\mathcal{A} = \langle A, Q, X, \delta, I, p \rangle$ , where:

- $A$  is a finite alphabet,
- $Q$  is a finite set of states,
- $X \subseteq Q$  is a set of parameters (or parameter-states),
- $\delta \subseteq (Q \setminus X) \times A \times Q \times Q$  is a transition relation,
- $I \subseteq Q$  is a set of initial states,
- $p : (Q \setminus X) \rightarrow \mathbb{N}$  is a priority function.

For  $X = \{x_1, x_2, \dots, x_k\}$ , we sometimes use the term-like notation  $\mathcal{A}(x_1, x_2, \dots, x_k)$  to explicitly enumerate the set of all parameters and to set their order. To talk about runs and, especially, the acceptance we need to instantiate the automaton.

Let, then,  $X = \{x_1, x_2, \dots, x_k\}$  and let  $L_1, L_2, \dots, L_k \subseteq T_{A,B}$  be languages, for some alphabet  $B$ . We note by  $\mathcal{A}(L_1, L_2, \dots, L_k)$  the instantiation of automaton  $\mathcal{A}$  with languages  $L_1, L_2, \dots, L_k$ . A **run** of instantiated automaton  $\mathcal{A}(L_1, L_2, \dots, L_k)$  on a tree (or context)  $t \in T_{A,B}$  is a labeled context  $\rho \in T_{Q,X}$  such that:

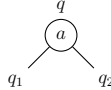
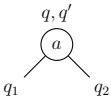
1.  $\text{dom}(\rho) \subseteq \text{dom}(t)$ ,
2.  $\varepsilon \in \text{dom}(\rho)$ ,
3.  $\rho(\varepsilon) \in I$ ,
4.  $\rho(v) \in Q \setminus X \implies vl, vr \in \text{dom}(\rho) \wedge (\rho(v), t(v), \rho(vl), \rho(vr)) \in \delta$ ,
5.  $\rho(v) \in X \implies vl \notin \text{dom}(\rho) \wedge vr \notin \text{dom}(\rho)$ .

A run  $\rho$  is **accepting** if:

1.  $\forall_{\alpha \in \{l,r\}^\omega} (\forall_n \alpha \upharpoonright_n \in \text{dom}(\rho)) \implies \rho(\alpha)$  satisfies the parity condition,
2.  $\forall_{v \in \{l,r\}^*} \bigwedge_{i=1}^k (\rho(v)=x_i \implies t_v \in L_i)$ .

The **language recognized** by instantiation  $\mathcal{A}(L_1, L_2, \dots, L_k)$  of the parametrized automaton is the set of those contexts  $t \in T_{A,B}$  on which there is an accepting run of  $\mathcal{A}(L_1, L_2, \dots, L_k)$  and is denoted by  $L(\mathcal{A}(L_1, L_2, \dots, L_k))$ .

**Definition 3.2.2 (Parity Automaton)** A parametrized tree automaton with the empty set of parameters is called a **parity tree automaton** and denoted  $\mathcal{A} = \langle A, Q, \delta, I, p \rangle$  (the empty set of parameters is omitted). Such an automaton does not need an instantiation (or there is the unique instantiation). The notions of a **run**, an **accepting run** and the **language recognized** by a parity automaton (denoted  $L(\mathcal{A})$ ) are inherited from the respective definitions for an instantiation of a parametrized automaton. Note that a run of a parity automaton is always a full tree, hence only points (3), (4) of the above definition of a run and point (1) of the definition of accepting run are relevant in this case.

In the sequel we denote a transition  $(q, a, q_1, q_2) \in \delta$  as . We also use a compressed notation, e.g. if  $(q, a, q_1, q_2), (q', a, q_1, q_2) \in \delta$  we write .

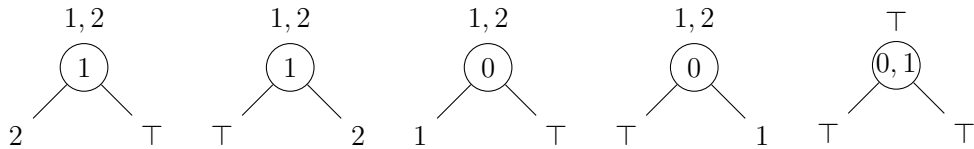
We use the notation  $\mathcal{A}_q$  for the automaton  $\mathcal{A}$  modified in such a way, that  $q$  becomes the only initial state. We also write  $\mathcal{A}_I$  if we want to put set  $I$  as initial states in  $\mathcal{A}$ .

**Definition 3.2.3 (Regular Tree Language)** A language  $L \subseteq T_A$  of full infinite binary trees is called **regular** if it is recognized by a parity tree automaton.

**Example 3.2.4** Consider the following language of full binary trees over alphabet  $\{0, 1\}$ :

$$EB = \{t \in T_{\{0,1\}} : \text{there exists a path in } t \text{ with infinitely many labels } 1\}$$

Note that  $EB$  is recognized by the following nondeterministic parity tree automaton  $\mathcal{EB} = \langle \{1, 0\}, \{1, 2, \top\}, \delta, \{1\}, p \rangle$ , where  $p = \{1 \mapsto 1, 2 \mapsto 2, \top \mapsto 2\}$  and  $\delta$  contains the following transitions:



Therefore,  $EB$  is regular.

Note that a parametrized automaton can be instantiated with arbitrary languages, not necessarily regular. We use this fact e.g. in Lemma 3.8.36



below, that is proven with no regularity assumption. If, on the other hand, languages  $L_1, L_2, \dots, L_k$  are regular, a parity automaton recognizing language  $L(\mathcal{A}(L_1, L_2, \dots, L_k))$  can be constructed using the following notion.

**Definition 3.2.5 (Composition of Automata)**

Let  $\mathcal{A} = \langle A, Q, X, \delta, I, p \rangle$  be a parametrized automaton. Let  $X = \{x_1, x_2, \dots, x_k\}$  for some  $k < \omega$ . Let  $\mathcal{L}_1 = \langle A, Q_1, \delta_1, I_1, p_1 \rangle$ ,  $\mathcal{L}_2 = \langle A, Q_2, \delta_2, I_2, p_2 \rangle$ ,  $\dots$ ,  $\mathcal{L}_k = \langle A, Q_k, \delta_k, I_k, p_k \rangle$  be parity tree automata. Assume, without loss of generality, that  $Q, Q_1, Q_2, \dots, Q_k$  are pairwise disjoint. Composed parity automaton  $\mathcal{A}(\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_k) := \langle A, Q', \delta', I', p' \rangle$  is defined as follows:

$$\begin{aligned} Q' &= Q_1 \cup Q_2 \cup \dots \cup Q_k \cup Q \setminus X \\ \delta' &= \delta_1 \cup \delta_2 \cup \dots \cup \delta_k \cup \{(q, a, q_1, q_2) : \\ &\quad \exists_{r_1, r_2} (q, a, r_1, r_2) \in \delta \\ &\quad \wedge ((r_1 \in Q \setminus X \wedge q_1 = r_1) \vee \bigvee_i (r_1 = x_i \wedge q_1 \in I_i)) \\ &\quad \wedge ((r_2 \in Q \setminus X \wedge q_2 = r_2) \vee \bigvee_i (r_2 = x_i \wedge q_2 \in I_i))\} \\ I' &= \bigcup \{I_i : x_i \in I\} \cup I \setminus X \\ p' &= p_1 \cup p_2 \cup \dots \cup p_k \cup p|_{Q \setminus X} \end{aligned}$$

**Remark 3.2.6**  $L(\mathcal{A}(\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_k)) = L(\mathcal{A}(L(\mathcal{L}_1), L(\mathcal{L}_2), \dots, L(\mathcal{L}_k)))$ .

We define subclasses of the class of nondeterministic parity automata. As will be argued later, contrary to the case of  $\omega$ -words, none of the subclasses is able to recognize all regular tree languages.

**Definition 3.2.7 (Deterministic Automaton)** A nondeterministic parity tree automaton  $\mathcal{A} = \langle A, Q, \delta, I, p \rangle$  is **deterministic** if  $|I| = 1$  and transition relation is a function  $\delta : (Q \times A) \rightarrow (Q \times Q)$ .

**Definition 3.2.8 (Unambiguous Automaton)** A nondeterministic parity tree automaton is **unambiguous** if it has at most one accepting run on each tree.

We note that, despite the property of a nondeterministic automaton to be unambiguous is not syntactic, it can be easily decided, by checking emptiness of an appropriately constructed product automaton, as closer discussed in [Col12, page 5].

The notion below is, in essence, a generalization of a parity tree automaton. However, it defines the same class of languages.

**Definition 3.2.9 (Alternating Automaton)** An *alternating parity tree automaton*<sup>1</sup> is a tuple  $\mathcal{A} = \langle A, Q_\exists, Q_\forall, \delta, q_I, p \rangle$ , where  $A$  is a finite alphabet,  $Q_\exists$  is a set of states of Player  $\exists$ ,  $Q_\forall$  is a set of states of Player  $\forall$ ,  $Q_\exists \cap Q_\forall = \emptyset$ , for  $Q := Q_\exists \cup Q_\forall$ ,  $\delta \subseteq Q \times A \times \{l, r, \varepsilon\} \times Q$  is a transition relation,  $q_I \in Q$  is an initial state and  $p : Q \rightarrow \mathbb{N}$  is a priority function.

For automaton  $\mathcal{A}$  and a tree  $t \in T_A$ , we define a parity game. The game is played by the two players  $\exists$  and  $\forall$ . A play in the game starts in the root of  $t$  in the initial state  $q_I$  of  $\mathcal{A}$  and is played down the tree along some path. If the current state belongs to  $Q_\exists$  then Player  $\exists$  moves, otherwise Player  $\forall$  moves. A move is an application of a transition, i.e. if the play is in a node  $v$  of the tree in state  $q \in Q_\Gamma$  for  $\Gamma \in \{\exists, \forall\}$  then Player  $\Gamma$  choses a transition  $(q, t(v), d, q') \in \delta$  for some  $d \in \{l, r, \varepsilon\}$  and  $q' \in Q$ . As a result the position of the play changes to  $vd$  and the state changes to  $q'$ . The player that cannot make a move loses immediately. If a play is infinite, then Player  $\exists$  wins if the sequence of priorities of consecutive states in the play satisfies the (word) parity condition; otherwise Player  $\forall$  wins.

A tree  $t \in T_A$  is **accepted** by the automaton if Player  $\exists$  has a winning strategy in the above game on  $t$ . Language **recognized** by  $\mathcal{A}$ , denoted by  $L(\mathcal{A})$ , consists of all trees accepted by  $\mathcal{A}$ .

**Theorem 3.2.10 ([MS84])** Alternating parity tree automata recognize exactly regular tree languages.

To complete the picture we recall the definition of a widely studied subclass of the alternating parity tree automata.

**Definition 3.2.11 (Weak Automaton)** A parity tree automaton  $\mathcal{A} = \langle A, Q, \delta, I, p \rangle$  is **weak** if for each  $(q, a, q_1, q_2) \in \delta$ ,  $p(q_1), p(q_2) \geq p(q)$ .

Although the definition of weakness makes sense for any of nondeterministic, deterministic, unambiguous and alternating types of automata, it is mostly studied for alternating ones. Therefore, in the sequel, **weak automaton** means “alternating weak automaton”, unless otherwise noted. We note that the weak variant of each of the mentioned types of automata has less expressive power than the unrestricted variant.

---

<sup>1</sup>Among many equivalent definitions of the notion we have selected the one presented e.g. in [ADMN08].

### 3.3 Complexity of Regular Tree Languages

**Theorem 3.3.1 (Rabin [Rab69])** *The complement of a regular tree language is a regular tree language<sup>2</sup>.*

Rabin's Theorem implies the following upper topological complexity bound for the class of regular tree languages.

**Fact 3.3.2** *Each regular tree language is in  $\Delta_2^1$ .*

Before we proceed with a proof, let us introduce the notation for the set of accepting runs of a parity tree automaton.

**Definition 3.3.3** *The following set  $T_{(\iota, \kappa)} \subseteq T_{\{\iota, \iota+1, \dots, \kappa\}}$  is called  $(\iota, \kappa)$ -**parity tree condition**.*

$$T_{(\iota, \kappa)} := \{t \in T_{\{\iota, \iota+1, \dots, \kappa\}} : \forall_{\alpha \in \{l, r\}^\omega} t(\alpha) \text{ satisfies the (word) parity condition}\}$$

**Lemma 3.3.4** *For each  $\iota, \kappa < \omega$ ,  $(\iota, \kappa)$ -parity tree condition is in  $\Pi_1^1$ .*

**Proof:**

Recall that, for a bounded set of priorities, the (word) parity condition is in  $\mathcal{BC}_2^0$  (see Theorem 2.2.1). Therefore,  $T_{(\iota, \kappa)} \in \Pi_1^1$ , as a co-projection (i.e. the complement of the projection) of the following Borel set:

$$\{(t, \alpha) \in T_{\{a, b\}} \times \{l, r\}^\omega : t(\alpha) \text{ does not satisfy the (word) parity condition}\}$$

□

**Proof (of Fact 3.3.2):**

By Lemma 3.3.4 and Remark 1.5.3, each regular tree language  $L$  is a  $\Sigma_2^1$ -set. By Rabin's Theorem, also the complement of  $L$  is in  $\Sigma_2^1$ , therefore,  $L \in \Delta_2^1$ . □

We note that this is not true that  $\Delta_2^1$  is the topological complexity of regular tree languages, because, as it was observed in the context of fixpoint logic by Bradfield [Bra03, Corollary 11] and also comes from stronger statements formulated in terms of tree automata i.a. by Gogacz, Michalewski, Mio, Skrzypczak [GMMS14] and by Finkel, Lacomte, Simonnet [FLS15], there are sets in  $\Delta_2^1$  that are not Wadge reducible to any regular tree language. However, the above upper bound is tight, as far as the projective hierarchy is concerned.

---

<sup>2</sup>The original Rabin's proof concerns automata with Muller acceptance condition, however, tree automata with Muller and parity condition have the same expressive power (see e.g. [Tho97, Theorem 6.1]).

**Fact 3.3.5** *There are regular tree languages outside topological class  $\sigma(\Sigma_1^1)$ .*

Here we only recall examples of regular languages outside  $\Pi_1^1$  and  $\Sigma_1^1$ . Based on these examples one can build an example proving Fact 3.3.5, but since a similar (but more general) construction is the subject of Section 3.8, we do not describe it here. The following was given as an example in an article by Niwiński and Walukiewicz (see [NW03], the Example in Section 4 and Lemma 9).

**Lemma 3.3.6** ([NW03]) *Language  $T_{(0,1)}$  is  $\Pi_1^1$ -complete.*

Note that for  $EB$  defined as in Example 3.2.4,  $EB = \overline{T_{(0,1)}}$ .

**Corollary 3.3.7** (of Lemma 3.3.6) *Language  $EB$  is  $\Sigma_1^1$ -complete.*

Similarly as in the case of word automata, tree automata were used to show decidability of MSO logic (the signature contained two successors in this case) [Rab69]. As an intermediate step decidability of the emptiness problem for automata was shown. The complexity of this problem is briefly discussed in Section 3.4.

## 3.4 Rabin-Mostowski Index

**Definition 3.4.1 (Index of a Graph)** *An **index** is a set of pairs of natural numbers. Let  $G$  be a directed graph with nodes labeled with natural numbers. We say that  $G$  is of index  $\{(\iota_1, \kappa_1), (\iota_2, \kappa_2), \dots, (\iota_k, \kappa_k)\}$ , if for each strongly connected component of  $G$ , the labels of its nodes fall into interval  $[\iota_j, \kappa_j]$  for some  $1 \leq j \leq k$ .*

**Definition 3.4.2 (Graph of an Automaton)**

*Let  $\mathcal{A}$  be a (parametrized) nondeterministic parity tree automaton. The **graph of automaton  $\mathcal{A}$** , denoted  $\text{Graph}(\mathcal{A})$ , is a directed graph whose vertices are the non-parameter states of  $\mathcal{A}$ , node labels are the priorities of states, and there is an edge from a vertex  $p$  to a vertex  $q$ , if there is a transition  $(p, a, q_1, q_2)$  in  $\mathcal{A}$  for some letter  $a$  and for  $q \in \{q_1, q_2\}$ .*

*For  $\mathcal{A}$  being an alternating parity tree automaton, the graph  $\text{Graph}(\mathcal{A})$  is defined similarly with the exception that there is an edge from a vertex  $p$  to a vertex  $q$ , if there is a transition  $(p, a, d, q)$  in  $\mathcal{A}$  for some letter  $a$  and  $d \in \{l, r, \varepsilon\}$ .*

**Definition 3.4.3 (Index of an Automaton)** *A parity tree automaton  $\mathcal{A}$  is of Rabin-Mostowski index  $I$  if  $\text{Graph}(\mathcal{A})$  is of index  $I$ .*

For convenience, we use a simplified notation  $(\iota, \kappa)$  for index  $\{(\iota, \kappa)\}$ .

**Definition 3.4.4 (Index Classes of Languages)** *Index class  $\mathcal{L}^{ndet}(I)$  (respectively  $\mathcal{L}^{det}(I)$ ,  $\mathcal{L}^{uamb}(I)$ ,  $\mathcal{L}^{alt}(I)$ ,  $\mathcal{L}^{weak}(I)$ ) is the class containing all languages recognized by nondeterministic (respectively deterministic, unambiguous, alternating, weak alternating) parity tree automata of index  $I$ .*

For convenience we write  $\mathcal{L}^\Gamma((\iota_1, \kappa_1), (\iota_2, \kappa_2), \dots, (\iota_k, \kappa_k))$ , omitting curly brackets, but remembering that an argument of the term  $\mathcal{L}^\Gamma()$  is actually a set.

We want to observe that it is enough to consider index classes of the form  $\mathcal{L}^\Gamma((0, \kappa))$ ,  $\mathcal{L}^\Gamma((1, \kappa + 1))$ ,  $\mathcal{L}^\Gamma((0, \kappa), (1, \kappa + 1))$  for  $\kappa < \omega$  and  $\Gamma \in \{ndet, det, uamb, alt, weak\}$ . For this we prove the following.

**Fact 3.4.5**

For any  $\Gamma \in \{ndet, det, uamb, alt, weak\}$  and any index  $\{I_1, I_2, \dots, I_k\}$ :

1.  $\mathcal{L}^\Gamma((\iota, \kappa), I_1, I_2, \dots, I_k) = \mathcal{L}^\Gamma((0, \kappa - \iota), I_1, I_2, \dots, I_k)$ , if  $\iota$  is even,
2.  $\mathcal{L}^\Gamma((\iota, \kappa), I_1, I_2, \dots, I_k) = \mathcal{L}^\Gamma((1, \kappa - \iota + 1), I_1, I_2, \dots, I_k)$ , if  $\iota$  is odd,
3.  $\mathcal{L}^\Gamma((\iota, \kappa), I_1, I_2, \dots, I_k) \subseteq \mathcal{L}^\Gamma((\iota', \kappa'), I_1, I_2, \dots, I_k)$ , if  $\kappa - \iota < \kappa' - \iota'$ ,
4.  $\mathcal{L}^\Gamma((\iota, \kappa), (\iota', \kappa'), I_1, I_2, \dots, I_k) = \mathcal{L}^\Gamma((\iota', \kappa'), I_1, I_2, \dots, I_k)$ , if  $\kappa - \iota < \kappa' - \iota'$ .

**Proof:**

Points (1), (2), (3), (4) come from an observation that if we shift the values of the priority function for all states of a given strongly connected component of the graph of an automaton by an even number the language recognized by the automaton does not change.  $\square$

**Remark 3.4.6** Let  $\mathcal{A}$  be a parametrized automaton of index  $I$ , with parameters  $x_1, x_2, \dots, x_k$ . Let  $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_k$  be parity automata of indexes  $I^{(1)}, I^{(2)}, \dots, I^{(k)}$ , respectively. Composed parity automaton  $\mathcal{A}(\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_k)$  is of index  $I \cup I^{(1)} \cup \dots \cup I^{(k)}$ .

Let us additionally define the following index classes of languages:

$$\Delta_\kappa^\Gamma = \mathcal{L}^\Gamma((0, \kappa - 1)) \cap \mathcal{L}^\Gamma((1, \kappa)) \quad (3.1)$$

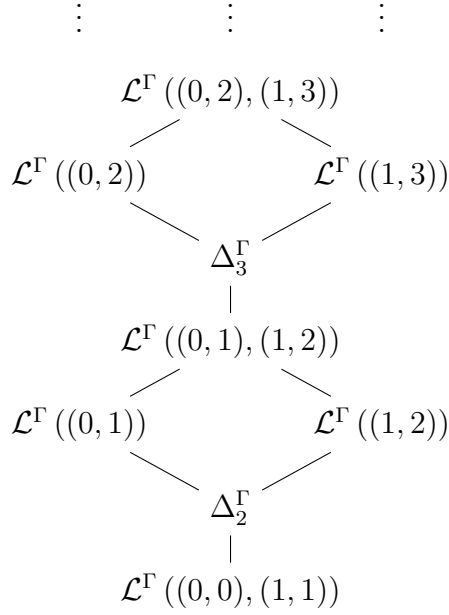


Figure 3.1: The Rabin-Mostowski index hierarchy (for  $\Gamma \in \{ndet, det, uamb, alt, weak\}$ ).

Index classes of languages constitute **Index Hierarchy**. Depending on type of index considered we have Nondeterministic Index Hierarchy, Alternating Index Hierarchy, etc. The inclusions between classes of each of the hierarchies are depicted in Figure 3.1. An important question that is asked about each of the hierarchies is whether the inclusions are strict.

The following was proven for the nondeterministic hierarchy by Niwiński [Niw86], for the weak hierarchy by Mostowski [Mos91], for the alternating hierarchy first by Bradfield [Bra99] and then by Arnold [Arn99]. For the deterministic hierarchy it follows from the analogous result for  $\omega$ -words by Wagner [Wag79].

**Theorem 3.4.7 (strictness [Niw86, Mos91, Bra99, Arn99, Wag79])**  
*Nondeterministic, weak, alternating and deterministic index hierarchies are strict, namely for  $\kappa > 0$  and  $\Gamma \in \{ndet, det, alt, weak\}$ :*

$$\begin{aligned} \Delta_\kappa^\Gamma &\subsetneq \mathcal{L}^\Gamma((0, \kappa - 1)) \\ \Delta_\kappa^\Gamma &\subsetneq \mathcal{L}^\Gamma((1, \kappa)) \end{aligned}$$

**Theorem 3.4.8 (Santocanale, Arnold [SA05, Theorem 2.2])**

For  $\kappa \geq 2$ :

$$\mathcal{L}^{alt}((0, \kappa - 1), (1, \kappa)) \subsetneq \Delta_{\kappa+1}^{alt}$$

**Theorem 3.4.9 (Rabin [Rab70])**

$$\mathcal{L}^{alt}((0, 0), (1, 1)) = \Delta_2^{alt}$$

**Definition 3.4.10** The dual index to index  $(\iota, \kappa)$  for  $\iota \in \{0, 1\}$  is denoted by  $\overline{(\iota, \kappa)}$  and defined as

$$\begin{aligned} \overline{(0, \kappa)} &:= (1, \kappa + 1) \\ \overline{(1, \kappa)} &:= (0, \kappa - 1) \end{aligned}$$

We use the implied notation  $\mathcal{L}^\Gamma(\overline{(\iota, \kappa)})$  also for index classes.

**Fact 3.4.11** If a language is in index class  $\mathcal{L}^{alt}((\iota, \kappa))$  then its complement is in index class  $\mathcal{L}^{alt}(\overline{(\iota, \kappa)})$ .

**Proof:**

Since parity games are determined (see e.g. [EJ91, Corollary 4.3]), it is enough to switch players and shift priorities by one in an alternating automaton recognizing a given language.  $\square$

We recall that alternating index classes admit complete languages. For  $W_{(\iota, \kappa)} \in \mathcal{L}^{alt}((\iota, \kappa))$  being the **game tree languages** used by Bradfield [Bra99] and Arnold [Arn99] to witness strictness of the alternating hierarchy, we have:

**Fact 3.4.12 ([Arn99])** If  $L \subseteq \mathcal{L}^{alt}((\iota, \kappa))$  then  $L \leq_w W_{(\iota, \kappa)}$ .

The above fact, together with the strictness theorem (Theorem 3.4.7), makes the alternating index hierarchy another topological complexity measure (although index classes are not formally topological complexity classes in the sense of the definition from Section 1.2).

**Fact 3.4.13** For  $\kappa \geq 2$ , classes  $\mathcal{L}^{ndet}((0, \kappa - 1), (1, \kappa))$  and  $\Delta_{\kappa+1}^{ndet}$  are not closed under complements.

**Proof:**

The proof of Theorem 3.4.7 for nondeterministic index by Niwiński [Niw86] relied on the fact that language  $T_{(\iota, \kappa)}$  is recognized by a nondeterministic automaton of index  $(\iota, \kappa)$ , but not by any nondeterministic automaton of

index  $\overline{(\iota, \kappa)}$ . Note that  $\overline{T_{(\iota, \kappa)}}$  is recognized by a nondeterministic automaton of index  $(1, 2)$  that guesses the path not satisfying the parity condition and the maximal odd priority occurring infinitely many times on the path, and verifies the guess. Hence, for  $\kappa \geq 2$ , the language  $\overline{T_{(0, \kappa)}}$  is in  $\mathcal{L}^{ndet}((0, \kappa - 1), (1, \kappa))$  and in  $\Delta_{\kappa+1}^{ndet}$ , while  $T_{(0, \kappa)}$  is not in any of those classes.  $\square$

Fact 3.4.13 makes the interpretation of Figure 3.1 for Nondeterministic Index Hierarchy a bit nonstandard. Therefore, the index hierarchy in its full form, as in the Figure 3.1, is studied almost only for the alternating case.

### 3.4.1 Relations Between Hierarchies

Since an important thread of the thesis is a comparison of different complexity measures, we note some facts about relations between index hierarchies.

#### Deterministic Index versus Nondeterministic Index

By definition, nondeterministic index classes include corresponding deterministic classes. Since the strictness of deterministic hierarchy comes from the strictness of the corresponding hierarchy for  $\omega$ -words and since each  $\omega$ -regular language is recognized by a Büchi automaton on  $\omega$ -words, there are languages arbitrary high in the deterministic index hierarchy that are in  $\mathcal{L}^{ndet}((1, 2))$ . On the other hand, the languages  $T_{(\iota, \kappa)}$  used by Niwiński [Niw86] to witness the strictness of the nondeterministic hierarchy, are all deterministic, and have the same deterministic and nondeterministic index.

#### Unambiguous Index versus Nondeterministic Index

Since each deterministic class is included in the respective unambiguous class which in turn is included in the respective nondeterministic class, the example of  $T_{(\iota, \kappa)}$  languages can be also used to observe that for each index class there is a language that has the same nondeterministic and unambiguous index. The languages can also be used to observe the following.

**Remark 3.4.14** *The unambiguous index hierarchy is strict.*

However, there are examples of languages that have greater unambiguous than nondeterministic index. An example will be seen in Section 3.7.

#### Deterministic Index versus Alternating Index

**Fact 3.4.15** (see e.g. [NW03, Theorem 6]) *Each language recognized by a deterministic tree automaton is in  $\mathcal{L}^{alt}((0, 1))$ .*



## Nondeterministic Index versus Alternating Index

As it was recalled before, nondeterministic and alternating parity tree automata recognize the same class of languages. However, the index classes do not coincide. By definition, alternating index classes include corresponding nondeterministic classes. The example of languages  $T_{(\iota, \kappa)}$  shows that there are languages arbitrary high in the nondeterministic index hierarchy that are in  $\mathcal{L}^{alt}((0, 1))$  (see Fact 3.4.15). On the other hand, the languages  $W_{(\iota, \kappa)}$  witnessing strictness of the alternating hierarchy have the same nondeterministic and alternating index.

## Weak Index versus Alternating Index

**Theorem 3.4.16 (Arnold and Niwiński [AN92])** *Weak automata recognize exactly the languages that are in  $\mathcal{L}^{alt}((0, 0), (1, 1))$ .*

## Algorithmic Considerations

As it was already mentioned in the previous section, the emptiness problem of tree automata is decidable. Standard approach to solving this algorithmic problem is to construct a parity game out of an automaton and find out whether an existential player has a winning strategy in this game. Although solving parity games is known to be in  $NP \cap \text{co-}NP$  complexity class, and many researchers believe they can be solved in polynomial time, no polynomial algorithm is known yet. However, subexponential algorithms solving the emptiness problem considered the most efficient until this year [Sch07], and the recent breakthrough quasi-polynomial time algorithm [CJK<sup>+</sup>17], both have time complexity expressed by an exponential function with exponent dependent only on the number of distinct priorities, not the number of game positions:

**Theorem 3.4.17 ([CJK<sup>+</sup>17])** *The emptiness problem for parity tree automata is decidable in time*

$$\mathcal{O}\left(n^{\lceil \log d \rceil + 6}\right),$$

where  $n$  is the number of states and  $d$  is the number of different priorities used in the automaton.

The above motivates the search for automata that use as small number of priorities as possible for recognizing a given language. This, in turn, makes the following algorithmic problem practically important.

Below **Type 1** and **Type 2** refer to types of automata, e.g. they can be “nondeterministic”, “deterministic”, “alternating”, etc.

**Type 1 Index Problem for Type 2 automata:**

Given a **Type 2** parity tree automaton  $\mathcal{A}$ ,  
calculate the smallest **Type 1** index class containing  $L(\mathcal{A})$ .

In general, i.e. for Type 2 being “nondeterministic” or “alternating”, the solution for the index problem is not known. There are some partial results, though. Colcombet, Kuperberg, Löding and Vanden Boom [CKLV13, Theorem 9] have shown that it is decidable whether for a given alternating automaton there exists a nondeterministic automaton of index  $(0, 1)$  (i.e. a co-Büchi automaton) that recognizes the same language. More progress have been achieved by considering subclasses of the class of regular languages. For example, all the variants of the index problem are solved for languages recognized by deterministic automata. Algorithms calculating position in the deterministic [NW98], the nondeterministic [NW05], and the weak [NW03, Mur08a] hierarchy are known for such languages. The same applies to so-called Game Automata [FMS16]. Some progress has been also made for unambiguous automata [MS14], that are the main topic of this chapter.

## 3.5 Subclasses of Known Topological Complexity

### 3.5.1 Deterministic Tree Languages

**Definition 3.5.1 (Deterministic Tree Language)** *A regular tree language is called **deterministic** if it is recognized by a deterministic parity tree automaton.*

Consider the following language:

$$LR = \{t : (t(l) = a \wedge t(r) = b) \vee (t(l) = b \wedge t(r) = a)\} \quad (3.2)$$

One can observe that the language cannot be recognized by a deterministic automaton, but can be easily recognized by a nondeterministic one. Therefore, deterministic tree languages constitute a proper subclass of the class of regular languages. We note that a stronger statement holds: the topological complexities (in the sense of Definition 1.4.1) of the classes differ.

**Fact 3.5.2** *The topological complexity of the class of deterministic tree languages is  $\Pi_1^1$ .*

First, note that each language  $T_{(\iota, \kappa)}$  from Definition 3.3.3 can be recognized by a deterministic automaton (of index  $(\iota, \kappa)$ ):

**Remark 3.5.3** *Language  $T_{(\iota, \kappa)}$  is deterministic.*

**Proof (of Fact 3.5.2):**

Recall that, by Lemma 3.3.4, the acceptance condition of parity tree automata is in  $\Pi_1^1$  projective class. By Remark 1.5.2, the same upper bound refers to the class of deterministic languages.

Language  $T_{(0,1)}$  provides the desired lower topological complexity bound, by Lemma 3.3.6 and Remark 3.5.3.  $\square$

The following fact can be seen as a consequence of Fact 3.5.2, but it can also be proven without use of topology that the complement of language  $T_{(0,1)}$ , i.e. the language  $EB$ , is not deterministic.

**Fact 3.5.4** *The class of deterministic tree languages is not closed under the complement.*

The latter is a bit unusual, since deterministic automata usually trivially complement. The issue in the case of tree automata comes from the fact that “on each branch” quantifier is hard-coded in the acceptance condition, and the dual cannot be used.

The following algorithmic question concerning topological complexity was solved for deterministic automata.

**Topological Complexity Problem:**

Given an automaton  $\mathcal{A}$ ,  
calculate the smallest boldface hierarchy class containing  $L(\mathcal{A})$ .

Niwiński and Walukiewicz [NW03] gave an effective characterization (decision procedure) of deterministic automata that recognize non-Borel (namely  $\Pi_1^1$ -complete) languages. They also show that if a deterministic language is Borel, then it is in  $\Pi_3^0$  and it is also recognized by a weak automaton. Murlak [Mur05] have added to this by providing algorithms deciding whether a given deterministic automaton recognizes language in  $\Pi_1^0$ ,  $\Sigma_1^0$ ,  $\Pi_2^0$ ,  $\Sigma_2^0$ . The two results combined give a procedure solving the topological complexity problem (restricted to  $\Pi_j^i$ ,  $\Sigma_j^i$  and  $\Delta_j^i$  classes) for deterministic languages.

Murlak [Mur08b] has shown that classes  $\mathcal{L}^{weak}((0,1))$ ,  $\mathcal{L}^{weak}((1,2))$ ,  $\mathcal{L}^{weak}((0,2))$ ,  $\mathcal{L}^{weak}((1,3))$  of the weak index hierarchy correspond exactly to the aforementioned Borel classes  $\Pi_1^0$ ,  $\Sigma_1^0$ ,  $\Pi_2^0$ ,  $\Sigma_2^0$  for deterministic languages. Therefore, the above algorithm can be used to solve the weak index problem for deterministic automata. As it was noted above, all the other considered variants of the index problem are also solved for deterministic languages.

Niwiński and Walukiewicz [NW03] have also shown that the following problem is decidable (and EXPTIME-complete).

**Determinizability Problem:**

Given a nondeterministic parity tree automaton  $\mathcal{A}$ ,  
decide whether the language recognized by  $\mathcal{A}$  is deterministic.

### 3.5.2 Büchi Tree Languages

Similarly as in the context of  $\omega$ -words we define:

**Definition 3.5.5** *A parity tree automaton of index  $(1, 2)$  is called a **Büchi tree automaton**.*

**Definition 3.5.6** *A language is called a **Büchi (tree) language** if it is recognized by a nondeterministic Büchi tree automaton.*

**Fact 3.5.7** *The topological complexity of the class of Büchi tree languages is  $\Sigma_1^1$ .*

First observe one of the facts that cause the uniqueness of Büchi automata, namely that Büchi acceptance condition is topologically less complex than the general parity tree condition.

**Lemma 3.5.8**  $T_{(1,2)} \in \Pi_2^0$ .

**Proof:**

For  $k < \omega$ , let  $T_k \subseteq T_{\{1,2\}}$  be the set of trees that have at least  $k$  labels 2 on each branch. For  $t \in T_k$ , let  $\hat{t}$  be the tree that is produced from  $t$  by removing all nodes that have  $k$  labels 2 above. Note that, by König's Lemma,  $\hat{t}$  is finite, because it contains only finite branches. Therefore,  $T_k$  is open as the following union of basic open sets:

$$T_k = \bigcup_{t \in T_k} G_{\hat{t}},$$

for  $G_s$  as in the definition of the topology on trees (1.5). Now,  $T_{(1,2)} = \bigcap_{k < \omega} T_k \in \Pi_2^0$ .  $\square$

**Proof (of Fact 3.5.7):**

By Lemma 3.5.8, each Büchi language is recognized by a nondeterministic automaton with a Borel acceptance condition so, by Remark 1.5.3, it is in  $\Sigma_1^1$ .

For the lower topological complexity bound it is enough to note that an automaton from Example 3.2.4 is a Büchi tree automaton, and to recall that  $E$  is  $\Sigma_1^1$ -complete, by Corollary 3.3.7.  $\square$

By the following result by Arnold and Niwiński, the nondeterministic and the alternating hierarchies coincide on the class  $(1, 2)$ .

**Theorem 3.5.9 ([AN92])** *A language is recognized by an alternating Büchi automaton if and only if it is recognized by a nondeterministic Büchi automaton.*

**Corollary 3.5.10** *If language  $L$  is recognized by an alternating Büchi automaton then  $L \in \Sigma_1^1$ .*

**Corollary 3.5.11** *If language  $L$  is recognized by an alternating automaton of index  $(0, 1)$  then  $L \in \Pi_1^1$ .*

Michał Skrzypczak and Igor Walukiewicz [SW16] have shown that a Büchi language is either Borel and recognized by a weak automaton, or  $\Sigma_1^1$ -complete, and that it can be algorithmically decided which of the two cases holds.

### 3.5.3 Weak Tree Languages

**Definition 3.5.12** *A regular tree language is called **weak** if it is recognized by a weak alternating tree automaton.*

Skurczyński [Sku93] has shown examples of  $\Pi_i^0$  and  $\Sigma_i^0$ -complete languages recognized by weak alternating automata of index  $(0, i)$  and  $(1, i + 1)$ , respectively. On the other hand, Duparc and Murlak [DM07] have shown that each language recognized by a weak automaton of index  $(0, i)$  is  $\Pi_i^0$  and, by duality, each language recognized by a weak automaton of index  $(1, i + 1)$  is  $\Sigma_i^0$ . We get:

**Theorem 3.5.13 ([Sku93, DM07])** *The topological complexity of  $\mathcal{L}^{weak}((0, i))$  is  $\Pi_i^0$ . The topological complexity of  $\mathcal{L}^{weak}((1, i + 1))$  is  $\Sigma_i^0$ .*

**Corollary 3.5.14** *The topological complexity of the class of weak tree languages is  $\bigcup_{i < \omega} \Sigma_i^0$ .*

Although this is conjectured that Borel hierarchy and weak index hierarchy coincide for weak languages, it is still possible that there exists a weak language on  $i$ th level of the Borel hierarchy that is not recognized by any weak automaton using  $i + 1$  priorities. As it was noted above, the coincidence holds if we restrict to weak languages that are also deterministic [Mur08b] (however, there are only 4 classes in that case).

## 3.6 Unambiguous Tree Languages

### 3.6.1 Basic Properties

**Definition 3.6.1 (Unambiguous Tree Language)** *A regular tree language is called **unambiguous** if it is recognized by an unambiguous parity tree automaton.*

By definition, each deterministic tree language is unambiguous and each unambiguous tree language is regular. In contrast to the case of  $\omega$ -words, it turns out that both the associated class inclusions are strict. For the first case it is a folklore fact and comes e.g. from an observation that the language  $LR$  from equation (3.2) is unambiguous.

**Fact 3.6.2** *Deterministic tree languages constitute a proper subclass of unambiguous tree languages.*

Strictness of the second inclusion is a more involved result that was first proven by Niwiński and Walukiewicz [NW96] using a result of Gurevich and Shelah [GS83] on lack of uniformization property of MSO over the binary tree. An elementary proof was given by Arnaud Carayol and Christof Löding [CL07], what was followed by a joined paper [CLNW10]. The following are examples of regular languages that are not unambiguous (i.e. are **ambiguous**).

$$\begin{aligned} E &:= \{t \in T_{\{0,1\}} : \exists_{v \in \{l,r\}^*} t(v) = 1\} \\ EB &- \text{ defined as in Example 3.2.4} \end{aligned} \tag{3.3}$$

Intuitively speaking, an automaton is not able to select unambiguously one of possibly many branches or nodes witnessing that a tree belongs to  $EB$  or  $E$ , respectively.

**Theorem 3.6.3 ([NW96])** *Unambiguous tree languages constitute a proper subclass of regular tree languages.*

Observe that the following language is unambiguous:

$$UB := \{t \in T_{\{0,1\}} : \exists!_{\alpha \in \{l,r\}^\omega} t(\alpha) \in N_2\},$$

where  $N_2$  defined as in Fact 1.2.7.

Turning towards a discussion of the topological complexity of unambiguous languages we start with recalling the following fact.

**Fact 3.6.4** *Set  $UB$  is in  $\Pi_1^1$ .*

**Proof:**

Note that:

$$UB = \{t \in T_{\{0,1\}} : \exists!_{\alpha \in \{l,r\}^\omega} (t, \alpha) \in B\} \quad (3.4)$$

$$\text{for } B = \{(t, \alpha) \in T_{\{0,1\}} \times \{l, r\}^\omega : t(\alpha) \in N_2\} \quad (3.5)$$

Observe that  $B \in \Pi_2^0$ , as an inverse image of  $N_2$  under a continuous function. A theorem by Lusin (see e.g. [Kec95, Theorem 18.11]) states that each set of the form as in equation (3.4) for  $B$  Borel is in  $\Pi_1^1$ .  $\square$

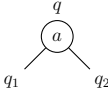
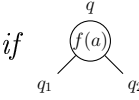
Recall that deterministic automata are capable of recognizing some  $\Pi_1^1$ -complete sets. Because  $UB$  seems to be typical and close to the very definition of unambiguous automata, before the article [Hum12] by the author of this thesis, it had been commonly believed that there was no unambiguous language topologically more complex than all deterministic languages. In Section 3.7 we recall the main result of the mentioned paper [Hum12], that falsifies the belief by showing an example of  $\Sigma_1^1$ -complete unambiguous language. The construction providing more and more complex unambiguous languages is given in sections 3.8, 3.9, 3.10. However, no progress is obtained in terms of upper topological complexity bound for unambiguous languages, what is discussed deeper in Section 3.12.

There is some uniqueness for unambiguous languages in terms of closure properties.

**Definition 3.6.5** *Given automata  $\mathcal{A}$ ,  $\mathcal{B}$ , a **product** of  $\mathcal{A}$  and  $\mathcal{B}$  is any automaton with state space being a product of sets of states of the two automata, and transition relation corresponding coordinate-wise to transition relations of  $\mathcal{A}$  and  $\mathcal{B}$ .*

**Fact 3.6.6** *The class of unambiguous languages is closed under:*

1. *intersection of two languages,*
2. *union of two disjoint languages,*
3. *inverse image under letter substitution. Precisely, for any  $f : A \rightarrow B$ , if  $L \subseteq T_B$  is recognized by unambiguous parity tree automaton  $\mathcal{L}$ , then  $\{t \in T_A : f \circ t \in L\} \subseteq T_A$  is recognized by a parity tree automaton  $\mathcal{M}$ , with state space and priority function the same as  $\mathcal{L}$  and with the set of*

transitions defined as follows:  is a transition in  $\mathcal{M}$  if and only if  is a transition in  $\mathcal{L}$ . The automaton  $\mathcal{M}$  is unambiguous.

**Proof (sketch):**

(1) Let unambiguous parity automata  $\mathcal{L}$  and  $\mathcal{M}$  recognize languages  $L$  and  $M$ , respectively. We construct an unambiguous automaton recognizing  $L \cap M$ . Let us consider a product of automata  $\mathcal{L}$  and  $\mathcal{M}$ . A run of such a product is a pair: a run of automaton  $\mathcal{L}$  and a run of automaton  $\mathcal{M}$ . In order for the product to recognize language  $L \cap M$  we would like to say that the product-run is accepting if and only if in both component-runs maximal priority occurring infinitely often is even. Although, the acceptance condition formulated like this is not a parity condition, we prove that it can be recognized by a deterministic parity tree automaton. First recall that a tree parity condition  $T_{(\iota, \kappa)}$  is recognized by a deterministic automaton. Deterministic tree automaton is, in essence, a deterministic word automaton that reads directions  $\{l, r\}$  and labels, in alternation (for details see the article by Niwiński and Walukiewicz [NW03]). Now, by McNaughton's determinization theorem and by closure of  $\omega$ -regular languages under intersection, we get that the conjunction of the parity conditions for the component-runs can also be recognized by a deterministic word parity automaton, hence also by a deterministic tree parity automaton. Now we let the deterministic automaton run on top of the run of the product automaton. We get a combined automaton (such a construction is called a cascade of automata [Col12]) that recognizes  $L \cap M$ .

It remains to observe that the constructed automaton is unambiguous. Note that each run of this automaton corresponds to a pair of runs of  $\mathcal{L}$  and  $\mathcal{M}$ , each pair of such runs corresponds to exactly one run of the resulting automaton, and a run of the resulting automaton is accepting only if the two component-runs are accepting. Since there is at most one pair of accepting runs of component-automata, there is at most one accepting run of the constructed automaton on each input tree.

(2) Let unambiguous parity automata  $\mathcal{L}$  and  $\mathcal{M}$  recognize languages  $L$  and  $M$ , respectively. We construct the so-called **disjoint union** of the pair of automata. Without loss of generality we may assume that state spaces of  $\mathcal{L}$  and  $\mathcal{M}$  are disjoint. The disjoint union automaton has the state space that is the union of state spaces of  $\mathcal{L}$  and  $\mathcal{M}$ , the transition relation is the union of the transition relations of  $\mathcal{L}$  and  $\mathcal{M}$ , and the set of initial states is



the union of the sets of initial states of  $\mathcal{L}$  and  $\mathcal{M}$ . Clearly, the automaton accepts a tree if and only if  $\mathcal{L}$  or  $\mathcal{M}$  accepts it.

Observe that the constructed automaton is unambiguous. Indeed, if there were two accepting runs on some tree  $t$ , it would mean that either one of the automata  $\mathcal{L}$ ,  $\mathcal{M}$  is not unambiguous or that the sets  $L$  and  $M$  are not disjoint.

(3) Note that each run (respectively accepting run) of automaton  $\mathcal{M}$  on  $t \in T_A$  is also a run (respectively an accepting run) of  $\mathcal{L}$  on  $f \circ t \in T_B$ . Therefore,  $\mathcal{M}$  recognizes language  $M = \{t \in T_A : f \circ t \in L\}$ .

Now suppose that there are two different accepting runs of  $\mathcal{M}$  on some tree  $t$ . Then they are also two different accepting runs of  $\mathcal{L}$  on  $f \circ t$ , what contradicts unambiguity of  $\mathcal{L}$ . Hence,  $\mathcal{M}$  is unambiguous.  $\square$

**Fact 3.6.7** *The class of unambiguous languages is not closed under:*

1. *union,*
2. *complementation.*

**Proof:**

(1) Bilkowski and Skrzypczak [BM13, Proposition 2] show that a union of two deterministic languages may not be unambiguous on the example of the language  $T_{\{0,1\}} \cup T_{\{1,2\}}$ .

(2) It is enough to recall that the language  $E$  defined by equation (3.3) is ambiguous and observe that  $\bar{E} = \{t \in T_{\{0,1\}} : \forall_{v \in \{l,r\}^*} t(v) = 0\}$  is a deterministic, hence also an unambiguous, language.  $\square$

**Definition 3.6.8 (Bi-Unambiguous Languages)** *Language  $L$  is **bi-unambiguous**<sup>3</sup> if  $L$  is unambiguous and  $\bar{L}$  is unambiguous.*

**Fact 3.6.9** *The class of bi-unambiguous languages is closed under:*

1. *union,*
2. *intersection,*

---

<sup>3</sup>The author of the thesis used to use the name “strongly unambiguous” for the same notion to follow Thomas Colcombet’s [Col12] notion of strongly unambiguous automata, [Hum12]. It was later questioned by Damian Niwiński whether there is anything “strong” in the languages (as opposed to the case of automata, where “strong unambiguity” was really strengthening of “unambiguity”). The name “bi-unambiguous” was proposed as one better reflecting the matter.

3. *complementation,*

4. *inverse image under letter substitution.*

**Proof:**

(1) If  $L, \bar{L}, M, \bar{M} \subseteq T_A$  are unambiguous languages then, by closure of unambiguous languages under intersection and disjoint union (see Fact 3.6.6),  $L \cup M$  and  $\bar{L} \cup \bar{M}$  are unambiguous, because:

$$\begin{aligned} L \cup B &= L \sqcup (M \cap \bar{L}), \\ \overline{L \cup B} &= \bar{L} \cap \bar{M}, \end{aligned}$$

where by  $\sqcup$  we designate union in which operands are disjoint.

(3) By the definition.

(2) By points (3), (1) and De Morgan laws.

(4) Follows from Fact 3.6.6(3), because  $\overline{f_*^{-1}(L)} = f_*^{-1}(\bar{L})$ , for  $f_* : T_A \rightarrow T_B$  being the function induced by an alphabet projection  $f : A \rightarrow B$ , namely  $f_*(t) = f \circ t$ .  $\square$

**Fact 3.6.10** *The class of bi-unambiguous languages is not closed under embedding in a space with larger alphabet.*

**Proof:**

It is enough to consider deterministic language  $L = \{t : \forall_{v \in \{l,r\}^*} t(v) = 0\}$ . This language as a subset of  $T_{\{0\}}$  is bi-unambiguous. The same language considered as a subset of  $T_{\{0,1\}}$  is still unambiguous, but its complement  $\bar{L} = E$  is the already known example of ambiguous language.  $\square$

From the algorithmic point of view, apart from the index problems presented before, the following decision problems seem the most interesting in the context of unambiguous languages:

**Unambiguity (Bi-unambiguity) Problem:**

Given a nondeterministic parity tree automaton,  
decide whether the language recognized by the automaton is unambiguous (bi-unambiguous).

According to the author's knowledge there is not much progress made in answering the question whether the unambiguity problem is decidable. Marcin Bilkowski and Michał Skrzypczak [BM13] have given a partial solution to the bi-unambiguity problem. They have shown that it is decidable if deterministic automaton is given as an input (i.e. it is decidable whether the

complement of a given deterministic automaton is unambiguous). Moreover, the authors conjecture that the decision procedure works also for the general bi-unambiguity problem.

Recall, for the contrast, that it is easily decidable whether a given automaton is unambiguous.

In this chapter we follow the approach that the discussion of the topological complexity of unambiguous automata may give some guidance in the search for algorithms for the above unsolved problems, similarly as it have given guidance to the proof of undecidability of MSO+U (see Section 1.5.4).

### 3.6.2 Topological Complexity of Unambiguous Büchi Automata

In this section we recall some partial results concerning the complexity of unambiguous automata. We start with a result by Olivier Finkel and Pierre Simonnet:

**Theorem 3.6.11 ([FS09, Corollary 4.14])** *A tree language recognized by an unambiguous Büchi automaton is Borel.*

The result was strengthened by Henryk Michalewski and Michał Skrzypczak:

**Theorem 3.6.12 ([MS14, Theorem 1])** *A tree language recognized by an unambiguous Büchi automaton is weak.*

In its general form the result states:

**Theorem 3.6.13 ([MS14, Theorem 1])** *For  $\kappa > 0$ :*

$$\begin{aligned}\mathcal{L}^{uamb}((0, 2\kappa)) &\subseteq \mathcal{L}^{alt}((0, 2\kappa - 1), (1, 2\kappa)) \\ \mathcal{L}^{uamb}((1, 2\kappa)) &\subseteq \mathcal{L}^{alt}((0, 2\kappa - 2), (1, 2\kappa - 1))\end{aligned}$$

## 3.7 Analytic-Complete Unambiguous Language

The result presented in this section was inspired by the work of Marcin Bilkowski [Bil10] published in an extended form in a paper co-authored by Michał Skrzypczak [BM13], and by the decidability result presented by Niwiński and Walukiewicz [NW03]. Bilkowski has shown that the complement of a deterministic language is unambiguous if and only if the language is recognized by a thin automaton, i.e. by an automaton that has only countably

many non-trivial paths in each accepting run. A path is trivial if, from some moment on, it is labeled only by all-accepting or all-rejecting states.

The deterministic automaton recognizing the complement of language  $G$  described below is thin in Bilkowski's sense, and has split property—the sufficient condition for the  $\Pi_1^1$ -hardness from the result of Niwiński and Walukiewicz [NW03]. The proofs presented in this section are self-contained, they do not use the results of Bilkowski, Niwiński and Walukiewicz.

We call a branch of a binary tree over the alphabet  $\{a, b\}$  **good** if:

1. it is labeled only with **a**'s,
2. it turns left infinitely many times.

Let:

$$G := \{t \in T_{\{a,b\}} : t \text{ has a good branch}\}$$

The following lemma is crucial for the unambiguity:

**Lemma 3.7.1** *If an infinite binary tree over the alphabet  $\{a, b\}$  has a good branch, then it has the left-most such branch, i.e. a good branch such that there is no good branch to the left.*

**Proof:**

Assume that a tree  $t$  has a good branch. The construction of the left-most good branch goes as follows. We start from the root. If we have constructed a prefix of the branch up to a node  $v$  we advance to the left descendant if there are good branches going through it. Otherwise we advance to the right descendant. Call the branch constructed by this procedure  $\rho(t)$ .

By the construction, it is clear that there is no good branch to the left of  $\rho(t)$ . Now we prove that  $\rho(t)$  is good. Note that during the construction we maintain the invariant that there is a good branch going through a considered node. In particular, all nodes we have selected are labeled with **a**, therefore, we only need to verify that  $\rho(t)$  turns left infinitely many times.

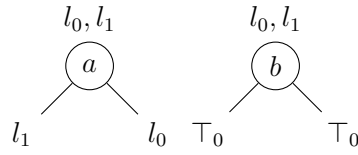
Suppose that  $\rho(t)$  turns left only finitely many times. Then there is a node  $v$  on the branch  $\rho(t)$  after which  $\rho(t)$  turns only right. Let us take a good branch going through  $v$ , and call it  $\sigma$ . By the assumption,  $\rho(t)$  is not good, so branches  $\rho(t)$  and  $\sigma$  diverge in some node  $w$ . Since  $w$  is below  $v$ ,  $\rho(t)$  goes right from  $w$  and  $\sigma$  goes left. Since  $\sigma$  is good, the construction should have selected the left descendant of  $w$ , but have selected the right one. That yields a contradiction, so  $\rho(t)$  turns left infinitely many times.  $\square$

Now let  $L = \overline{G}$ .

**Proposition 3.7.2** *Language  $G$  is recognized by an unambiguous automaton and its complement  $L$  is recognized by a deterministic automaton. In particular,  $G$  is bi-unambiguous.*

**Proof:**

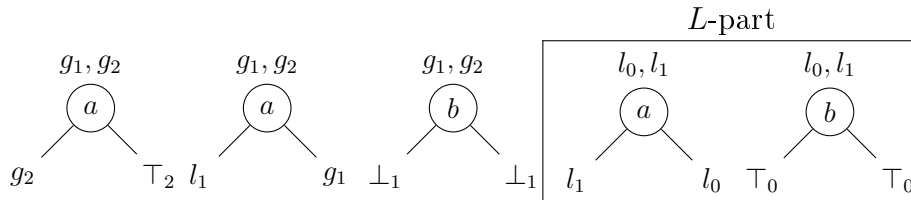
Let us start with constructing a deterministic automaton  $\mathcal{L}$  recognizing  $L$ . It has 3 states: states  $l_0$  and  $l_1$  (subscript indicates the priority) occur on paths that have had only  $a$ 's so far; state  $\top_0$  is all-accepting (i.e. self-looping with priority 0). Initial state is  $l_1$  and the transitions are as follows:



The automaton uses priorities  $\{0, 1\}$ . Note that, for a given tree  $t$ , the run of  $\mathcal{L}$  on  $t$  has a branch with infinitely many priorities 1 on it if and only if there is an  $a$ -labeled branch turning left infinitely often in  $t$ . Therefore the automaton recognizes  $L$ .

Thanks to Lemma 3.7.1, to prove unambiguity of  $G$  it is enough to construct an automaton  $\mathcal{G}$  that guesses the left-most good branch and verifies correctness of the guess. The idea is that the automaton goes along a branch labeled with  $a$ 's, proving that the branch turns left infinitely many times, and proving that everything that diverges to the left of the branch does not have a good branch (i.e. belongs to  $L$ ), and not caring what happens to the right of the branch.

The automaton  $\mathcal{G}$  uses  $\mathcal{L}$  as a component. It has 7 states and uses priorities 0, 1, 2. States  $g_1$  and  $g_2$  (again, subscript indicates the priority) are used to track the branch; states of  $\mathcal{L}$ —to prove non-existence of a good branch in a subtree; state  $\top_2$  is all-accepting, and state  $\perp_1$  is all-rejecting. The initial state is  $g_1$ . The automaton uses the following transitions:



It is not hard to see that presented automaton implements described idea, therefore accepts if and only if a given tree has a good branch. It is unambiguous, because it only can accept by labeling the left-most good branch with  $g$  states.  $\square$

As we can see the automaton  $\mathcal{L}$  is of index  $(0, 1)$ . The automaton  $\mathcal{G}$  uses priorities  $\{0, 1, 2\}$ , however, the L-part of the automaton, that uses priorities  $\{0, 1\}$ , constitutes a strongly connected component, while the remaining part of the automaton uses only priorities  $\{1, 2\}$ . As a result we get:

**Remark 3.7.3**  $G \subseteq \mathcal{L}^{uamb}((0, 1), (1, 2))$

**Proposition 3.7.4** *Set  $G$  is  $\Sigma_1^1$ -complete.*

**Proof:**

To prove the hardness we continuously reduce the set  $\text{IF}^1$  of  $\mathbb{N}$ -branching trees with an infinite branch to our set  $G$  (see Fact 2.4.4). We construct a reducing function  $f : T^{\mathbb{N}} \rightarrow T_{\{a,b\}}$ . Fix a tree  $t \in T^{\mathbb{N}}$ . Put labels  $a$  to the root and all right descendant nodes in the tree  $f(t)$ . For each node  $n_1 n_2 n_3 \dots n_k$  of tree  $t$  we put label  $a$  to the node  $r^{n_1} l r^{n_2} l r^{n_3} l \dots r^{n_k} l$  in  $f(t)$ . Remaining left descendant nodes obtain label  $b$ .

Note that  $f(t)$  has an  $a$ -labeled branch that turns left infinitely many times (i.e. a good branch) if and only if  $t$  has an infinite branch. Therefore

$$f(t) \in G \iff t \in \text{IF}^1$$

So  $f$  indeed reduces  $\text{IF}^1$  to  $G$ .

Function  $f$  is continuous, because the labels at the  $n$ th level of the tree  $f(t)$  are determined by the finite part of a tree  $t$ , namely the part in  $\{1, \dots, n\}^{\leq n}$ .

The upper topological complexity bound for set  $G$  comes from Proposition 3.7.6, that will be proven later, and from Fact 3.5.7.  $\square$

Therefore, we have proven the following:

**Theorem 3.7.5** *There is a bi-unambiguous language of infinite trees that is  $\Sigma_1^1$ -complete.*

Thanks to Theorem 3.6.11 we know that no unambiguous Büchi automaton recognizes language  $G$ . Proposition 3.7.4 implies that  $G$  is not a  $\Pi_1^1$  set, therefore, by Corollary 3.5.11, it cannot be recognized by any (even alternating) automaton of index  $(0, 1)$ . As a result we obtain that the use of 3 priorities is necessary for unambiguous automaton to recognize  $G$ . On the other hand observe that:

**Proposition 3.7.6** *Language  $G$  is recognized by a nondeterministic Büchi automaton.*

**Proof:**

It suffices to remove  $L$ -part from the unambiguous automaton  $\mathcal{G}$  presented in the proof of Proposition 3.7.2, replacing  $l_1$  with  $\top_2$  in other transitions, to obtain needed nondeterministic automaton. The only purpose of that part was to make sure that we select the left-most good branch. We do not need this if we do not care about the number of accepting runs.  $\square$

From this observation we obtain that Theorem 3.6.11 by Finkel and Simonnet is tight in the following sense:

**Corollary 3.7.7** *There is a language of non-Borel topological complexity that is on one hand unambiguous, and on the other hand Büchi.*

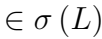
## 3.8 Sigma-Lifting Operation

In this section we introduce an operation on tree languages that preserves bi-unambiguity and increases the topological complexity. More precisely, the operation applied to a language  $L$  produces a language that is topologically more complex than the whole sigma algebra generated by the sets reducible to  $L$ .

First we give an intuition of the operation. Let  $L$  be a given language. To construct language  $\sigma(L)$  we consider trees that correspond to formulas of a logic with negation and countable disjunction. Atomic formulas are trees with  $\sharp$  in the root. Negation is represented by a tree with  $\neg$  in the root, whose left subtree is disregarded and right subtree corresponds to the negated formula. Countable disjunction is represented as a tree with whole left-most branch labeled with  $\vee$ , where subtrees diverging right from this branch correspond to the disjuncts. An atomic formula evaluates to **TRUE** if and only if the right subtree of  $\sharp$ -labeled node is in  $L$ . The evaluation of disjunction and negation is standard. Language  $\sigma(L)$  consists of those trees that correspond to formulas that evaluate to **TRUE**. As usual, formulas are required to be well-founded, meaning that there is no infinite nesting in a formula (but nesting depth does not have to be globally bounded for a formula). Figure 3.2 gives an example of a tree in  $\sigma(L)$ .

The formal definitions follow.

First we give a fixpoint definition of a pair of parametrized languages  $P^\sigma(\mathbf{t}, \mathbf{ff}, \top), N^\sigma(\mathbf{t}, \mathbf{ff}, \top) \subseteq T_{\{\neg, \vee\}, \{\mathbf{t}, \mathbf{f}, \top\}}$ . Intuitively  $P$  and  $N$  stand for “positive” and “negative”; parameters  $\mathbf{t}$  and  $\mathbf{ff}$  stand for “TRUE” and “FALSE”. Parameter  $\top$  will always be replaced with a whole space, which is different for different alphabets—this is why we need it to be a parameter.



112



The operator  $\Phi : \left( \mathbb{P} \left( T_{\{\neg, \vee\}, \{\mathbf{t}, \mathbf{f}, \top\}} \right) \right)^2 \rightarrow \left( \mathbb{P} \left( T_{\{\neg, \vee\}, \{\mathbf{t}, \mathbf{f}, \top\}} \right) \right)^2$  is defined using parametrized languages  $\Phi_P, \Phi_N \subseteq T_{\{\vee, \neg\}, \{p, n, \mathbf{t}, \mathbf{f}, \top\}}$  by:

$$(\mathbf{P}, \mathbf{N}) \xrightarrow{\Phi} (\Phi_P [p \mapsto \mathbf{P}, n \mapsto \mathbf{N}], \Phi_N [p \mapsto \mathbf{P}, n \mapsto \mathbf{N}])$$

where:

$$\Phi_P := \left\{ \underbrace{\left( \begin{array}{c} \vee \\ \swarrow \quad \searrow \\ \vee \quad \square_{x_1} \\ \vdots \quad \vdots \\ \vee \quad \square_{x_n} \end{array} \right)}_{\Phi_{P_\vee}} : \begin{array}{l} \forall_i x_i \in \{p, n\} \\ \exists_i x_i = p \end{array} \right\} \cup \underbrace{\left\{ \begin{array}{c} \neg \\ \swarrow \quad \searrow \\ \top \quad n \end{array} \right\}}_{\Phi_{P_\neg}} \cup \underbrace{\left\{ \begin{array}{c} \mathbf{t} \end{array} \right\}}_{\Phi_{P_\top}} \quad (3.6)$$

$$\Phi_N := \left\{ \underbrace{\left( \begin{array}{c} \vee \\ \swarrow \quad \searrow \\ \vee \quad \square_n \\ \vdots \quad \vdots \\ \vee \quad \square_n \end{array} \right)}_{\Phi_{N_\vee}} \right\} \cup \underbrace{\left\{ \begin{array}{c} \neg \\ \swarrow \quad \searrow \\ \top \quad p \end{array} \right\}}_{\Phi_{N_\neg}} \cup \underbrace{\left\{ \begin{array}{c} \mathbf{ff} \end{array} \right\}}_{\Phi_{N_F}} \quad (3.7)$$

The set of pairs of languages over a fixed alphabet ordered by coordinate-wise inclusion forms a complete lattice. Moreover, operation  $\Phi$  is monotonic—this comes from monotonicity of substitution and union. Therefore, by Knaster-Tarski Fixpoint Theorem,  $\Phi$  has the least fixed point. Let:

$$(P^\sigma(\mathbf{t}, \mathbf{ff}, \top), N^\sigma(\mathbf{t}, \mathbf{ff}, \top)) := \text{Fix}(\Phi) \quad (3.8)$$

Now we want to prove that under some assumption on languages  $T, F, K$ , languages  $P^\sigma(T, F, K)$  and  $N^\sigma(T, F, K)$  are disjoint. To achieve this we first note a general fact about sets defined by fixpoints:

**Lemma 3.8.1** *Let  $X$  be a complete lattice. Let  $Y \subseteq X$  (a property of elements of  $X$ ) be such that:*

1.  $\perp \in Y$  (for  $\perp$  being the least element of  $X$ ),
2.  $Y$  is closed, in the sense that for each ordinal  $\beta$  and each increasing (not necessarily strictly) sequence  $\{x_\alpha\}_{\alpha < \beta}$ , if  $\{x_\alpha\}_{\alpha < \beta} \subseteq Y$  then  $\sup_{\alpha < \beta} x_\alpha \in Y$ .

Let  $f : X \rightarrow X$  be such a function that:

1.  $f$  is monotonic,
2.  $f$  preserves property  $Y$ , i.e. if  $f(Y) \subseteq Y$ .

Then  $\text{Fix}(f) \in Y$ .

**Proof:**

First note that  $\text{Fix}(f) = f^\alpha(\perp)$  for some ordinal  $\alpha$ , where:

- $f^0 = \text{id}$ ,
- $f^{\gamma+1} = f \circ f^\gamma$ ,
- for  $\beta$  being a limit ordinal, for each  $x \in X$ ,  $f^\beta(x) = \sup_{\gamma < \beta} f^\gamma(x)$ .

Therefore it suffices to prove that  $f^\beta(\perp) \in Y$ , for each ordinal  $\beta$ . We do it by induction on  $\beta$ .

For  $\beta = 0$  we have  $f^\beta(\perp) = \perp \in Y$  by the assumption on  $Y$ .

Successor step of the induction comes directly from the assumption that  $f$  preserves  $Y$ : if  $f^\beta(x) \in Y$  then  $f^{\beta+1}(x) = f(f^\beta(x)) \in Y$ .

Let now  $\beta$  be a limit ordinal. By the monotonicity of  $f$ , the sequence  $\{f^\gamma(\perp)\}_{\gamma < \beta}$  is increasing. By the inductive assumption,  $\{f^\gamma(\perp)\}_{\gamma < \beta} \subseteq Y$ , therefore, by closedness of  $Y$ ,  $f^\beta(\perp) = \sup_{\gamma < \beta} f^\gamma(\perp) \in Y$ . This concludes the limit step of the induction.  $\square$

Now we prove a lemma that is very specifically tight to our needs. Since it will be reused later, we prove it in a bit more general form than it is necessary for this section.

**Lemma 3.8.2** *Let  $f : \left(\mathbb{P}\left(T_{B,\{x_1,x_2,\dots,x_k\}}\right)\right)^2 \rightarrow \left(\mathbb{P}\left(T_{B,\{x_1,x_2,\dots,x_k\}}\right)\right)^2$  be a monotonic function on pairs of parametrized languages with  $k \geq 0$  parameters. Let  $L_1, L_2, \dots, L_k, M$  be languages such that, for any  $G, H$ , for  $(G', H') := f(G, H)$ , and for the substitution:*

$$\alpha = \{x_1 \mapsto L_1, x_2 \mapsto L_2, \dots, x_k \mapsto L_k\},$$

*if  $G[\alpha], H[\alpha], M$  are pairwise disjoint then  $G'[\alpha], H'[\alpha], M$  are pairwise disjoint. Then for  $(G_0, H_0) := \text{Fix}(f)$ , languages  $G_0[\alpha], H_0[\alpha], M$  are pairwise disjoint.*

**Proof:**

We intend to use Lemma 3.8.1, for  $Y$  being a set of pairs of languages  $(G, H)$  such that  $G[\alpha]$ ,  $H[\alpha]$ ,  $M$  are pairwise disjoint. Function  $f$  is monotonic and preserves  $Y$  by the assumption of the lemma we prove. Of course  $\perp = (\emptyset, \emptyset) \in Y$ , because each substitution performed on the empty set returns the empty set, and the empty set is disjoint with any set (including the empty set itself).

We need to prove that  $Y$  is closed in the sense as in Lemma 3.8.1. Let  $\{(G_\gamma, H_\gamma)\}_{\gamma < \beta}$  be a coordinate-wise increasing sequence such that for each  $\gamma < \beta$ ,  $(G_\gamma, H_\gamma) \in Y$ . Assume, towards a contradiction, that  $(G_\beta, H_\beta) := \sup_{\gamma < \beta} (G_\gamma, H_\gamma) \notin Y$ . It means that one of the two cases holds:

1. there exists a tree  $t$  and contexts  $g \in G_\beta$  and  $h \in H_\beta$  such that  $t$  is context  $g$  with some substitutions done according to  $\alpha$  and  $t$  is also context  $h$  with some substitutions done according to  $\alpha$ ,
2. there exists a tree  $t \in M$  and context  $g \in G_\beta \cup H_\beta$  such that  $t$  is context  $g$  with some substitutions done according to  $\alpha$ .

For the first case, note that, since set of pairs of sets is ordered by coordinate-wise inclusion,  $\sup_{\gamma < \beta} (G_\gamma, H_\gamma) = \left( \bigcup_{\gamma < \beta} G_\gamma, \bigcup_{\gamma < \beta} H_\gamma \right)$ . Therefore, there are some  $\gamma_1 < \beta$ ,  $\gamma_2 < \beta$  such that  $g \in G_{\gamma_1}$  and  $h \in H_{\gamma_2}$ . Since  $\{(G_\gamma, H_\gamma)\}$  is increasing,  $g \in G_{\max(\gamma_1, \gamma_2)}$  and  $h \in H_{\max(\gamma_1, \gamma_2)}$ , and we obtain a contradiction with the assumption that  $G_{\max(\gamma_1, \gamma_2)}[\alpha]$  and  $H_{\max(\gamma_1, \gamma_2)}[\alpha]$  are disjoint.

For the second case it is enough to note that there is some  $\gamma < \beta$  such that  $g \in G_\gamma \cup H_\gamma$ , what gives a contradiction with the assumption on  $M$  being disjoint with both  $G_\gamma[\alpha]$  and  $H_\gamma[\alpha]$ .  $\square$

**Proposition 3.8.3** *Let sets  $T$  and  $F$  be disjoint and such that for  $t \in T \cup F$ ,  $t(\varepsilon) \notin \{\neg, \vee\}$ . Let  $K$  arbitrary. Then sets  $P^\sigma(T, F, K)$  and  $N^\sigma(T, F, K)$  are disjoint.*

**Proof:**

We intend to use Lemma 3.8.2 for  $f = \Phi$ ,  $k = 3$ ,  $L_1 = T$ ,  $L_2 = F$ ,  $L_3 = K$ , and  $M = \emptyset$ . Therefore, it is enough to show that for  $T$  and  $F$  as in the statement we prove, for any  $\mathbf{P}, \mathbf{N} \subseteq T_{\{\neg, \vee\}, \{\mathbf{t}, \mathbf{f}, \mathbf{\top}\}}$ , if  $\mathbf{P}(T, F, K) \cap \mathbf{N}(T, F, K) = \emptyset$ , then for  $(\mathbf{P}', \mathbf{N}') = \Phi(\mathbf{P}, \mathbf{N})$  also  $\mathbf{P}'(T, F, K) \cap \mathbf{N}'(T, F, K) = \emptyset$ .

Below we use sets  $\Phi_{P_\vee}$ ,  $\Phi_{P_\neg}$ ,  $\Phi_{P_T}$ ,  $\Phi_{N_\vee}$ ,  $\Phi_{N_\neg}$ ,  $\Phi_{N_F}$ , as used to define operator  $\Phi$ . Set  $\Phi_{P_\vee}[p \mapsto \mathbf{P}, n \mapsto \mathbf{N}]$  is disjoint with  $\Phi_{N_\vee}[p \mapsto \mathbf{P}, n \mapsto \mathbf{N}]$  thanks to disjointness of  $\mathbf{P}$  and  $\mathbf{N}$ . The same argument gives disjointness of

$\Phi_{P_-}[p \mapsto \mathbf{P}, n \mapsto \mathbf{N}, \top \mapsto K]$  and  $\Phi_{N_-}[p \mapsto \mathbf{P}, n \mapsto \mathbf{N}, \top \mapsto K]$ . Sets  $\Phi_{P_T}[\textcolor{blue}{t} \mapsto T, \textcolor{blue}{f} \mapsto F]$  and  $\Phi_{N_F}[\textcolor{blue}{t} \mapsto T, \textcolor{blue}{f} \mapsto F]$  are disjoint thanks to the disjointness of  $T$  and  $F$ . To observe the disjointness of the remaining pairs of components (e.g.  $\Phi_{P_V}$  and  $\Phi_{N_-}$ ;  $\Phi_{P_V}$  and  $\Phi_{N_F}$ ; etc.) it is enough to look at the labels in the root of given subtrees.  $\square$

Now we are ready to define a sigma-lifting operation.

Let  $L$  be a language of trees over a nonempty alphabet  $A$ . Let us fix an arbitrary letter  $a \in A$ , and let

$$A_\sigma = A \cup \{\vee, \neg, \#\}$$

(we do not require that  $\{\vee, \neg, \#\}$  and  $A$  are disjoint). Now we extend  $L$  to a language over  $A_\sigma$ . The way we do it will play a role when we discuss unambiguity.

Let  $\pi_a : A_\sigma \rightarrow A$  be the following alphabet projection:

$$\pi_a(x) = \begin{cases} a & \text{if } x \notin A \\ x & \text{otherwise} \end{cases}$$

A tree mapping  $\widehat{\pi}_a : T_{A_\sigma} \rightarrow T_A$  is defined by mapping each label according to  $\pi_a$ , i.e.:

$$\widehat{\pi}_a(t) = \pi_a \circ t \quad (3.9)$$

It is crucial for our construction that  $\widehat{\pi}_a^{-1}(\overline{L}) = \overline{\widehat{\pi}_a^{-1}(L)}$ .

Finally, we use the following substitution:

$$\lambda_a^L := \left\{ \textcolor{blue}{t} \mapsto T_{A_\sigma} \begin{array}{c} \textcircled{\#} \\ \swarrow \quad \searrow \\ \widehat{\pi}_a^{-1}(L) \end{array}, \textcolor{blue}{ff} \mapsto T_{A_\sigma} \begin{array}{c} \textcircled{\#} \\ \swarrow \quad \searrow \\ \widehat{\pi}_a^{-1}(\overline{L}) \end{array}, \textcolor{blue}{\top} \mapsto T_{A_\sigma} \right\} \quad (3.10)$$

to define:

$$\sigma_a(L) := P^\sigma [\lambda_a^L] \quad (3.11)$$

Note that the definition implements the idea presented in Figure 3.2, with this exception that the alphabet projection is omitted in the figure. Namely, Definition (3.11) expects subtrees diverging to the right from  $\sharp$ -labeled nodes to belong to  $\hat{\pi}_a^{-1}(L)$  or  $\hat{\pi}_a^{-1}(\overline{L})$ , instead of  $L$  and  $\overline{L}$ .

Since the choice of  $a$  hardly ever affects properties that we are interested in, we usually simply write  $\sigma(L)$ , instead of  $\sigma_a(L)$ , and  $\lambda^L$  instead of  $\lambda_a^L$ .

From Proposition 3.8.3 we derive:

### Corollary 3.8.4

$$P^\sigma \left[ \lambda_a^L \right] \cap N^\sigma \left[ \lambda_a^L \right] = \emptyset$$

As it was discussed above, trees from  $P^\sigma$  and from  $N^\sigma$  correspond to boolean formulas. Now, we define several notions that come from this correspondence.

In the following definitions let  $t \in T_{B,C}$  for some alphabets  $B, C$  such that  $\{\vee, \neg\} \subseteq B$ , and  $C \cap \{\vee, \neg\} = \emptyset$  ( $C$  can be empty).

**Definition 3.8.5** We call  $v \in \{l, r\}^*$  a **boolean path** in  $t$  if for all  $w \prec v$ :

- $t(w) \in \{\vee, \neg\}$ ,
- if  $t(w) = \neg$  then  $wr \prec v$ , i.e.  $v$  turns right after each label  $\neg$ ,
- if  $t(w) = \vee$  and  $wl \prec v$ , then  $t(wl) = \vee$ .

For  $D \subseteq (B \cup C) \setminus \{\vee, \neg\}$ , we call a boolean path  $v$   **$D$ -correct** in  $t$  if it satisfies:

- $t(v) \in \{\vee, \neg\} \cup D$ ,
- if  $t(v) = \vee$  then  $t(vl) = \vee$ .

If  $D = \{x\}$  for some  $x$ , then we write  $x$ -correct instead of  $\{x\}$ -correct.

**Definition 3.8.6** We call  $\alpha \in \{l, r\}^\omega$  a **boolean branch** in  $t$  if each prefix of  $\alpha$  is a boolean path. A boolean branch  $\alpha$  is **correct** in  $t$  if  $\alpha \in \{l, r\}^* l^\omega$ , i.e. if it turns right only finitely many times.

We sometimes abuse this terminology and call correct boolean branches  **$D$ -correct**, although the definition does not depend on  $D$ . On the other hand, if  $D$  is clear from the context, we may simply write correct instead of  $D$ -correct for a boolean path.

A path (or a branch) is  **$D$ -incorrect** (or **incorrect**, respectively) if it is not  $D$ -correct (or correct, respectively).

**Remark 3.8.7 (Suffix and concatenation closure)**

1. The set of boolean paths is closed under concatenation, in the following sense.

If  $v$  is a boolean path in  $t$   
and  $w$  is a boolean path (or branch) in  $t_v$ ,  
then  $vw$  is a boolean path (respectively branch) in  $t$ .

Since the definition of a boolean branch cares only about the local properties (labels and successors), also infinite concatenation of boolean paths yields a boolean branch.

2. The set of boolean paths is closed under suffixes, in the following sense.

If  $v$  is a boolean path (or branch) in  $t$   
and  $v = xy$  for  $x \in \{l, r\}^*$  and  $y \in \{l, r\}^{\leq \omega}$ ,  
then  $y$  is a boolean path (or branch) in  $t_x$ .

**Remark 3.8.8** Correctness is a **prefix-independent** property of boolean paths and branches, in the following sense.

If  $vw$  is a boolean branch or path in  $t$  then:  
 $vw$  is  $D$ -correct in  $t$  if and only if  $w$  is  $D$ -correct in  $t_v$ .

We introduce one more set that will play an important role until the end of the chapter. Let, as above,  $\{\vee, \neg\} \subseteq B$ ,  $C \cap \{\vee, \neg\} = \emptyset$ , and let  $D \subseteq (B \cup C) \setminus \{\vee, \neg\}$ .

$$W^D := \left\{ \begin{array}{l} t \in T_{B,C} : \\ \text{all boolean paths in } t \text{ are } D\text{-correct} \quad (\textbf{proper shape}) \\ \text{and all boolean branches in } t \text{ are correct} \quad (\textbf{well-foundedness}) \end{array} \right\} \quad (3.12)$$

If  $D = \{x\}$  for some  $x$ , than we write  $W^x$  instead of  $W^{\{x\}}$ .

We sometimes write  $W^D = W_{B,C}^D$ , for  $W^D \subseteq T_{B,C}$  and  $W^D = W_B^D$ , for  $W^D \subseteq T_B = T_{B,\emptyset}$ .

**Remark 3.8.9** If  $v$  is a boolean path in  $t \in W^D$  and  $t(v) = \vee$ , then for  $n \in \mathbb{N}$ ,  $t(vl^n) = \vee$ .

**Proposition 3.8.10**

$$P^\sigma \cup N^\sigma \subseteq W^{\{\mathbf{t}, \mathbf{f}\}}$$

**Proof:**

Knaster-Tarski Theorem (see e.g. [Tar55]) implies that if  $f$  is a monotonic function on a complete lattice  $(X, \leq)$ , then  $Fix(f) = \inf\{X : \Phi(X) \leq X\}$ . Therefore, to prove the proposition it suffices to show that  $\Phi(W^{\{\mathbf{t}, \mathbf{f}\}}, W^{\{\mathbf{t}, \mathbf{f}\}}) \subseteq (W^{\{\mathbf{t}, \mathbf{f}\}}, W^{\{\mathbf{t}, \mathbf{f}\}})$ , where  $\subseteq$  designates coordinate-wise inclusion. Indeed, if  $(W^{\{\mathbf{t}, \mathbf{f}\}}, W^{\{\mathbf{t}, \mathbf{f}\}}) \in \{(\mathbf{P}, \mathbf{N}) : \Phi(\mathbf{P}, \mathbf{N}) \subseteq \mathbf{P}, \mathbf{N}\}$ , then  $(P^\sigma, N^\sigma) = Fix(\Phi) \subseteq (W^{\{\mathbf{t}, \mathbf{f}\}}, W^{\{\mathbf{t}, \mathbf{f}\}})$ , i.e.  $P^\sigma \subseteq W^{\{\mathbf{t}, \mathbf{f}\}}$  and  $N^\sigma \subseteq W^{\{\mathbf{t}, \mathbf{f}\}}$ , so  $P^\sigma \cup N^\sigma \subseteq W^{\{\mathbf{t}, \mathbf{f}\}}$ .

We show that languages  $\Phi_P[p \mapsto W^{\{\mathbf{t}, \mathbf{f}\}}, n \mapsto W^{\{\mathbf{t}, \mathbf{f}\}}]$  and  $\Phi_N[p \mapsto W^{\{\mathbf{t}, \mathbf{f}\}}, n \mapsto W^{\{\mathbf{t}, \mathbf{f}\}}]$  consist of trees in which each boolean path is  $\{\mathbf{t}, \mathbf{ff}\}$ -correct and each boolean branch is correct. Take  $t \in \Phi_P[p \mapsto W^{\{\mathbf{t}, \mathbf{f}\}}, n \mapsto W^{\{\mathbf{t}, \mathbf{f}\}}] \cup \Phi_N[p \mapsto W^{\{\mathbf{t}, \mathbf{f}\}}, n \mapsto W^{\{\mathbf{t}, \mathbf{f}\}}]$ . Then one of the three cases holds:

1.  $t(\varepsilon) = \neg$  and  $t_r \in W^{\{\mathbf{t}, \mathbf{ff}\}}$ . The path containing only the root is  $\{\mathbf{t}, \mathbf{ff}\}$ -correct in  $t$ . All other potential boolean paths and branches in  $t$  are of the form  $rv$  for  $v$  being a boolean path or branch in  $t_r$ . They are  $\{\mathbf{t}, \mathbf{ff}\}$ -correct, because correctness is prefix-independent and all boolean paths and branches in  $t_r$  are  $\{\mathbf{t}, \mathbf{ff}\}$ -correct.
2.  $t(\varepsilon) \in \{\mathbf{t}, \mathbf{ff}\}$ . Then  $\varepsilon$  is the only boolean path in  $t$ , and it is  $\{\mathbf{t}, \mathbf{ff}\}$ -correct.
3. For each  $k \geq 0$ ,  $t(l^k) = \vee$  and  $t_{l^k r} \in W^{\{\mathbf{t}, \mathbf{f}\}}$ . Each boolean path or branch is either contained in the leftmost branch—then it is  $\{\mathbf{t}, \mathbf{ff}\}$ -correct; or finishes in some subtree  $t_{l^k r}$ —then it is also  $\{\mathbf{t}, \mathbf{ff}\}$ -correct, by prefix-independence.

□

**Corollary 3.8.11** *For any alphabet  $A$ , for  $D \subseteq A \setminus \{\vee, \neg\}$  and any languages  $L, M, N \subseteq T_A$  such that if  $t \in L \cup M$  then  $t(\varepsilon) \in D$ , the following holds.*

$$P^\sigma(L, M, N) \cup N^\sigma(L, M, N) \subseteq W^D$$

**Proof:**

If  $t \in P^\sigma(L, M, N) \cup N^\sigma(L, M, N)$  then there is  $t' \in P^\sigma \cup N^\sigma$  such that  $t$  is  $t'$  with each occurrence of  $\mathbf{t}$  replaced with a tree from  $L$ , each occurrence of  $\mathbf{ff}$  replaced with a tree from  $M$ , and each occurrence of  $\top$  replaced with a tree from  $N$ . Recall that, by Proposition 3.8.10,  $t' \in W^{\{\mathbf{t}, \mathbf{f}\}}$ .

Suppose, towards a contradiction, that  $t$  has a  $D$ -incorrect boolean path or branch  $\alpha$ .

If  $\alpha$  is a branch, then either it is entirely included in  $t'$ , then it is correct—a contradiction; or there is a boolean path  $v \prec \alpha$  in  $t'$  such that  $t'(v) \in \{\mathbf{t}, \mathbf{ff}, \top\}$ . If  $t'(v) = \top$  then  $v$  is a  $\{\mathbf{t}, \mathbf{ff}\}$ -incorrect path in  $t'$ —a contradiction. If  $t'(v) \in \{\mathbf{t}, \mathbf{ff}\}$ , then  $t(v) \in D$  (by the assumption on languages  $L$  and  $M$ ) and  $\alpha$  is not a boolean branch in  $t$ , because no label different than  $\vee$  and  $\neg$  can occur on a boolean branch. Therefore, all the possible cases for  $\alpha$  being a branch lead to a contradiction.

If  $\alpha$  is a path, then, again, it could potentially be entirely included in  $t'$ . Then, since  $\alpha$  is  $D$ -incorrect and  $\{\mathbf{t}, \mathbf{ff}\}$ -correct,  $t'(\alpha) \in \{\mathbf{t}, \mathbf{ff}\}$ . Then  $t(\alpha) \in D$ , so  $\alpha$  is  $D$ -correct in  $t$ —a contradiction. Therefore, there is a boolean path  $v \prec \alpha$  in  $t'$  such that  $t'(v) \in \{\mathbf{t}, \mathbf{ff}\}$ . Then, as above,  $t(v) \in D$  and  $\alpha$  is not a boolean path in  $t$ , what yields a contradiction. □

We define the following relation on the set  $W^D$ :

$$s \sqsubset_D t \iff s = t_w \text{ for some boolean path } w \in \{l, r\}^* r \text{ in } t$$

We show that  $\sqsubset_D$  is a strict well-founded order on  $W^D$ . Suppose that  $t^1 \sqsupset_D t^2 \sqsupset_D t^3 \sqsupset_D \dots$  is an infinite  $\sqsubset_D$ -decreasing sequence of trees satisfying well-foundedness and proper shape. Then:

$$t^2 = t^1_{w_1 r}, \quad t^3 = t^2_{w_2 r}, \quad t^4 = t^3_{w_3 r}, \quad \dots$$

for some boolean paths  $w_1 r, w_2 r, w_3 r, \dots$ . By closure under the infinite concatenation,  $w_1 r w_2 r w_3 r \dots$  is a boolean branch in  $t^1$  turning right infinitely many times. This contradicts well-foundedness assumption on boolean branches of  $t^1$ .

The above also proves the antisymmetry of the relation, because if  $t_1 \sqsubset_D t_2 \sqsubset_D t_1$ , then the alternating sequence  $t_1, t_2, t_1, t_2, \dots$  is an infinite decreasing sequence satisfying the above assumptions.

Transitivity of  $\sqsubset_D$  comes immediately from closure of boolean paths under concatenation.

Let us note, that:

**Remark 3.8.12** *If  $t \in W^D$  and  $w$  is a boolean path in  $t$ , then  $t_w \in W^D$ .*

This is because boolean paths and branches in  $t_w$  are suffixes of boolean paths and branches in  $t$ , and correctness of boolean paths and branches is prefix-independent.

For any tree  $t$ ,  $\varepsilon$  is a boolean path. If this path is correct then  $t(\varepsilon) \in \{\vee, \neg\} \cup D$ . If  $t(\varepsilon) \in D$  then there are no more boolean paths or branches, so, in particular, there is no tree  $s \sqsubset_D t$ . If  $t(\varepsilon) \in \{\vee, \neg\}$  then  $r$  is a boolean path in  $t$  as well. If additionally  $t \in W^D$  then, by the above remark and the definition of  $\sqsubset_D$ ,  $t_r \sqsubset_D t$ . Finally, we obtain:

**Remark 3.8.13** *If  $t \in W^D$ , then  $t$  is a minimal element of  $\sqsubset_D$  if and only if  $t(\varepsilon) \in D$ .*

Now we prove that a kind of inverse of Corollary 3.8.11 holds.

**Proposition 3.8.14** *Let  $B, C$  be any alphabets such that  $\{\vee, \neg\} \subseteq B$  and  $\{\vee, \neg\} \cap C = \emptyset$ . Let  $D \subseteq (B \cup C) \setminus \{\vee, \neg\}$ . Let  $L, M \subseteq T_{B,C}$  be languages such that for  $t \in T_{B,C}$ , if  $t(\varepsilon) \in D$  then  $t \in L \cup M$ . Then:*

$$W_{B,C}^D \subseteq P^\sigma(L, M, T_{B,C}) \cup N^\sigma(L, M, T_{B,C})$$

**Proof:**

Knowing that relation  $\sqsubset_D$  is well-founded on  $W^D$ , we prove that each tree from  $W^D$  is in  $(P^\sigma \cup N^\sigma)(L, M, T_{B,C}) = P^\sigma(L, M, T_{B,C}) \cup N^\sigma(L, M, T_{B,C})$  by an induction with respect to  $\sqsubset_D$ .



If  $t$  is a minimal element of  $\sqsubset_D$  then, by Remark 3.8.13,  $t(\varepsilon) \in D$ . Then  $t \in L \cup M$ . Let  $t \in L$ . Thanks to the construction of  $\Phi_{P_T}$  (see equation (3.6)), and by the assumption on languages  $L$  and  $M$ , we have:

$$t \in \Phi_P \left[ \begin{array}{l} p \mapsto \emptyset, \\ n \mapsto \emptyset, \\ \textcolor{blue}{t} \mapsto L, \\ \textcolor{blue}{ff} \mapsto M, \\ \textcolor{blue}{\top} \mapsto \emptyset \end{array} \right] \subseteq \Phi_P \left[ \begin{array}{l} p \mapsto P^\sigma(L, M, T_{B,C}), \\ n \mapsto N^\sigma(L, M, T_{B,C}), \\ \textcolor{blue}{t} \mapsto L, \\ \textcolor{blue}{ff} \mapsto M, \\ \textcolor{blue}{\top} \mapsto T_{B,C} \end{array} \right] = P^\sigma(L, M, T_{B,C})$$

where the first inclusion comes from the monotonicity of  $\Phi$  and the equality comes from the fact that the pair  $(P^\sigma(L, M, T_{B,C}), N^\sigma(L, M, T_{B,C}))$  is a fixpoint of  $\Phi[\textcolor{blue}{t} \mapsto L, \textcolor{blue}{ff} \mapsto M, \textcolor{blue}{\top} \mapsto T_{B,C}]$ .

The prove that if  $t \in M$  then  $t \in N^\sigma(L, M, T_{B,C})$  is analogous.

If  $t \in W^D$  is not  $\sqsubset_D$ -minimal then  $t(\varepsilon) \in \{\vee, \neg\}$ .

Let us take  $t \in W^D$  such that  $t(\varepsilon) = \neg$ . Then, thanks to the above notes concerning  $\sqsubset_D$  relation,  $t_r \in W^D$  and  $t_r \sqsubset_D t$ . By the inductive assumption,  $t_r \in (P^\sigma \cup N^\sigma)(L, M, T_{B,C})$ . Therefore, using  $\Phi_{P_\neg}$  or  $\Phi_{N_\neg}$  we obtain:

$$t \in (\Phi_P \cup \Phi_N) \left[ \begin{array}{l} p \mapsto P^\sigma(L, M, T_{B,C}), \\ n \mapsto N^\sigma(L, M, T_{B,C}), \\ \textcolor{blue}{t} \mapsto L, \\ \textcolor{blue}{ff} \mapsto M, \\ \textcolor{blue}{\top} \mapsto T_{B,C} \end{array} \right] = (P^\sigma \cup N^\sigma)(L, M, T_{B,C})$$

If  $t \in W^D$  is such that  $t(\varepsilon) = \vee$ , then, by proper shape assumption, whole leftmost branch is labeled with  $\vee$ 's. Then for each  $k \geq 0$ ,  $l^k r$  is a boolean path, so  $t_{l^k r} \in W^D$  and  $t_{l^k r} \sqsubset_D t$ . By the inductive assumption, for each  $k \geq 0$ ,  $t_{l^k r} \in (P^\sigma \cup N^\sigma)(L, M, T_{B,C})$ . If there is such  $k$  that  $t_{l^k r} \in P^\sigma(L, M, T_{B,C})$ , then using  $\Phi_{P_\vee}$ , we get:

$$t \in \Phi_P \left[ \begin{array}{l} p \mapsto P^\sigma(L, M, T_{B,C}), \\ n \mapsto N^\sigma(L, M, T_{B,C}), \\ \textcolor{blue}{t} \mapsto L, \\ \textcolor{blue}{ff} \mapsto M, \\ \textcolor{blue}{\top} \mapsto T_{B,C} \end{array} \right] \subseteq (P^\sigma \cup N^\sigma)(L, M, T_{B,C})$$

Otherwise for each  $k$ ,  $t_{l^k r} \in N^\sigma(L, M, T_{B,C})$ , therefore, using  $\Phi_{N_\vee}$ , we get:

$$t \in \Phi_N \left[ \begin{array}{l} p \mapsto P^\sigma(L, M, T_{B,C}), \\ n \mapsto N^\sigma(L, M, T_{B,C}), \\ \textcolor{blue}{t} \mapsto L, \\ \textcolor{blue}{ff} \mapsto M, \\ \textcolor{blue}{\top} \mapsto T_{B,C} \end{array} \right] \subseteq (P^\sigma \cup N^\sigma)(L, M, T_{B,C})$$

□

The following equality will be important for the construction of automata in the following sections (recall that we use  $\sqcup$  as a union sign that additionally designates that the components are disjoint):

**Corollary 3.8.15** *For a nonempty alphabet  $A$  and a language  $L \subseteq T_A$ :*

$$W_{A_\sigma}^\# = P^\sigma [\lambda^L] \sqcup N^\sigma [\lambda^L]$$

**Proof:**

Note that  $t \in \lambda^L(\textcolor{blue}{t}) \cup \lambda^L(\textcolor{blue}{f})$  if and only if  $t(\varepsilon) = \#$ . Additionally recall that  $\{\vee, \neg, \#\} \subseteq A_\sigma$ . Therefore, by Corollary 3.8.11,  $P^\sigma [\lambda^L] \cup N^\sigma [\lambda^L] \subseteq W_{A_\sigma}^\#$  and, by Proposition 3.8.14,  $W_{A_\sigma}^\# \subseteq P^\sigma [\lambda^L] \cup N^\sigma [\lambda^L]$ . Sets  $P^\sigma [\lambda^L]$  and  $N^\sigma [\lambda^L]$  are disjoint by Corollary 3.8.4. □

We conclude the section with one more definition using relation  $\sqsubset_D$ .

Since relation  $\sqsubset_D$  is well-founded on  $W^D$ , we can use it to define a rank in a standard way. Let for a tree  $t \in W^D$ :

$$\text{rank}_D(t) = \sup\{\text{rank}_D(s) + 1 : s \sqsubset_D t\}$$

The rank of a tree is always a countable ordinal. This is because trees smaller than  $t$  with respect to  $\sqsubset_D$  are subtrees of  $t$ ,  $t$  has countably many subtrees, whereas the cofinality of  $\omega_1$  is  $\omega_1$ .

### 3.8.1 Topological Properties

First we prove the property of the sigma-lifting operation that it was designed for and that explains its name.

**Theorem 3.8.16** *Let  $L \subseteq T_A$ . If  $L$  is hard for a topological complexity class  $\mathbf{K}$  (e.g.  $\mathbf{K} = \{K : K \leq_w L\}$ ), then  $\sigma(L)$  is hard for the sigma-algebra<sup>4</sup>  $\sigma(\mathbf{K})$ .*

**Proof:**

For the sake of the proof we strengthen the claim slightly. We use substitution  $\lambda_a^L$  as in the definition of  $\sigma(L)$  (see equation (3.10)), and sets:

$$\begin{aligned} P &:= P^\sigma [\lambda_a^L] = \sigma_a(L) \\ N &:= N^\sigma [\lambda_a^L] \\ W &:= W_{A_\sigma}^\# = P \cup N \quad (\text{by Corollary 3.8.15}) \end{aligned}$$

---

<sup>4</sup>Formally,  $\sigma(K)$  is defined for  $K$  being a set of subsets of some space (see Section 1.1), while  $\mathbf{K}$  is not necessarily a set. Here, we say that a subset  $L \subseteq X$  of a space  $X$  belongs to  $\sigma(\mathbf{K})$  if it belong to the sigma algebra generated by the subsets of  $X$  that are in class  $\mathbf{K}$ .

Let  $X, Y$  be topological spaces, and let  $K \subseteq X, L_1 \subseteq Y, L_2 \subseteq Y$ . We say that a function  $f : X \rightarrow Y$  **reduces  $K$  to the pair  $(L_1, L_2)$**  if:

$$\begin{aligned} f(x) \in L_1 &\iff x \in K \\ f(x) \in L_2 &\iff x \notin K \end{aligned}$$

If  $L_1$  and  $L_2$  are disjoint, it is equivalent to:

$$\begin{aligned} x \in K &\implies f(x) \in L_1 \\ x \notin K &\implies f(x) \in L_2 \end{aligned}$$

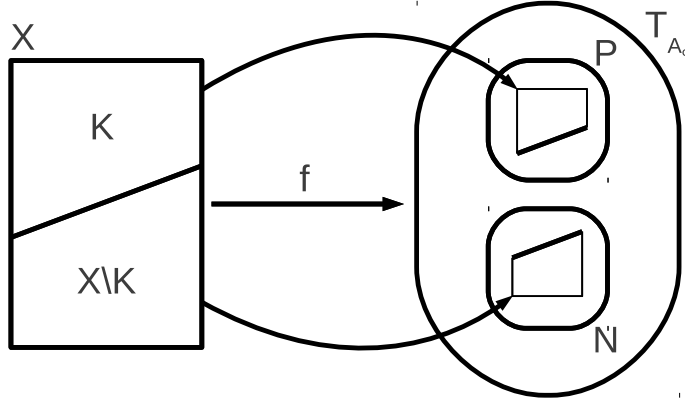


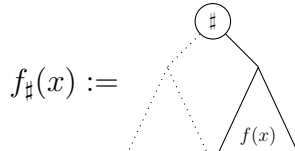
Figure 3.3: A function  $f$  reduces a set  $K$  to a pair  $(P, N)$ .

For the proof of the theorem it is enough to show that the class of sets continuously reducible to pair  $(P, N)$  contains all sets from  $\mathbf{K}$ , and is closed under complementation and countable unions.

**(sets from  $\mathbf{K}$ )** Each set  $K \subseteq X$  from class  $\mathbf{K}$  is continuously reducible to  $L$ . So there is a continuous function  $f : X \rightarrow T_A$  such that:

$$\begin{aligned} f(x) \in L &\iff x \in K, \quad \text{or equivalently:} \\ f(x) \in T_A \setminus L &\iff x \notin K. \end{aligned}$$

Let us put:



(the left subtree does not matter,  
we put an arbitrary fixed tree there)

The function is clearly continuous, because  $f$  is continuous. We show that  $f_{\#} : X \rightarrow T_{A_\sigma}$  is a needed reduction of  $K$  to  $(P, N)$ . If  $x \in K$  then, by

**(closure under the complement)** Take any set  $K \subseteq X$  continuously reducible to the pair  $(P, N)$ , and let  $f : X \rightarrow T_{A_\sigma}$  be an appropriate reducing function. We construct a reduction  $\bar{f} : X \rightarrow T_{A_\sigma}$  of  $\bar{K}$  to  $(P, N)$  by putting:

$\bar{f}(x) :=$

$$\begin{array}{ccccccc} x \in \overline{K} & \iff & x \notin K & \iff & f(x) \in N & \implies & \overline{f}(x) \in P \\ x \notin \overline{K} & \iff & x \in K & \iff & f(x) \in P & \implies & \overline{f}(x) \in N \end{array}$$

Diagram illustrating the evaluation of a function  $f(x)$  using a sequence of nested function calls. The diagram shows a sequence of nodes labeled  $V$  (representing function calls) connected by arrows, forming a chain. Each node  $V$  is associated with a function call  $f_i(x)$  shown in a triangle. The sequence starts with  $f_3(x)$  at the bottom, followed by  $f_2(x)$ ,  $f_1(x)$ , and  $f_0(x)$  at the top. The expression  $f(x) :=$  is shown to the left of the chain.

We show one more topological property of the operation for further convenience.

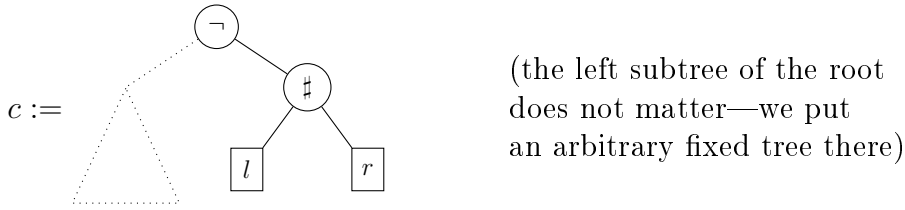
**Proposition 3.8.17** *For a nonempty alphabet  $A$ , any  $a \in A$ , and any language  $L \subseteq T_A$ :*

$$\sigma_a(L) \equiv_w \sigma_a(\overline{L})$$

**Proof:**

Since  $\overline{\overline{L}} = L$ , it suffices to prove that for any  $L$ ,  $\sigma_a(L) \leq_w \sigma_a(\overline{L})$ .

As a function that reduces  $\sigma(L)$  to  $\sigma(\overline{L})$ , we take a function  $f$  that works as the identity with an exception that each  $\sharp$ -labeled node  $v$  that does not have any  $\sharp$ -labeled node above is replaced with a context:



In the sense that  $f(t)_v$  is the context  $c$  with the hole  $l$  substituted with  $t_{vl}$  and the hole  $r$  substituted with  $t_{vr}$ .

First we prove that  $W^\sharp = f^{-1}(W^\sharp)$ .

( $\supseteq$ ) We show that if there is a  $\sharp$ -incorrect path or branch in  $t$  then there is a  $\sharp$ -incorrect path or branch in  $f(t)$ .

Suppose that there is an incorrect boolean branch  $\alpha$  in  $t$ . Since there is no  $\sharp$  labels on a boolean branch, this branch is untouched by  $f$ , and  $\alpha$  is also an incorrect boolean branch in  $f(t)$ .

Let us now assume that there is a  $\sharp$ -incorrect path  $v$  in  $t$ . One of the two possible reasons for this may be that  $t(v) \notin \{\vee, \neg, \sharp\}$ . In such a case, the path  $v$  is not changed by  $f$  and  $v$  is a  $\sharp$ -incorrect path in  $f(t)$ . The other reason may be that  $t(v) = \vee$  and  $t(vl) \neq \vee$ . If  $t(vl) \neq \sharp$ , then the whole path  $vl$  is not changed by  $f$ , and  $v$  is a  $\sharp$ -incorrect path in  $f(t)$ . If  $t(vl) = \sharp$ , then the node  $vl$  is replaced with  $c$  by  $f$ , and  $f(t)(vl) = \neg \neq \vee$ , so  $v$  is also a  $\sharp$ -incorrect path in  $f(t)$ .

( $\subseteq$ ) We show that if there is a  $\sharp$ -incorrect path or branch in  $f(t)$  then there is a  $\sharp$ -incorrect path or branch in  $t$ .

If there is an incorrect boolean branch in  $f(t)$  then it does not cross any context  $c$ , because if it were crossing  $c$  then it would go through a  $\sharp$ -labeled node and boolean branches cannot have  $\sharp$ -labeled nodes. Therefore, the branch is also included in  $t$  and has the same labels, so  $t$  has an incorrect branch.

Now assume that there is a  $\sharp$ -incorrect path  $v$  in  $f(t)$ . One of the two possible reasons for this may be that  $f(t)(v) \notin \{\vee, \neg, \sharp\}$ . Since all boolean paths that enter  $c$  end in  $\neg$  or  $\sharp$ -labeled node,  $v$  does not intersect any context

$c$ , therefore, it is included in  $t$  and is a  $\sharp$ -incorrect path there. If  $v$  is  $\sharp$ -incorrect because  $f(t)(v) = \vee$  and  $f(t)(vl) \neq \vee$ , then  $v$  does not intersect any context  $c$ , but it is possible that there is a context  $c$  rooted in node  $vl$ . Then if the context  $c$  comes from the replacement done by  $f$  then  $t(vl) = \sharp$  and  $v$  is a  $\sharp$ -incorrect path in  $t$ ; otherwise  $v$  is entirely included in  $t$  and is  $\sharp$ -incorrect there. As a result, in all the cases  $v$  is also a  $\sharp$ -incorrect path in  $t$ .

We have proven that  $t \in W^\sharp$  if and only if  $f(t) \in W^\sharp$ . Recall that  $W^\sharp = P^\sigma [\lambda_a^L] \sqcup N^\sigma [\lambda_a^L]$ . Now we only need to prove that for  $t \in W^\sharp$ :

$$t \in P^\sigma [\lambda_a^L] \iff f(t) \in P^\sigma [\lambda_a^{\bar{L}}] \quad (3.13)$$

I.e., referring to the intentional logic-like semantics of trees in  $W^\sharp$ , among trees expressing well-shaped formulas the value of a formula is preserved by  $f$ .

Thanks to the restriction to the trees in  $W^\sharp$ , we can prove equivalence (3.13) by the induction over  $\text{rank}_\sharp(t)$ .

The base of the induction involves trees with  $\sharp$  in the root (see Remark 3.8.13). Such a tree  $t$  is in  $P^\sigma [\lambda_a^L]$  if and only if  $t_r \in \widehat{\pi}_a^{-1}(L)$ . Since  $f(t)$  is context  $c$  with subtree  $t_r$  (unchanged) plugged into hole  $r$ ,  $f(t) \in P^\sigma [\lambda_a^{\bar{L}}]$  if and only if  $t_r \in \widehat{\pi}_a^{-1}(\overline{\bar{L}}) = \widehat{\pi}_a^{-1}(L)$  (see the value of  $\lambda_a^{\bar{L}}(\text{ff})$  in equation (3.10)). Therefore,  $t \in P^\sigma [\lambda_a^L]$  if and only if  $f(t) \in P^\sigma [\lambda_a^{\bar{L}}]$ .

The step of the induction is straightforward: function  $f$  is compositional, it does not change the shape of tree in  $\vee$ - or  $\neg$ -labeled node (nor in a  $\vee$ -labeled leftmost branch), so the evaluation of subtrees simply propagates.  $\square$

### 3.8.2 Automata Theoretic Properties

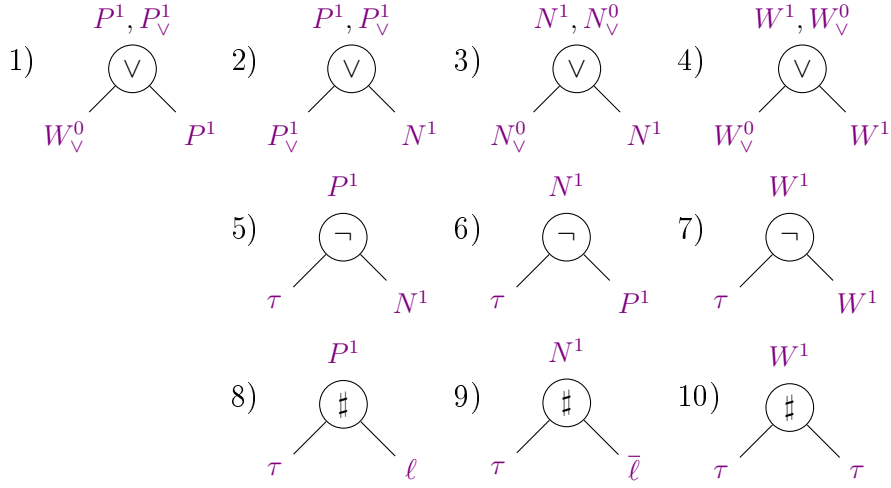
Now we prove that the sigma-lifting operation preserves bi-unambiguity.

**Theorem 3.8.18** *If a language  $L \subseteq T_A$  is bi-unambiguous and  $a \in A$ , then  $\sigma_a(L)$  is bi-unambiguous.*

Using an unambiguous automaton  $\mathcal{L}$  recognizing a language  $L \subseteq T_A$ , and an unambiguous automaton  $\bar{\mathcal{L}}$  recognizing  $\bar{L}$ , we construct an unambiguous automaton  $\mathcal{C}^{\mathcal{L}, \bar{\mathcal{L}}}$  recognizing  $\sigma_a(L)$ . The alphabet of the automaton is  $A_\sigma = A \cup \{\vee, \neg, \sharp\}$ . In the further part of the section we will construct an unambiguous automaton recognizing  $\overline{\sigma_a(L)}$ , what will conclude the proof of bi-unambiguity.

First we modify automata  $\mathcal{L}$  and  $\overline{\mathcal{L}}$  according to projection  $\widehat{\pi}_a$  to work with alphabet  $A_\sigma$ . Namely, let  $\mathcal{L}'$  (respectively  $\overline{\mathcal{L}}'$ ) be the same as  $\mathcal{L}$  (respectively  $\overline{\mathcal{L}}$ ), except that it treats letters outside  $A$  as if they were  $a$ . Then  $\mathcal{L}'$  recognizes  $\widehat{\pi}_a^{-1}(L)$ , and  $\overline{\mathcal{L}}'$  recognizes  $\widehat{\pi}_a^{-1}(\overline{L})$ . Note that the languages recognized by the two automata are complements in  $T_{A_\sigma}$ .

Now we construct a parametrized automaton  $\mathcal{C}(\ell, \overline{\ell}, \tau)$ . Apart from the parameter-states, the automaton uses states:  $P^1, P_\vee^1, N^1, N_\vee^0, W^1, W_\vee^0$ , where upper index denotes the priority of a state. States  $P^1, N^1, W^1$  are meant to recognize languages  $P^\sigma, N^\sigma, W^\sharp$  used in the definition of  $\sigma(L)$  (see equations (3.8), (3.12)), respectively. The initial state is  $P^1$ . The transitions of  $\mathcal{C}$  are as follows:



Now we put  $\mathcal{C}^{\mathcal{L}, \overline{\mathcal{L}}} := \mathcal{C}(\mathcal{L}', \overline{\mathcal{L}}', \{\top^0\})$ , where  $\{\top^0\}$  is a one-state automaton accepting every tree in  $T_{A_\sigma}$  (state  $\top^0$  is a self-looping state with even priority).

In the proofs in this section we use substitution  $\lambda_a^L$ , and languages  $P, N, W$ , as in the previous section. Let us recall their definitions for easier reference:

$$\lambda_a^L := \left\{ \begin{array}{l} \mathbf{tt} \mapsto T_{A_\sigma} \begin{array}{c} \bigcirc \# \\ \swarrow \quad \searrow \\ \widehat{\pi}_a^{-1}(L) \end{array}, \mathbf{ff} \mapsto T_{A_\sigma} \begin{array}{c} \bigcirc \# \\ \swarrow \quad \searrow \\ \widehat{\pi}_a^{-1}(\overline{L}) \end{array}, \top \mapsto T_{A_\sigma} \end{array} \right\}$$

$$\begin{array}{lll} P & := & P^\sigma [\lambda_a^L] = \sigma_a(L) \\ N & := & N^\sigma [\lambda_a^L] \\ W & := & W_{A_\sigma}^\sharp = P \cup N \end{array}$$

**Lemma 3.8.19** *If automaton  $\mathcal{L}$  recognizes  $L$  and automaton  $\overline{\mathcal{L}}$  recognizes  $\overline{L}$  then:*

$$\begin{aligned} L\left(\mathcal{C}_{P^1}^{\mathcal{L}, \overline{\mathcal{L}}}\right) &= P, \\ L\left(\mathcal{C}_{N^1}^{\mathcal{L}, \overline{\mathcal{L}}}\right) &= N, \\ L\left(\mathcal{C}_{W^1}^{\mathcal{L}, \overline{\mathcal{L}}}\right) &= W. \end{aligned}$$

**Proof:**

Note that, by the construction of transitions of  $\mathcal{C}^{\mathcal{L}, \overline{\mathcal{L}}}$ , in a run of  $\mathcal{C}^{\mathcal{L}, \overline{\mathcal{L}}}$  starting from either of states  $P^1$ ,  $N^1$ ,  $W^1$  on a tree  $t \in T_{A_\sigma}$ , each boolean path (or branch) is labeled only with states  $P^1$ ,  $P_\vee^1$ ,  $N^1$ ,  $N_\vee^0$ ,  $W^1$ ,  $W_\vee^0$ .

Using this fact we prove that each tree  $t \notin W$  is not accepted from any of states  $P^1$ ,  $N^1$ ,  $W^1$ . By the definition of  $W^\sharp$ , for such a tree one of the three holds:

1. there is a boolean path  $w$  such that  $t(w) \notin \{\vee, \neg, \sharp\}$ ,
2. there is a boolean path  $w$  such that  $t(w) = \vee$  and  $t(wl) \neq \vee$ ,
3. there is a boolean branch  $\alpha$  that turns right infinitely many times.

In the first case each run of automaton  $\mathcal{C}^{\mathcal{L}, \overline{\mathcal{L}}}$  from either of states  $P^1$ ,  $N^1$ ,  $W^1$  gets stuck in node  $w$ , because there is no transition from any of states  $P^1$ ,  $P_\vee^1$ ,  $N^1$ ,  $N_\vee^0$ ,  $W^1$ ,  $W_\vee^0$  by any letter different than  $\vee$ ,  $\neg$  and  $\sharp$ .

In the second case, node  $wl$  obtains a state label  $P_\vee^1$ ,  $N_\vee^0$ , or  $W_\vee^0$  in each run of  $\mathcal{C}^{\mathcal{L}, \overline{\mathcal{L}}}$  on  $t$ . There is no transition from any of these states for any letter different than  $\vee$ , so the run is stuck.

For the third case we note that in each run of  $\mathcal{C}^{\mathcal{L}, \overline{\mathcal{L}}}$  on  $t$ , state  $P^1$ ,  $N^1$ , or  $W^1$  occurs on branch  $\alpha$  after each turning right. Therefore, priority 1 of the automaton occurs infinitely often on this path, and the run is rejecting.

For  $t \in W$  we prove, by an induction on the rank of  $t$ , that  $t \in P$  if and only if  $t$  is accepted from state  $P^1$  and that  $t \in N$  if and only if  $t$  is accepted from state  $N^1$ .

For  $\text{rank}_\sharp(t) = 0$ , by Remark 3.8.13,  $t(\varepsilon) = \sharp$ , and then:

- from state  $W^1$ : tree  $t$  is accepted immediately—all-accepting state  $\top^0$  is assigned to both successors of the root;
- from state  $P^1$ : tree  $t$  is accepted if and only if  $t_r \in \widehat{\pi}_a^{-1}(L)$  (see transition (8)). By the construction of  $\Phi_P$  (see equation (3.6)) and the shape of the language  $\lambda_a^L(\sharp)$  (see equation (3.10)), if  $t(\varepsilon) = \sharp$  then  $t \in P$  if and only if  $t_r \in \widehat{\pi}_a^{-1}(L)$ . Therefore,  $t$  is accepted if and only if  $t \in P$ ;



- from state  $N^1$ : tree  $t$  is accepted if and only if  $t_r \in \widehat{\pi}_a^{-1}(\overline{L})$  (see transition (9)). By the construction of  $\Phi_N$  (see equation (3.7)) and the shape of the language  $\lambda_a^L(\mathbf{ff})$  (see equation (3.10)), if  $t(\varepsilon) = \sharp$  then  $t \in N$  if and only if  $t_r \in \widehat{\pi}_a^{-1}(\overline{L})$ . Therefore,  $t$  is accepted if and only if  $t \in N$ .

If  $\text{rank}_{\sharp}(t) > 0$ , then tree  $t$  is of one of the forms:

1. It has  $\neg$  in the root and subtree  $t_r \in W$  is a tree of smaller rank. Then, by the inductive assumption,  $t_r$  is accepted from  $W^1$ , so  $t$  is accepted from  $W^1$  ((7) is the only transition from  $W^1$  for letter  $\neg$ ).

Such a tree is accepted from state  $P^1$  if and only if  $t_r$  is accepted from state  $N^1$  ((5) is the only transition from  $P^1$  for letter  $\neg$ ), so, by the inductive assumption, if and only if  $t_r \in N$ . By the fixpoint definition of  $P^\sigma$ ,  $t \in P$  if and only if  $t_r \in N$  (see  $\Phi_{P^-}$ ), so among trees  $t \in W$  with  $\neg$  in the root exactly those are accepted from state  $P^1$  that belong to  $P$ .

We similarly prove analogous claim for state  $N^1$  and set  $P$ .

2. The whole leftmost branch is labeled with  $\vee$ , and for each  $n$ ,  $t_{l^n r} \in W$  is a tree of smaller rank than  $t$ . Then, by the inductive assumption, for each  $n$ , tree  $t_{l^n r}$  is accepted from state  $W^1$ . Then, since there is a run of  $\mathcal{C}^{\mathcal{L}, \overline{\mathcal{L}}}$  from state  $W^1$  on  $t$  that labels whole leftmost branch (except the root) with  $W_\vee^0$ , and each node of the form  $l^n r$  with state  $W^1$  (see transition (4)), also  $t$  is accepted from  $W^1$ .

Let us now consider a run of  $\mathcal{C}^{\mathcal{L}, \overline{\mathcal{L}}}$  from state  $N^1$  on such a tree  $t$ . The leftmost branch (except the root) in such a run is always labeled with state  $N_\vee^0$ , and nodes  $l^n r$ —with state  $N^1$  (transition (3) is the only one from state  $N^1$  or  $N_\vee^0$  for letter  $\vee$ ). Hence, the tree is accepted if and only if each of trees  $t_{l^n r}$  is accepted from  $N^1$ , what, by the inductive assumption, holds if and only if  $t_{l^n r} \in N$  for each  $n$ , i.e. if and only if  $t \in N$  (see  $\Phi_{N_\vee}$ ).

Now consider a run from state  $P^1$ . Such a run could assign state  $P_\vee^1$  to all nodes on the leftmost branch (using transition (2)), but then it would not be accepting, because  $P_\vee^1$  has an odd priority. Therefore, in order to accept, the automaton has to use transition (1) in some node on the leftmost branch. As a result, the tree can only be accepted if one of subtrees  $t_{l^n r}$  is accepted from state  $P^1$ , what, by the inductive assumption, holds if and only if this subtree is in  $P$ .

Now we show an accepting run from state  $P^1$  on tree  $t$  with  $\vee$  at the leftmost branch and at least one subtree  $t_{l^n r}$  in  $P$ . It uses transition

(1) as soon as possible, i.e. in the node  $l^n$  for the least  $n$  such that  $t_{l^n r}$  belongs to  $P$ . By the inductive assumption,  $t_{l^n r}$  is accepted from state  $P^1$ . In this run all nodes  $l^m r$  for  $m < n$  are labeled with state  $N^1$ . By the inductive assumption and the fact that  $P$  and  $N$  partition  $W$  (see Corollary 3.8.15), subtrees  $t_{l^m r}$  for  $m < n$  are accepted from  $N^1$ . For  $m \geq 1$ , nodes  $l^{n+m}$  obtain state label  $W_\vee^0$ , and nodes  $l^{n+m} r$  label  $W^1$  in the run (see transition (4)). Since subtrees  $t_{l^{n+m} r}$  belong to  $W$ , they are accepted from state  $W^1$ , from the inductive assumption. Also the leftmost branch of the run satisfies the parity condition, since  $W_\vee^0$  has priority 0. As a result whole run is accepting, and we have proven that  $t$  is accepted from state  $P^1$  if and only if one of the subtrees  $t_{l^n r}$  is in  $P$ , i.e. if and only if  $t \in P$  (see  $\Phi_{P_\vee}$  in equation (3.6)).

□

Since  $\sigma_a(L) = P$ , we obtain in particular:

**Corollary 3.8.20** *Automaton  $\mathcal{C}^{\mathcal{L}, \bar{\mathcal{L}}}$  recognizes language  $\sigma(L)$ .*

Now we use Lemma 3.8.19 in the proof of the following:

**Lemma 3.8.21** *If automata  $\mathcal{L}$  and  $\bar{\mathcal{L}}$  are unambiguous and  $s$  is a single state of  $\mathcal{C}^{\mathcal{L}, \bar{\mathcal{L}}}$ , then the automaton  $\mathcal{C}_s^{\mathcal{L}, \bar{\mathcal{L}}}$  is unambiguous.*

**Proof:**

First observe that if automata  $\mathcal{L}$  and  $\bar{\mathcal{L}}$  are unambiguous then also automata  $\mathcal{L}'$  and  $\bar{\mathcal{L}}'$  are unambiguous, by Fact 3.6.6(3).

Component  $\{\top^0\}$  is obviously unambiguous.

Now assume, to the contrary, that there are two different accepting runs  $\rho_1$  and  $\rho_2$  of automaton  $\mathcal{C}_s^{\mathcal{L}, \bar{\mathcal{L}}}$  on some tree  $t$ . Let  $v$  be such a node of  $t$  that runs  $\rho_1$  and  $\rho_2$  differ on  $v$ , but they are the same on each prefix of  $v$ . Note that  $v \neq \varepsilon$ , because  $\mathcal{C}_s^{\mathcal{L}, \bar{\mathcal{L}}}$  has only one initial state. Note that  $v$  cannot be in the subtree where the runs are already in one of the components  $\mathcal{L}'$ ,  $\bar{\mathcal{L}}'$  or  $\{\top^0\}$ , because then there would be two accepting runs on the subtree, which is impossible since  $\mathcal{L}'$ ,  $\bar{\mathcal{L}}'$  and  $\{\top^0\}$  are unambiguous.

The only nondeterminism in  $\mathcal{C}(\ell, \bar{\ell}, \tau)$  part of automaton  $\mathcal{C}_s^{\mathcal{L}, \bar{\mathcal{L}}}$  occurs in state  $P^1$  or  $P_\vee^1$  for label  $\vee$ . Therefore, node  $v$  is labeled with  $\vee$  and, without loss of generality, run  $\rho_1$  takes transition (1) and run  $\rho_2$  takes transition (2) in this node. As a result  $\rho_1$  assigns state  $P^1$  to node  $vr$ , while  $\rho_2$  assigns state  $N^1$  to this node. By Lemma 3.8.19 and Corollary 3.8.4, languages accepted from states  $P^1$  and  $N^1$  are disjoint, so at least one of the runs is rejecting, what yields a contradiction. □

**Corollary 3.8.22 (of Lemma 3.8.21 and Corollary 3.8.20)** *If a language  $L \subseteq T_A$  is bi-unambiguous then language  $\sigma(L) \subseteq T_{A_\sigma}$  is unambiguous.*

Now we need to show that language  $\overline{\sigma(L)}$  is also unambiguous.

Recall that, by Corollary 3.8.15,  $\overline{\sigma(L)}$  is the disjoint union of  $\overline{W}$  and  $N$ . Since the class of unambiguous languages is closed under disjoint unions, and  $N$  is recognized by an unambiguous automaton  $\mathcal{C}_{N^1}^{\mathcal{L}, \overline{\mathcal{L}}}$ , we only need to show how to recognize trees in  $\overline{W}$  unambiguously.

Similarly as we have done for the set  $G$  in Lemma 3.7.1, we proceed by proving the following:

**Lemma 3.8.23** *If  $t \notin W^D$ , then  $t$  has the rightmost  $D$ -incorrect boolean path or branch.*

**Proof:**

Take  $t \in \overline{W^D}$ . By the definition of  $W^D$ ,  $t$  has a  $D$ -incorrect boolean path or branch (we call both of them paths for the sake of this proof). We construct the rightmost  $D$ -incorrect boolean path starting from the root, maintaining the following invariants: 1) whenever we descend to some subtree, it has a  $D$ -incorrect boolean path, 2) the path constructed so far (call it  $\rho$ ) is a boolean path, 3) there is no  $D$ -incorrect boolean path  $\eta$  such that for  $v$  being a common prefix of  $\rho$  and  $\eta$ ,  $vl \preceq \rho$  and  $vl \not\preceq \eta$ . The initial one-node path is boolean (as  $\varepsilon$  is a boolean path in each tree), and the invariants are satisfied.

We consider the following cases:

1. If label in the current node is  $\neg$  then we descend right. We prove that this move maintains invariants. First observe that  $r$  is a boolean path in the current subtree (subtree rooted in the current node), so invariant (2) is maintained by closure of boolean paths under concatenation. Now, if current subtree has an incorrect boolean path, then this path has to turn right in  $\neg$ -labeled node (path  $\varepsilon$  is not incorrect), and right subtree also has an incorrect boolean path (by prefix-independence), so invariant (1) is maintained. Invariant (3) is trivially maintained, since we turn right.
2. If the current node is labeled with  $\vee$  and the right subtree has an incorrect path, then we descend right. It trivially maintains invariants (1) and (3). It also maintains invariant (2), because  $r$  is a boolean path in  $\vee$ -rooted tree.
3. If the current node is labeled with  $\vee$ , the right subtree has no incorrect boolean path, and the left successor node is labeled with something

different than  $\vee$ , then we finish the construction—the path constructed so far is the desired incorrect boolean path.

4. If the current node is labeled with  $\vee$ , the right subtree has no incorrect boolean path, and the left successor node is labeled with  $\vee$ , then we descend left. In this case the root of the subtree does not constitute an incorrect boolean path. Since the current subtree has an incorrect boolean path and the right subtree does not have one, the left subtree has to have an incorrect boolean path (again by prefix-independence, because both  $r$  and  $l$  are boolean paths in the current subtree). Therefore, invariant (1) is maintained in this case. Invariant (2) follows from the fact that the left successor, that we descend to, is labeled with  $\vee$ . Invariant (3) is maintained, because the right subtree has no  $D$ -incorrect boolean path and the current path is not  $D$ -incorrect.
5. If the current node is labeled with something outside  $\{\vee, \neg\} \cup D$ , then we finish the construction—the path constructed so far is the desired incorrect boolean path.

By invariant 1, at any step of the construction, the current node cannot be labeled with a symbol from  $D$  (trees with a label from  $D$  in the root do not have  $D$ -incorrect paths). Therefore, the cases considered above exhaust the possibilities.

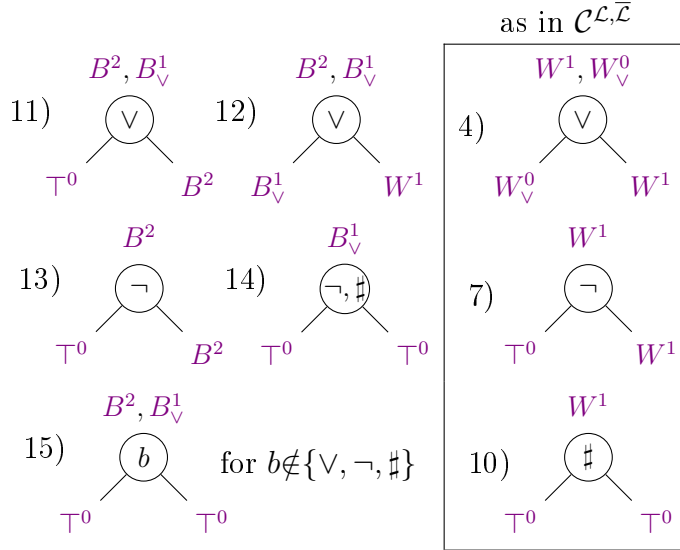
Let us denote the constructed path by  $\rho$ . By invariant (3), there is no  $D$ -incorrect boolean path to the right from  $\rho$ . It suffices to prove that  $\rho$  is indeed a  $D$ -incorrect boolean path.

If  $\rho$  is finite, then it was finished in case 3 or case 5 of the construction. If it was case 3 then  $\rho$  is a boolean path ending with label  $\vee$  such that  $t(\rho l) \neq \vee$ , so it is an incorrect boolean path. In case 5  $\rho$  is a boolean path ending with label different than  $\vee, \neg$ , and not in  $D$ , so it is also  $D$ -incorrect.

Let us now consider the case of  $\rho$  infinite. Suppose additionally, to the contrary, that  $\rho$  has finitely many turnings right. Then  $\rho = \rho_0 l^\omega$ , for some  $\rho_0 \in \{l, r\}^*$  such that for each  $n \geq 0$ ,  $t(\rho_0 l^n) = \vee$  ( $\rho$  is a boolean branch). However, by invariant (1), subtree  $t_{\rho_0}$  has an incorrect boolean path. Since  $l^\omega$  is not incorrect in this subtree, there has to be some incorrect path diverging right in some point  $l^n$ . It contradicts invariant (3), therefore,  $\rho$  turns right infinitely many times, so it is an incorrect boolean branch.  $\square$

Now we are ready to construct an unambiguous automaton  $\mathcal{B}$  recognizing language  $\overline{W} = \overline{W}_{A_\sigma}^\sharp$ . It selects the rightmost  $\sharp$ -incorrect boolean path and verifies the selection.

The automaton uses states  $B^2$ ,  $B_\vee^1$ ,  $W^1$ ,  $W_\vee^0$ ,  $\top^0$ . States  $B^2$  and  $B_\vee^1$  are used to track the path, states  $W^1$  and  $W_\vee^0$  serve showing non-existence of an incorrect boolean path in a subtree, and  $\top^0$  is an all-accepting state, as before. Again, upper index designates the priority of a state. The initial state is  $B^2$ . The transitions are as follows:



**Lemma 3.8.24** *Automaton  $\mathcal{B}$  recognizes language  $\overline{W}$ .*

**Proof:**

First note that, in any run  $\rho$  of automaton  $\mathcal{B}$  on tree  $t \in T_{A_\sigma}$ , states  $B^2$  and  $B_\vee^1$  mark a single path starting from the root, and they cannot occur outside this one path. Let us call this path  $\beta_\rho$ . The path descends downwards only if the conditions of a boolean branch are satisfied: no transition extends this path in case of label different than  $\vee$  and  $\neg$ ; it turns right after label  $\neg$ ; if label different than  $\vee$  occurs after turning left in  $\vee$ -labeled node, the path is terminated (transition (12) is then followed by (14) or (15)). As a result, either the path is infinite—then it is a boolean branch; or  $\beta_\rho$  is a boolean path; or  $\beta_\rho = wl$  for  $w \in \{l, r\}^*$ , where  $w$  is a boolean path.

Let us consider any branch  $\eta$  containing path  $\beta_\rho$ . The construction of the transitions of the automaton implies that the priorities of states on this branch satisfy the parity condition if and only if one of the three holds:

1.  $\eta = \beta_\rho$ , so  $\beta_\rho$  is a boolean branch labeled only with states  $B^2$  and  $B_\vee^1$ . Because of the priorities,  $B^2$  has to occur infinitely often and it occurs only after turning right;
2.  $\beta_\rho = wl$  for some boolean path  $w \in \{l, r\}^*$ , where  $t(w) = \vee$  and  $t(wl) \neq \vee$ —transition (14) or transition (15) is taken in node  $wl$ , that

assign all-accepting state  $\top^0$  to both successor nodes, so in particular to the one contained in  $\eta$  that extends  $wl$  in this case;

3.  $t(\beta_\rho) \notin \{\vee, \neg, \sharp\}$ —transition (15) is taken in node  $\beta_\rho$ , all-accepting state is assigned to both successor nodes, as above.

The first case witnesses existence of an incorrect boolean branch, while the latter two correspond to  $\sharp$ -incorrect boolean paths. This shows that a run cannot be accepting if a tree does not contain any  $\sharp$ -incorrect path or branch.

Now we need to prove that there always is an accepting run  $\rho$  on a tree with an incorrect path or branch. We use Lemma 3.8.23, and consider a run  $\rho$  that marks the rightmost incorrect path with  $B$ -states, i.e.  $\beta_\rho$  contains the rightmost incorrect boolean path. Then, by the above observations,  $\rho$  is accepting on all branches containing  $\beta_\rho$ . Additionally, every subtree that diverges to the right from  $\beta_\rho$  contains no incorrect branch, i.e. is in  $W$ . Then we can use Lemma 3.8.19 (remember that  $W$ -part of automaton  $\mathcal{B}$  is exactly the same as in automaton  $\mathcal{C}^{\mathcal{L}, \bar{\mathcal{L}}}$ ), to conclude that those subtrees can be accepted from state  $W^1$ . Since  $\rho$  assigns state  $W^1$  to nodes descending to the right from  $\beta_\rho$  (see transition (12)), the subtrees are accepted. The subtrees descending to the left from  $\beta_\rho$  are always accepted, because transitions (11) and (13) assign state  $\top^0$  to them.  $\square$

**Lemma 3.8.25** *Automaton  $\mathcal{B}$  is unambiguous.*

**Proof:**

The only nondeterminism in automaton  $\mathcal{B}$  occurs in states  $B^2$ ,  $B_\vee^1$  for label  $\vee$ . Two transitions are possible from such configurations: transition (11) assigns state  $B^2$  to the right successor, while transition (12) assigns state  $W^1$  to the right successor. Analogously as in the proof of Lemma 3.8.21, we use the fact that languages  $L(\mathcal{B}_{B^2}) = \bar{W}$  and  $L(\mathcal{B}_{W^1}) = W$  are disjoint (see Lemma 3.8.24 and Lemma 3.8.19), to conclude that only one of the transitions can be used in an accepting run from a given node in a given tree.  $\square$

**Proof (of Theorem 3.8.18):**

If language  $L$  is bi-unambiguous and unambiguous automata  $\mathcal{L}$  and  $\bar{\mathcal{L}}$ , recognize  $L$  and  $\bar{L}$ , respectively, then:

- Language  $\sigma(L)$  is recognized by unambiguous automaton  $\mathcal{C}^{\mathcal{L}, \bar{\mathcal{L}}}$  (by Corollary 3.8.20 and Lemma 3.8.21),

- Language  $\overline{\sigma(L)}$  is unambiguous, as a disjoint union of language recognized by unambiguous automaton  $\mathcal{C}_{N^1}^{\mathcal{L}, \bar{\mathcal{L}}}$  and language recognized by unambiguous automaton  $\mathcal{B}$  (by lemmas 3.8.19, 3.8.24, 3.8.21 and 3.8.25).

Therefore, language  $\sigma(L)$  is bi-unambiguous.  $\square$

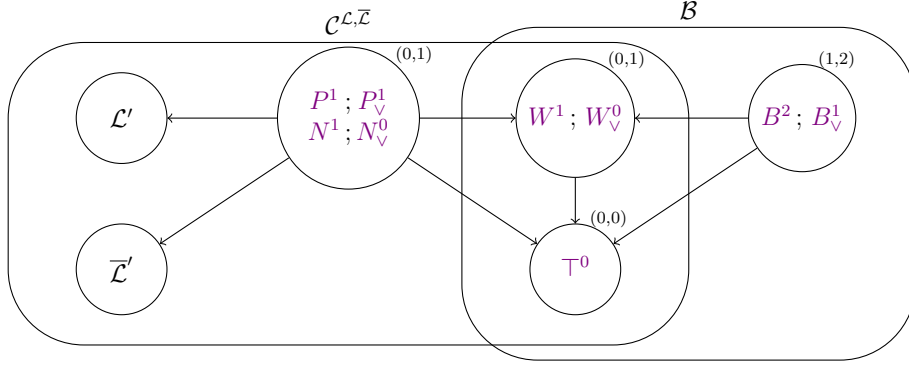


Figure 3.4: Diagram of strongly connected components of automata  $\mathcal{C}^{\mathcal{L}, \bar{\mathcal{L}}}$  and  $\mathcal{B}$ .

Looking at the diagram of strongly connected components of the constructed automata depicted in Figure 3.4, we conclude that:

**Remark 3.8.26** *If a language  $L$  is recognized by an unambiguous (respectively nondeterministic) automaton of index  $I$ , and  $\bar{L}$  is recognized by an unambiguous (respectively nondeterministic) automaton of index  $J$ , then  $\sigma(L)$  is recognized by an unambiguous (respectively nondeterministic) automaton of index  $\{(0, 1)\} \cup I \cup J$ , and  $\overline{\sigma(L)}$  is recognized by an unambiguous (respectively nondeterministic) automaton of index  $\{(0, 1), (1, 2)\} \cup I \cup J$ .*

### 3.8.3 Iteration Potential

At the first glance it is not clear whether  $\sigma(\sigma(L)) \neq \sigma(L)$ . It occurs that, under relatively weak assumptions, not only this is true, but also iterating sigma operation yields topologically harder and harder sets. The assumptions were inspired by the property introduced in a paper by Arnold and Niwiński [AN07]:

**Definition 3.8.27** *Let  $X$  be a metric space. We say that a set  $L \subseteq X$  has a **Reduction Contractability Property**<sup>5</sup> if for each metric space  $Y$  and*

<sup>5</sup>Arnold and Niwiński do not use this name, but they have introduced the notion.

each set  $K \subseteq Y$  such that  $L \leq_w K$ , there is a contracting<sup>6</sup> reduction of  $L$  to  $K$ .

Under this definition, Lemma 2 of the mentioned paper [AN07] states that game languages  $W_{(\iota, \kappa)}$  have Reduction Contractability Property.

Now we show where the contractability can be used. The following also comes from the paper [AN07]:

**Proposition 3.8.28** *Let  $X$  be a complete metric space, and let  $L \subseteq X$ . If  $L$  has Reduction Contractability Property then  $\bar{L} \not\leq_w L$ .*

**Proof:**

Let us assume, towards contradiction, that  $\bar{L}$  continuously reduces to  $L$ . Then, by the same reduction,  $L$  continuously reduces to  $\bar{L}$ . Since  $L$  has Reduction Contractability Property, there is a contracting reduction  $r$  of  $L$  to  $\bar{L}$ . Space  $X$  is complete and  $r : X \rightarrow X$  is a contraction, so, by Banach Fixpoint Theorem, it has a fixed point  $x \in X$ . Since  $r$  is a reduction, we have:

$$x \in L \iff f(x) \in \bar{L} \iff x \in \bar{L},$$

where the latter comes from the fact, that  $f(x) = x$ . We have obtained a contradiction.  $\square$

In this section we show that, in a sense, sigma-lifting operation preserves Reduction Contractability Property. Precisely we only prove that a stronger property is preserved. The proof is an adaptation of the proof by Arnold and Niwiński of their lemma [AN07, Lemma 2].

**Definition 3.8.29 (Stretching)** *Let  $(X, d)$  be a metric space, let  $L \subseteq X$ , and let  $\{a_n\}$  be a sequence of natural numbers. A mapping  $s : X \rightarrow X$  is a **stretching** of  $L$  with respect to  $\{a_n\}$  if it reduces  $L$  to itself and satisfies for each  $k \geq 0$  and each  $t_1, t_2 \in X$ :*

$$d(t_1, t_2) \leq 2^{-k} \implies d(s(t_1), s(t_2)) \leq 2^{-a_k} \quad (3.14)$$

*We also say that  $s$  **stretches**  $L$  with respect to  $\{a_n\}$ .*

Note that if  $\{a_n\}$  tends to infinity then a stretching with respect to  $\{a_n\}$  is continuous.

The name stretching comes from the context of trees, and the following note explains its etymology.

---

<sup>6</sup>See Definition 1.1.5.



**Remark 3.8.30** *If  $X$  is a space of trees, inequality 3.14 is equivalent to saying that the first  $a_k$  levels of the resulting tree are determined by the first  $k$  levels of an argument-tree.*

**Definition 3.8.31 (Stretchability)** *Let  $(X, d)$  be a metric space. A language  $L \subseteq X$  is **stretchable** if for each sequence  $\{a_n\}$  of natural numbers, there is a stretching of  $L$  with respect to  $\{a_n\}$ .*

In our context, stretchability is indeed sufficient for Reduction Contractability. Namely:

**Proposition 3.8.32** *If  $A$  is a finite alphabet, and language  $L \subseteq T_A$  is stretchable, then  $L$  has Reduction Contractability Property.*

**Proof:**

Take  $L \subseteq T_A$  as in the claim of the proposition. Let  $Y$  be an arbitrary metric space,  $K \subseteq Y$ , and let  $f : T_A \rightarrow Y$  be a continuous reduction of  $L$  to  $K$ . Now recall that each continuous mapping from a compact space is uniformly continuous<sup>7</sup>. Since  $A$  is finite, space  $T_A$  is compact; so, in particular, there is a function  $\delta_f : (0, \infty) \rightarrow (0, \infty)$ , such that for each  $\varepsilon > 0$ :

$$d_{T_A}(t_1, t_2) \leq 2^{-\delta_f(\varepsilon)} \implies d_Y(f(t_1), f(t_2)) \leq \varepsilon,$$

where  $d_{T_A}$  is the metric in  $T_A$ , and  $d_Y$  is the metric in  $Y$ .

Since  $L$  is stretchable, we can take a stretching  $s$  of  $L$  with respect to sequence  $a_k = \lceil \delta_f(2^{-k-1}) \rceil$ . We show that  $f \circ s$  is a needed contracting reduction of  $L$  to  $K$ . For each  $k \geq 0$  we have:

$$\begin{aligned} d_{T_A}(t_1, t_2) \leq 2^{-k} &\implies d_{T_A}(s(t_1), s(t_2)) \leq 2^{-\lceil \delta_f(2^{-k-1}) \rceil} \leq 2^{-\delta_f(2^{-k-1})} \\ &\implies d_Y(f(s(t_1)), f(s(t_2))) \leq 2^{-k-1} \end{aligned}$$

Since the set of all possible positive values of  $d_{T_A}$  is  $\{2^{-k} : k \in \mathbb{N}\}$ , and since  $s$  reduces  $L$  to  $L$ , and  $f$  reduces  $L$  to  $K$ ,  $f \circ s$  is a contracting reduction of  $L$  to  $K$  with constant  $\frac{1}{2}$ .

Figure 3.5 presents the above argument for  $Y$  being a tree space. □

**Theorem 3.8.33** *If  $A$  is a finite nonempty alphabet and a language  $L \subseteq T_A$  is stretchable then  $\sigma(L)$  is also stretchable.*

---

<sup>7</sup>See e.g. [Kur72, XVI.5, Theorem 4].

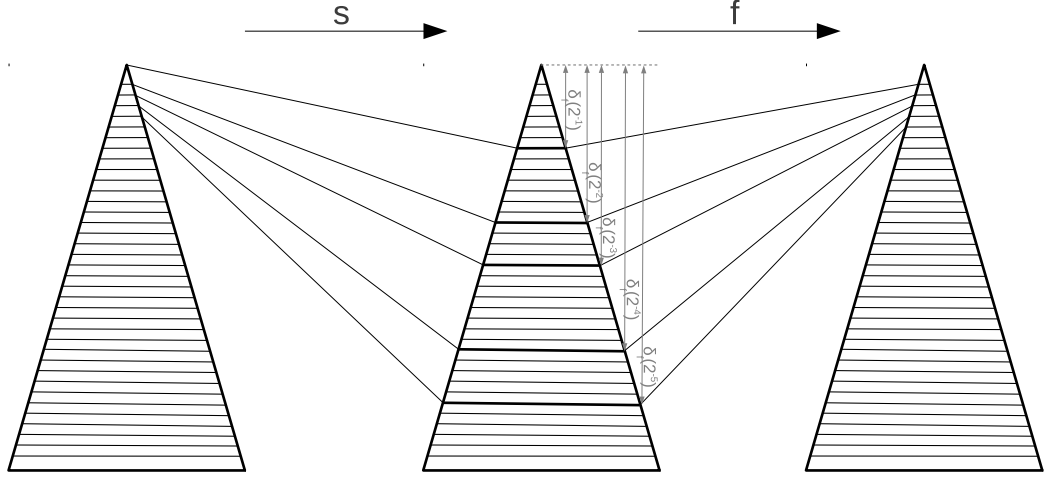


Figure 3.5: The use of stretching  $s$  to make  $f$  contracting. The first  $k$  levels of  $f(t)$  are determined by  $\lceil \delta_f(2^{-k}) \rceil$  levels of  $t$ , while the first  $\lceil \delta_f(2^{-k-1}) \rceil$  levels of  $s(t)$  are determined by  $k$  levels of  $t$ . As a result the first  $k + 1$  levels of  $f(s(t))$  are determined by the first  $k$  levels of  $t$ .

It may seem that the easiest way to prove the above theorem is to use a compositionality of stretchability. One might want to have a lemma that says that if a parametrized language  $L(x_1, x_2, \dots, x_k)$  is stretchable, and languages  $L_1, L_2, \dots, L_k$  are stretchable, then language  $L[x_1 \mapsto L_1, x_2 \mapsto L_2, \dots, x_k \mapsto L_k]$  is stretchable. However, we conjecture that such a general statement is not true. Intuitively, one of the problems here is that if we do not know where the parts coming from languages  $L_i$  are plugged, we have no clue what reduction to apply: the one for language  $L$  or one of the ones for  $L_i$  languages. If we even know that the latter, then for which of  $L_i$  languages. In general, the reductions for all the considered languages can be different, and it may not be possible to find the common reduction for them.

We now introduce notions that allow us to state the lemma as abstract and general to be reusable in the sequel, and as specific to be provable. We also want the lemma to provide a way to prove stretchability by looking only at an automaton recognizing a language. The statement of the lemma to occur may look complex, but the set of assumptions is natural and well tight to our setting.

First we introduce the following definition:

**Definition 3.8.34 (Acceptance Profile)** *Let  $\mathcal{A}$  be a parity tree automaton (potentially parametrized) over an alphabet  $A$ , with state set  $Q$ . Let  $S \subseteq Q$  be any subset of states. For a context  $c$  with holes  $v_1, v_2, \dots, v_k$*

we define the ***S-acceptance profile*** of  $c$  in  $\mathcal{A}$  as the following relation  $R \subseteq S \times (\mathbb{N} \times Q)^k$ :

$$(p, m_1, q_1, m_2, q_2, \dots, m_k, q_k) \in R \iff \begin{array}{l} \text{There exists a run } \rho \text{ of } \mathcal{A} \text{ on} \\ c \text{ from state } p \text{ that is accepting} \\ \text{on all branches that are entirely} \\ \text{contained in } c, \text{ and for each } i, \\ \rho(v_i) = q_i \text{ and the maximal pri-} \\ \text{ority on the path from the root} \\ \text{of } c \text{ to } v_i \text{ is } m_i. \end{array}$$

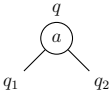
If  $S = Q$  then the above is simply called ***A-acceptance profile*** of  $c$ .

It can be seen that the acceptance profile of a context stores enough information about the context to study accepting runs of  $\mathcal{A}$  on trees that contain it.

**Definition 3.8.35 (Simultaneous Stretchability)** Sets  $L_1, L_2, \dots, L_k$  are ***simultaneously stretchable*** if for each sequence  $\{a_n\}$  there is a function that stretches each of the sets with respect to  $\{a_n\}$ .

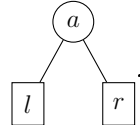
**Lemma 3.8.36 (Stretchability is Quasi-Compositional)** Let  $L_1, L_2, \dots, L_k \subseteq T_A$ . Let  $\mathcal{M} = \langle A, Q, \{x_1, x_2, \dots, x_k\}, \delta, I, o \rangle$  be a parametrized automaton. Let  $S = Q \setminus \{x_1, x_2, \dots, x_k\}$ . Assume that:

1.  $\{x_1, x_2, \dots, x_k\} \cap I = \emptyset$ .
2. Positions on which parameter states occur in a run are determined by the label of the parent node and by the direction to which the node descends from the parent. Formally, there is a function  $p : (A \times \{l, r\}) \rightarrow$

$\{0, 1\}$  such that for each transition  in  $\mathcal{M}$ :

$$\begin{aligned} q_1 \in \{x_1, x_2, \dots, x_k\} &\iff p(a, l) = 1 \\ \wedge \quad q_2 \in \{x_1, x_2, \dots, x_k\} &\iff p(a, r) = 1. \end{aligned}$$

3. For each  $a \in A$  and each  $n < \omega$  there is a two-hole context  $c_a^n$  with holes  $l$  and  $r$  at the level at least  $n$ , that has the same *S-acceptance profile* in

$\mathcal{M}(L_1, L_2, \dots, L_k)$  as context: .

Additionally we require that if  $v = v_1 v_2 \dots v_d$  is a hole in  $c_a^n$  then for each  $0 \leq m \leq d-2$ ,  $p(c_a^n(v_1 v_2 \dots v_m), v_{m+1}) = 0$ , i.e. none of parameter-states can occur on the path from the root to a hole (except possibly in the hole itself).

4. For each  $n < \omega$ , there is a one-hole context  $c_0^n$  with the hole at the level at least  $n$ , such that the states reachable in the hole from initial states of  $\mathcal{M}$  are exactly the initial states. Formally, the  $I$ -acceptance profile of  $c_0^n$  is a relation  $R \subseteq I \times \mathbb{N} \times Q$  that projected to the third coordinate equals  $I$ .

Additionally we require that if  $v = v_1 v_2 \dots v_d$  is a hole in  $c_0^n$  then for each  $0 \leq m \leq d-1$ ,  $p(c_0^n(v_1 v_2 \dots v_m), v_{m+1}) = 0$ , i.e. none of parameter-states can occur on the path from the root to the hole (including the hole itself).

5.  $L_1, L_2, \dots, L_k$  are simultaneously stretchable.

Then language  $M := L(\mathcal{M}(L_1, L_2, \dots, L_k))$  is stretchable.

**Proof:**

Let languages  $L_1, L_2, \dots, L_k$  and an automaton  $\mathcal{M}$  be as in the statement of the lemma. For a given sequence  $\{a_n\}$ , let  $s_{\{a_n\}}^L$  be a simultaneous stretching of all languages  $L_1, L_2, \dots, L_k$  with respect to sequence  $\{a_n\}$ . The existence of such a stretching is guaranteed by assumption 5 of the lemma. Let, additionally,  $p$  be a function determining transitions to parameter-states, as in assumption 2 of the lemma.

Take any sequence of natural numbers  $\{a_n\}$ . We show a stretching  $s_{\{a_n\}} : T_A \rightarrow T_A$  of  $M$  with respect to  $\{a_n\}$ . For  $t \in T_A$ , we construct  $s_{\{a_n\}}(t)$  from the root downwards, putting  $c_0^n$  (from assumption 4) first, then replacing each node of  $t$  labeled with  $a$  with appropriate context  $c_a^n$  (from assumption 3) until reaching a position  $v$  for which  $p$  is 1, when we plug  $s_{\{b_n\}}^L(t_v)$ , for sequence  $\{b_n\}$  appropriately calculated from  $\{a_n\}$ . The details follow.

For a sequence  $\alpha$ , let us define a  $k$ -shift  $\alpha^{(k)}$  of this sequence by  $\alpha_n^{(k)} = \alpha_{n+k}$ .

For a sequence  $\alpha$ , stretching  $s_\alpha$  can be defined by equations:

$$s_\alpha(x) = \begin{array}{c} \triangle \\ c_0^{\alpha_0} \\ \triangle \\ \bar{s}_{\alpha(1)}(x) \end{array}$$

$$\bar{s}_\beta(y) =$$

$$h_\gamma^b(z) = \begin{cases} \bar{s}_{\gamma(1)}(z) & \text{if } b = 0 \\ s_\gamma^L(z) & \text{if } b = 1 \end{cases}$$

Note that this set of equations uniquely defines function  $s_\alpha$ . This is because in the right hand side of the equation for  $\bar{s}_\beta$  at least one level of the tree is fixed—precisely  $\beta_0$  levels of context  $c_{y(\varepsilon)}^{\beta_0}$ .

Now we prove that constructed function  $s_{\{a_n\}}$  stretches trees as desired. By a straightforward induction over the levels we obtain that, as long as function  $p$  returns 0, a node at level  $k$  (assuming the root is at level 0) is replaced with a context that has at least  $a_{k+1}$  levels fixed (holes are below this level) by  $s_{\{a_n\}}$ . Additionally the first  $a_0$  levels are fixed independently on an argument-tree, because  $c_0^{a_0}$  is put as an initial part of the resulting tree. As a result, in such a case the first  $k$  levels of an argument-tree  $t$  determine  $\sum_{i=0}^k a_i$  levels of  $s_{\{a_n\}}(t)$ . Since  $\{a_n\}$  is a sequence of positive numbers,  $\sum_{i=0}^k a_i \geq a_k$ , therefore, by Remark 3.8.30, we get the stretching condition for the paths on which function  $p$  returns only 0.

If function  $p$  returns 1 for the first time on some path of  $t$  in a node  $v$ , then subtree  $t_v$  turns into  $s_{\{a_n\}^{(|v|)}}^L(t_v)$  in  $s_{\{a_n\}}(t)$ . Since  $s_{\{a_n\}^{(|v|)}}^L$  is a stretching with respect to  $\{a_n\}^{(|v|)}$ , the first  $k$  levels of  $t_v$  determine the first  $a_{|v|+k}$  levels of  $s_{\{a_n\}^{(|v|)}}^L(t_v)$ . Remember that  $v$  is the first occurrence of the value 1 for  $p$  on the path, so  $s_{\{a_n\}^{(|v|)}}^L(t_v)$  is plugged at the level at least  $\sum_{i=0}^{|v|} a_i$  into  $s_{\{a_n\}}(t)$ . As a result, when we restrict to the nodes inside  $s_{\{a_n\}^{(|v|)}}^L(t_v)$ , at least  $a_{|v|+k} + \sum_{i=0}^{|v|} a_i \geq a_{|v|+k}$  levels of  $s_{\{a_n\}}(t)$  are determined by  $|v| + k$  levels of  $t$ . This again corresponds to the condition for stretchability given by Remark 3.8.30, hence we have proven that  $s_{\{a_n\}}$  stretches trees accordingly.

Now we need to prove that  $s_{\{a_n\}}$  reduces  $M$  to itself. Equivalently, for each  $t \in T_A$ ,  $t$  is accepted by  $\mathcal{M}(L_1, L_2, \dots, L_k)$  if and only if  $s_{\{a_n\}}(t)$  is

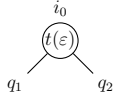
accepted by  $\mathcal{M}(L_1, L_2, \dots, L_k)$ .

( $\Rightarrow$ )

This direction of the proof is quite straightforward. Let us assume that there is an accepting run  $\rho$  of  $\mathcal{M}(L_1, L_2, \dots, L_k)$  on  $t$ . We construct an accepting run  $\rho'$  on  $s_{\{a_n\}}(t)$  based on  $\rho$ . Let us construct the run from the root. Recall that  $s_{\{a_n\}}(t)$  is built of appropriate contexts  $c_a^n, c_0^n$  and of subtrees coming from applications of  $s_\alpha^L$  (for appropriate sequences  $\alpha$ ) to subtrees of  $t$ . We maintain the invariant that the root of a context  $c_a^n$  or a subtree  $s_\alpha^L$  being a building block of  $s_{\{a_n\}}(t)$  obtains the same state label in  $\rho'$  as the corresponding node of  $t$  obtains in  $\rho$ .

Let  $i_0$  be the initial state of the run  $\rho$ . The initial part of  $s_{\{a_n\}}(t)$  is the context  $c_0^{a_0}$ . By assumption 4, there is a run on this context starting from some initial state of  $\mathcal{M}(L_1, L_2, \dots, L_k)$  that assigns  $i_0$  to the hole. We put this run at the beginning of  $\rho'$ .

There is  $c_{t(\varepsilon)}^{a_1}$  plugged below  $c_0^{a_0}$  in  $s_{\{a_n\}}(t)$ . The root of this context is labeled with state  $i_0$  in  $\rho'$ , so the declared invariant holds. If  $\rho$  takes a

transition  in the root, then, thanks to assumption 3, there is a run

on  $c_{t(\varepsilon)}^{a_1}$  starting from  $i_0$  that is accepting on all branches contained in  $c_{t(\varepsilon)}^{a_1}$ , assigns  $q_1$  to the left hole,  $q_2$  to the right hole, has no priority greater than the priorities of  $i_0$  and  $q_1$  on the path from the root to the left hole and has no priority greater than the priorities of  $i_0$  and  $q_2$  on the path from the root to the right hole. We use this run on  $c_{t(\varepsilon)}^{a_1}$  in  $\rho'$ . This maintains the invariant for the successor nodes of the root. We use the same construction for both successor nodes, their successors, and so on, as long as we do not meet a node for which function  $p$  returns 1.

If function  $p$  returns 1 for a considered node  $v$  (precisely for the label of the parent of  $v$  and for the direction into which  $v$  descends from the parent), then, by assumption 2,  $\rho(v) = x_i$  for some  $i$ . Since  $\rho$  is accepting, this means that  $t_v \in L_i$ . Remember that, by the construction, this subtree is turned into  $s_{\{a_n\}(|v|)}^L(t_v)$  in  $s_{\{a_n\}}(t)$ . Since  $s_{\{a_n\}(|v|)}^L$  is a reduction of  $L_i$  to itself,  $s_{\{a_n\}(|v|)}^L(t_v) \in L_i$ . Therefore, subtree  $s_{\{a_n\}(|v|)}^L(t_v)$  is accepted from  $x_i$  and this is the state that  $\rho'$  assigns to the the root of this subtree, by the invariant.

Let us show that the constructed run is accepting. As it was already noted, acceptance condition holds on the branches entirely contained in one of the contexts  $c_a^n$  or  $c_0^n$ . The same for the paths that end up in some of the subtrees coming from applications of  $s_\alpha^L$ —run on such paths ends in one of  $x_i$  states, and the appropriate subtree is accepted from this state. We are

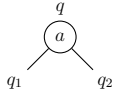
left with showing that the acceptance condition holds on branches that cross infinitely many contexts  $c_a^n$ . Each such branch in  $s_{\{a_n\}}(t)$  comes from some branch in  $t$  and the run  $\rho'$  on such branch has the same maximal priority occurring infinitely often as the corresponding branch in  $\rho$ . Therefore, the acceptance condition is satisfied on all branches of  $s_{\{a_n\}}(t)$ , and the run is accepting.

( $\Leftarrow$ )

This direction of the proof is only a bit more subtle. Now we assume that there is an accepting run  $\rho$  of  $\mathcal{M}(L_1, L_2, \dots, L_k)$  on  $s_{\{a_n\}}(t)$  and we construct an accepting run  $\rho'$  on  $t$  based on  $\rho$ . As above, reduction  $s_{\{a_n\}}$  sets the decomposition of  $s_{\{a_n\}}(t)$  into contexts  $c_a^n, c_0^n$  and subtrees coming from application of  $s_\alpha^L$  (for appropriate sequences  $\alpha$ ), that correspond to nodes and subtrees of  $t$ .

Consider the run  $\rho$  on the context  $c_0^{a_0}$  being an initial part of  $s_{\{a_n\}}(t)$ . Since, by assumption 4, none of the parameter-states  $x_1, x_2, \dots, x_k$  can occur on the path from the root to the hole, there is a state  $i_0$  that is assigned by  $\rho$  to the hole of the context. Also by assumption 4,  $i_0$  is initial. We start  $\rho'$  from state  $i_0$ .

Let us now assume that we have constructed run  $\rho'$  on  $t$  until a point when it assigns a state  $q \in S$  to an  $a$ -labeled node  $v$  that is replaced with a context  $c = c_a^n$  in  $s_{\{a_n\}}(t)$ . Let us also assume that so far we have maintained an invariant that  $\rho'$  assigns to a node the same state that  $\rho$  assigns to the root of the context or subtree that corresponds to this node in  $s_{\{a_n\}}(t)$  (in particular, all ancestor-nodes of  $v$  have corresponding contexts in  $s_{\{a_n\}}(t)$ ). This in particular means that the state assigned by  $\rho$  to the root of the considered context  $c$  is  $q$ . We use the part of assumption 3 that requires non-existence of any of states  $x_i$  on the path from the root to a hole, to note that there are states  $q_1, q_2$  assigned by  $\rho$  to the respective holes of the context. Thanks to the first part of assumption 3, there is a transition



in  $\mathcal{M}(L_1, L_2, \dots, L_k)$ . We put this transition onto  $v$  in  $\rho'$ . This maintains the invariant. Note that assumption 3 implies that there is no priority greater than both the priorities of  $q$  and  $q_1$  (respectively  $q_2$ ) on the path from the root of  $c$  to the left (respectively right) hole of  $c$  in  $\rho$ .

Let us now assume that the above part of the construction of run  $\rho'$  assigns state  $x_i$  for some  $i$  to a node  $v$ . By assumption 2, it means that function  $p$  returns 1 for node  $v$ . Therefore,  $t_v$  was turned into  $c_v := s_{\{a_n\}}^L(|v|)(t_v)$  in  $s_{\{a_n\}}(t)$ . Moreover, by the maintained invariant,  $\rho$  assigns state  $x_i$  to the root of  $c_v$ . Since  $\rho$  is accepting,  $c_v$  is accepted from  $x_i$ , hence  $c_v \in L_i$ . Therefore,

since  $s_{\{a_n\}^{(|v|)}}^L$  is a reduction of  $L_i$  to  $L_i$ , also  $t_v \in L_i$ , so  $t_v$  is accepted from  $x_i$ .

The important point of this construction is that, thanks to the strong assumptions 2, 3, 4 of the lemma, holes of the contexts defined on  $s_{\{a_n\}}(t)$  by  $s_{\{a_n\}}$  that are assigned states  $x_i$  are uniformly determined for all runs and correspond exactly to the positions in  $t$  for which function  $p$  returns 1. As a result, each run on  $s_{\{a_n\}}(t)$  terminates with one of parameter-states  $x_i$  in exactly those places where the construction have put respective  $s_{\{a_n\}^{(|v|)}}^L(t_v)$  subtrees. Note that here we use the assumption that  $s_{\{a_n\}^{(|v|)}}^L$  is a simultaneous stretching for languages  $L_1, L_2, \dots, L_k$ , because it is not determined which of the states  $x_i$  is assigned to a given hole.

The acceptance of the constructed run  $\rho'$  is justified exactly the same way as in the other direction of the proof.  $\square$

The proof of the theorem is now a straightforward application of the lemma.

**Proof (of Theorem 3.8.33):**

Take  $L \subseteq T_A$  as in the claim of the theorem, and any  $a \in A$ .

Recall that automaton  $\mathcal{C}^{\mathcal{L}, \bar{\mathcal{L}}} = \mathcal{C}(\mathcal{L}', \bar{\mathcal{L}}', \{\top^0\})$  recognizes  $\sigma_a(L)$ , where:  $\mathcal{L}'$  recognizes  $\hat{\pi}_a^{-1}(L)$ ,  $\bar{\mathcal{L}}'$  recognizes  $\hat{\pi}_a^{-1}(\bar{L})$ , and one-state component  $\{\top^0\}$  recognizes  $T_{A_\sigma}$ . Therefore, by Remark 3.2.6:

$$L(\mathcal{C}^{\mathcal{L}, \bar{\mathcal{L}}}) = L(\mathcal{C}(\hat{\pi}_a^{-1}(L), \hat{\pi}_a^{-1}(\bar{L}), T_{A_\sigma}))$$

We apply Lemma 3.8.36 to automaton  $\mathcal{C}(\hat{\pi}_a^{-1}(L), \hat{\pi}_a^{-1}(\bar{L}), T_{A_\sigma})$ .

Since none of parameter-states of  $\mathcal{C}$  is initial, assumption 1 of Lemma 3.8.36 is satisfied.

Now we show that the automaton satisfies assumption 2 of the lemma. For this we define function  $p : (A \times \{l, r\}) \rightarrow \{0, 1\}$  as follows:

$$p(b, d) = 1 \iff b = \sharp \vee (b = \neg \wedge d = l)$$

This reflects the fact that automaton  $\mathcal{C}$  enters one of the parameter-states  $\ell, \bar{\ell}, \tau$  always after seeing label  $\sharp$  and in the left successor of  $\neg$ -labeled node, and it does not enter any of these components in any other case.

Let  $\{a_n\}$  be any sequence of natural numbers and let  $s_{\{a_n\}} : T_A \rightarrow T_A$  be a stretching of  $L$  with respect to  $\{a_n\}$ . Such a stretching exists by the assumption of the theorem. Note that  $s'_{\{a_n\}} : T_{A_\sigma} \rightarrow T_{A_\sigma}$  defined by  $s'_{\{a_n\}} = s_{\{a_n\}} \circ \hat{\pi}_a$  is a stretching of  $\hat{\pi}_a^{-1}(L)$ , because  $\hat{\pi}_a^{-1}(L) \cap T_A = L$ . Since  $\hat{\pi}_a^{-1}(\bar{L}) = \hat{\pi}_a^{-1}(\bar{L})$ , the same function also stretches  $\hat{\pi}_a^{-1}(\bar{L})$ . Additionally



Now we need to construct contexts that are required in assumptions 4 and 3 of the lemma. First we fix any tree  $t_0 \in T_A$ . We use the following tree as a building block in our construction:

Note that  $t_N$  is accepted by  $\mathcal{C}(\hat{\pi}_a^{-1}(L), \hat{\pi}_a^{-1}(\overline{L}), T_{A_\sigma})$  from states  $N^1$ ,  $W^1$  and  $\tau$ , and rejected from each of the states  $P^1$ ,  $P_\vee^1$ ,  $N_\vee^0$ ,  $W_\vee^0$ .

[illegible]

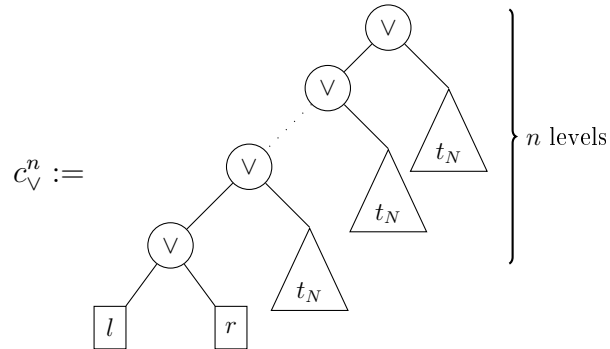
145

starts with state  $P^1$  (the only initial state of  $\mathcal{C}$ ), then it has to assign state  $P^1$  to the hole—otherwise it would end up labeling whole left-most branch with states  $P^1$  and  $P_\vee^1$ , and the labeling of this branch would not satisfy the acceptance condition. Hence, the acceptance profile of  $c_0^n$  projected to the last coordinate consists of the only initial state of  $\mathcal{C}$ . Additionally the hole occurs below level  $n$ , and no parameter-state can occur on the path from the root to the hole, hence context  $c_0^n$  satisfies conditions described in assumption 4 of Lemma 3.8.36.

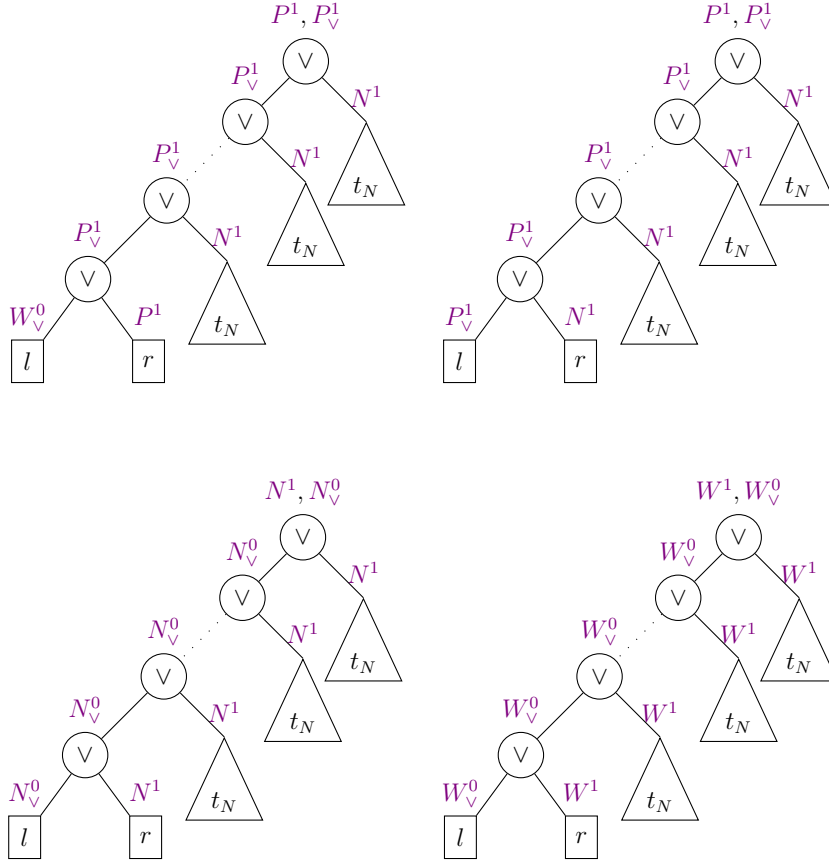
Now, for each  $n$  and each label  $b \in A_\sigma$ , we need to define a context  $c_b^n$  satisfying conditions from assumption 3 of the lemma. Note that the set of non-parameter states of  $\mathcal{C}$  is  $S := \{P^1, P_\vee^1, N^1, N_\vee^0, W^1, W_\vee^0\}$ . We will consider  $S$ -acceptance profiles of the contexts. It will be clear, from the construction of the transitions of the automaton, that no parameter-state can occur on the path from the root to a hole in any of the considered contexts, so we will not mention this any more.

For labels other than  $\vee$ ,  $\neg$  and  $\sharp$ , a run, when in one of the states from  $S$ , gets stuck. Therefore, the  $S$ -acceptance profile of one-node context with such a label is empty. Hence, for  $b \in A_\sigma \setminus \{\vee, \neg, \sharp\}$ , we can take any context with  $b$  in the root and holes at the required level as  $c_b^n$ .

Now we present contexts for labels  $\vee$ ,  $\neg$ , and  $\sharp$ , together with all possible accepting runs on those contexts starting with states from  $S$ . It is easy to verify that those runs are the only ones accepting on all branches entirely contained in the contexts and that the resulting  $S$ -acceptance profiles are the same as acceptance profiles of one-node contexts labeled with  $\vee$ ,  $\neg$ , or  $\sharp$ , respectively.

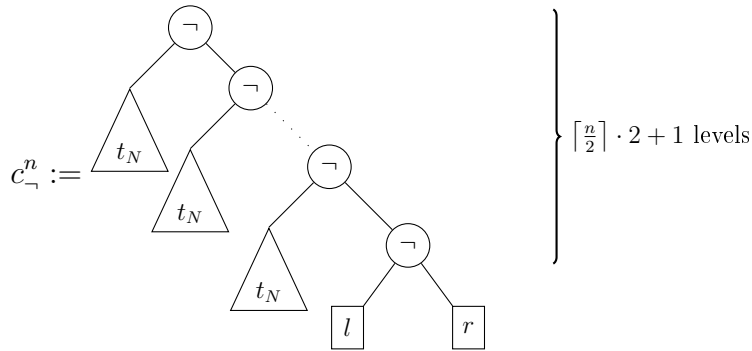


The accepting runs on  $c_{\vee}^n$ :

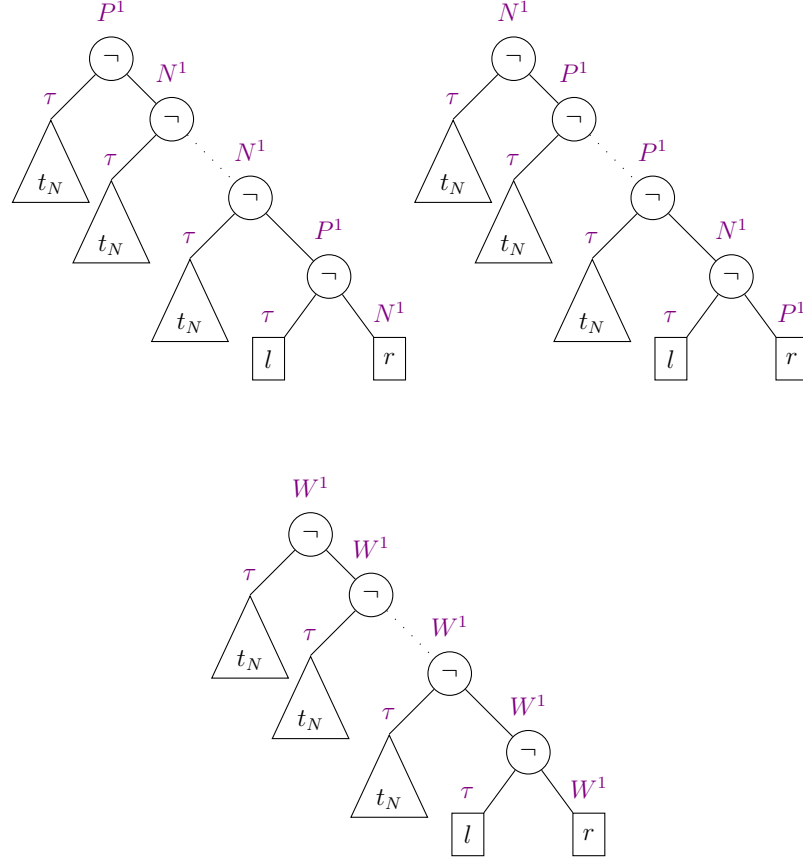


The runs correspond respectively to transitions (1), (2), (3), (4) over  $\vee$  in the automaton  $\mathcal{C}$ . Hence, the context  $c_{\vee}^n$  has the same  $S$ -acceptance profile as the context with one  $\vee$ -labeled inner node.

Now the next context:

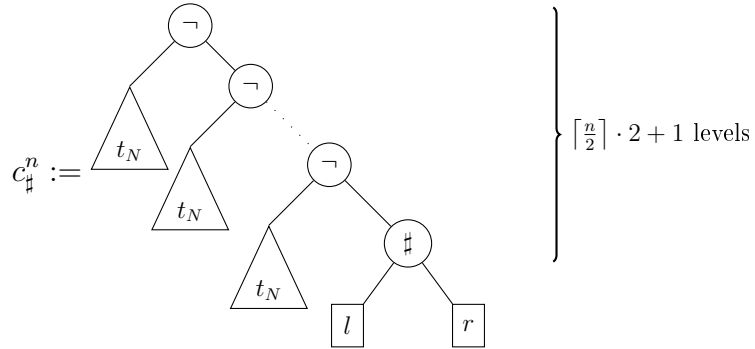


The accepting runs on  $c_{\neg}^n$ :

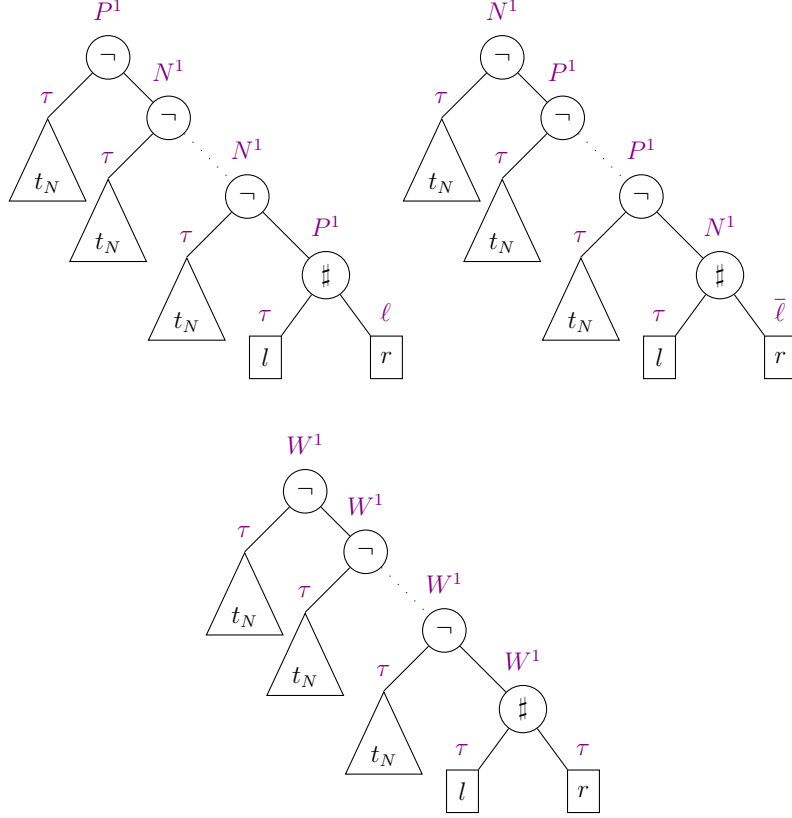


The runs correspond respectively to transitions (5), (6), (7) over  $\neg$  in the automaton  $\mathcal{C}$ . Hence, the context  $c_{\neg}^n$  has the same  $S$ -acceptance profile as the context with one  $\neg$ -labeled inner node.

And the last context:



The accepting runs on  $c_{\sharp}^n$ :



These runs correspond respectively to transitions (8), (9), (10) over  $\sharp$  in the automaton  $\mathcal{C}$ . Hence, the context  $c_{\sharp}^n$  has the same  $S$ -acceptance profile as the context with one  $\sharp$ -labeled inner node.

The presented contexts provide satisfaction of assumption 3 of Lemma 3.8.36—the last assumption to be proved. Therefore, the claim of the lemma holds for language  $\sigma_a(L)$ , and  $\sigma_a(L)$  is stretchable (for any  $a$ ).  $\square$

Finally, we come to the iteration mentioned in the title of the section.

**Theorem 3.8.37** *If  $A$  is a finite nonempty alphabet and  $L \subseteq T_A$  is stretchable then, for each  $n \geq 0$ ,  $\sigma^n(L) <_w \sigma^{n+1}(L)$ .*

**Proof:**

Inequality  $\sigma^n(L) \leq_w \sigma^{n+1}(L)$  is obvious (e.g. by Theorem 3.8.16).

For  $n = 0$ ,  $\sigma^n(L) = L \subseteq T_A$  and for  $n > 0$ ,  $\sigma^n(L) \subseteq T_{A_\sigma}$  (remember that we did not assume that  $\vee$ ,  $\neg$ , and  $\sharp$  are new symbols). By straightforward induction from Theorem 3.8.33 we obtain that for each  $n \geq 0$ ,  $\sigma^n(L)$  is stretchable, so, by Proposition 3.8.32, it has Reduction

Contractability Property. Since spaces  $T_A$  and  $T_{A\sigma}$  are complete, by Proposition 3.8.28,  $\overline{\sigma^n(L)} \not\leq_w \sigma^n(L)$ . On the other hand, by Theorem 3.8.16, we have  $\overline{\sigma^n(L)} \leq_w \sigma^{n+1}(L)$ . As a result we obtain that  $\sigma^{n+1}(L) \not\leq_w \sigma^n(L)$ .  $\square$

### 3.9 Lower Bound Pushed Up

Since the empty set  $\emptyset$  is obviously stretchable (each function reduces  $\emptyset$  to  $\emptyset$ ) and bi-unambiguous (trivial one-state, hence unambiguous, automata recognize both  $\emptyset$  and  $\overline{\emptyset}$ ), by Theorem 3.8.37 and Theorem 3.8.18, languages  $\sigma^n(\emptyset)$  constitute a strictly increasing (with respect to the Wadge order) sequence of bi-unambiguous languages.

$$\emptyset <_w \sigma(\emptyset) <_w \sigma^2(\emptyset) <_w \sigma^3(\emptyset) <_w \dots \quad (3.15)$$

Each consecutive language is topologically harder than the whole sigma-algebra generated by all the sets reducible to the previous language in the sequence (by Theorem 3.8.16).

Now we locate this sequence in the context of previously known examples.

For this we need to estimate the topological complexity of set  $\sigma(\emptyset)$ . Since sigma-algebra  $\sigma(\{\emptyset\})$  is still a trivial set, namely  $\sigma(\{\emptyset\}) = \{\emptyset, \overline{\emptyset}\}$ , it is not enough to use Theorem 3.8.16. We prove the following:

**Proposition 3.9.1** *Set  $\overline{\sigma(\emptyset)}$  is  $\Sigma_1^1$ -complete.*

**Proof:**

**(Hardness)** Thanks to Proposition 3.8.17, we can prove that  $\overline{\sigma(T_{\{\vee, \neg, \# \}})} \subseteq T_{\{\vee, \neg, \# \}}$  is  $\Sigma_1^1$ -hard, instead.

To achieve this we continuously reduce set  $\text{IF}^1 \subseteq T^{\mathbb{N}}$  of ill-founded trees on  $\mathbb{N}$ , which is  $\Sigma_1^1$ -complete (see Fact 2.4.4), to the set  $\overline{\sigma(T_{\{\vee, \neg, \# \}})}$ . Recall that  $T^{\mathbb{N}}$  is a set of unlabeled  $\mathbb{N}$ -branching trees. A tree is ill-founded if it has an infinite branch.

We construct a continuous reduction  $f : T^{\mathbb{N}} \rightarrow T_{\{\vee, \neg, \# \}}$ . Let  $t \in T^{\mathbb{N}}$  and let us note that each node  $v \in \{l, r\}^*$  of a tree from  $T_{\{\vee, \neg, \# \}}$  is of the form:

$$l^{n_1} r l^{n_2} r \dots l^{n_k} r l^{n_{k+1}}$$

for some  $k \geq 0$ , and some  $n_1, n_2, \dots, n_k, n_{k+1} \geq 0$ . We define:

$$f(t)(l^{n_1} r l^{n_2} r \dots l^{n_k} r l^{n_{k+1}}) = \begin{cases} \vee & \text{if } n_1 n_2 \dots n_k \in \text{dom}(t) \\ \# & \text{otherwise} \end{cases}$$

Since each node of  $f(t)$  depends on one node of  $t$ , the function is continuous.

Now we prove that  $f(t) \in \overline{\sigma\left(T_{\{\vee, \neg, \sharp\}}\right)} \iff t \in \text{IF}^1$ .

If  $t \in \text{IF}^1$  then there exists an infinite branch in  $t$ , hence, by the definition of  $f$ , there is an infinite  $\vee$ -labeled branch  $\alpha$  in  $f(t)$  that turns right infinitely many times. Branch  $\alpha$  is then an  $\sharp$ -incorrect boolean branch, so  $f(t) \in \overline{W^\sharp} \subseteq \overline{\sigma\left(T_{\{\vee, \neg, \sharp\}}\right)}$ .

If  $t \notin \text{IF}^1$  then all branches in  $t$  are finite, and it is not hard to see that, by the definition of  $f$ , all boolean paths and branches in  $f(t)$  are  $\sharp$ -correct. Therefore,  $t \in W^\sharp$ , and we are left with checking whether  $f(t) \in P^\sigma\left[\lambda^{T_{\{\vee, \neg, \sharp\}}}\right]$  or  $f(t) \in N^\sigma\left[\lambda^{T_{\{\vee, \neg, \sharp\}}}\right]$  (see Corollary 3.8.15).

Note that trees in  $f(T^\mathbb{N})$  only have labels  $\vee$  and  $\sharp$ . We prove by induction on  $\text{rank}_\sharp$  that if a tree in  $W^\sharp$  has only labels  $\vee$  and  $\sharp$  then it is in  $P^\sigma\left[\lambda^{T_{\{\vee, \neg, \sharp\}}}\right]$ .

If  $\text{rank}_\sharp(t) = 0$ , then  $t(\varepsilon) = \sharp$  (see Remark 3.8.13). In such a case  $t \in P^\sigma\left[\lambda^{T_{\{\vee, \neg, \sharp\}}}\right]$  if and only if  $t_r \in \widehat{\pi}_a^{-1}\left(T_{\{\vee, \neg, \sharp\}}\right) = T_{\{\vee, \neg, \sharp\}}$ . Therefore, the claim holds for each tree of rank 0.

If  $\text{rank}_\sharp(t) > 0$ , then the whole left-most branch is labeled with  $\vee$  (remember that we consider only trees in  $W^\sharp \cap T_{\{\vee, \sharp\}}$ ). By Remark 3.8.12, all the subtrees  $t_{l^{n_r}}$  are in  $W^\sharp$ . They also have only labels  $\vee$  and  $\sharp$ , therefore, by the inductive assumption, they are in  $P^\sigma\left[\lambda^{T_{\{\vee, \neg, \sharp\}}}\right]$ . Hence,  $t \in P^\sigma\left[\lambda^{T_{\{\vee, \neg, \sharp\}}}\right]$ .

We have shown that if  $t \notin \text{IF}^1$  then  $f(t) \in P^\sigma\left[\lambda^{T_{\{\vee, \neg, \sharp\}}}\right] \subseteq \overline{\sigma\left(T_{\{\vee, \neg, \sharp\}}\right)}$ , so  $f(t) \in \overline{\sigma\left(T_{\{\vee, \neg, \sharp\}}\right)}$ .

**(Upper Bound)** By Corollary 3.5.10, it is enough to prove that  $\overline{\sigma(\emptyset)}$  is recognized by an alternating Büchi automaton.

Let us note that language  $\overline{\sigma(\emptyset)}$  is the (non disjoint) union of the following languages:

1.  $\overline{W^\sharp}$ ,
2. Trees on which player  $\forall$  wins a game satisfying the following conditions:
  - (a) A configuration of the game consists of a position in a tree and a state from set  $\{P, N\}$ .
  - (b) A play starts in the root of a tree in state  $P$ .
  - (c) If a current node of a tree is labeled with  $\vee$ , then the possible moves are to one of the successor nodes. A move is chosen by

player  $\exists$  if in state  $P$  and by player  $\forall$  if in state  $N$ . Both possible moves from a  $\vee$ -labeled node preserve state.

- (d) There is only one possible move from a  $\neg$ -labeled node. The move descends to the right successor and switches a state.
- (e) When a play reaches  $\sharp$ -labeled node it is won by player  $\forall$  if in state  $P$ , and by player  $\exists$  if in state  $N$ .
- (f) Additionally none of the players can move left infinitely many times in a row.

Note that point 2 does not specify the winning condition for plays that turn right infinitely many times. This is because all trees on which such a play is possible are considered in point 1.

It is not hard to see that for  $t \in W^\sharp$ , player  $\forall$  has a winning strategy in a game satisfying conditions from point 2 if and only if  $t \in N^\sigma [\lambda^\emptyset]$ . Remember that  $\overline{\sigma(\emptyset)} = \overline{W^\sharp} \cup N^\sigma [\lambda^\emptyset]$ .

Now we prove that both the above languages are recognized by a Büchi automaton.

By Lemma 3.8.24, language  $\overline{W^\sharp}$  is recognized by the automaton  $\mathcal{B}$ . Automaton  $\mathcal{B}$  is not a Büchi automaton. It uses priorities 0, 1, 2. Note that the  $W$ -part of the automaton was introduced only to ensure unambiguity. If we do not care for unambiguity, we can eliminate this part (precisely we replace it with the all-accepting state  $\top^0$ ) and what we are left with is an automaton that tracks incorrect path or branch and uses only priorities 1 and 2, hence is Büchi.

Now we observe that a game satisfying conditions from point 2 can be implemented by an alternating Büchi automaton. First consider finite plays. Recall that such a play always ends in a  $\sharp$ -labeled node. In an automaton this situation can be handled by entering self-looping state of the automaton after seeing label  $\sharp$ . The state is accepting or rejecting depending on the state of a play: if a play is in state  $P$  then the state of the automaton is rejecting (odd priority), otherwise it is accepting (even priority).

For infinite plays we need to ensure that none of the players turns left from some moment on (as required by condition (2f)). We do it by assigning to a state a priority that is not preferable for the player that makes choice from this state, i.e. states in which player  $\exists$  makes a choice have odd priority and those in which player  $\forall$  makes a choice have even priority. It makes each play that is played by one player from some moment on, without crossing a  $\sharp$ -labeled node, losing for this player. Since players alternate only in  $\neg$ -labeled nodes and a play always turns right there (see condition (2d)), the above in particular causes that a player that turns left in infinitely many consecutive



moves loses. This also makes some plays that turn infinitely many times right losing, but we do not care who wins such plays in this construction because they all are handled in point 1.

Note that in the above discussion we only refer to parity of the priorities of states, not the actual priority value. This means that the above construction can be implemented by both an automaton of index  $(0, 1)$  and by an automaton of index  $(1, 2)$ . Note that the languages recognized by the two automata (the one of index  $(0, 1)$  and the one of index  $(1, 2)$ ) may be different, but they are the same if intersected with  $W^\sharp$ .  $\square$

Thanks to the above proposition we see that  $\overline{\sigma(\emptyset)}$  is topologically equivalent (Wadge equivalent) to the set  $G$  from Section 3.7 (see Proposition 3.7.4). The construction of the language  $L(\mathcal{C})$  in Section 3 of the paper that the set  $G$  comes from [Hum12] is not described as a generic operation, but, in fact, it is an application of an operation equivalent to  $\sigma$  to  $G$ . As a result we get:

$$\begin{aligned} G &\equiv_w \overline{\sigma(\emptyset)} \\ L(\mathcal{C}) &\equiv_w \sigma\left(\overline{\sigma(\emptyset)}\right) \equiv_w \sigma^2(\emptyset) \end{aligned}$$

According to author's knowledge, the two sets,  $G$  and  $L(\mathcal{C})$ , were the most topologically complex bi-unambiguous languages known before introduction of the construction presented in this thesis.

We only note for the completeness, that it can be easily proven looking at the automaton from Section 2 of the paper [Hum12] and using Lemma 3.8.36 that language  $G$  is stretchable. Hence, the sequence (3.15) could be alternatively built based on language  $G$ —it would only cause one-level shift.

We finish this section by putting the above sequence into the context of index hierarchies. By Remark 3.8.26, we have:

**Remark 3.9.2** *For any  $n \geq 0$ , both languages  $\sigma^n(\emptyset)$  and  $\overline{\sigma^n(\emptyset)}$  are of unambiguous index  $\mathcal{L}^{uamb}((0, 1), (1, 2))$ .*

For  $n \geq 2$ , the above remark gives the tight index-complexity bound even in the context of alternating index hierarchy. To prove this we recall that alternating automata of index  $(0, 1)$  and of index  $(1, 2)$  can recognize only sets in  $\Pi_1^1$  and  $\Sigma_1^1$ , respectively (by Corollaries 3.5.11 and 3.5.10). By Proposition 3.9.1 and by duality,  $\sigma(\emptyset)$  is  $\Pi_1^1$ -complete, therefore, by Proposition 3.8.37,  $\sigma^2(\emptyset) \notin \Pi_1^1$ . On the other hand, since, by Remark 3.8.17,  $\sigma^2(\emptyset) \equiv_w \sigma\left(\overline{\sigma(\emptyset)}\right)$  and since  $\overline{\sigma(\emptyset)}$  is  $\Sigma_1^1$ -complete, again by Proposition 3.8.37,  $\sigma^2(\emptyset) \notin \Sigma_1^1$ . The fact that  $\sigma^2(\emptyset) \notin \Pi_1^1 \cup \Sigma_1^1$  immediately implies that  $\overline{\sigma^2(\emptyset)} \notin \Pi_1^1 \cup \Sigma_1^1$ . Finally we obtain:

**Remark 3.9.3** For  $n \geq 2$ , neither language  $\sigma^n(\emptyset)$  nor language  $\overline{\sigma^n(\emptyset)}$  is recognized by an alternating automaton of any of the indexes  $(0, 1)$ ,  $(1, 2)$ .

Looking at the sequence presented in inequality (3.15), it is natural to ask whether it can be extended to the level  $\omega$  and potentially beyond. This question is addressed in the next section.

### 3.10 Limit Step

The main goal of this section is to define such an operation  $\sigma^\omega$  that for each  $n$ ,  $\sigma^n(L) <_w \sigma^\omega(L)$ . The intuition behind the construction of this operation is best understood when we think of the automaton recognizing language  $\sigma^n(L)$  that comes from the construction presented in Section 3.8.2. Such an automaton consists of  $n$  nested copies of the automaton  $\mathcal{C}$ . After seeing label  $\sharp$  it passes from one copy to the “deeper” one, or to the automaton recognizing the complement of the “deeper” language, i.e. the disjoint union of automaton  $\mathcal{C}_{N^1}$  and the automaton  $\mathcal{B}$ . The idea of the automaton for  $\sigma^\omega(L)$  is that it has one component similar to  $\mathcal{C}$  that, after seeing  $\sharp$ , loops to itself. The full picture is a bit more complex, though, because one has to make a decision about how to deal with branches with infinitely many  $\sharp$ 's. The decision affects properties of the resulting language. One of the properties of interest is bi-unambiguity.

Since  $\sigma^\omega$  is meant to be an operation on languages, an input language  $L$  also plays a role in the construction. For this we extend an alphabet with letter  $\sharp^1$ , that plays the same role as  $\sharp$  plays in the definition of  $\sigma(L)$ . As a result, in  $\sigma^\omega(L)$  there are trees with possibly many layers of formula-like contexts separated with  $\sharp$ 's where an occurrence of  $\sharp^1$ -labeled node means that no more layers will occur on the given path. The subtrees rooted in  $\sharp^1$ -labeled nodes are tested whether they belong to  $L$  or to the complement.

Before we proceed with a formal definition let us fix an input language. Let, then,  $A$  be a finite alphabet, and let  $L \subseteq T_A$  be an arbitrary set of trees. The alphabet of the language  $\sigma^\omega(L)$  is  $A_\omega := A \cup \{\vee, \neg, \sharp, \sharp^1\}$ . Similarly as in the case of the construction for  $\sigma$ , we do not require that  $A$  and  $\{\vee, \neg, \sharp, \sharp^1\}$  are disjoint.

The definitions in this section use languages  $P^\sigma$  and  $N^\sigma$  defined in equation (3.8), and languages  $W^D$  from equation (3.12), for  $D$  being a subset of the alphabet. We also use notation:

$$B_{E,X}^D := T_{E,X} \setminus W_{E,X}^D$$

If  $X = \emptyset$  then we write  $B_{E,X}^D = B_E^D$ . If the space  $T_{E,X}$  in which we complement is clear from the context, we write  $B_{E,X}^D = B^D$ .

The operator  $\Psi : \left( \mathbb{P} \left( T_{A_\omega, \{\mathfrak{t}^\omega, \mathfrak{f}^\omega, \top^\omega\}} \right) \right)^2 \rightarrow \left( \mathbb{P} \left( T_{A_\omega, \{\mathfrak{t}^\omega, \mathfrak{f}^\omega, \top^\omega\}} \right) \right)^2$  is defined as follows:

where:

The desired languages are now defined as the least fixpoint:

To define  $\sigma_a^\omega(L)$  we need to use alphabet projection similar to the one used for the definition of operation  $\sigma$ . For  $a \in A$ , tree mapping  $\widehat{\pi_a^{(\omega)}} : T_{A_\omega} \rightarrow T_A$  is the application of the alphabet projection:

to each node of a tree, i.e.:

We use the following substitution:

to define:

Similarly as for  $\sigma$  operation, we sometimes omit  $a$  and simply write  $\sigma^\omega(L)$ .

155

**Remark 3.10.1**

$$P^\omega \cup N^\omega \subseteq W_{A_\omega, \{\sharp, \mathbf{t}^\omega, \mathbf{f}^\omega, \top^\omega\}}$$

What, in turn, implies:

**Corollary 3.10.2**

$$P^\omega [\delta_a^L] \cup N^\omega [\delta_a^L] \subseteq W_{A_\omega}^{\{\sharp, \sharp^1\}}$$

It occurs that the inverse inclusion does not hold. The reason, as anticipated before, lies in some of the branches with infinitely many  $\sharp$ 's.

We are going to give a precise characterization of the gap between  $P^\omega [\delta_a^L] \cup N^\omega [\delta_a^L]$  and  $W_{A_\omega}^{\{\sharp, \sharp^1\}}$ . This will be crucial for the construction of the automaton recognizing the complement of  $\sigma_a^\omega(L)$ . We define several notions that extend the vocabulary introduced in Section 3.8 to talk about trees in  $T_{B,C}$  for  $\{\neg, \vee\} \subseteq B$ .

**Definition 3.10.3 (Closed Boolean Path)** *Let  $B, C$  be alphabets such that  $\{\vee, \neg\} \subseteq B$  and  $C$  arbitrary (possibly empty), and let  $D \subseteq (B \cup C) \setminus \{\vee, \neg\}$ . A boolean path  $v$  in tree  $t \in T_{B,C}$  is called  **$D$ -closed** if  $t(v) \in D$ .*

If  $D = \{x\}$  for some  $x$ , then we write  $x$ -closed instead of  $\{x\}$ -closed.

**Remark 3.10.4** *A boolean path  $v$  is  $D$ -closed in  $t$  if and only if*

1.  *$v$  is maximal (i.e. there is no boolean path extending  $v$  in  $t$ ), and*
2.  *$v$  is  $D$ -correct.*

For  $P^\sigma$  and  $N^\sigma$  defined by equation (3.8) we note:

**Corollary 3.10.5 (of Remark 3.10.4 and Proposition 3.8.10)** *Let  $t \in P^\sigma \cup N^\sigma$ , let  $v \in \text{dom}(t)$ . Then  $t(v) \in \{\mathbf{t}, \mathbf{f}\}$  if and only if  $v$  is a  $\{\mathbf{t}, \mathbf{f}\}$ -closed boolean path.*

**Definition 3.10.6** *A branch  $\rho$  in a tree  $t$  is called a  **$\sharp$ -chain** if  $\rho = w_1 r w_2 r \dots w_n r \dots$ , where for each  $i$ ,  $w_i$  is a  $\sharp$ -closed boolean path in  $t_{w_1 r w_2 r \dots w_{i-1} r}$ .*

**Definition 3.10.7** *A  $\sharp$ -chain  $\rho$  is called **proper** if  $\forall_{v \prec_\rho} t_v \in W^{\{\sharp, \sharp^1\}}$ .*

Now we are ready to define the set that will occur to fill the gap mentioned:

$$H = \{t \in T_{A_\omega} : \exists_\rho \rho \text{ is a proper } \sharp\text{-chain in } t\}$$

**Proposition 3.10.8**

$$W^{\{\#, \#^1\}} = P^\omega [\delta_a^L] \sqcup N^\omega [\delta_a^L] \sqcup H,$$

by what we mean that  $W^{\{\#, \#^1\}} = P^\omega [\delta_a^L] \cup N^\omega [\delta_a^L] \cup H$  and that sets  $P^\omega [\delta_a^L]$ ,  $N^\omega [\delta_a^L]$ ,  $H$  are pairwise disjoint.

The rest of this section is devoted to the proof of this proposition.

**Definition 3.10.9** Let  $A$  be an arbitrary alphabet. Path  $v$  is a  **$\#$ -boolean path** in  $t \in T_{A_\omega}$  if, for some  $k \geq 0$ ,  $v = v_0 r v_1 r \dots r v_k$ , where for each  $l < k$ ,  $v_l$  is a  $\#$ -closed boolean path (in  $t_{v_0 r \dots r v_{l-1} r}$ ), and  $v_k$  is a boolean path (in  $t_{v_0 r \dots r v_{k-1} r}$ ).

The path is  **$\{\#, \#^1\}$ -correct** (or simply **correct**) if  $v_k$  is  $\{\#, \#^1\}$ -correct.

The path is  **$\#^1$ -closed** (or simply **closed**) if  $v_k$  is a  $\#^1$ -closed boolean path.

We note, for further reference, that:

**Remark 3.10.10** If  $v$  is a  $\#$ -boolean path in  $t$ , then for  $w \prec v$ ,  $t(w) \in \{\vee, \neg, \#\}$ .

Since each  $\#$ -boolean path ends with a (possibly empty) boolean path and it also starts with a boolean path, by the concatenation-closure of boolean paths we get:

**Remark 3.10.11 (concatenation-closure)** The set of  $\#$ -boolean paths is closed under concatenation, in the following sense:

if  $v$  is a  $\#$ -boolean path in  $t$  and  $w$  is a  $\#$ -boolean path in  $t_v$ ,  
then  $vw$  is an  $\#$ -boolean path in  $t$ .

**Remark 3.10.12** A finite prefix of a  $\#$ -chain is a  $\#$ -boolean path. A concatenation of a  $\#$ -boolean path and a  $\#$ -chain is a  $\#$ -chain.

We also have an analogue of Remark 3.10.4:

**Remark 3.10.13** A  $\#$ -boolean path  $v$  is  $\#^1$ -closed in  $t$  if and only if

1.  $v$  is maximal (i.e. there is no  $\#$ -boolean path extending  $v$  in  $t$ ), and
2.  $v$  is  $\{\#, \#^1\}$ -correct.

**Remark 3.10.14** If  $\rho$  is a  $\#$ -chain in  $t$ , then for  $w \prec \rho$ ,  $t(w) \in \{\vee, \neg, \#\}$ .

As a consequence of the suffix-closure of boolean paths we get:

**Remark 3.10.15 (suffix-closure)**

1. The set of  $\sharp$ -boolean paths is suffix-closed.

Precisely, if  $v$  is a  $\sharp$ -boolean path in  $t$  and  $w \prec v$ , then  $w^{-1}v$  is a  $\sharp$ -boolean path in  $t_w$ .

2. Correctness is suffix-closed.

Precisely, if  $v$  is a  $\{\sharp, \sharp^1\}$ -correct  $\sharp$ -boolean path in  $t$  and  $w \prec v$ , then  $w^{-1}v$  is a  $\{\sharp, \sharp^1\}$ -correct  $\sharp$ -boolean path in  $t_w$ .

3. The set of  $\sharp$ -chains is suffix-closed.

Precisely, if  $\rho$  is a  $\sharp$ -chain in  $t$  and  $v \prec \rho$ , then  $v^{-1}\rho$  is a  $\sharp$ -chain in  $t_v$ .

4. The set of proper  $\sharp$ -chains is suffix-closed.

Precisely, if  $\rho$  is a proper  $\sharp$ -chain in  $t$  and  $v \prec \rho$ , then  $v^{-1}\rho$  is an proper  $\sharp$ -chain in  $t_v$ .

By the definition of the set  $H$ , we get:

**Remark 3.10.16**  $H \cap B_{A_\omega}^{\{\sharp, \sharp^1\}} = \emptyset$

**Lemma 3.10.17** Let  $\mathbf{P}, \mathbf{N} \subseteq T_{A_\omega, \{\mathbf{t}^\omega, \mathbf{ff}^\omega, \top^\omega\}}$  and let  $\alpha : \{\mathbf{t}^\omega, \mathbf{ff}^\omega, \top^\omega\} \rightarrow T_{A_\omega}$  be any substitution such that for  $t \in \alpha(\mathbf{t}^\omega) \cup \alpha(\mathbf{ff}^\omega)$ ,  $t(\varepsilon) = \sharp^1$ . Let  $(\mathbf{P}', \mathbf{N}') := \Psi(\mathbf{P}, \mathbf{N})$ , for  $\Psi$  as defined by equation (3.16). Then:

$$(\mathbf{P}[\alpha] \cup \mathbf{N}[\alpha]) \cap H = \emptyset \implies (\mathbf{P}'[\alpha] \cup \mathbf{N}'[\alpha]) \cap H = \emptyset.$$

**Proof:**

Recall that  $(\mathbf{P}', \mathbf{N}') = \Psi(\mathbf{P}, \mathbf{N}) = (P^\sigma[\eta^{\mathbf{P}, \mathbf{N}}], N^\sigma[\eta^{\mathbf{P}, \mathbf{N}}])$ . Assume, towards a contradiction, that there exists a tree  $t \in P^\sigma[\eta^{\mathbf{P}, \mathbf{N}}][\alpha] \cup N^\sigma[\eta^{\mathbf{P}, \mathbf{N}}][\alpha]$  that has a branch  $\rho$  that is a proper  $\sharp$ -chain.

Note that  $t$  is a context  $t' \in P^\sigma \cup N^\sigma$  with appropriate substitutions done according to  $\eta^{\mathbf{P}, \mathbf{N}}$  and  $\alpha$ . First observe that branch  $\rho$  cannot be entirely contained in  $t'$ . This is because a  $\sharp$ -chain contains nodes labeled with  $\sharp$  (precisely, it has infinitely many such nodes) and  $t' \in P^\sigma \cup N^\sigma \subseteq T_{\{\vee, \neg\}, \{\mathbf{t}, \mathbf{f}, \top\}}$ .

Therefore, only a prefix  $v$  of  $\rho$  is contained in  $t'$ , and  $v$  is a hole in  $t'$ . Note that, by definitions of  $P^\sigma$  and  $N^\sigma$  (precisely by definition of  $\Phi$ ), label  $\top$  can occur only in the left successor of a node labeled with  $\sharp$  or  $\neg$ . On the other hand, by the definition of a  $\sharp$ -chain,  $\rho$  turns right in any node labeled with any of these labels. Therefore,  $t'(v) \in \{\mathbf{t}, \mathbf{ff}\}$  and  $t_v \in \eta^{\mathbf{P}, \mathbf{N}}(\mathbf{t})[\alpha] \cup \eta^{\mathbf{P}, \mathbf{N}}(\mathbf{ff})[\alpha]$ . Hence,  $t_v$  is a context  $t'' \in \eta^{\mathbf{P}, \mathbf{N}}(\mathbf{t}) \cup \eta^{\mathbf{P}, \mathbf{N}}(\mathbf{ff})$  with appropriate substitutions done according to  $\alpha$ . Consider two cases coming from the definition of  $\eta^{\mathbf{P}, \mathbf{N}}$ :

1.  $t''(\varepsilon) \in \{\mathbf{tt}^\omega, \mathbf{ff}^\omega\}$ —In this case  $t_v \in \alpha(\mathbf{tt}^\omega) \cup \alpha(\mathbf{ff}^\omega)$ , so, by the assumption on  $\alpha$ ,  $t(v) = \sharp^1$ . By Remark 3.10.14, label  $\sharp^1$  cannot occur on a  $\sharp$ -chain, so we get a contradiction.
2.  $t''(\varepsilon) = t(v) = \sharp$  and  $t_{vr} \in \mathbf{P}[\alpha] \cup \mathbf{N}[\alpha] \cup B_{A_\omega}^{\{\sharp, \sharp^1\}}$ — This also yields a contradiction, because, by suffix-closure (Remark 3.10.15),  $(vr)^{-1}\rho$  is a proper  $\sharp$ -chain in  $t_{vr}$ , while, by the assumption and by Remark 3.10.16, none of the sets  $\mathbf{P}[\alpha]$ ,  $\mathbf{N}[\alpha]$ ,  $B_{A_\omega}^{\{\sharp, \sharp^1\}}$  can have a tree with such a branch.

□

**Lemma 3.10.18** *Let  $\mathbf{P}, \mathbf{N} \subseteq T_{A_\omega, \{\mathbf{tt}^\omega, \mathbf{ff}^\omega, \top^\omega\}}$  and let  $\alpha : \{\mathbf{tt}^\omega, \mathbf{ff}^\omega, \top^\omega\} \rightarrow T_{A_\omega}$  be any substitution such that  $\alpha(\mathbf{tt}^\omega) \cap \alpha(\mathbf{ff}^\omega) = \emptyset$  and that for  $t \in \alpha(\mathbf{tt}^\omega) \cup \alpha(\mathbf{ff}^\omega)$ ,  $t(\varepsilon) = \sharp^1$ . Let  $(\mathbf{P}', \mathbf{N}') := \Psi(\mathbf{P}, \mathbf{N})$ . Then:*

*If sets  $\mathbf{P}[\alpha]$ ,  $\mathbf{N}[\alpha]$  and  $B_{A_\omega}^{\{\sharp, \sharp^1\}} \cup H$  are pairwise disjoint, then sets  $\mathbf{P}'[\alpha]$ ,  $\mathbf{N}'[\alpha]$  and  $B_{A_\omega}^{\{\sharp, \sharp^1\}} \cup H$  are pairwise disjoint.*

**Proof:**

Observe that none of the labels from  $\text{dom}(\alpha)$  occurs in a context from  $P^\sigma \cup N^\sigma$ . By this and by the definition of  $\Psi$  we get:

$$\begin{aligned} \mathbf{P}'[\alpha] &= P^\sigma [\eta^{\mathbf{P}, \mathbf{N}}] [\alpha] = P^\sigma [\alpha \circ \eta^{\mathbf{P}, \mathbf{N}}] \\ \mathbf{N}'[\alpha] &= N^\sigma [\eta^{\mathbf{P}, \mathbf{N}}] [\alpha] = N^\sigma [\alpha \circ \eta^{\mathbf{P}, \mathbf{N}}] \end{aligned}$$

where the composition of substitutions is meant to have the following semantics:

For  $\beta : X \rightarrow T_{A, Y}$ ,  $\alpha : Y \rightarrow T_A$ ,  $x \in X$ , we have:  $\alpha \circ \beta(x) = \beta(x)[\alpha]$ .

Thanks to the above, we can prove the disjointness of  $\mathbf{P}'[\alpha]$  and  $\mathbf{N}'[\alpha]$  using Proposition 3.8.3, provided we show that  $T := \alpha \circ \eta^{\mathbf{P}, \mathbf{N}}(\mathbf{tt})$  and  $F := \alpha \circ \eta^{\mathbf{P}, \mathbf{N}}(\mathbf{ff})$  are disjoint, and that for each  $t \in T \cup F$ ,  $t(\varepsilon) \notin \{\vee, \neg\}$ . The latter is true because, by the definition of  $\eta^{\mathbf{P}, \mathbf{N}}$  and assumption on  $\alpha$ , for  $t \in T \cup F$ ,  $t(\varepsilon) \in \{\sharp, \sharp^1\}$ .

For the disjointness of  $T$  and  $F$  observe that, since trees in  $B_{A_\omega}^{\{\sharp, \sharp^1\}}$  have no holes, we have:

$$\begin{aligned} T &= \begin{array}{c} \textcircled{\sharp} \\ \swarrow \quad \searrow \\ \top^\omega \quad \mathbf{P}[\alpha] \end{array} \cup \alpha(\mathbf{tt}^\omega) \\ F &= \begin{array}{c} \textcircled{\sharp} \\ \swarrow \quad \searrow \\ \top^\omega \quad \mathbf{N}[\alpha] \cup B_{A_\omega}^{\{\sharp, \sharp^1\}} \end{array} \cup \alpha(\mathbf{ff}^\omega) \end{aligned}$$

By the assumption on  $\alpha$ , for  $t \in \alpha(\mathbf{tt}^\omega) \cup \alpha(\mathbf{ff}^\omega)$ ,  $t(\varepsilon) = \sharp^1$ , and sets  $\alpha(\mathbf{tt}^\omega)$ ,  $\alpha(\mathbf{ff}^\omega)$  are disjoint. Sets  $\mathbf{P}[\alpha]$  and  $\mathbf{N}[\alpha] \cup B_{A_\omega}^{\{\sharp, \sharp^1\}}$  are disjoint also disjoint, by the assumption. Therefore,  $T$ ,  $F$  are disjoint, and we get the disjointness of  $\mathbf{P}'[\alpha]$  and  $\mathbf{N}'[\alpha]$ .

Since for  $t \in T \cup F$ ,  $t(\varepsilon) \in \{\sharp, \sharp^1\}$ , by Corollary 3.8.11,  $\mathbf{P}'[\alpha] \cup \mathbf{N}'[\alpha]$  and  $B_{A_\omega}^{\{\sharp, \sharp^1\}}$  are disjoint. The disjointness of  $\mathbf{P}'[\alpha] \cup \mathbf{N}'[\alpha]$  and  $H$  comes from the assumed disjointness of  $\mathbf{P}[\alpha] \cup \mathbf{N}[\alpha]$  and  $H$  by Lemma 3.10.17. This concludes the proof of the required pairwise disjointness.  $\square$

We conclude the disjointness discussion with:

**Proposition 3.10.19** *Sets  $P^\omega[\delta^L]$ ,  $N^\omega[\delta^L]$ ,  $B_{A_\omega}^{\{\sharp, \sharp^1\}}$ ,  $H$  are pairwise disjoint.*

**Proof:**

Observe that  $\Psi$  is a monotonic operation, as a composition of unions and substitutions, that are always monotonic. Also observe that for each  $t \in \delta^L(\mathbf{tt}^\omega) \cup \delta^L(\mathbf{ff}^\omega)$ ,  $t(\varepsilon) = \sharp^1$  and that  $\delta^L(\mathbf{tt}^\omega) \cap \delta^L(\mathbf{ff}^\omega) = \emptyset$ . Therefore, by Lemma 3.10.18, we can use Lemma 3.8.2 to obtain pairwise disjointness of sets  $P^\omega[\delta^L]$ ,  $N^\omega[\delta^L]$  and  $B_{A_\omega}^{\{\sharp, \sharp^1\}} \cup H$ .

Sets  $B_{A_\omega}^{\{\sharp, \sharp^1\}}$  and  $H$  are disjoint by Remark 3.10.16.  $\square$

**Proposition 3.10.20**  $W^{\{\sharp, \sharp^1\}} \setminus H \subseteq P^\omega[\delta^L] \cup N^\omega[\delta^L]$ .

Before we proceed with a proof, we introduce a strict order on  $T_{A_\omega}$ .

$$t \sqsubset_\omega s \iff \begin{aligned} & t = s_{vr} \text{ for some } \sharp\text{-boolean path } vr \\ & \wedge \forall_{w \prec v} s_w \notin B_{A_\omega}^{\{\sharp, \sharp^1\}} \cup H \end{aligned} \quad (3.20)$$

We prove that  $\sqsubset_\omega$  is a strict well-founded order.

Transitivity comes from the concatenation-closure of  $\sharp$ -boolean paths.

We now prove well-foundedness. Let  $t^1 \sqsubset_\omega t^2 \sqsubset_\omega t^3 \sqsubset_\omega \dots$  be an infinite  $\sqsubset_\omega$ -decreasing sequence of trees in  $T_{A_\omega}$ . Let  $w_1r, w_2r, \dots$  be  $\sharp$ -boolean paths, such that:

$$t^2 = t^1_{w_1r}, \quad t^3 = t^2_{w_2r}, \quad t^4 = t^3_{w_3r}, \dots$$

Then  $\rho = w_1rw_2rw_3r\dots$  is such a branch that for each  $v \prec \rho$ ,  $t^1_v \notin B_{A_\omega}^{\{\sharp, \sharp^1\}} \cup H$  and:

1. either  $\rho$  starting from some point on contains only labels  $\vee$  and  $\neg$ . Then there is  $k \geq 1$  such that  $\rho' = (w_1rw_2r\dots rw_{k-1}r)^{-1}\rho$  is a boolean branch in  $t^k$ . Since  $\rho'$  turns right infinitely many times, it is incorrect and  $t^k \in B_{A_\omega}^{\{\sharp, \sharp^1\}}$ , what yields a contradiction;



2. or  $\rho$  is an infinite  $\sharp$ -chain in  $t^1$ . Since for each  $v \prec \rho$ ,  $t_v^1 \notin B_{A_\omega}^{\{\sharp, \sharp^1\}} \cup H$ ,  $\rho$  is a proper  $\sharp$ -chain, and  $t^1 \in H$ , what yields a contradiction.

Transitivity and well-foundedness imply antisymmetry, so we have proven that  $\sqsubset_\omega$  is a strict well-founded order.

Note that:

**Lemma 3.10.21** *A tree  $t$  is a minimal element of  $\sqsubset_\omega$  if and only if  $t \in B_{A_\omega}^{\{\sharp, \sharp^1\}} \cup H$  or  $t(\varepsilon) = \sharp^1$ .*

**Proof:**

If  $t \in B_{A_\omega}^{\{\sharp, \sharp^1\}} \cup H$  then there is no tree less than  $t$  with respect to  $\sqsubset_\omega$ , by the definition. If  $t(\varepsilon) = \sharp^1$ , then by Remark 3.10.10,  $\varepsilon$  is the only  $\sharp$ -boolean path in  $t$ , therefore, there is no tree less than  $t$  with respect to  $\sqsubset_\omega$ . We have proven the right-to-left implication.

Assume  $t \notin B_{A_\omega}^{\{\sharp, \sharp^1\}} \cup H$  and  $t(\varepsilon) \neq \sharp^1$ . Then  $t(\varepsilon) \in \{\vee, \neg, \sharp\}$ . Therefore,  $r$  is a  $\sharp$ -boolean path. Since the only  $v \prec r$  is  $v = \varepsilon$ , and since  $t_\varepsilon \notin B_{A_\omega}^{\{\sharp, \sharp^1\}} \cup H$ ,  $t_r \sqsubset_\omega t$ . Therefore,  $t$  is not minimal.  $\square$

**Lemma 3.10.22** *Let  $t \in W^{\{\sharp, \sharp^1\}} \setminus H$ . If  $v \in \{l, r\}^* r$  is a boolean path in  $t$ , then  $t_v \in W^{\{\sharp, \sharp^1\}} \setminus H$  and  $t_v \sqsubset_\omega t$ .*

**Proof:**

Let us take any  $t \in W^{\{\sharp, \sharp^1\}} \setminus H$  and a boolean path  $v \in \{l, r\}^* r$ . Recall that  $t \in W^{\{\sharp, \sharp^1\}}$  implies  $t_v \in W^{\{\sharp, \sharp^1\}}$ , by Remark 3.8.12.

Assume now, towards a contradiction, that  $t_v \in H$ . Let  $\rho$  be a proper  $\sharp$ -chain in  $t_v$ . By concatenation-closure of boolean paths,  $v\rho$  is a  $\sharp$ -chain in  $t$ . By 3.8.12, for any  $w \prec v$ ,  $t_w \in W^{\{\sharp, \sharp^1\}}$ , therefore  $v\rho$  is a proper  $\sharp$ -chain, and  $t \in H$ , what yields a contradiction.

Now since each  $w \prec v$  is a boolean path, we can apply what we have already proven to obtain that  $t_w \in W^{\{\sharp, \sharp^1\}} \setminus H$ . Therefore, by the definition,  $t_v \sqsubset_\omega t$ .  $\square$

**Proof (of Proposition 3.10.20):**

We prove the stated inclusion by the induction with respect to the order  $\sqsubset_\omega$ .

Induction basis: if  $t$  is  $\sqsubset_\omega$ -minimal and  $t \in W^{\{\sharp, \sharp^1\}} \setminus H$  then  $t(\varepsilon) = \sharp^1$ . By the definitions of  $P^\sigma$ ,  $N^\sigma$ ,  $\eta^{\mathbf{P}, \mathbf{N}}$ , and  $\delta^L$ , if  $t_r \in \widehat{\pi_a^{(\omega)}}^{-1}(L)$  then  $t \in P^\omega[\delta^L]$ , and if  $t_r \in \widehat{\pi_a^{(\omega)}}^{-1}(\overline{L})$  then  $t \in N^\omega[\delta^L]$ . Therefore, since  $\widehat{\pi_a^{(\omega)}}^{-1}(L) \cup \widehat{\pi_a^{(\omega)}}^{-1}(\overline{L}) = T_{A_\omega}$ ,  $t \in P^\omega[\delta^L] \cup N^\omega[\delta^L]$ .

For the inductive step assume that  $t$  is not  $\sqsubseteq_\omega$ -minimal, i.e.  $t \in W^{\{\#, \#^1\}} \setminus H$  and  $t(\varepsilon) \neq \#^1$ , and consider the following cases:

1.  $t(\varepsilon) \notin \{\vee, \neg, \#, \#^1\}$ —then  $t \in B_{A_\omega}^{\{\#, \#^1\}}$ , so this is in fact not a possible case here;
2.  $t(\varepsilon) = \#$ —then  $r$  is an  $\#$ -boolean path and  $t_r \sqsubseteq_\omega t$ . Hence, by the inductive assumption,  $t_r \in P^\omega[\delta^L] \cup N^\omega[\delta^L] \cup B_{A_\omega}^{\{\#, \#^1\}} \cup H$ . Consider the cases:
  - (a)  $t_r \in P^\omega[\delta^L]$ —by the definition of  $\Psi$  (remember that  $(P^\omega, N^\omega) = \Psi(P^\omega, N^\omega)$ ),  $t \in P^\omega[\delta^L]$ ,
  - (b)  $t_r \in N^\omega[\delta^L] \cup B_{A_\omega}^{\{\#, \#^1\}}$ —by the definition of  $\Psi$ ,  $t \in N^\omega[\delta^L]$ ,
  - (c)  $t_r \in H$ —there is a proper  $\#$ -chain in  $t_r$ . In such a case  $r\rho$  is also a proper  $\#$ -chain in  $t$ , so  $t \in H$ , what contradicts the assumption of the inductive step;
3.  $t(\varepsilon) = \neg$ —path  $r$  is a boolean path in  $t$ , and by assumption  $t \in W^{\{\#, \#^1\}} \setminus H$ , so, by Lemma 3.10.22,  $t_r \in W^{\{\#, \#^1\}} \setminus H$ . Therefore, by the inductive assumption,  $t_r \in P^\omega[\delta^L] \cup N^\omega[\delta^L]$ . Hence, by the definition of  $P^\sigma$  and  $N^\sigma$ ,  $t \in N^\omega[\delta^L] \cup P^\omega[\delta^L]$ ;
4.  $t(\varepsilon) = \vee$ —since  $t \in W^{\{\#, \#^1\}}$ , the whole left-most branch  $l^\omega$  is labeled with  $\vee$ , and for each  $n \geq 0$ ,  $l^n r$  is a boolean path. Therefore, by Lemma 3.10.22, for each  $n \geq 0$ ,  $t_{l^n r} \in W^{\{\#, \#^1\}} \setminus H$ , so, by the inductive assumption, for each  $n \geq 0$ ,  $t_{l^n r} \in P^\omega[\delta^L] \cup N^\omega[\delta^L]$ . Hence, by the definition of  $P^\sigma$  and  $N^\sigma$ ,  $t \in P^\omega[\delta^L] \cup N^\omega[\delta^L]$ .

□

We are ready to prove the main proposition of this section.

**Proof (of Proposition 3.10.8):**

Sets  $P^\omega[\delta^L]$ ,  $N^\omega[\delta^L]$ ,  $H$  partition  $W^{\{\#, \#^1\}}$ , by Proposition 3.10.19 and Proposition 3.10.20. □

In the sequel we refer to the partition of space  $T_{A_\omega}$  that is equivalent to the statement of the proposition.

**Corollary 3.10.23** *For a nonempty alphabet  $A$ , for  $L \subseteq T_A$ :*

$$T_{A_\omega} = P^\omega[\delta^L] \sqcup N^\omega[\delta^L] \sqcup B_{A_\omega}^{\{\#, \#^1\}} \sqcup H$$

### 3.10.1 Automaton

The goal of this section is to prove the following.

**Theorem 3.10.24** *If a language  $L$  is bi-unambiguous then the language  $\sigma^\omega(L)$  is bi-unambiguous.*

Corollary 3.10.23 gives a decomposition of space  $T_{A_\omega}$  that allows us to construct unambiguous automata recognizing  $\sigma^\omega(L)$  and its complement.

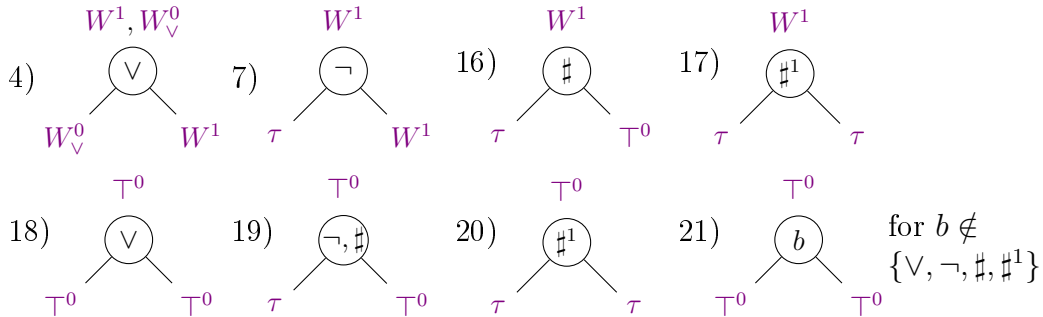
Let  $A$  be an arbitrary nonempty alphabet, and  $a \in A$  be an arbitrary letter. Let us fix a language  $L \subseteq T_A$ . Let  $\mathcal{L}$  be an automaton recognizing  $L$  and  $\bar{\mathcal{L}}$  be an automaton recognizing  $\bar{L}$ .

Analogously as during the construction of automaton recognizing  $\sigma(L)$  (see Section 3.8.2), we modify automata  $\mathcal{L}$  and  $\bar{\mathcal{L}}$  to work over alphabet  $A_\omega$ . Let then  $\mathcal{L}'$  (respectively  $\bar{\mathcal{L}}'$ ) be the same as  $\mathcal{L}$  (respectively  $\bar{\mathcal{L}}$ ), except that it treats letters outside  $A$  as if they were  $a$ . Then  $\mathcal{L}'$  recognizes  $\widehat{\pi_a^{(\omega)}}^{-1}(L)$ , and  $\bar{\mathcal{L}}'$  recognizes  $\widehat{\pi_a^{(\omega)}}^{-1}(\bar{L})$ . Note that the languages recognized by the two automata are complements in  $T_{A_\omega}$ .

We approach constructing automaton  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$  that recognizes language  $\sigma^\omega(L)$  by constructing component automata that recognize components of the partition given by 3.10.23. As already anticipated in Section 3.10, while constructing the automata, we use extended and modified versions of components of automata  $\mathcal{C}^{\mathcal{L}, \bar{\mathcal{L}}}$  and  $\mathcal{B}$  used in section 3.8.2.

We start with a construction of an automaton  $\mathcal{W}^\omega$ , that recognizes language  $W^{\{\#, \#^1\}}$ . The automaton is the same as the component  $\mathcal{C}_{W^1}^{\mathcal{L}, \bar{\mathcal{L}}}$  of  $\mathcal{C}^{\mathcal{L}, \bar{\mathcal{L}}}$ , except that it treats both,  $\#$  and  $\#^1$ , as  $\mathcal{C}_{W^1}^{\mathcal{L}, \bar{\mathcal{L}}}$  treats  $\#$ . We first define an automaton  $\mathcal{W}^1(\tau)$  with one parameter  $\tau$ . We maintain numeration of the states of automaton  $\mathcal{C}$ .

$\mathcal{W}^1$ :



The initial state is  $W^1$ .

Now we define  $\mathcal{W}^\omega = \mathcal{W}^1(\{\top^0\})$ , for  $\{\top^0\}$  being a one-state automaton accepting every tree<sup>8</sup>.

Consider the following letter projection  $\theta : A_\omega \rightarrow A_\sigma$ :

$$\theta(a) = \begin{cases} \# & \text{if } a = \#^1 \\ a & \text{otherwise} \end{cases}$$

The tree homomorphism  $\widehat{\theta} : T_{A_\omega} \rightarrow T_{A_\sigma}$  is defined by  $\widehat{\theta}(t) = \theta \circ t$ .

Observe that the automaton  $\mathcal{W}^\omega$  is the result of application to the automaton  $\mathcal{C}_{W^1}^{\mathcal{L}, \bar{\mathcal{L}}}$  of the construction presented in Fact 3.6.6(3) for inverse image of letter projection  $\widehat{\theta}$ . Therefore, by Lemma 3.8.19 and since  $W_{A_\omega}^{\{\#, \#^1\}} = \widehat{\theta}^{-1}(W_{A_\sigma}^\#)$ , we have:

**Remark 3.10.25**  $L(\mathcal{W}^\omega) = W^{\{\#, \#^1\}}$ .

Additionally observe that:

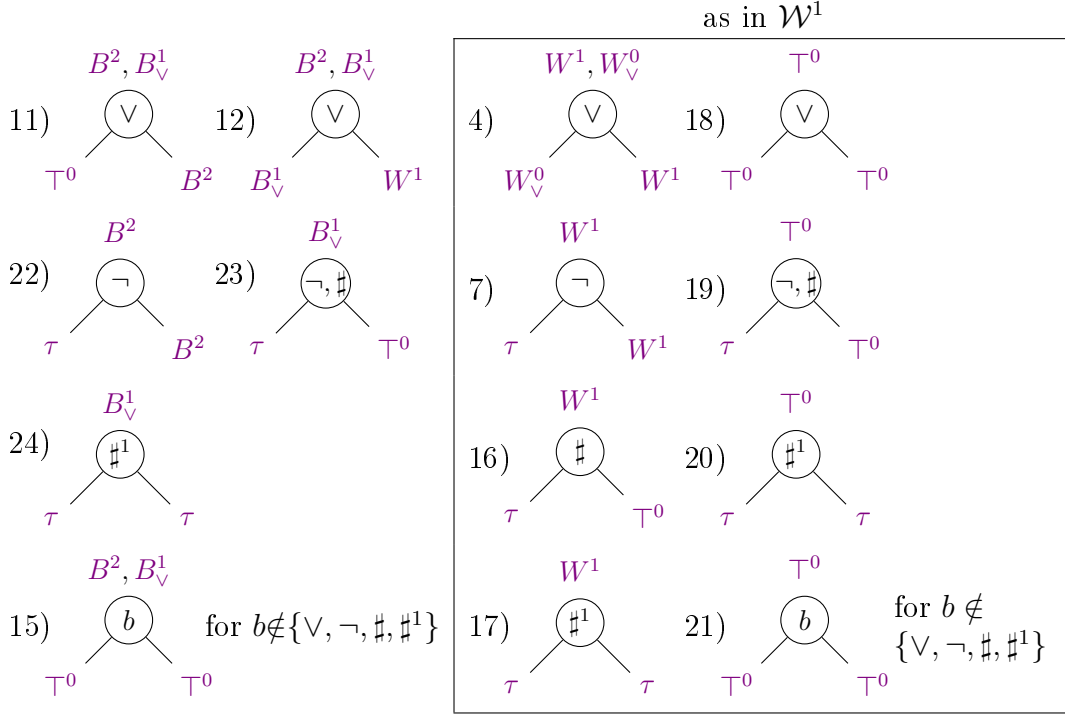
**Remark 3.10.26** Automaton  $\mathcal{W}^\omega$  is deterministic of index  $(0, 1)$ .

We construct automaton  $\mathcal{B}^\omega$ , recognizing language  $B_{A_\omega}^{\{\#, \#^1\}}$ , from automaton  $\mathcal{B}$  the same way as we have constructed automaton  $\mathcal{W}^\omega$  from  $\mathcal{C}_{W^1}^{\mathcal{L}, \bar{\mathcal{L}}}$ . Namely,  $\mathcal{B}^\omega$  acts as  $\mathcal{B}$ , except that it treats both,  $\#$  and  $\#^1$ , as  $\mathcal{B}$  treats  $\#$ . Again, we start with defining an automaton  $\mathcal{B}^1(\tau)$  with one parameter. We maintain the numeration of the states of automaton  $\mathcal{B}$ .

$\mathcal{B}^1$ :

---

<sup>8</sup>The fact that automaton  $\mathcal{W}^\omega$  consists of two components,  $\mathcal{W}^1$  and  $\{\top^0\}$ , will be important in Section 3.10.3, where we prove stretchability using Lemma 3.8.36.



The initial state is  $B^2$ .

Let  $\mathcal{B}^\omega = \mathcal{B}^1(\{\top^0\})$ .

As above, by Lemma 3.8.24, we get:

**Remark 3.10.27**  $L(\mathcal{B}^\omega) = B_{A_\omega}^{\{\#, \#^1\}}$ .

Since automaton  $\mathcal{B}$  is unambiguous, by Fact 3.6.6(3), we observe that:

**Remark 3.10.28** Automaton  $\mathcal{B}^\omega$  is unambiguous of index  $\{(0, 1), (1, 2)\}$ .

Let us now construct automaton  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$  recognizing  $\sigma^\omega(L)$ .

First we define an automaton  $\mathcal{O}(\ell, \bar{\ell}, \tau)$  with three parameters. Apart from the states of the automaton  $\mathcal{B}^1$ , and states  $P^1, P_\vee^1, N^1, N_\vee^0$  used by the automaton  $\mathcal{C}$ , the automaton  $\mathcal{O}$  uses state  $F_\vee^0$  ( $F$  stands for “formula”) that is intended to recognize  $P^\omega[\delta^L] \cup N^\omega[\delta^L]$ . The initial state of the automaton is  $P^1$ . Superscript in state symbols designates priority. The transitions are presented on Figure 3.6.

Let  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}} = \mathcal{O}(\mathcal{L}', \bar{\mathcal{L}}', \{\top^0\})$ .

To show that the automaton recognizes the appropriate language, we prove the following:

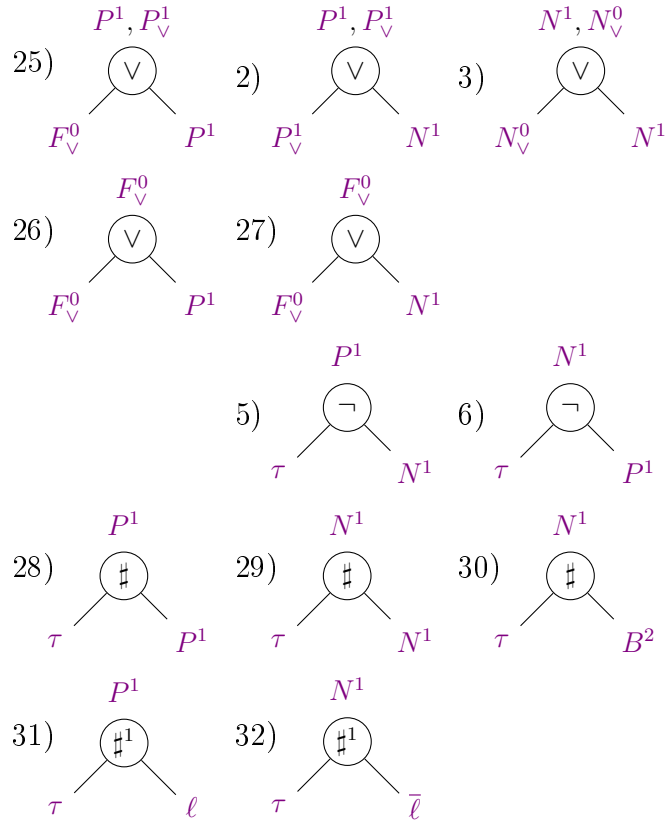


Figure 3.6: Transitions of automaton  $\mathcal{O}$ . Transition leading to  $B^2$  leads to the embedded automaton  $\mathcal{B}^1(\tau)$ . The numbering of transitions coming directly from automaton  $\mathcal{C}$  is preserved.

**Lemma 3.10.29** *If  $L \subseteq T_A$ , an automaton  $\mathcal{L}$  recognizes  $L$  and  $\bar{\mathcal{L}}$  recognizes  $\bar{L}$ , then:*

$$\begin{aligned} L \left( \mathcal{O}_{P^1}^{\mathcal{L}, \bar{\mathcal{L}}} \right) &= P^\omega [\delta_a^L] \\ L \left( \mathcal{O}_{N^1}^{\mathcal{L}, \bar{\mathcal{L}}} \right) &= N^\omega [\delta_a^L] \end{aligned}$$

**Proof:**

Let  $F_\vee := (P^\omega [\delta_a^L] \cup N^\omega [\delta_a^L]) \cap \{t : t(\varepsilon) = \vee\}$ .

We prove that for each  $t \in T_{A_\omega}$ :

$t$  is accepted by  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$  from state  $P^1$  if and only if  $t \in P^\omega [\delta_a^L]$  (3.21)

$t$  is accepted by  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$  from state  $N^1$  if and only if  $t \in N^\omega [\delta_a^L]$  (3.22)

$t$  is accepted by  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$  from state  $F_\vee^0$  if and only if  $t \in F_\vee$  (3.23)

The proof is by induction (simultaneous for the three equivalences) with respect to relation  $\sqsubset_\omega$  on  $T_{A_\omega}$ .

By Remark 3.10.21, for the induction basis we need to consider such  $t \in T_{A_\omega}$  that  $t \in H$ ,  $t \in B_{A_\omega}^{\{\sharp, \sharp^1\}}$ , or  $t(\varepsilon) = \sharp^1$ .

Consider a run  $\rho$  of  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$  on a tree  $t \in H$  from one of the states  $P^1$ ,  $N^1$ ,  $F_\vee^0$ . In such a tree there exists a proper  $\sharp$ -chain  $\alpha$ . Let us analyze what states are assigned to nodes of  $\alpha$  in run  $\rho$ . If state  $B^2$  occurs on  $\alpha$  then the run is rejecting, because, by definition, no subtree rooted in a node on a proper  $\sharp$ -chain can belong to  $B_{A_\omega}^{\{\sharp, \sharp^1\}}$  and, by Remark 3.10.27,  $L(\mathcal{O}_{B^2}^{\mathcal{L}, \bar{\mathcal{L}}}) = B_{A_\omega}^{\{\sharp, \sharp^1\}}$ . Let us now consider the case when no state  $B^2$  occurs on  $\alpha$  in  $\rho$ . Then by the shape of the transitions of  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$ , only states  $P^1$ ,  $P_\vee^1$ ,  $N^1$ ,  $N_\vee^0$ ,  $F_\vee^0$  can occur on  $\alpha$  in this run. Additionally after each label  $\sharp$  one of the states  $P^1$ ,  $N^1$  occurs. Since both these states have priority 1 and no greater priority occurs on  $\alpha$ , the parity condition is not satisfied on  $\alpha$  and  $\rho$  is rejecting. Therefore, no tree from  $H$  is accepted from any of the states  $P^1$ ,  $N^1$ ,  $F_\vee^0$ . Hence, equivalences (3.22), (3.23), (3.23) are satisfied for  $t \in H$ , because, by Proposition 3.10.19,  $t \notin P^\omega [\delta_a^L] \cup N^\omega [\delta_a^L]$ .

Now consider  $t \in B_{A_\omega}^{\{\sharp, \sharp^1\}}$ . By Proposition 3.10.19,  $t \notin P^\omega [\delta_a^L] \cup N^\omega [\delta_a^L]$ , therefore, in order for equivalences (3.22), (3.23), (3.23) to be satisfied,  $t$  should not be accepted from any of states  $P^1$ ,  $N^1$ ,  $F_\vee^0$ . Let us consider a run  $\rho$  of automaton  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$  from one of the states  $P^1$ ,  $N^1$ ,  $F_\vee^0$  on  $t$ . Since  $t \in B_{A_\omega}^{\{\sharp, \sharp^1\}}$ , it has a  $\{\sharp, \sharp^1\}$ -incorrect boolean path or branch  $\alpha$ . The path is labeled with states  $P^1$ ,  $P_\vee^1$ ,  $N^1$ ,  $N_\vee^0$ ,  $F_\vee^0$ . Since  $\alpha$  is  $\{\sharp, \sharp^1\}$ -incorrect, one of the following holds:

- $\alpha$  is infinite, labeled with  $\vee$  and  $\neg$ , and turns right infinitely many

times—then state  $N^1$  or  $P^1$  occurs after each turning right, so the parity condition is not satisfied on the branch,

- a label different than  $\vee, \neg, \sharp, \sharp^1$  occurs at the end of the path—then the run is stuck at the end of  $\alpha$ , because there is no transition from any of  $P^1, P_\vee^1, N^1, N_\vee^0, F_\vee^0$  over a letter different than  $\vee, \neg, \sharp, \sharp^1$ ,
- label different than  $\vee$  occurs in the left successor of last node of  $\alpha$  that is labeled with  $\vee$ —then the run is stuck in  $\alpha l$ , because one of the states  $P_\vee^1, N_\vee^0, F_\vee^0$  is assigned to this node and there are only transitions over  $\vee$  from each of these states.

As a consequence, there is no run from any of states  $P^1, N^1, F_\vee^0$  that is accepting on an  $\{\sharp, \sharp^1\}$ -incorrect path or branch, so  $t$  is not accepted from those states—as required.

If  $t(\varepsilon) = \sharp^1$  then  $t \in P^\omega[\delta_a^L]$  if and only if  $t_r \in \widehat{\pi_a^{(\omega)}}^{-1}(L)$ , that is if and only if  $t_r$  is accepted by automaton  $\mathcal{L}'$ , i.e. if and only if  $t$  is accepted from  $P^1$ , since transition (31) is the only one from state  $P^1$ . Similarly,  $t \in N^\omega[\delta_a^L]$  if and only if  $t_r \in \widehat{\pi_a^{(\omega)}}^{-1}(\bar{L})$  that is if and only if  $t_r$  is accepted by automaton  $\bar{\mathcal{L}}'$ , i.e. if and only if  $t$  is accepted from  $N^1$ , since transition (32) is the only one from state  $N^1$ . Since there is no transition over  $\sharp^1$  from state  $F_\vee^0$ , and trivially (because of the label in the root)  $t$  does not belong to the set  $F_\vee$ , we have concluded the inductive hypothesis for case when  $t(\varepsilon) = \sharp^1$ .

Now, as an inductive step, it is enough to consider  $t \in W^{\{\sharp, \sharp^1\}} \setminus H$  such that  $t(\varepsilon) \neq \sharp^1$ . Since such  $t$  has no  $\{\sharp, \sharp^1\}$ -incorrect boolean path, in particular,  $\varepsilon$  is a  $\{\sharp, \sharp^1\}$ -correct boolean path, so one of the three cases holds:

1.  $t(\varepsilon) = \neg$ —then  $t$  is not accepted from state  $F_\vee^0$ , because there is no transition over  $\neg$  from this state. Since, on the other hand,  $t$  does not belong to  $F_\vee$ , equivalence (3.23) is satisfied.

Tree  $t$  is accepted from  $P^1$  if and only if  $t_r$  is accepted from  $N^1$  (transition (5) is the only transition from  $P^1$  over  $\neg$ ). Note that  $r$  is a boolean path, so  $t_r \sqsubset_\omega t$ , by Lemma 3.10.22. Then, by the inductive assumption,  $t_r$  is accepted from  $N^1$  if and only if  $t_r \in N^\omega[\delta_a^L]$ , so if and only if  $t \in P^\omega[\delta_a^L]$ , by the way how  $P^\omega$  and  $N^\omega$  are defined using  $P^\sigma$  and  $N^\sigma$  (see equation (3.16)). Hence, equivalence (3.22) is satisfied in this case.

Tree  $t$  is accepted from  $N^1$  if and only if  $t_r$  is accepted from  $P^1$  (transition (6) is the only transition from  $N^1$  over  $\neg$ ). Equivalence (3.23) is satisfied by exactly analogous argument as for the state  $P^1$  and equivalence (3.22).



2.  $t(\varepsilon) = \vee$ —then the whole left-most branch is labeled with  $\vee$  (otherwise there would be an  $\{\sharp, \sharp^1\}$ -incorrect boolean path in  $t$ ). Moreover, for each  $n \geq 0$ ,  $t_{l^n r}$  is a boolean path, so  $t_{l^n r} \sqsubset_\omega t$ , by Lemma 3.10.22, and we can use an inductive assumption for each of those subtrees.

First, analyze all possible runs on a tree with the whole left-most branch labeled with  $\vee$  from one of the states  $P^1$ ,  $N^1$ ,  $F_\vee^0$ . Note that the left-most branch is assigned states  $P^1$ ,  $P_\vee^1$ ,  $N^1$ ,  $N_\vee^0$ ,  $F_\vee^0$  in such a run. All the nodes diverging right from the branch are assigned state  $P^1$  or  $N^1$ . This is enough to conclude that if there is  $n \geq 0$  such that  $t_{l^n r}$  is accepted from neither  $P^1$  nor  $N^1$ , then  $t$  is accepted from neither of states  $P^1$ ,  $N^1$ ,  $F_\vee^0$ .

Now consider a run from state  $N^1$ . Note that it has to use transition (3) on each node of the left-most branch. Therefore, it is accepting if and only if each subtree  $t_{l^n r}$  is accepted from  $N^1$ .

A run from state  $F_\vee^0$  can use any of transitions (26), (27) in any node of the left-most branch. Therefore, there is an accepting run from this state if and only if each subtree  $t_{l^n r}$  is accepted from  $P^1$  or  $N^1$ .

For a run from state  $P^1$  there is a bit more of choice. In order to be accepting, a run cannot use only transition (2) on the whole left-most branch (parity condition on this branch would not be satisfied). Therefore, such a run can use this transition zero or finite number of times, then it has to use transition (1), and then it reaches state  $F_\vee^0$  for which we have already discussed the acceptance. As a result,  $t$  is accepted from state  $P^1$  if and only if there is  $n$  for which  $t_{l^n r}$  is accepted from state  $P^1$  and for each  $n \geq 0$ ,  $t_{l^n r}$  is accepted from  $P^1$  or  $N^1$ .

The above discussion of possible runs implies, in particular, that, if  $t(\varepsilon) = \vee$ , then language accepted from state  $F_\vee^0$  is a union of languages accepted from states  $P^1$  and  $N^1$ .

Now we confront this acceptance discussion with the other hand, i.e. with definitions of the languages  $P^\omega[\delta_a^L]$ ,  $N^\omega[\delta_a^L]$ ,  $F_\vee$ .

Since  $N^\omega[\delta_a^L]$  is equal the set  $N^\sigma$  with variables replaced with appropriate languages, we can infer from the fixpoint definition (see equation (3.8)) of set  $N^\sigma$  (without even touching the substituted part), that,  $t \in N^\omega[\delta_a^L]$  if and only if for each  $n$ ,  $t_{l^n r} \in N^\omega[\delta_a^L]$ . Now we use the inductive assumption for the subtrees and obtain that  $t \in N^\omega[\delta_a^L]$  if and only if each of the subtrees  $t_{l^n r}$  is accepted from state  $N^1$ . By the above discussion concerning accepting runs, this is exactly the case when there is a run from  $N^1$  on  $t$ , so equivalence (3.23) is satisfied.

Language  $P^\omega[\delta_a^L]$  is equal the set  $P^\sigma$  with variables replaced with appropriate languages, but to consider belonging to this language we need to use also the fact about  $N^\omega[\delta_a^L]$  being  $N^\sigma$  (substituted), because  $P^\sigma$  is defined using both  $P^\sigma$  and  $N^\sigma$  (see equation (3.6)). By the fixpoint definition of sets  $P^\sigma$  and  $N^\sigma$ ,  $t \in P^\omega[\delta_a^L]$  if and only if there is  $n$  such that  $t_{l^n r} \in P^\omega[\delta_a^L]$  and for each  $n$ ,  $t_{l^n r} \in P^\omega[\delta_a^L]$  or  $t_{l^n r} \in N^\omega[\delta_a^L]$ , i.e., by the inductive assumption, if and only if each of the subtrees  $t_{l^n r}$  is accepted from state  $P^1$  or state  $N^1$  and there is  $n$  such that  $t_{l^n r}$  is accepted from state  $P^1$ . This corresponds exactly to the outcome of the discussion of acceptance cases from state  $P^1$  and equivalence (3.22) is satisfied.

Now equivalence (3.23) comes from the, already noted, fact that the language accepted from state  $F_\vee^0$  is a union of languages accepted from states  $P^1$  and  $N^1$ .

3.  $t(\varepsilon) = \sharp$ —then  $t$  is not accepted from state  $F_\vee^0$ , because there is no transition over  $\sharp$  from this state. Such trees also do not belong to the language  $F_\vee$ , so equivalence (3.23) is satisfied.

Note that if  $t(\varepsilon) = \sharp$  then  $r$  is a  $\sharp$ -boolean path in  $t$ . Since additionally  $t \in W^{\{\sharp, \sharp^1\}} \setminus H$ ,  $t_r \sqsubset_\omega t$  (see equation (3.20) for the definition of  $\sqsubset_\omega$ ). Now consider all possible cases for the right subtree of the root.

Since transitions (28), (29), and (30) are the only ones from states  $P^1$  and  $N^1$  over the letter  $\sharp$ ,  $t$  is accepted from  $P^1$  is and only if  $t_r$  is accepted from state  $P^1$ , and  $t$  is accepted from  $N^1$  is and only if  $t_r$  is accepted from one of the states  $N^1$ ,  $B^2$ . On the other hand, by the construction of  $\eta^{\mathbf{P}, \mathbf{N}}(\mathbf{t}^\omega)$  used to define languages  $N^\omega$  and  $P^\omega$  (see equation (3.17)),  $t \in P^\omega[\delta_a^L]$  if and only if  $t_r \in P^\omega[\delta_a^L]$ , and  $t \in N^\omega[\delta_a^L]$  if and only if  $t_r \in N^\omega[\delta_a^L]$  or  $t_r \in B_{A_\omega}^{\{\sharp, \sharp^1\}}$ . Therefore, it is enough to prove that:

$$t_r \in L\left(\mathcal{O}_{P^1}^{\mathcal{L}, \overline{\mathcal{L}}}\right) \iff t_r \in P^\omega[\delta_a^L] \quad (3.24)$$

$$t_r \in L\left(\mathcal{O}_{N^1}^{\mathcal{L}, \overline{\mathcal{L}}}\right) \iff t_r \in N^\omega[\delta_a^L] \quad (3.25)$$

$$t_r \in L\left(\mathcal{O}_{B^2}^{\mathcal{L}, \overline{\mathcal{L}}}\right) \iff t_r \in B_{A_\omega}^{\{\sharp, \sharp^1\}} \quad (3.26)$$

Equivalence (3.26) holds by Remark 3.10.27, while equivalences (3.24), (3.25) hold by the inductive assumption.

The above enumeration of cases concludes the proof of the inductive step.

□

**Corollary 3.10.30** *If  $L \subseteq T_A$ , automaton  $\mathcal{L}$  recognizes  $L$  and  $\overline{\mathcal{L}}$  recognizes  $\overline{L}$ , then automaton  $\mathcal{O}^{\mathcal{L}, \overline{\mathcal{L}}}$  recognizes language  $\sigma^\omega(L)$ .*

**Lemma 3.10.31** *If automata  $\mathcal{L}$  and  $\overline{\mathcal{L}}$  are unambiguous, and  $s$  is a single state of  $\mathcal{O}^{\mathcal{L}, \overline{\mathcal{L}}}$ , then automaton  $\mathcal{O}_s^{\mathcal{L}, \overline{\mathcal{L}}}$  is unambiguous.*

**Proof:**

First we note that if  $\mathcal{L}$  and  $\overline{\mathcal{L}}$  are unambiguous then  $\mathcal{L}'$  and  $\overline{\mathcal{L}}'$  are unambiguous, by Fact 3.6.6(3).

Now assume, towards contradiction, that there is a tree  $t$  that admits two distinct accepting runs  $\rho_1$  and  $\rho_2$  of automaton  $\mathcal{O}_s^{\mathcal{L}, \overline{\mathcal{L}}}$ . Since  $\mathcal{O}_s^{\mathcal{L}, \overline{\mathcal{L}}}$  has only one initial state, there is a node  $v$  such that on the path to this node the runs are equal and in the node the runs use different transitions. The state that the runs assign to  $v$  cannot be in  $\mathcal{L}'$  or  $\overline{\mathcal{L}}'$  because this would contradict unambiguity of the automata. We consider all cases where there is more than one transition from a given state for a given letter:

- state  $P^1$  or  $P_\vee^1$ , letter  $\vee$ , transitions: (25), (2). One of them assigns state  $P^1$  to the right successor, while the other assigns  $N^1$  there. Since the languages accepted from the two states are disjoint (by Lemma 3.10.29 and Proposition 3.10.19), it is impossible that both the transitions can be used in two accepting runs on the same tree;
- state  $F_\vee^0$ , letter  $\vee$ , transitions: (26), (27). Again, at least one of them cannot be used in an accepting run since languages accepted from states  $P^1$  and  $N^1$  are disjoint;
- state  $N^1$ , letter  $\sharp$ , transitions: (29), (30). Here we use the fact that languages accepted from  $N^1$  and  $B^2$  are disjoint (by Proposition 3.10.19);
- state  $B^2$  or  $B_\vee^1$ , letter  $\vee$ , transitions: (11), (12). Here we use the disjointness of languages  $B_{A_w}^{\{\sharp, \sharp^1\}}$  and  $W^{\{\sharp, \sharp^1\}}$ , accepted from  $B^2$  and  $W^1$ , respectively (see Remark 3.10.27 and Remark 3.10.25).

We have come to a contradiction in all the cases so there are no such distinct accepting runs.  $\square$

Since our final goal is bi-unambiguity of  $\sigma^\omega(L)$ , we need to show that the complement of the language can also be recognized by an unambiguous automaton. Lemma 3.10.29 and Remark 3.10.27 show that automaton  $\mathcal{O}^{\mathcal{L}, \overline{\mathcal{L}}}$  provides a way to recognize most of the component languages mentioned in Corollary 3.10.23. The only missing component is the language  $H$ . Before we proceed with the construction of the automaton, we prove a kind of uniqueness property of trees in  $H$ .

**Lemma 3.10.32** *If  $t \in H$  then there exists the right-most proper  $\sharp$ -chain in  $t$ .*

**Proof:**

The proof goes similarly as the proof of Lemma 3.8.23. We start with the root and proceed downwards to construct a branch  $\beta$  that will occur to be the right-most of proper  $\sharp$ -chains. We maintain the invariant that the path constructed so far is a  $\sharp$ -boolean path and that the subtree rooted in the current node (the node that the path constructed so far ends in) is in  $H$ . Note that the path  $\varepsilon$  that we start with satisfies the invariants, since  $t \in H \subseteq W^{\{\sharp, \sharp^1\}}$ . Having constructed the path until a node  $v$ , we proceed as follows:

1. If  $t(v) = \neg$  then we extend the path being constructed rightwards. Note that this extension maintains invariants. Indeed, if  $t_v$  has a proper  $\sharp$ -chain then this chain has to start with  $r$  if the root is labeled with  $\neg$ , so  $t_{vr}$  also has a proper  $\sharp$ -chain (by suffix-closure of proper  $\sharp$ -chains). Moreover, boolean paths are closed under concatenations, so if  $v$  ends with a boolean path (as a  $\sharp$ -boolean path) then  $vr$  does as well.
2. If  $t(v) = \sharp$  then we extend the path being constructed rightwards. This extension also maintains invariants. This is because if  $t(v) = \sharp$  then  $v$  ends with a  $\sharp$ -closed boolean path, and in a  $\sharp$ -boolean path there is a turning right after each  $\sharp$ -closed boolean path. So the argument goes as above.
3. If  $t(v) = \vee$  and  $t_{vr} \in H$  then we descend rightwards. The second invariant is trivially maintained. The first one is maintained, again, by closure of boolean paths under concatenation.
4. If  $t(v) = \vee$  and  $t_{vr} \notin H$  then we extend the path leftwards. First note that if  $t(vl) \neq \vee$  then  $\varepsilon$  is an  $\{\sharp, \sharp^1\}$ -incorrect boolean path in  $t_v$ , so  $t_v \notin W^{\{\sharp, \sharp^1\}}$ , so  $t_v \notin H$ —a contradiction with the second invariant being satisfied for  $v$ . Therefore,  $t(vr) = \vee$  and  $l$  is a boolean path in  $t_v$ , so, by closure of boolean paths under concatenation, the first invariant is maintained.

Since the second invariant is satisfied for  $v$ , there is a proper  $\sharp$ -chain  $\alpha$  in  $t_v$ . If  $\alpha$  starts with  $r$  then, by closure of proper  $\sharp$ -chains under suffixes,  $t_{vr} \in H$ —a contradiction with an assumption of this bullet point. Therefore,  $\alpha$  starts with  $l$  and, again by suffix-closure,  $t_{vl} \in H$ , so the second invariant is satisfied.

The above cases exhaust the possibilities, because if  $t(v) \notin \{\vee, \neg, \sharp\}$  then, by Remark 3.10.14,  $t \notin H$ , what contradicts the second invariant being satisfied for  $v$ . Let the branch  $\beta$  be the one that comes from infinitely many applications of the above cases.

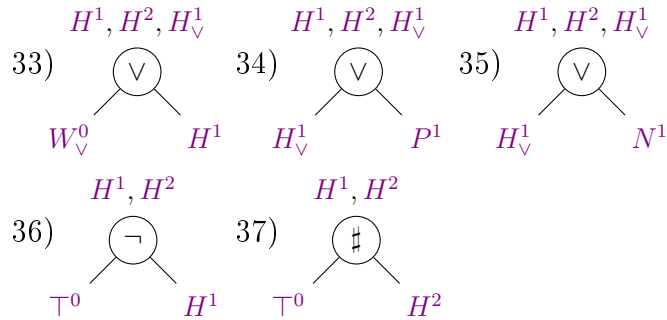
It is clear, from the construction, that there is no proper  $\sharp$ -chain to the right from  $\beta$ . We show that  $\beta$  is a proper  $\sharp$ -chain. First observe that each prefix of  $\beta$  is a  $\sharp$ -boolean path, by the first invariant. This implies that  $\beta$  consists of  $\sharp$ -closed boolean paths, separated from each other by  $r$ 's, with possibly an unclosed boolean branch at the end. Let then assume that there indeed is such  $v$ , that  $v^{-1}\beta$  is a boolean branch. Since  $t_v$  does not have an  $\{\sharp, \sharp^1\}$ -incorrect boolean path (by the second invariant  $t_v \in H \subseteq W^{\{\sharp, \sharp^1\}}$ ), then there is  $w$  such that  $(vw)^{-1}\beta = l^\omega$ . On the other hand, by the second invariant for  $vw$ , there is a proper  $\sharp$ -chain  $\gamma$  in  $t_{vw}$ . Since, in particular, such a chain turns right infinitely many times,  $\gamma$  diverges from  $l^\omega$  in some node  $u$  of  $t_{vw}$ . Then  $t_{vwur} \in H$  and  $\beta$  should have turned right in node  $vwu$ , by the construction, and it has not—a contradiction. Therefore,  $\beta$  is an infinite  $\sharp$ -chain. It is proper, because for each  $v \prec \beta$  the second invariant is satisfied, and  $v \in H \subseteq W^{\{\sharp, \sharp^1\}}$ .  $\square$

We are ready to construct an automaton recognizing language  $T_{A_\omega} \setminus \sigma^\omega(L)$ . Recall that, by Corollary 3.10.23:

$$T_{A_\omega} \setminus \sigma^\omega(L) = N^\omega [\delta_a^L] \sqcup H \sqcup B_{A_\omega}^{\{\sharp, \sharp^1\}}$$

We construct an automaton  $\mathcal{H}^{\mathcal{L}, \bar{\mathcal{L}}}$  that recognizes this language.

Automaton  $\mathcal{H}^{\mathcal{L}, \bar{\mathcal{L}}}$  contains whole automaton  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$ . Apart from the states of  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$  it uses states  $H^1, H^2, H_\vee^1$ , that serve recognizing language  $H$ , and transitions:



The set of initial states of  $\mathcal{H}^{\mathcal{L}, \bar{\mathcal{L}}}$  is  $\{N^1, H^1, B^2\}$ .

**Lemma 3.10.33**  $L(\mathcal{H}_{H^1}^{\mathcal{L}, \bar{\mathcal{L}}}) = H$

**Proof:**

First note that in each full run  $\rho$  of  $\mathcal{H}^{\mathcal{L}, \bar{\mathcal{L}}}$  on any tree  $t$  from state  $H^1$  there is exactly one branch that is labeled with states  $H^1$ ,  $H^2$ ,  $H_\vee^1$  (we will call them  $H$ -states). The way how  $H$ -states are propagated by transitions correspond to the shape of boolean paths, with the exception that if label  $\sharp$  is met state  $H^2$  occurs and the  $H$ -labeled path is continued to the right. As a result, each interval between two occurrences of state  $H^2$  corresponds to a  $\sharp$ -closed boolean path. The parity condition is satisfied on the  $H$ -labeled branch if and only if  $\sharp$  occurs infinitely many times, i.e. if and only if the branch is a  $\sharp$ -chain. We show that the whole run is accepting if and only if the branch is proper.

Assume that  $t \in H$ . We show that there is an accepting run  $\rho$  of  $\mathcal{H}^{\mathcal{L}, \bar{\mathcal{L}}}$  on  $t$  from  $H^1$ . This run labels the right-most proper  $\sharp$ -chain  $\eta$  with  $H$ -states. As we have noted above, such a run is accepting on  $\eta$ . We show how to fill other parts of the run to make it accepting.

First consider subtree that diverges left from  $\eta$  in some node  $v \prec \eta$ . If  $t(v) = \neg$  or  $t(v) = \sharp$  then state  $\top^0$  is assigned to  $vl$  (transition (36) or (37) was used in  $v$ ), so the subtree is clearly accepted. Now consider the case when  $t(v) = \vee$ . Note that since  $\eta$  is a proper  $\sharp$ -chain,  $t_v \in W^{\{\sharp, \sharp^1\}}$ , so  $\varepsilon$  is a  $\sharp$ -closed boolean path, so  $l$  is a boolean path,  $t_{vl} \in W^{\{\sharp, \sharp^1\}}$  and  $t(vl) = \vee$ . Run  $\rho$  uses transition (33) in  $v$ , so  $vl$  has the state label of  $W_\vee^0$ . Note that  $W_\vee^0$  has exactly the same set of transitions as  $W^1$  over the letter  $\vee$ , so the subtree  $t_{vl}$  can be accepted from  $W_\vee^0$ , by Remark 3.10.25.

Now consider subtree that diverges right from  $\eta$  in some node  $v \prec \eta$ . Note that it can only happen if  $t(v) = \vee$ , because only transitions over  $\vee$  propagate  $H$ -states to the left. Hence, path  $r$  is boolean in  $t_v$  and, since  $\eta$  is a proper  $\sharp$ -chain,  $t_v \in H \subseteq W^{\{\sharp, \sharp^1\}}$ , so  $t_{vr} \in W^{\{\sharp, \sharp^1\}}$ , by Remark 3.8.12. Then, by Proposition 3.10.20,  $t_{vr} \in H$ ,  $t_{vr} \in P^\omega[\delta_a^L]$  or  $t_{vr} \in N^\omega[\delta_a^L]$ . However, since  $\eta$  was selected the right-most proper  $\sharp$ -chain,  $t_{vr} \notin H$ . Therefore, by Lemma 3.10.29,  $t_{vr}$  is accepted from state  $P^1$  or from  $N^1$ . Then  $\rho$  can accept the subtree by using transition (34) or (35), respectively, in node  $v$ .

Now assume that  $t \notin H$ , and assume that there is an accepting run  $\rho$  of  $\mathcal{H}^{\mathcal{L}, \bar{\mathcal{L}}}$  from  $H^1$  on  $t$ . Let us call  $\eta$  the branch that  $\rho$  assigns  $H$ -states to. As noted above in order for  $\rho$  to be accepting on  $\eta$  it needs to be a  $\sharp$ -chain. Since  $t \notin H$ ,  $\eta$  is not proper. Therefore, there is  $v \prec \eta$  such that  $t_v \notin W^{\{\sharp, \sharp^1\}}$ . Then there is an  $\{\sharp, \sharp^1\}$ -incorrect boolean path  $w$  in  $t_v$ . There are three cases to be considered for  $w$ :

1.  $w$  diverges left from  $\eta$ , i.e. there are  $u, x$  such that  $w = ulx$ ,  $vu \prec \eta$ ,  $vul \not\prec \eta$ —then  $ul$  is a boolean path in  $t_v$  (as a prefix of a boolean path), so  $t(vu) = \vee$  and transition (33) was used in  $vu$ , so state  $W_\vee^0$  is

assigned to  $vul$ . Boolean path  $x$  is  $\{\#, \#^1\}$ -incorrect in  $t_{vul}$  (as a suffix of a  $\{\#, \#^1\}$ -incorrect path), so  $t_{vul} \notin W^{\{\#, \#^1\}}$  and  $t_{vul}$  is not accepted from  $W_\vee^0$ , because a subset of  $W^{\{\#, \#^1\}}$  is accepted from this state;

2.  $w$  diverges right from  $\eta$ , i.e. there are  $u, x$  such that  $w = urx$ ,  $vu \prec \eta$ ,  $vur \not\prec \eta$ —then one of the transitions (34), (35) was used in node  $vu$  (as the only ones that propagate an  $H$ -state to the left), and state  $P^1$  or state  $N^1$  is assigned to  $vur$ . Boolean path  $x$  is  $\{\#, \#^1\}$ -incorrect in  $t_{vur}$  (as a suffix of a  $\{\#, \#^1\}$ -incorrect path), so  $t_{vur} \notin W^{\{\#, \#^1\}}$ . By Corollary 3.10.2 and Lemma 3.10.29,  $t_{vur}$  is not accepted from any of states  $P^1, N^1$ ;
3.  $vw \prec \eta$ —the only case when a prefix of a  $\#$ -closed boolean path is a  $\{\#, \#^1\}$ -incorrect boolean path is when the prefix ends in a node with label  $\vee$  whose left successor has label different than  $\vee$ . Therefore,  $t(vw) = \vee$ ,  $t(vwl) \neq \vee$  and  $vwl \not\prec \eta$ . In such a case  $\rho$  uses transition (33) in  $vw$ , therefore, state  $W_\vee^0$  is assigned to  $vwl$ . Since there is no transition over label different than  $\vee$  from state  $W_\vee^0$ , the run is stuck.

In all the cases we have obtained a contradiction with the assumption that the run is accepting, so there is no accepting run from state  $H^1$  on tree  $t \notin H$ .  $\square$

**Corollary 3.10.34** *If  $L \subseteq T_A$ , automaton  $\mathcal{L}$  recognizes  $L$  and  $\overline{\mathcal{L}}$  recognizes  $\overline{L}$ , then  $\mathcal{H}^{\mathcal{L}, \overline{\mathcal{L}}}$  recognizes language  $\sigma^\omega(L)$ .*

**Proof:**

Since the initial states of automaton  $\mathcal{H}^{\mathcal{L}, \overline{\mathcal{L}}}$  are  $N^1, H^1, B^2$ , the claim comes from Corollary 3.10.23, Lemma 3.10.29, Remark 3.10.27, and Lemma 3.10.33.  $\square$

**Lemma 3.10.35** *If automata  $\mathcal{L}$  and  $\overline{\mathcal{L}}$  are unambiguous then automaton  $\mathcal{H}^{\mathcal{L}, \overline{\mathcal{L}}}$  is unambiguous.*

**Proof:**

Note that, by Corollary 3.10.23, Lemma 3.10.29, Remark 3.10.27, and Lemma 3.10.33, languages accepted from the initial states of  $\mathcal{H}^{\mathcal{L}, \overline{\mathcal{L}}}$  are pairwise disjoint. Therefore, if there are two distinct accepting runs on the same tree, they start with the same state. As in the previous unambiguity proofs, we consider a node  $v$  in such runs where they use different transitions, but used the same transitions on the path to the node. In particular, in such a

case, the runs assign the same state to  $v$ . Thanks to the Lemma 3.10.31 we know that it is not possible for any of the states of automaton  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$ , so the runs assign one of  $H$ -states to  $v$ . The only nondeterministic choice for these states is by letter  $\vee$ . Three transitions are possible from any of  $H$ -states over letter  $\vee$ : (33), (34), (35). They assign states  $H^1$ ,  $P^1$ ,  $N^1$ , respectively, to the right successor. Since the three states have pairwise disjoint accepted languages, at most one of them can be used in an accepting run on a given tree.  $\square$

**Proof (of Theorem 3.10.24):**

Let  $L \subseteq T_A$ , let unambiguous automaton  $\mathcal{L}$  recognizes  $L$  and let unambiguous automaton  $\bar{\mathcal{L}}$  recognizes  $\bar{L}$ . By Corollary 3.10.30 and Lemma 3.10.31, unambiguous automaton  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$  recognizes  $\sigma^\omega(L)$ . By Corollary 3.10.34 and Lemma 3.10.35, unambiguous automaton  $\mathcal{H}^{\mathcal{L}, \bar{\mathcal{L}}}$  recognizes  $\overline{\sigma^\omega(L)}$ .  $\square$

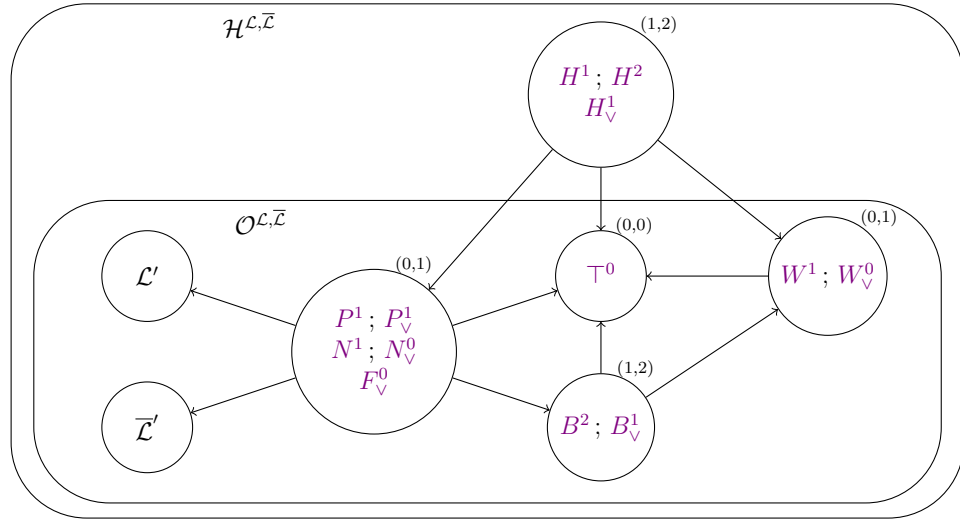


Figure 3.7: Strongly connected components of automata  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$  and  $\mathcal{H}^{\mathcal{L}, \bar{\mathcal{L}}}$ .

Looking at the diagram of strongly connected components depicted in Figure 3.7, we conclude that:

**Remark 3.10.36** *If a language  $L$  is recognized by an unambiguous (respectively nondeterministic) automaton of index  $I$ , and  $\bar{L}$  is recognized by an unambiguous (respectively nondeterministic) automaton of index  $J$ , then  $\sigma^\omega(L)$  and  $\overline{\sigma^\omega(L)}$  are recognized by unambiguous (respectively nondeterministic) automata of index  $\{(0, 1), (1, 2)\} \cup I \cup J$ .*



### 3.10.2 Topological Properties


In this section we prove that the operation has the planned topological complexity, namely:

**Theorem 3.10.37** *Let  $A$  be a nonempty alphabet, let  $L \subseteq T_A$  and let  $a \in A$ . Then for each  $n \geq 0$ ,  $\sigma_a^n(L) \leq_w \sigma_a^\omega(L)$ .*

**Proof:**

For each  $n$  we show a continuous function  $f_n$  reducing  $\sigma^n(L)$  to  $\sigma^\omega(L)$ . The definition is inductive.

For  $n = 0$  we define  $f_0 : T_A \rightarrow T_{A_\omega}$  as:

$f_0(t)$  = 
 (the left subtree does not matter,  
we put an arbitrary fixed tree there)

By the fixpoint definition of  $P^\omega$  and  $N^\omega$  sets (namely by equation (3.19)) we get:

$$t \in \sigma^0(L) = L \implies f_0(t) \in P^\omega[\delta_a^L] = \sigma^\omega(L) \quad (3.27)$$

$$t \in \overline{\sigma^0(L)} = \overline{L} \implies f_0(t) \in N^\omega[\delta_a^L] \subseteq \overline{\sigma^\omega(L)} \quad (3.28)$$

because  $M \subseteq \widehat{\pi_a^{(\omega)}}^{-1}(M)$  for each  $M \subseteq T_A$  (see equation (3.18)).

Recall that operation  $\sigma$  is defined using projection  $\hat{\pi}_a : T_{A_\sigma} \rightarrow T_A$ , which maps letters from  $A_\sigma \setminus A$  to  $a$  (see equation (3.9)). Note that this projection is trivial whenever  $\{\vee, \neg, \sharp\} \subseteq A$ . Therefore, if we iterate application of  $\sigma$ , then only for the innermost application, the projection is different than identity (alphabet of language  $\sigma(L)$  already contains  $\vee, \neg$  and  $\sharp$ ). To avoid confusion we will use a separate notation for  $\hat{\pi}_a$  on each iteration level. Let then  $\pi_a^{(n)}$  be the following projection of the space of set  $\sigma^{n+1}(L)$  onto the space of set  $\sigma^n(L)$ :

$$\pi_a^{(n)} = \begin{cases} id : T_{A_\sigma} \rightarrow T_{A_\sigma} & \text{if } n > 0 \\ \widehat{\pi}_a : T_{A_\sigma} \rightarrow T_A & \text{if } n = 0 \end{cases}$$

Let us fix an arbitrary tree  $t_B$  from  $T_{\{\vee, \neg, \#, \#1\}} \cap B_{A_\omega}^{\{\#, \#1\}}$ . For example let:

$$t_B :=$$

where  $\neg$ -labeled subtrees are subtrees with all nodes labeled with  $\neg$ .

The definition of  $f_n : T_{A_\sigma} \rightarrow T_{A_\omega}$  for  $n > 0$  follows. Let  $f_n(t)$  be tree  $t$  with two kinds of modifications performed (the kinds are mutually exclusive, by definition):

1. Subtrees to the right from the end of  $\sharp$ -closed boolean paths are replaced according to  $f_{n-1}$ . Namely, if  $v$  is a  $\sharp$ -closed boolean path in  $t$  then:  $f_n(t)_{vr} = f_{n-1} \circ \pi_a^{(n-1)}(t_{vr})$ . Note that no  $\sharp$ -closed boolean path is a prefix of a different  $\sharp$ -closed boolean path so there is no ambiguity in the definition.
2. If  $v$  is a  $\sharp^1$ -closed boolean path (it is possible that  $\sharp^1 \in A$ —we do not want to restrict this) then we replace the subtree of node  $v$  with  $t_B$ , i.e.  $f_n(t)_v = t_B$  in this case.

Now we prove by induction on  $n \geq 0$  the following properties, that in particular imply that  $f_n$  are appropriate reductions:

$$\begin{aligned} t \in \sigma^{n+1}(L) = P^\sigma[\lambda_a^L] &\implies f_{n+1}(t) \in P^\omega[\delta_a^L] = \sigma^\omega(L) \\ t \in N^\sigma[\lambda_a^L] &\implies f_{n+1}(t) \in N^\omega[\delta_a^L] \\ t \in B^\sharp &\implies f_{n+1}(t) \in B_{A_\omega}^{\{\sharp, \sharp^1\}} \end{aligned}$$

Since sets  $P^\sigma[\lambda_a^L]$ ,  $N^\sigma[\lambda_a^L]$ ,  $B^\sharp$  cover the whole space  $T_{A_\sigma}$ , the above implications entail that  $f_n(t) \notin H$ .

First consider  $t \in B^\sharp$ , i.e.  $t$  has a  $\sharp$ -incorrect boolean path or branch  $\alpha$ . If  $\alpha$  is infinite or if it is finite and  $t(\alpha) \neq \sharp^1$ , then the path is also  $\{\sharp, \sharp^1\}$ -incorrect and the replacements do not touch  $\alpha$  (no  $\{\sharp, \sharp^1\}$ -closed boolean path is a prefix of a  $\{\sharp, \sharp^1\}$ -incorrect boolean path), so  $\alpha$  is also a  $\{\sharp, \sharp^1\}$ -incorrect boolean path in  $f_{n+1}(t)$ , and  $f_{n+1}(t) \in B_{A_\omega}^{\{\sharp, \sharp^1\}}$ . If  $t(\alpha) = \sharp^1$  then the path above node  $\alpha$  is not changed by  $f_{n+1}$  and  $f_{n+1}(t)_\alpha = t_B$ . In such a case  $\alpha$  is a boolean path in  $f_{n+1}(t)$  and  $\varepsilon$  is a  $\{\sharp, \sharp^1\}$ -incorrect boolean path in  $f_{n+1}(t)_\alpha$ , so  $\alpha$  is a  $\{\sharp, \sharp^1\}$ -incorrect boolean path in  $f_{n+1}(t)$ , and  $f_{n+1}(t) \in B_{A_\omega}^{\{\sharp, \sharp^1\}}$ .

Let now  $t \in P^\sigma[\lambda_a^L]$ . It means that there is a context  $t^P \in P^\sigma$  such that  $t$  is  $t^P$  with leaves  $\mathbf{t}$  replaced with some trees with  $\sharp$  in the root and a tree from  $\pi_a^{(n)-1}(\sigma^n(L))$  as the right subtree and leaves  $\mathbf{f}$  with some trees with  $\sharp$  in the root and a tree from  $\pi_a^{(n)-1}(\overline{\sigma^n(L)})$  as the right subtree (see equation (3.10)). Let us consider the effect of application of  $f_{n+1}$  to  $t$ . Since  $t$  does not have  $\sharp$ -incorrect paths, only modification (1) is applied. This modification does not touch the part of  $t$  coming from  $t^P$ , but for each  $v$  such that  $t^P(v) \in \{\mathbf{t}, \mathbf{f}\}$ ,  $t_{vr}$  is replaced with  $f_n(\pi_a^{(n)}(t_{vr}))$ . Recall that if  $t^P(v) = \mathbf{t}$

then  $t_{vr} \in \pi_a^{(n)^{-1}}(\sigma^n(L))$ , therefore:

$$f_{n+1}(t_{vr}) \in f_n \left( \pi_a^{(n)} \left( \pi_a^{(n)^{-1}}(\sigma^n(L)) \right) \right) = f_n(\sigma^n(L)) \subseteq \sigma^\omega(L) = P^\omega[\delta_a^L] \quad (3.29)$$

where the last inclusion comes from inductive assumption for  $n > 0$  and from equation (3.27) for  $n = 0$ . If  $t^P(v) = \mathbf{ff}$  then  $t_{vr} \in \pi_a^{(n)^{-1}}(\overline{\sigma^n(L)})$ , therefore:

$$\begin{aligned} f_{n+1}(t_{vr}) &\in f_n \left( \pi_a^{(n)} \left( \pi_a^{(n)^{-1}}(\overline{\sigma^n(L)}) \right) \right) = f_n(\overline{\sigma^n(L)}) \\ &= \begin{cases} f_n(N^\sigma[\lambda_a^L] \cup B^\#) & \text{for } n > 0 \\ f_n(\overline{L}) & \text{for } n = 0 \end{cases} \\ &\subseteq \begin{cases} N^\omega[\delta_a^L] \cup B_{A_\omega}^{\{\#, \#^1\}} & \text{for } n > 0 \\ N^\omega[\delta_a^L] & \text{for } n = 0 \end{cases} \\ &\subseteq N^\omega[\delta_a^L] \cup B_{A_\omega}^{\{\#, \#^1\}} \end{aligned} \quad (3.30)$$

where the first inclusion comes from inductive assumption for  $n > 0$  and from equation (3.28) for  $n = 0$ .

Since for each node  $v$  such that  $t^P(v) \in \{\mathbf{tt}, \mathbf{ff}\}$ ,  $f_{n+1}(t)(v) = \#$  and  $f_{n+1}(t_{vr})$  is as in (3.29) and (3.30), by Corollary 3.10.5, we get:

$$f_{n+1}(t) \in P^\sigma \left[ \mathbf{tt} \mapsto \begin{array}{c} \textcircled{\#} \\ \swarrow \quad \searrow \\ T_{A_\omega} \quad P^\omega[\delta_a^L] \end{array}, \mathbf{ff} \mapsto \begin{array}{c} \textcircled{\#} \\ \swarrow \quad \searrow \\ T_{A_\omega} \quad N^\omega[\delta_a^L] \cup B_{A_\omega}^{\{\#, \#^1\}} \end{array}, \top \mapsto T_{A_\omega} \right] \quad (3.31)$$

Now note that, since  $(P^\omega, N^\omega) = \text{Fix}(\Psi)$  and by the definition of  $\eta^{\mathbf{P}, \mathbf{N}}$  (equation (3.17)), we get:

$$P^\omega = P^\sigma[\eta^{P^\omega, N^\omega}] \supseteq P^\sigma \left[ \mathbf{tt} \mapsto \begin{array}{c} \textcircled{\#} \\ \swarrow \quad \searrow \\ \top^\omega \quad P^\omega \end{array}, \mathbf{ff} \mapsto \begin{array}{c} \textcircled{\#} \\ \swarrow \quad \searrow \\ \top^\omega \quad N^\omega \cup B_{A_\omega}^{\{\#, \#^1\}} \end{array}, \top \mapsto \top^\omega \right] \quad (3.32)$$

Since  $\mathbf{tt}^\omega, \mathbf{ff}^\omega, \top^\omega$  do not occur in trees from  $P^\sigma$  (although may occur in trees from  $P^\omega$  or  $N^\omega$ ), (3.32) implies:

$$P^\omega[\delta_a^L] \supseteq P^\sigma \left[ \mathbf{tt} \mapsto \begin{array}{c} \textcircled{\#} \\ \swarrow \quad \searrow \\ T_{A_\omega} \quad P^\omega[\delta_a^L] \end{array}, \mathbf{ff} \mapsto \begin{array}{c} \textcircled{\#} \\ \swarrow \quad \searrow \\ T_{A_\omega} \quad N^\omega[\delta_a^L] \cup B_{A_\omega}^{\{\#, \#^1\}} \end{array}, \top \mapsto T_{A_\omega} \right] \quad (3.33)$$

By (3.31) and (3.33) we get:

$$f_{n+1}(t) \in P^\omega[\delta_a^L] \quad (3.34)$$

If  $t \in N^\sigma [\lambda_a^L]$ , then  $t$  is equal to a context  $t^N \in N^\sigma$  with  $\mathbf{t}$  and  $\mathbf{f}$  labeled holes replaced with trees from the same languages as for  $t^P$ , above. Therefore, exactly the same sequence of arguments leads to the conclusion that:

$$f_{n+1}(t) \in N^\omega [\delta_a^L]$$

□

**Corollary 3.10.38** *If  $A$  is a finite nonempty alphabet and  $L \subseteq T_A$  is stretchable then, for each  $n \geq 0$ ,  $\sigma^n(L) <_w \sigma^\omega(L)$ .*

**Proof:**

By Theorem 3.10.37, for each  $n$ ,  $\sigma^n(L) \leq_w \sigma^\omega(L)$ . If for some  $n$ ,  $\sigma^\omega(L) \leq_w \sigma^n(L)$ , then  $\sigma^{n+1}(L) \leq_w \sigma^\omega(L) \leq_w \sigma^n(L)$ , what contradicts Theorem 3.8.37, by stretchability of  $L$ . □

### 3.10.3 Stretchability

In this section we prove that limit-step operation  $\sigma^\omega$  preserves stretchability.

**Theorem 3.10.39** *If  $A$  is a finite nonempty alphabet and a language  $L \subseteq T_A$  is stretchable then  $\sigma^\omega(L)$  is also stretchable.*

**Proof:**

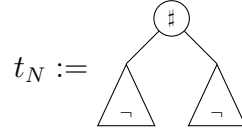
We want to apply Lemma 3.8.36. To do this we recall that the automaton  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$  is composed using components:  $\mathcal{L}'$ ,  $\bar{\mathcal{L}}'$ ,  $\{\mathbf{T}^0\}$ , by  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}} = \mathcal{O}(\mathcal{L}', \bar{\mathcal{L}}', \{\mathbf{T}^0\})$ . The components recognize languages  $\widehat{\pi_a^{(\omega)}^{-1}}(L)$ ,  $\widehat{\pi_a^{(\omega)}^{-1}}(\bar{L})$ ,  $T_{A_\omega}$ , respectively. Note that if a function  $f : T_A \rightarrow T_A$  stretches  $L$  with respect to a sequence  $\{a_n\}$ , then  $f \circ \widehat{\pi_a^{(\omega)}^{-1}}$  stretches each of the languages  $\widehat{\pi_a^{(\omega)}^{-1}}(L)$ ,  $\widehat{\pi_a^{(\omega)}^{-1}}(\bar{L})$ ,  $T_{A_\omega}$  with respect to  $\{a_n\}$ . Therefore, the languages are simultaneously stretchable and assumption 5 of the lemma is satisfied.

The following function determines places where the automaton enters one of the three components, as required by assumption 2 of the lemma:

$$p(a, d) = 1 \iff (a = \#^1) \vee (a \in \{\neg, \#\} \wedge d = l)$$

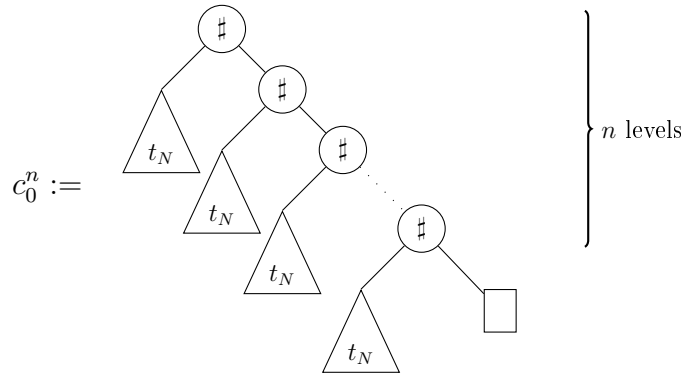
None of the parameter states is initial in  $\mathcal{O}$ , so assumption 1 of the lemma is also satisfied.

We are left with showing contexts as in assumptions 4 and 3 of the lemma. First we fix a tree that is always (regardless of  $L$ ) accepted from states  $N^1$ ,  $T^0$  and  $\tau$ . Let then:

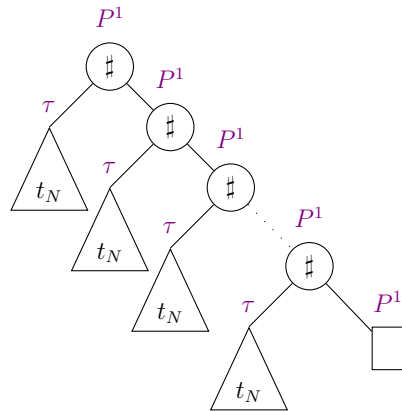


where the  $\neg$ -labeled subtrees are subtrees with all nodes labeled with  $\neg$ .

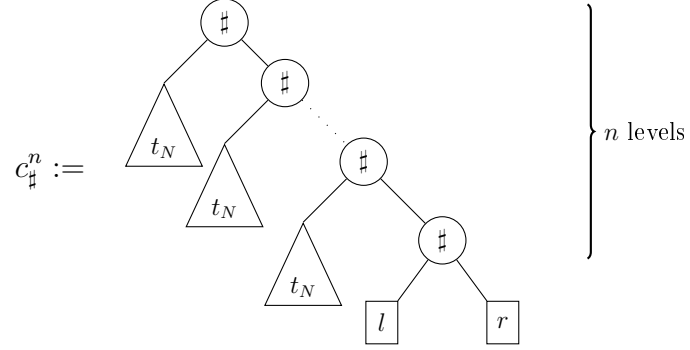
As a context  $c_0^n$  required in assumption 4 we take:



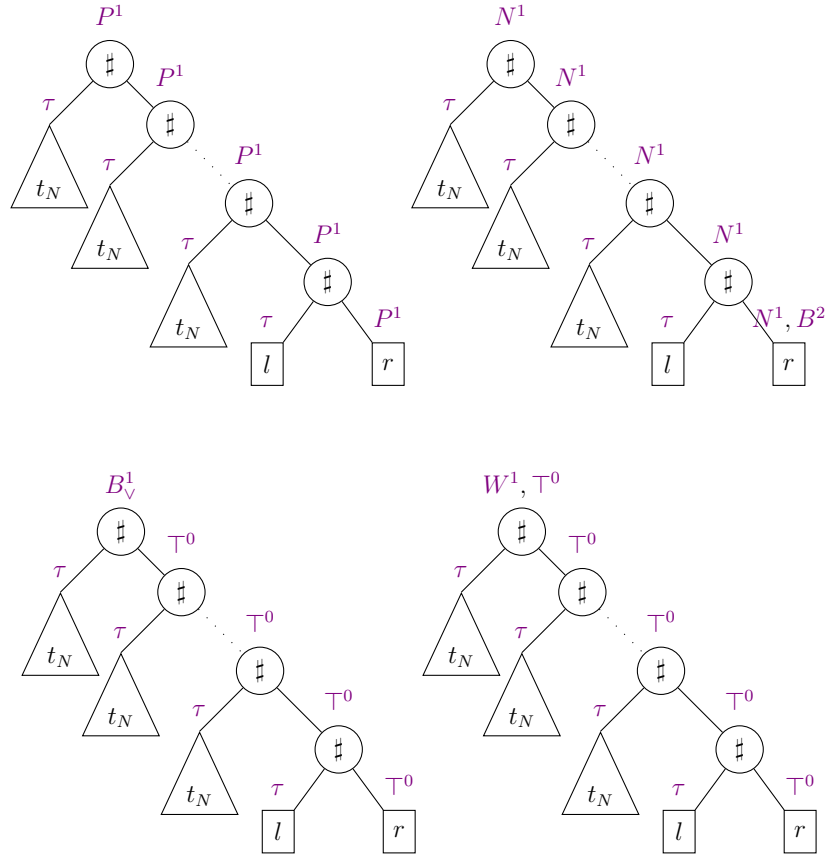
We immediately present all possible runs on such a context from the initial state of  $\mathcal{O}$ :



We use a similar context (or actually a series of contexts) for  $c_{\#}^n$ :

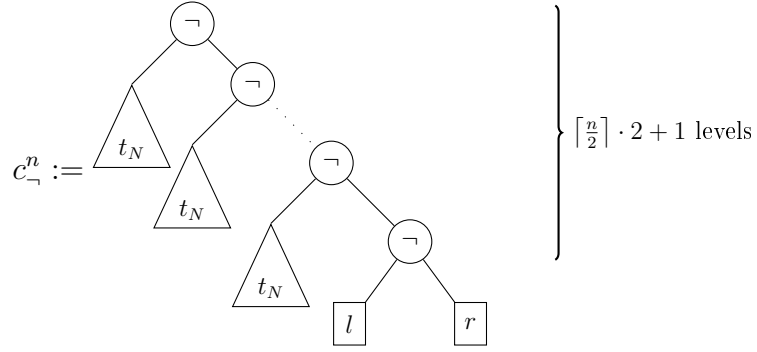


All the possible runs on this context follow:

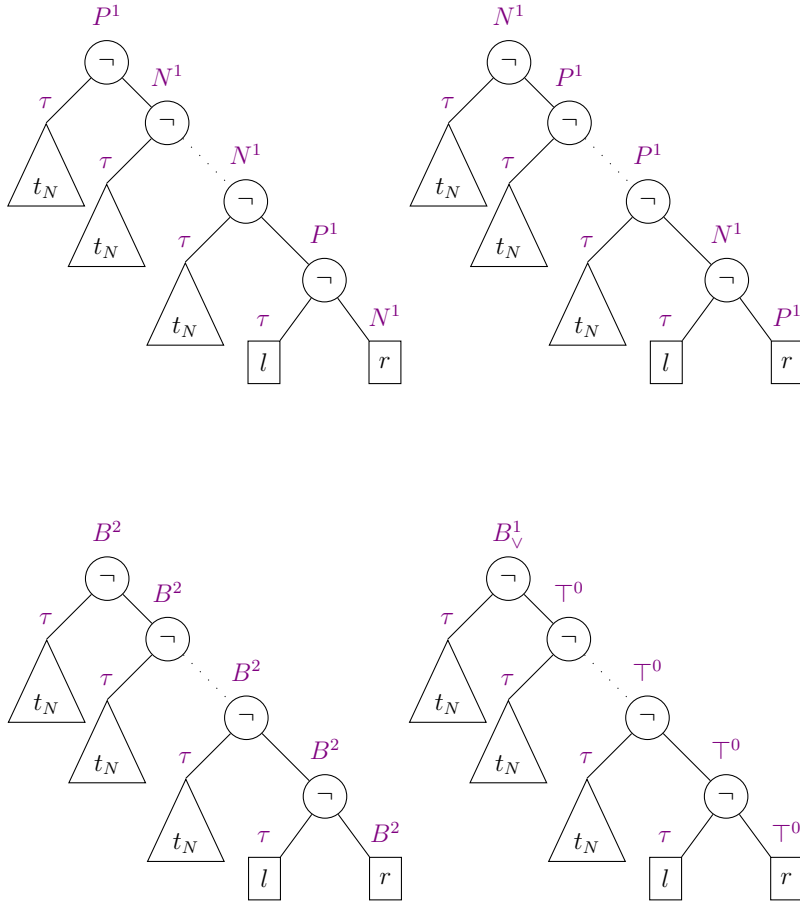


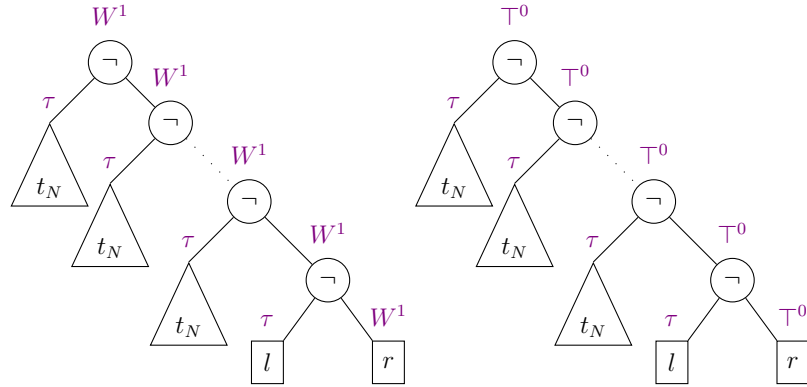
They correspond exactly to the transitions (28), (29), (30), (23), (16), (19) of  $\mathcal{O}$  over the letter  $\#$ .

Now we present the sequence of contexts for letter  $\neg$ :



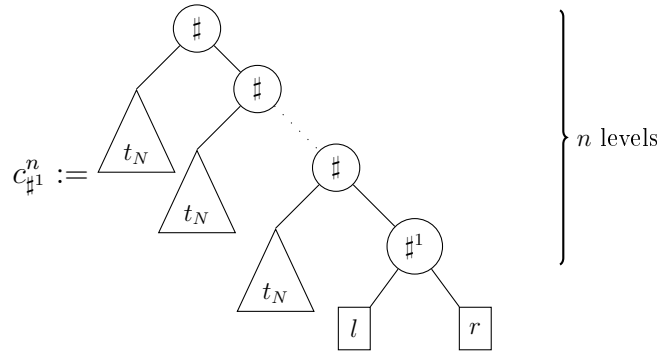
The runs on it:



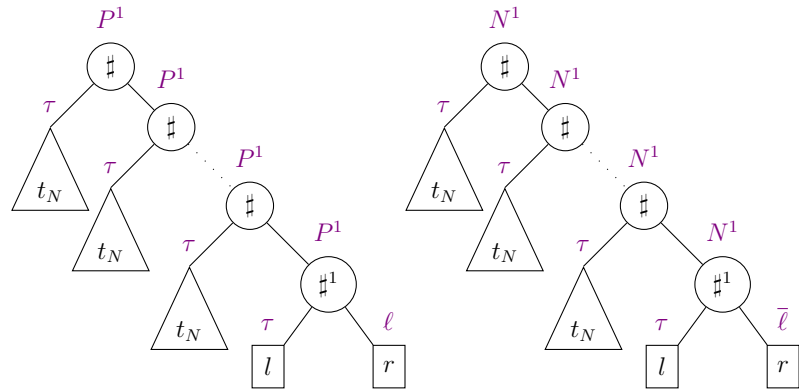


They correspond exactly to the transitions (5), (6), (22), (23), (7), (19) of  $\mathcal{O}$  over the letter  $\neg$ .

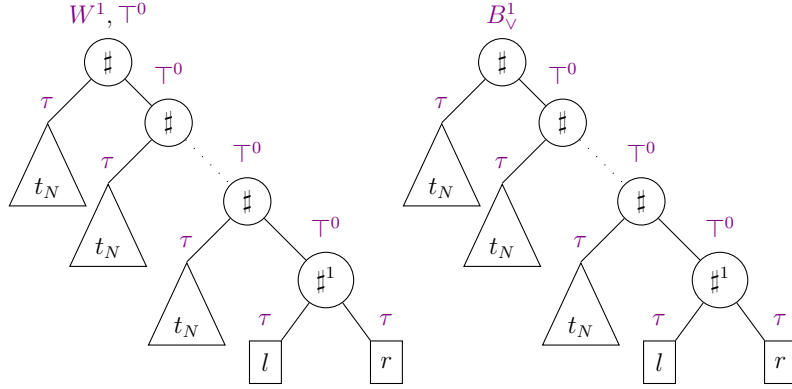
Now the contexts for  $\#^1$ :



All the runs on it:

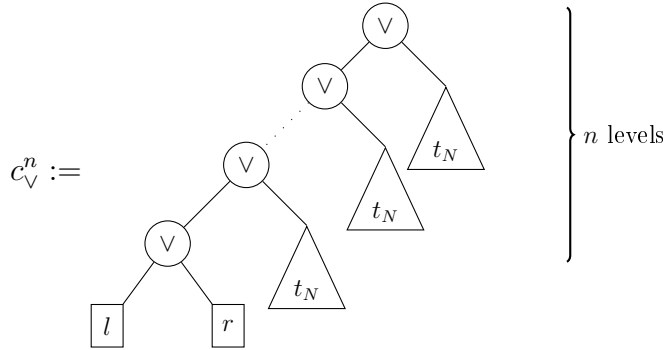




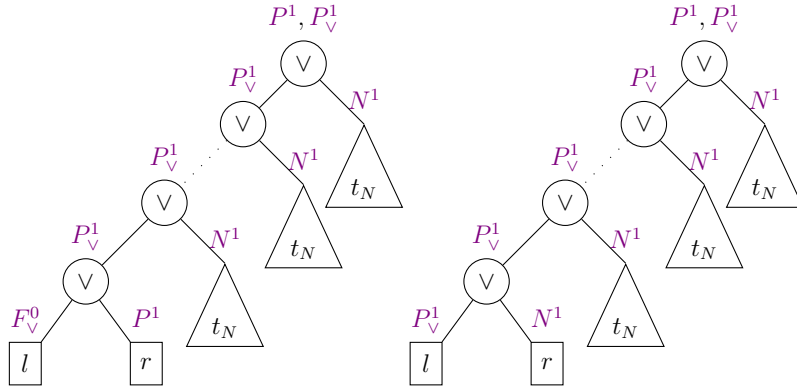


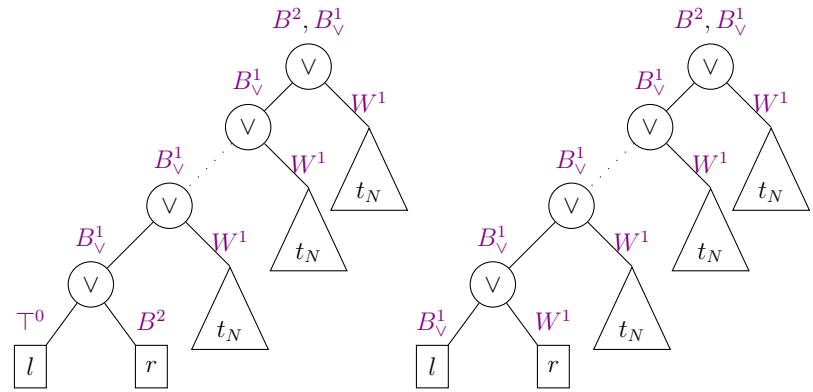
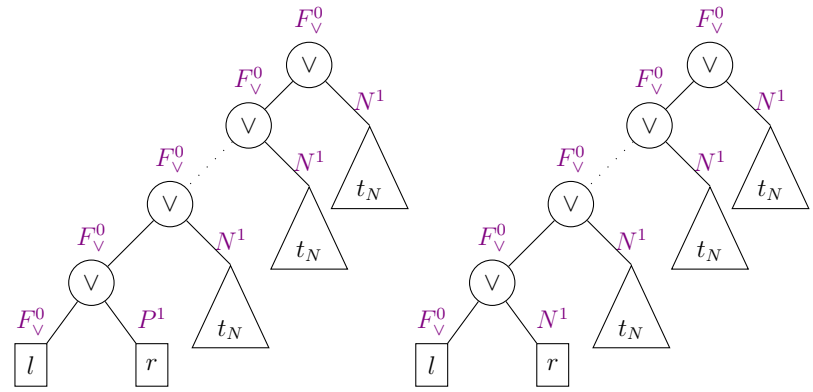
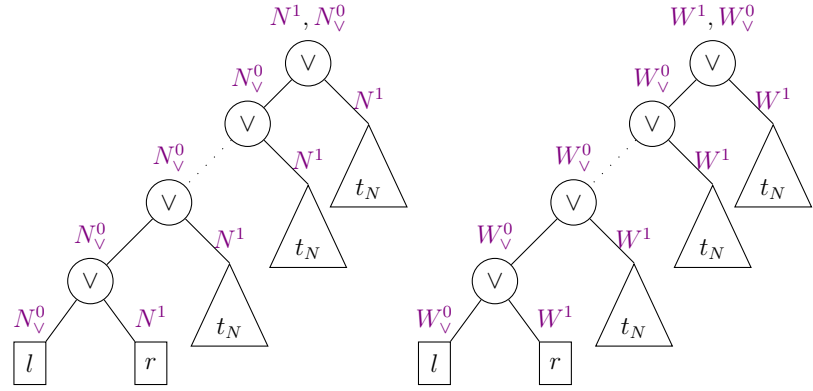
One could note that also transition (30) could be taken in the root of this context, but then the run would stuck, because there is no transition from  $B^2$  over any of letters  $\#$ ,  $\#^1$ . The runs depicted above correspond to transitions (31), (32), (17), (20), (24) of automaton  $\mathcal{O}$  over the letter  $\#^1$ .

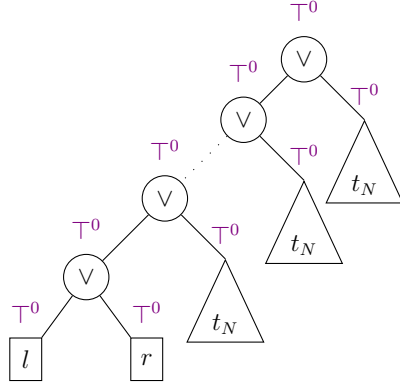
Now the context for  $\vee$ :



All the runs on it:

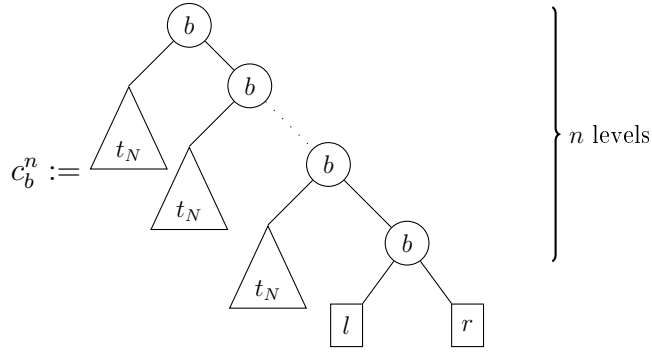




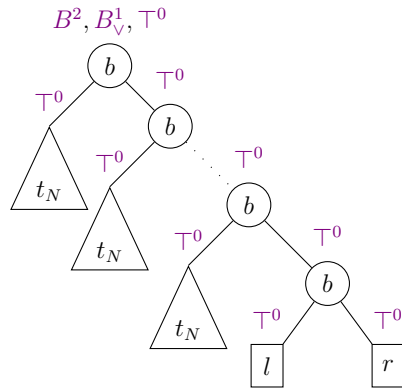


The runs correspond to transitions (25), (2), (3), (4), (26), (27), (11), (12), (18) of automaton  $\mathcal{O}$ . Those are all the possible transitions over the letter  $\vee$ .

Now let  $b \in A_\omega \setminus \{\vee, \neg, \sharp, \sharp^1\}$ . We put:



All possible runs on it follow.



This corresponds to the fact that transitions (15), (21) is the only one over  $b$ .

We have constructed all the contexts required by assumptions 3 and 4 of Lemma 3.8.36, so we have shown that all the assumptions are satisfied. Therefore, we can use the lemma to obtain that:

$$\mathcal{O}\left(\widehat{\pi_a^{(\omega)}}^{-1}(L), \widehat{\pi_a^{(\omega)}}^{-1}(\bar{L}), T_{A_\omega}\right) = \sigma^\omega(L) \quad \text{is stretchable.}$$

□

### 3.11 The Hierarchy

Since we have proven that operation  $\sigma^\omega$ :

1. has the required topological hardness property ( $\forall_n \sigma^n(L) <_w \sigma^\omega(L)$ ),
2. preserves bi-unambiguity, and
3. preserves stretchability,

we conclude that, similarly to operation  $\sigma$ , the operation can be iterated and that the two operations,  $\sigma$  and  $\sigma^\omega$ , can be alternated to obtain more and more complex sets. This leads to the observation, that the sequence presented in equation (3.15) can be extended to obtain the following strictly increasing (with respect to the Wadge order) sequence of bi-unambiguous languages:

$$\begin{aligned} & \emptyset <_w \sigma(\emptyset) <_w \sigma^2(\emptyset) <_w \sigma^3(\emptyset) <_w \dots \\ & <_w \sigma^\omega(\emptyset) <_w \sigma(\sigma^\omega(\emptyset)) <_w \sigma^2(\sigma^\omega(\emptyset)) <_w \sigma^3(\sigma^\omega(\emptyset)) <_w \dots \\ & <_w \sigma^\omega(\sigma^\omega(\emptyset)) <_w \sigma(\sigma^\omega(\sigma^\omega(\emptyset))) <_w \sigma^2(\sigma^\omega(\sigma^\omega(\emptyset))) <_w \dots \\ & \vdots \\ & <_w (\sigma^\omega)^k(\emptyset) <_w \sigma((\sigma^\omega)^k(\emptyset)) <_w \sigma^2((\sigma^\omega)^k(\emptyset)) <_w \dots \\ & <_w (\sigma^\omega)^{k+1}(\emptyset) <_w \dots \\ & \vdots \end{aligned} \tag{3.35}$$

The length of the above sequence is  $\omega^2$  (elements are numbered with all the ordinals less than  $\omega^2$ ). All languages in the sequence are in the unambiguous index class  $\mathcal{L}^{uamb}((0, 1), (1, 2))$ .

Note that an increasing sequence of bi-unambiguous languages similar to the one shown in (3.35) can be built starting from any stretchable bi-unambiguous language.

## 3.12 Further Work and Discussion

### 3.12.1 A Vain Attempt

The author of this thesis have attempted to extend the sequence (3.35) to the length  $\omega^\omega$ . It was done by the introduction of  $\omega$  operations  $\sigma^{\omega^i}$  similar to the operation  $\sigma^\omega$ . If we think of operation  $\sigma^\omega$  as of  $\omega$  iterations of operation  $\sigma$ , then the new operations would correspond to  $\omega^2, \omega^3, \dots$  iterations of  $\sigma$ .

Recall that the construction of  $\sigma^\omega$  introduces a symbol  $\sharp^1$ . In the definition of  $\sigma$  we have considered a notion of ill-foundedness, namely a tree with an incorrect boolean branch cannot belong to  $\sigma(L)$ . An incorrect boolean branch is, roughly speaking, a branch turning right infinitely many times with  $\vee$  and  $\neg$  labels and without  $\sharp$  labels. The operation  $\sigma^\omega$  introduces another type of ill-foundedness. This one is connected with the existence of proper  $\sharp$ -chain, i.e., roughly speaking, a branch turning right infinitely many times labeled with  $\vee, \neg$  and  $\sharp$ , with no label  $\sharp^1$ . Each operation  $\sigma^{\omega^i}$  introduces a symbol  $\sharp^i$  and, together with it, a new type of ill-foundedness. The most complex point of this construction is fixing a relation between the types of ill-foundedness. Recall that in the case of  $\sigma^\omega$ , a  $\sharp$ -chain is proper only if there is no incorrect boolean branch diverging from it. Similar solutions lead to the definition of  $\sigma^{\omega^i}$  that was considered by the author.

Unfortunately an automaton considered by the author as a candidate for recognizing language  $\sigma^{\omega^i}(L)$  (a natural extension of  $\mathcal{O}^{\mathcal{L}, \bar{\mathcal{L}}}$ ) occurred not to be unambiguous for  $i > 1$ . In order to obtain this conclusion the author have written a Python script that translated the unambiguity question for language  $\sigma^{\omega^2}(\emptyset)$  into a parity game that was then solved by PGSolver—a parity game solving tool by Oliver Friedmann and Martin Lange [FL15].

### 3.12.2 Difference Hierarchy

In the already cited paper by Finkel and Simonnet [FS09], the difference hierarchy of analytic sets is considered. The authors show a sequence of regular tree languages hard for the levels  $D_\alpha(\Sigma_1^1)$  of the hierarchy, for  $\alpha < \omega^\omega$ . Then, on page 10, they note that the languages (even the first one in the sequence) are not recognized by unambiguous tree automata.

The base for the construction by Finkel and Simonnet is the set  $EB$  as defined in Example 3.2.4. If we look at the construction there, we see that we can instead use the unambiguous set  $G$  from Section 3.7 as a base, and hardness results still hold. Actually the proofs remain exactly the same, since they only use  $\Sigma_1^1$ -hardness of the basic set.

Now we note that each automaton built during the construction is unam-

biguous if we use the unambiguous automaton  $\mathcal{G}$  recognizing  $G$  and the unambiguous automaton  $\mathcal{L}$  recognizing its complement (i.e. the set  $L$  from Section 3.7) as basic building blocks. Indeed, for a given tree  $t$ , the automaton described in the proof of Lemma 4.5 in the paper [FS09] always selects the path corresponding to the smallest ordinal  $\omega^{n-1} \cdot a_{n-1} + \omega^{n-2} \cdot a_{n-2} + \dots + \omega \cdot a_1 + a_0$  for which the tree  $t_{l^{a_{n-1}} r^{l^{a_{n-2}} r \dots r l^{a_0}}}$  belongs to  $G$ , and proves that all paths corresponding to smaller ordinals end up in something that does not belong to  $G$  (i.e. belongs to  $L$ ). The unambiguous automaton  $\mathcal{G}$  is used to verify belonging to  $G$  and  $\mathcal{L}$  is used to verify belonging to  $L$ .

As a result, using example languages  $G$  and  $L$  from this thesis and the construction from the paper [FS09] by Finkel and Simonnet, we get a sequence of unambiguous languages hard for the classes  $D_\alpha(\Sigma_1^1)$ , for  $\alpha < \omega^\omega$ . Note that, since each set at any countable level of the difference hierarchy of analytic sets is, by definition, in  $\sigma(\Sigma_1^1)$ , all the languages in the sequence continuously reduce to the language  $\sigma(G)$ . Therefore, the constructions presented in this thesis are able to produce bi-unambiguous languages of higher topological complexity than all the languages in the sequence considered by Finkel and Simonnet.

### 3.12.3 Unambiguous vs Regular

In this section we discuss possible upper topological complexity bounds for unambiguous languages. As noted before, the thesis does not present any results in this regard. We only enumerate some open questions.

First observe that it is still possible that the topological complexity of nondeterministic and unambiguous languages is the same. Let us formalize it in the following.

**Question 3.12.1** *Is it true that: For each regular language  $L$  of infinite trees there is such an unambiguous (or even bi-unambiguous) language  $U$  that  $L$  is Wadge reducible to  $U$ ?*

If the answer to the above question is negative, then there is a spectrum of possibilities. Note that all the unambiguous languages presented in this thesis are of alternating index  $\mathcal{L}^{alt}((0, 1), (1, 2))$ . According to the author's knowledge the answer to the following question is not known.

**Question 3.12.2** *Are all unambiguous tree languages in the alternating index class  $\mathcal{L}^{alt}((0, 1), (1, 2))$ ?*

Or more generally:

**Question 3.12.3** *Is there an alternating index class that contains all unambiguous (bi-unambiguous) languages? If yes, then what is the smallest such class?*

Note that negative answer to the last question does not contradict potential negative answer to Question 3.12.1, because Fact 3.4.12 works only one direction, and there may be languages outside  $\mathcal{L}^{alt}((\iota, \kappa))$  reducible to  $W_{(\iota, \kappa)}$ . An answer to Question 3.12.3 would fill the missing paragraph in Section 3.4.1 about relations between the hierarchies.

### 3.12.4 Wadge Hierarchy

As it was noted before, the finest way to measure the difference in the topological complexity of unambiguous and regular languages is by the use of the Wadge hierarchy. This approach, though, when applied to non-Borel sets, usually uses an extension of ZFC with an appropriate axiom of determinacy. The thesis of Kevin Fournier [Fou16, Section 1.3.1] gives a summary of results on the Wadge hierarchy of non-Borel languages using full axiom of determinacy. The measure applied there (as well as e.g. in the work by Murlak and Duparc [Mur06, DM07]) is the length of the Wadge hierarchies induced by languages in a given class. Some lower bound for this length is also given for unambiguous automata in the paper by Duparc, Fournier and the author of this thesis [DFH15]. The paper shows a sequence of unambiguous languages of strictly increasing Wadge degree of length  $\varphi^2(0)$ , which is much larger than the length of the Wadge hierarchy for deterministic languages discovered by Murlak [Mur06], and much smaller than the length of the longest such sequence known for regular languages, which is  $\varphi_\omega(0)$  (given by Fournier [Fou16, Section 6.3]).

It would be interesting to compare the sequence of unambiguous languages from the mentioned paper [DFH15] with the sequence given in this chapter. It is not easy, since they use different approaches. The one from the paper is very long and described in terms of the Wadge degrees using axiom of co-analytic determinacy. The sequence described in this chapter is much shorter, but the gap between the elements is much bigger, since each step concerns going beyond the sigma-algebra generated by the previous ones. Additionally in the thesis we have decided not to use axioms beyond ZFC—which makes the comparison harder.

The two sequences have significant similarities, though. They both are built from bi-unambiguous languages (although the paper does not state it explicitly) that can be recognized by automata of index  $\{(0, 1), (1, 2)\}$  (the paper mentions index  $(0, 2)$ , but it can be observed that it is actually

$\{(0, 1), (1, 2)\}$ ). In particular the class  $\mathcal{L}^{alt}((0, 1), (1, 2))$  is a common upper topological complexity bound for both the examples.

### 3.12.5 Bi-unambiguous vs Unambiguous

Recall that the unambiguous index hierarchy is strict (see Remark 3.4.14). One can ask:

**Question 3.12.4** *Is there an unambiguous index class that contains all bi-unambiguous languages?*

The following question is somehow related to Question 3.12.1 and Question 3.12.3.

**Question 3.12.5** *What is the difference in the topological complexity of unambiguous and bi-unambiguous languages?*



# Notations

$\omega$	the smallest infinite ordinal, page 18
$\omega_1$	the smallest uncountable ordinal, page 18
$ A $	the cardinality (number of elements) of a set $A$ , page 18
$\overline{A}$	the complement of a set $A$ , page 18
$\inf A$	the infimum of a subset $A$ of an ordered set $X$ , i.e. the greatest element of $X$ less than or equal to each element of $A$ , page 118
$\mathbb{P}(A)$	the power set of a set $A$ , i.e. the set of subsets of $A$ , page 18
$\bigsqcup_{i \in I} A_i$	the disjoint union of a family of sets. The semantics of this operator is the same as of the ordinary union operator, we use it to indicate that the sets in the family are pairwise disjoint, page 31
$A \sqcup B$	the disjoint union of sets $A$ and $B$ . As an operator it is an ordinary union, it only additionally informs that the operands are disjoint. If used in multi-argument union, like $A_1 \sqcup A_2 \sqcup \dots \sqcup A_k$ , it designates pairwise disjointness, page 106
$\bigcap_{n < \omega}^{\rightarrow} L_n$	the intersection of the (not necessarily strictly) decreasing sequence $\{L_n\}_{n < \omega}$ of sets. Semantics of this operator is identical as of ordinary intersection. It is only used to indicate that for each $n < m$ , $L_m \subseteq L_n$ , page 31
$\pi_i(A)$	the projection of a subset $A$ of a product space on the $i$ 'th coordinate, page 29
$BC(A)$	the smallest algebra containing all elements of $A$ , page 18
$\sigma(A)$	the smallest sigma-algebra containing all elements of $A$ , page 18

$\exists_x^\infty \varphi(x)$	there exists infinitely many $x$ for which $\varphi$ holds, page 30
$\forall_x^\infty \varphi(x)$	for all except finitely many $x$ , $\varphi$ holds, page 30
$\exists^{fin}$	the existential weak second order quantifier, i.e. the one ranging over finite subsets of a domain, page 37
$\forall^{fin}$	the universal weak second order quantifier, i.e. the one ranging over finite subsets of a domain, page 37
$\exists!_x \varphi(x)$	there exists exactly one $x$ for which $\varphi$ holds, page 41
$\varphi[\alpha]$	a formula $\varphi$ with a substitution $\alpha$ applied, page 36
$\mathbb{1}_X$	the characteristic function $\mathbb{1}_X : D \rightarrow \{0, 1\}$ of a subset $X \subseteq D$ of a set $D$ , page 37
$\text{dom}(f)$	the domain of a (partial) function $f$ , page 22
$\text{id}$	an identity function, page 114
$A \multimap B$	for $f : A \multimap B$ , $f$ is a partial function from a set $A$ to $B$ , page 86
$f \upharpoonright_A$	restriction of the domain of a function, namely, for $f : X \rightarrow Y$ , if $A \subseteq X$ , then $f \upharpoonright_A : A \rightarrow Y$ is defined by $f \upharpoonright_A(x) = f(x)$ for each $x \in A$ , page 89
$B_d(x, r)$	the ball with center $x$ and radius $r$ in metric $d$ (we may write $B(x, r)$ if the metric is clear from the context), page 19
$\tau Y$	relative topology: for topological space $(X, \tau)$ and for $Y \subseteq X$ , $\tau Y = \{U \cap Y : U \in \tau\}$ , page 19
$X \simeq Y$	topological spaces $X$ and $Y$ are homeomorphic, page 19
$\mathbb{N}^\omega$	the Baire space, page 24
$\mathcal{C}$	the Cantor space $\mathcal{C} = \{0, 1\}^\omega$ , page 24
$A \equiv_w B$	a set $A$ is Wedge-equivalent to a set $B$ , i.e. they are mutually continuously reducible to each other, page 33
$A \leq_w B$	a set $A$ is continuously reducible (Wedge-reducible) to a set $B$ , page 27
$\text{Fix}(\Phi)$	the least fixed point of an operator $\Phi$ , page 113

$A^*$	the set of all finite words over an alphabet $A$ , page 21
$A^\omega$	the set of all $\omega$ -words over an alphabet $A$ , page 21
$A^{\leq\omega}$	the set of finite and infinite words over alphabet $A$ , i.e. $A^{\leq\omega} = A^* \cup A^\omega$ , page 21
$A^+$	the set of all finite nonempty words over an alphabet $A$ , page 54
$wv$	the concatenation of words $w$ and $v$ , page 21
$LM$	the concatenation of sets of words, $LM = \{wv : w \in L, v \in M\}$ , page 21
$L^{-1}M$	the left quotient of language $M$ w.r.t. language $L$ : $L^{-1}M = \{v : \exists_{u \in L} \exists_{w \in M} uv = w\}$ . We also write $v^{-1}w$ to designate a suffix of $w$ (the element of $\{v\}^{-1}\{w\}$ ) if we know that $v \preceq w$ , page 26
$LM^{-1}$	the right quotient of language $L$ w.r.t. language $M$ : $LM^{-1} = \{v : \exists_{w \in L} \exists_{u \in M} vu = w\}$ . We also write $wv^{-1}$ to designate a prefix of $w$ (the element of $\{w\}\{v\}^{-1}$ ) if we know that $v$ is a suffix of $w$ , page 27
$L^*$	the Kleene star of a set $L$ of words, i.e. $L^* = \bigcup_{i \geq 0} L^i$ , page 21
$L^\omega$	the set of infinite concatenations of words from $L$ , i.e. $L^\omega = \{w_0w_1w_2 \dots : \forall_{i \in \omega} w_i \in L\}$ , page 21
$v \preceq w$	the prefix order, i.e. $w = vu$ for some $u \in A^*$ , where $A$ is an alphabet, page 22
$v \prec w$	the strict prefix order, page 22
$\varepsilon$	the empty sequence (or word), page 21
$ w $	the length of a sequence $w$ , page 21
$\alpha \upharpoonright_n$	the prefix of length $n$ of a (finite or infinite) sequence $\alpha$ , page 22
$T^B$	the set of all unlabeled trees on $B$ ( $B$ -branching trees), page 22
$T_A^B$	the set of all full $B$ -branching trees over $A$ ( $A$ -labeled), page 22
$T_A^{B, \text{fin}}$	the set of all finite $B$ -branching $A$ -labeled trees, page 22
$T_A^{\text{fin}}$	the set of all finite binary $A$ -labeled trees, page 22

$T_A$	the set of all full binary $A$ -labeled trees, page 22
$[t]$	the set of infinite branches of a tree $t \in T^X$ , i.e. $[t] := \{\alpha \in X^\omega : \forall_{n < \omega} \alpha \upharpoonright_n \in t\}$ , page 56
$t(\alpha)$	the sequence of labels of a tree $t$ along a branch $\alpha$ , page 22
$t_v$	the subtree of tree $t$ rooted in node $v$ , page 27
$t \upharpoonright_w$	the section of a multi-branching tree $t$ along a word $w$ on the first branching coordinate, page 55
$L[\alpha]$	a substitution $\alpha$ applied to a parametrized language $L$ of trees, page 86
$\mathcal{A}_q$	an automaton $\mathcal{A}$ with initial state changed to $q$ ( $q$ may also be a set of states), page 88
$\mathcal{L}^\Gamma(I)$	an index class of languages. A language is in index class $\mathcal{L}^\Gamma(I)$ , for $\Gamma \in \{ndet, det, uamb, alt, weak\}$ , if there is a nondeterministic (respectively deterministic, unambiguous, alternating, weak) automaton of index $I$ that recognizes it, page 93
$\Delta_\kappa^\Gamma$	a delta index class of languages, $\Delta_\kappa^\Gamma = \mathcal{L}^\Gamma((0, \kappa - 1)) \cap \mathcal{L}^\Gamma((1, \kappa))$ , page 93
$\overline{(\iota, \kappa)}$	the dual index to an index $(\iota, \kappa) \in \{0, 1\} \times \omega$ , $\overline{(\iota, \kappa)} = (1 - \iota, \kappa + (-1)^\iota)$ , page 95
$\mathcal{O}(f(n))$	asymptotic upper bound notation, namely, $g(n) = \mathcal{O}(f(n))$ means that there exist such constants $c, n_0$ , that for $n > n_0$ , $g(n) \leq c \cdot f(n)$ , page 48

# Bibliography

- [Add04] J.W. Addison. Tarski’s theory of definability: common themes in descriptive set theory, recursive function theory, classical pure logic, and finite-universe logic. *Annals of Pure and Applied Logic*, 126(1–3):77–92, 2004. Provinces of logic determined. Essays in the memory of Alfred Tarski. Parts I, {II} and {III}.
- [ADMN08] André Arnold, Jacques Duparc, Filip Murlak, and Damian Niwiński. On the topological complexity of tree languages. In *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, pages 9–28, 2008.
- [AN92] André Arnold and Damian Niwiński. Fixed point characterization of weak monadic logic definable sets of trees. In *Tree Automata and Languages*, pages 159–188. 1992.
- [AN07] André Arnold and Damian Niwiński. Continuous separation of game languages. *Fundamenta Informaticae*, 81(1-3):19–28, 2007.
- [Arn99] André Arnold. The  $\mu$ -calculus alternation-depth hierarchy is strict on binary trees. *ITA*, 33(4/5):329–340, 1999.
- [BC06] Mikołaj Bojańczyk and Thomas Colcombet. Bounds in  $\omega$ -regularity. In *LICS*, pages 285–296, 2006.
- [BGMS14] Mikołaj Bojańczyk, Tomasz Gogacz, Henryk Michalewski, and Michał Skrzypczak. On the decidability of MSO+U on infinite trees. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Proceedings, Part II*, pages 50–61, Copenhagen, Denmark, 2014.
- [Bil10] Marcin Bilkowski. personal communication, 2010.
- [BL69] J. Richard Büchi and Lawrence H. Landweber. Definability in the monadic second-order theory of successor. *J. Symb. Log.*, 34(2):166–170, 1969.

- [BM13] Marcin Bilkowski and Michał Skrzypczak. Unambiguity and uniformization problems on infinite trees. In *Computer Science Logic 2013, CSL 2013*, pages 81–100, Torino, Italy, 2013.
- [Boj04] Mikołaj Bojańczyk. A bounding quantifier. In *Computer Science Logic*, volume 3210 of *Lecture Notes in Computer Science*, pages 41–55, 2004.
- [Boj10] Mikołaj Bojańczyk. Beyond  $\omega$ -regular languages. In *STACS*, pages 11–16, 2010.
- [Boj11] Mikołaj Bojańczyk. Weak MSO with the unbounding quantifier. *Theory Comput. Syst.*, 48(3):554–576, 2011.
- [BPT16] Mikołaj Bojańczyk, Paweł Parys, and Szymon Toruńczyk. The MSO+U theory of  $(N, <)$  is undecidable. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016*, pages 21:1–21:8, Orléans, France, 2016.
- [Bra99] Julian C. Bradfield. Fixpoint alternation: Arithmetic, transition systems, and the binary tree. *ITA*, 33(4/5):341–356, 1999.
- [Bra03] Julian C. Bradfield. Fixpoints, games and the difference hierarchy. *ITA*, 37(1):1–15, 2003.
- [BT09] Mikołaj Bojańczyk and Szymon Toruńczyk. Deterministic automata and extensions of Weak MSO. In *FSTTCS*, pages 73–84, 2009.
- [BT12] Mikołaj Bojańczyk and Szymon Toruńczyk. Weak MSO+U over infinite trees. In *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012*, pages 648–660, Paris, France, 2012.
- [Büc60] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundle. Math.*, 6:66–92, 1960.
- [Büc62] J. Richard Büchi. On a decision method in restricted second-order arithmetic. In *Proc. 1960 Int. Congr. for Logic, Methodology and Philosophy of Science*, pages 1–11, 1962.
- [CDFM09] Jérémie Cabessa, Jacques Duparc, Alessandro Facchini, and Filip Murlak. The Wadge hierarchy of max-regular languages. In

*IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009*, pages 121–132, IIT Kanpur, India, 2009.

- [CJK<sup>+</sup>17] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *STOC 2017 Theory Fest: 49th Annual ACM Symposium on the Theory of Computing*, Montreal, Canada, 2017.
- [CKLV13] Thomas Colcombet, Denis Kuperberg, Christof Löding, and Michael Vanden Boom. Deciding the weak definability of Büchi definable tree languages. In *Computer Science Logic 2013, CSL 2013*, pages 215–230, Torino, Italy, 2013.
- [CL07] Arnaud Carayol and Christof Löding. MSO on the infinite binary tree: Choice and order. In *CSL*, pages 161–176, 2007.
- [CLNW10] Arnaud Carayol, Christof Löding, Damian Niwiński, and Igor Walukiewicz. Choice functions and well-orderings over the infinite binary tree. *Central European Journal of Mathematics*, 8:662–682, 2010.
- [Col12] Thomas Colcombet. Forms of determinism for automata (invited talk). In *STACS*, pages 1–23, 2012.
- [DFH15] Jacques Duparc, Kevin Fournier, and Szczepan Hummel. On unambiguous regular tree languages of index  $(0, 2)$ . In *24th EACSL Annual Conference on Computer Science Logic, CSL 2015*, pages 534–548, Berlin, Germany, 2015.
- [DM07] Jacques Duparc and Filip Murlak. On the topological complexity of weakly recognizable tree languages. In *Fundamentals of Computation Theory, 16th International Symposium, FCT 2007, Proceedings*, pages 261–273, Budapest, Hungary, 2007.
- [Dup01] Jacques Duparc. Wadge hierarchy and Veblen hierarchy part I: Borel sets of finite rank. *J. Symb. Log.*, 66(1):56–86, 2001.
- [EJ91] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy. In *Foundations of Computer Science*, pages 368–377, 1991.
- [Elg61] Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the AMS*, 98:21–52, 1961.

- [Eme90] E. Allen Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B)*, pages 995–1072. 1990.
- [Far01] Berndt Farwer.  $\omega$ -automata. In Grädel et al. [GTW02], pages 3–20.
- [FL15] Oliver Friedmann and Martin Lange. PGSolver [computer software]. <https://github.com/tcsprojects/pgsolver.git>, February 2015. Version: 3.3.
- [FLS15] Olivier Finkel, Dominique Lecomte, and Pierre Simonnet. An upper bound on the complexity of recognizable tree languages. *RAIRO - Theor. Inf. and Applic.*, 49(2):121–137, 2015.
- [FMS16] Alessandro Facchini, Filip Murlak, and Michał Skrzypczak. Index problems for game automata. *ACM Trans. Comput. Log.*, 17(4):24:1–24:38, 2016.
- [Fou16] Kevin Fournier. *The Wadge Hierarchy: Beyond Borel Sets*. PhD thesis, Université Paris Diderot and Université de Lausanne, 2016.
- [FS09] Olivier Finkel and Pierre Simonnet. On recognizable tree languages beyond the Borel hierarchy. *Fundamenta Informaticae*, 95(2-3):287–303, 2009.
- [GMMS14] Tomasz Gogacz, Henryk Michalewski, Matteo Mio, and Michał Skrzypczak. Measure properties of game tree languages. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014. Proceedings, Part I*, pages 303–314, Budapest, Hungary, 2014.
- [GS83] Yuri Gurevich and Saharon Shelah. Rabin’s uniformization problem. *J. Symb. Log.*, 48(4):1105–1119, 1983.
- [GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.
- [GZ78] Wojciech Guzicki and Paweł Zbierski. *Podstawy teorii mnogości*. Państwowe Wydawnictwo Naukowe, 1978.



- [HMT10] Szczepan Hummel, Michał Skrzypczak, and Szymon Toruńczyk. On the topological complexity of  $\text{MSO}+\text{U}$  and related automata models. In *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010. Proceedings*, pages 429–440, Brno, Czech Republic, 2010.
- [HS12] Szczepan Hummel and Michał Skrzypczak. The topological complexity of  $\text{MSO}+\text{U}$  and related automata models. *Fundamenta Informaticae*, 119(1):87–111, 2012.
- [Hum12] Szczepan Hummel. Unambiguous tree languages are topologically harder than deterministic ones. In *GandALF*, pages 247–260, 2012.
- [JGP99] Edmund M. Clarke Jr., Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999.
- [Kec95] Alexander S. Kechris. *Classical Descriptive Set Theory*, volume 156 of *Graduate Texts in Mathematics*. Springer-Verlag, 1995.
- [Kur72] Kazimierz Kuratowski. *Introduction to Set Theory and Topology*. PWN, Warszawa, second edition, 1972.
- [KV98] Orna Kupferman and Moshe Y. Vardi. Verification of fair transition systems. *Chicago J. Theor. Comput. Sci.*, 1998, 1998.
- [McN66] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521–530, 1966.
- [Mos91] Andrzej Włodzimierz Mostowski. Hierarchies of weak automata and weak monadic formulas. *Theoretical Computer Science*, 83(2):323–335, 1991.
- [MS84] David E. Muller and Paul E. Schupp. Alternating automata on infinite objects, determinacy and Rabin’s theorem. In *Automata on Infinite Words, Ecole de Printemps d’Informatique Théorique*, pages 100–107, Le Mont Dore, 1984.
- [MS14] Henryk Michalewski and Michał Skrzypczak. Unambiguous Büchi is weak. *CoRR*, abs/1401.4025, 2014.
- [Mur05] Filip Murlak. On deciding topological classes of deterministic tree languages. In *Computer Science Logic, 19th International*

*Workshop, CSL 2005, 14th Annual Conference of the EACSL, Proceedings*, pages 428–441, Oxford, UK, 2005.

- [Mur06] Filip Murlak. The Wadge hierarchy of deterministic tree languages. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Proceedings, Part II*, pages 408–419, Venice, Italy, 2006.
- [Mur08a] Filip Murlak. *Effective Topological Hierarchies of Recognizable Tree languages*. PhD thesis, University of Warsaw, 2008.
- [Mur08b] Filip Murlak. Weak index versus Borel rank. In *STACS 2008, 25th Annual Symposium on Theoretical Aspects of Computer Science, Proceedings*, pages 573–584, Bordeaux, France, 2008.
- [Niw86] Damian Niwiński. On fixed-point clones (extended abstract). In *ICALP*, pages 464–473, 1986.
- [NW96] Damian Niwiński and Igor Walukiewicz. Ambiguity problem for automata on infinite trees. unpublished note, 1996.
- [NW98] Damian Niwiński and Igor Walukiewicz. Relating hierarchies of word and tree automata. In *STACS 98, 15th Annual Symposium on Theoretical Aspects of Computer Science, Proceedings*, pages 320–331, Paris, France, 1998.
- [NW03] Damian Niwiński and Igor Walukiewicz. A gap property of deterministic tree languages. *Theor. Comput. Sci.*, 1(303):215–231, 2003.
- [NW05] Damian Niwiński and Igor Walukiewicz. Deciding nondeterministic hierarchy of deterministic tree automata. *Electr. Notes Theor. Comput. Sci.*, 123:195–208, 2005.
- [PP04] Dominique Perrin and Jean-Éric Pin. *Infinite Words*. Elsevier, 2004.
- [Rab69] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the AMS*, 141:1–23, 1969.
- [Rab70] Michael O. Rabin. Weakly definable relations and special automata. *Mathematical Logic and Foundations of Set Theory*, pages 1–23, 1970.

- [Rog01] Markus Roggenbach. Determinization of Büchi-automata. In Grädel et al. [GTW02], pages 43–60.
- [SA05] Luigi Santocanale and André Arnold. Ambiguous classes in mu-calculi hierarchies. *Theoretical Computer Science*, 333(1-2):265–296, 2005.
- [Saf88] Shmuel Safra. On the complexity of omega-automata. In *29th Annual Symposium on Foundations of Computer Science*, pages 319–327, White Plains, NY, USA, 1988.
- [Sch07] Sven Schewe. Solving parity games in big steps. In *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science, 27th International Conference, Proceedings*, pages 449–460, New Delhi, India, 2007.
- [Sku93] Jerzy Skurczyński. The Borel hierarchy is infinite in the class of regular sets of trees. *Theoretical Computer Science*, 112(2):413–418, 1993.
- [Sou17] Michail J. Souslin. Sur une definition des ensembles B sans nombres tranfinis. *Comptes Rendus Acad. Sci. Paris*, 164:88–91, 1917.
- [Sri98] Sashi M. Srivastava. *A Course on Borel Sets*, volume 180 of *Graduate Texts in Mathematics*. Springer-Verlag, 1998.
- [SW16] Michal Skrzypczak and Igor Walukiewicz. Deciding the topological complexity of Büchi languages. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*, pages 99:1–99:13, Rome, Italy, 2016.
- [Tar55] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.*, 5:285–309, 1955.
- [Tho97] 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.
- [TL93] Wolfgang Thomas and Helmut Lescow. Logical specifications of infinite computations. In *REX School/Symposium*, pages 583–621, 1993.

- [Wad72] William W. Wadge. Degrees of complexity of subsets of the Baire space. *Notices of the American Mathematical Society*, 19:714–715, 1972.
- [Wad84] William W. Wadge. *Reducibility and determinateness on the Baire space*. PhD thesis, University of California, Berkeley, 1984.
- [Wag79] Klaus Wagner. On  $\omega$ -regular sets. *Information and Control*, 43(2):123–177, 1979.