Università degli Studi di Torino
Dipartimento di Matematica G. Peano

Université de Lausanne
Faculté des HEC - Institut des Systèmes d'Information

Regular tree languages in the first two levels of the Borel hierarchy

Advisors:
Prof. Luca Motto Ros
Prof. Jacques Duparc

Candidate:
Filippo Cavallari

Acknowledgements

In general I do not like to dedicate a whole section of a scientific text, like a PhD thesis, to acknowledgements. In this way, it would seem that they are something mandatory and formal, something that has to be there, among the chapters of a thesis, as if "Acknowledgements" was a planned section. I think that the best way to thank someone is with actions, by doing something for them, and that gratitude does not necessarily have to be connected to a written text. In spite of that, the relationships between myself and the people who have given me some help is too asymmetric and I already know that I will not be able to give them back even a small part of what I received. Therefore, the only thing I can do is to proceed with the usual procedure and to list all the people who deeply supported me during this difficult path.

First of all I want to express my gratitude to my two advisors: Luca Motto Ros (my advisor in Turin) and Jacques Duparc (my advisor in Lausanne). Two great people. Very different one from the other, but both very important for me. In particular, I want to thank Luca for his tireless backing, for teaching me the tools of Descriptive Set Theory and for inviting in Turin some researchers whom I could work with. I want to thank Jacques for teaching me the techniques of Automata Theory and for giving me the chance to establish a co-tutorship with the University of Lausanne, a very positive experience for me.

The people I am going to thank now are listed in accordance with the cities who welcomed me, one after the other. First Turin, then Lausanne, finally Warsaw.

My life in Turin was extraordinary: I made a lot of experiences at work as well as outside the University. At the University of Turin I found a very stimulating environment. The Mathematical Logic group of Turin is very

active: a lot of seminars, workshops and many initiatives are constantly organised and this helped me to keep my research prolific. So I want to express my gratitude to this group, all the professors who work there and all the researchers, even those who had just a small chat with me. In particular special thanks go to Alessandro Andretta, Riccardo Camerlo and Matteo Viale, three serious professors who keep the Mathematical Logic group very active.

Obviously I have to thank also my PhD colleagues whom I met in Turin: their constant support was extremely important for me. Many thanks also to the colleagues of my "same-field" (Mathematical Logic): Giorgio Audrito and Filippo Calderoni. Moreover I want to warmly thank Eleonora, always close to me, from the beginning to the end of this difficult path. And I continue with Luisa, Martina, Chiara, Alessandro and Carlo.

Now let us move to Lausanne. Also the experience in Lausanne was terrific. I could say that my stay in Lausanne has been my real first abroad-life long-term experience. The Swiss atmosphere is very particular: Lausanne is nice and clean, the University of Lausanne is well organised and the University campus of Unil offers a very stimulating place where to work. I have to thank two brilliant people who made my stay in Lausanne really pleasant. Huge (really huge!) thanks go to Gianluca Basso. Gianluca was fundamental for me: with his constant help and support to me he was like a guide for me in the new reality of Lausanne. Moreover I want to thank Louis Vuillemier, a nice office companion with whom I could talk about Mathematics and more.

Finally Warsaw. My experience in Warsaw is maybe the most important moment of my PhD, in scientific terms. In Warsaw I found an astonishing research group formed by many experts of Automata Theory, a very significative research area for my PhD. I have visited Warsaw several times during these three years and a half and I met several very interesting people. First of all I want to thank Henryk Michalewski, a serious colleague and also a nice friend. I want to thank Mikołaj Bojańczyk, an extraordinary professor, very friendly and with a great style, whom I esteem very much. And then I want to thank Filip Murlak, Szczepan Hummel and all the other members of the group.

I left the most important thanks at the end. I want to warmly thank Michał Skrzypczak for all he has given me. Michał was first of all a true

friend, then an extraordinary teacher and finally a reliable colleague. Michał taught me a lot of instruments and techniques of Automata Theory and his teachings were fundamental for me to conclude this PhD at best. Despite his skills and competence, I have never perceived any haughtiness or annoyance from him, not even a hint. In fact, his collaboration has always been affectionate, friendly and extremely professional. In short: great friend, great teacher, great researcher.

A bit last minute, thank also to the two referees of my thesis: Olivier Finkel and Matteo Mio. Thank for accepting to read my thesis and for the positive reports. And final thanks to Valérie Chavez for accepting to take part in my committee.

Concludo con dei ringraziamenti di carattere affettivo che esulano dal contesto universitario. Ho, infatti, avuto a fianco a me tante persone che mi hanno sempre sostenuto in questo lungo e difficile percorso. Prima di tutto la mia splendida famiglia: i miei fantastici genitori per la fiducia accordatami sempre in maniera costante, e il mio unico (in tutti i sensi) fratello, grande esempio di incrollabile determinazione per me. Inoltre ringrazio l'amica e collega Arianna per aver condiviso con me tutti i momenti in cui mi sono dovuto confrontare con difficili scelte determinanti per il futuro lavorativo. Momenti delicati, ma che si affrontano meglio con una presenza così preziosa al proprio fianco. Un ultimo personale ringraziamento a mia cugina Emilia per tutto l'aiuto che mi ha dato nell'affinare la mia padronanza della lingua inglese. E infine tutti gli altri amici e parenti a me cari che non ho citato. Grazie di cuore a tutti, davvero!

Symbols

Symbol	Meaning
$\subset$	Strict Inclusion.
$\subseteq$	Inclusion.
$\cap, \bigcap$	Intersection.
$\cup, \bigcup$	Union.
$\sqcup$	Disjoint union.
$\lvert A \rvert$	Cardinality of a set A.
$\mathcal{P}(A)$	Power set of A.
$\emptyset$	Empty set.
A^c	The complement of A.
$A \times B$	The cartesian product of A and B.
$A \setminus B$	The difference between A and B.
$\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$	Standard numerical sets.
$\mathbb{R}^+$	Positive real numbers.
$\mathrm{dom}(f)$	Domain of a function f.
$\mathrm{ran}(f)$	Range of a function f.
f^{-1}	Inverse function of f.
ω	First infinite ordinal.
ω_1	First uncountable ordinal.

List of Automata

Class of Automata	Corresponding definition
Word parity non-deterministic	Definition 1.37.
Word parity deterministic	Definition 1.42.
Tree parity non-deterministic	Definition 2.4.
Tree parity deterministic	Definition 2.8.
Tree parity alternating	Definition 2.13 and 2.14.
Tree weak-alternating	Definition 2.19.
Tree guidable	Definition 2.22.
Tree weak non-deterministic	Definition 2.24.

Introduction

The area of research of this thesis lies between Mathematical Logic and Computer Science: more precisely, this thesis is concerned with connections between Descriptive Set Theory and Automata Theory. Descriptive Set Theory is a branch of Set Theory, that is in turn one of the four big areas in which Mathematical Logic is split (the other ones being Proof Theory, Computability Theory and Model Theory), while Automata Theory is a very important branch of Computer Science. In spite of their different origin these two areas have many interesting connections.

To understand these connections better let us start with considering a general problem in Mathematics. When we work inside a set X and consider subsets of X, a natural issue is to select a class of subsets of X that are, in some sense, "*tractable*", where for the moment we use this word without a formal definition but just with an intuitive idea. For example, when we work inside the set of real numbers $\mathbb{R}$, the role of *tractable* sets could be taken over by Lebesgue-measurable sets, or by sets with the Baire Property, or by other classes, depending on the context we are working in.

Descriptive Set Theory deals with the so-called *Polish spaces*, i.e. topological spaces whose the topology is completely metrizable and separable. A well-know example of a Polish space is the *Cantor space*, denoted by 2^ω, that is the space of the infinite words over the alphabet $2 = \{0, 1\}$. The Cantor space is a canonical example of perfect space and is one of the best known uncountable Polish spaces. We will see that the Cantor space also plays an important role in Automata Theory.

Given a Polish space X, we can select a quite natural collection of *tractable* subsets, namely the *Borel* sets of X. These are the subsets that belong to the *Borel hierarchy*, a topological hierarchy where the simplest sets are the

open and closed sets (i.e. the simplest possible sets in topological terms) and then sets are stratified according to how difficult it is to obtain them by starting with open sets and using countable unions, countable intersections and complement. Therefore, at the first level of the hierarchy we find open and closed sets, at the second level countable intersections of open sets and countable unions of closed sets, and so on. Informally we could say that these sets are "definable" in easy ways: if X is a Polish space and $B \subseteq X$ is a Borel subset of X then we have a good definition of B that uses infinitary Boolean operations and starts with open sets.

Once selected a favourite class of *tractable* sets, usually one can also define a measure of "relative complexity" between the sets inside it. In this way we have *tractable* and *non tractable* sets, and among the tractable sets we can define when a set is more *complex* than another one (and so less tractable).

In the case of Borel sets we have a natural "candidate" for this measure of relative complexity, that is the *Borel rank*. We say that α is the *Borel rank* of a subset B if the α-th level is the lowest where B appears in the hierarchy. When we work with uncountable Polish spaces the Borel hierarchy is a non-collapsing hierarchy of length ω_1 (the first uncountable ordinal). Therefore, among the Borel sets of an uncountable Polish space (e.g. the Cantor space), we can define that A is more complex than B if the Borel rank of A is higher than B. Actually we can also consider more sophisticated measures of complexity, e.g. the Borel hierarchy is refined by the difference hierarchy that in turn is refined by the Wadge hierarchy. For the moment we focus on the Borel hierarchy and the Borel rank.

Now let us move on to Automata Theory. The general framework where we work is the space of infinite words and the space of infinite trees. Let us start with words: fix an alphabet A, i.e. a finite set of symbols, and consider the space A^ω of the infinite words over the alphabet A. Subsets of A^ω are called *word languages*. In this context the role of *tractable* sets is played by *regular* languages.

The concept of *regularity* is quite delicate and it deserves some explanation. Broadly speaking, a set of words is *regular* if we have a reasonable description of it, which means a list of properties that the words it contains satisfy (e.g. "in every word of the set there are finitely many a", "there is no b", "there are four a", and so on). More formally, this "description" has to be

formalised in a suitable logic. It turns out that the right logic to work with is Monadic Second Order Logic, denoted by the acronym MSO: this logic allows second order quantifications that are needed to express many properties related to sets of words. A natural question that arises is why this logic is a "good" choice.

Let us consider for a moment what happens with subsets of $\mathbb{N}$ in the context of Computability Theory. Let us call a set *intuitively computable* if we do have a precise algorithm that allows to say, given a natural number n, whether n is in our set or not. In Computability Theory it is widely accepted that a subset $A \subseteq \mathbb{N}$ is *computable* if there exists a Turing machine $\mathcal{M}$ that *recognises*[1] A. Moreover, it turns out that the subsets of $\mathbb{N}$ recognised by Turing machines are the same as the ones recognised by a lot of different other formalisms, like modern programming languages (C, C++, Java, etc) and such class of sets is stable under basic theoretical operations, like complementation, union, intersection, and so on. All these facts support the so-called *Church-Turing Thesis*: the very strong conjecture that every set that is *intuitively computable* is recognisable by a Turing machine. In other words and from a different point of view Church-Turing Thesis claims that Turing machines are the current strongest computational model that we can implement.

In the case of regular languages, there is not such a strong thesis to justify the choice of the logic, but, in spite of that, also in this case the definition is supported by some mathematical facts. Indeed, it turns out that MSO is equivalent to several classes of *automata*, that essentially are Turing Machines with some constraints. Word parity automata, word Muller automata, word Rabin automata, etc, (see Theorem 1.51 for a complete list of models) are all formalisms that capture exactly regular languages and therefore we can conclude that this class is not tied to a specific formalism, but it is a very steady collection of sets which, moreover, is closed under many operations and many formalisms.

Also in the case of regular sets we can consider a measure of *relative complexity*. Indeed, many of the classes of automata that capture regular languages use *priorities*, i.e. natural numbers assigned to every state of an

[1] The word "recognises" has a formal definition that we do not explain now (for more details see for example [Eil74], [Pap93] and [Soa87]).

automaton. The more priorities an automaton uses, the more complex it is. Hence, the amount of priorities used by automata is a good candidate of measure of relative complexity. We can say that a regular language L is more complicated than another language L' if there exists an automaton $\mathcal{A}$ that recognises L' such that every automaton that recognises L uses more priorities than $\mathcal{A}$. In this way we can *stratify* regular languages in a hierarchy according to this measure.

So far, we observed that in both Descriptive Set Theory and Automata Theory, we have two corresponding classes of *tractable* sets, namely Borel sets and regular languages, and two measures of relative complexity, namely the Borel rank and the priorities used by automata. But the space of infinite words over a finite alphabet A is homeomorphic to the Cantor space 2^ω. Hence it is an uncountable Polish space carrying its own notions of Borel hierarchy and Borel rank. By this fact, some interesting questions about the interplay of the two classes of tractable sets arise. We are interested in studying the relationship between Borel sets and regular languages and, more deeply, between Borel rank and amount of priorities. One of the first results in this direction was proved in [Lan69] by Landweber in 1969: all regular word languages lie inside the first three levels of the Borel hierarchy, so one notion of tractable sets (regular languages) is strictly included into the other one (Borel sets). Moreover, it has been proved (see Section 1.6), that the priorities used by parity *deterministic* automata are strongly connected to levels of the Wadge hierarchy (see Theorem 1.56) which is a sophisticated refinement of the Borel hierarchy.

The space of infinite trees is homeomorphic to the Cantor space 2^ω as well, but in the case of trees the situation is less clear than for words. First of all in this case not all regular languages are Borel sets. Moreover, it is not clear if there is a suitable class of automata that fits the Borel rank of regular tree languages. Thank to some works like [DM07] and [CMS17] it emerges that the class of the so-called *tree weak-aternating automata* (see Definition 2.19) could play this role. The tree languages recognised by them are all Borel (and regular) and there are evidences that the number of priorities used by them could correspond to the Borel rank of the languages they recognise. Two open problems related to this class are:

1. Verify whether they are enough to capture all the Borel regular tree languages.

2. Understand the relationship between priorities used by weak-alternating automata and Borel rank.

Regarding the second point, a reasonable conjecture is that a regular tree language is recognised by a weak-alternating automaton that uses n priorities if and only if it is in the n-th level of the Borel hierarchy. The direction from left to right has been proved in [DM07], while we are far from a proof of the other direction: despite the fact that this conjecture is expected to be true by the community working in this area, there is no single n for which it has been verified. Indeed, it turns out that even for very small n's many totally non trivial difficulties arise when attacking the problem; clarifying the situation in those cases could thus help in finding a way to verify the whole conjecture. The goal of this thesis is precisely to study the role of regularity in the first two levels of the Borel hierarchy on the space of infinite trees labelled over a finite alphabet.

Organisation of the thesis The first chapter of the thesis introduces all the concepts that have informally been mentioned in this introduction. There we formally define the space of infinite words over a finite alphabet and we study this space in topological terms and in terms of regularity. In the last section of the chapter we present the state of the art for word languages.

In the second chapter we introduce tree languages and automata for infinite trees. We give a complete presentation of *alternating automata*, that constitute a class of automata that capture regular tree languages and have some advantages with respect to standard parity automata. Moreover, we will present the state of the art for the space of trees with an overview on the main open problems and conjectures related to Borel regular tree languages.

In the third chapter we start our "climbing" of the Borel hierarchy by presenting a complete analysis of the first level of it. The results in this chapter can be considered folklore, but the proofs that we present are particularly intuitive and, to the best of the author's knowledge, have not appeared elsewhere. In particular we prove the following:

Theorem 0.1. *A regular tree language L is recognised by a weak-alternating automaton that uses only two priorities if and only if it is in the first level of the Borel hierarchy.*

Moreover we present a new use of *guidable automata* related to open sets of infinite trees. Guidable automata are a modern formalism that captures regular languages, but the status of this class of automata is still not very clear. The characterisation for open sets that we give using guidable automata is a small result that could help in understanding the role that this class of automata can assume in the context of regular tree languages.

In the fourth chapter we keep climbing the hierarchy and we analyse the topological levels between the first and the second level of the Borel hierarchy. All the results of this chapter are contained in [BCPS18] and are summed up by the following theorem:

Theorem 0.2. *The following holds:*

1. *Let Γ be a Wadge degree with finite Wadge rank. Then it is decidable if a regular tree language L belongs to Γ.*
2. *It is decidable if a regular tree language L is a Boolean combination of open sets.*
3. *It is decidable if a regular tree language L is in the Borel class $\mathbf{\Delta}^0_2$.*

While the first result is new, the second one was already proved in [BP12], but here we polish and simplify the proofs. The third result was already stated in [FM14], but unfortunately the proof in that work is not correct, so here we fix it.

Finally, the fifth chapter is dedicated to a complete characterisation of the second level of the Borel hierarchy:

Theorem 0.3. *It is decidable if a regular tree language L belongs to the second level of the Borel hierarchy. Moreover, a regular language L is in the second level of the Borel hierarchy if and only if it is recognised by a weak-alternating automaton that uses exactly three priorities.*

The chapter contains all the results proved in [CMS17].

Contents

Chapter 1

Preliminaries

We recall some basic preliminary notions coming from both Descriptive Set Theory and Automata Theory that will be useful throughout the thesis.

The axiomatic framework where we work in is ZFC, the Zermelo-Fraenkel set theory ZF with the Axiom of Choice AC.

1.1 Descriptive Set Theory

Descriptive Set Theory has an important role in our context because it provides some natural hierarchies measuring the complexity of sets of infinite objects read by our classes of automata. We split this section into four different subsections: in the first one we define Polish spaces and in the other ones we see the main hierarchies considered for this class of spaces. For more details about this section we refer the reader to [Kec95].

1.1.1 Polish Spaces

We denote a topological space by the pair (X, τ). If τ is understood from the context, we just write X and suppress τ from the notation.

Definition 1.1. Let (X, τ) be a topological space. A subset $Y \subseteq X$ is *dense* (in X) if $U \cap Y \neq \emptyset$ for any non empty open set $U \in \tau$. The topology τ is *separable* if there exists a subset of X which is countable and dense in X.

Definition 1.2. Let (X, τ) be a topological space. The topology τ is *metrizable* if there exists a metric d on X such that the topology generated by d, denoted by τ_d, coincides with τ.

Definition 1.3. Let (X, d) be a metric space. A sequence $(x_n)_{n\in\mathbb{N}}$ is a *Cauchy sequence* if the following condition holds:

$$\forall \varepsilon \in \mathbb{R}^+ \exists m \; \forall n_1, n_2 \geq m \; \big(d(x_{n_1}, x_{n_2}) < \varepsilon\big),$$

where m, n_1 and n_2 range over $\mathbb{N}$. A metric space (X, d) (or its metric d) is *complete* if every Cauchy sequence converges (the converse is always true in a metric space).

Definition 1.4. Let (X, τ) be a topological space. The topology τ is *completely metrizable* if there exists a complete metric d on X such that $\tau_d = \tau$.

Now we have all the notions we need to define Polish spaces.

Definition 1.5. Let (X, τ) be a topological space. We say that X is a *Polish space* (or τ is a *Polish topology*) if τ is completely metrizable and separable.

Example 1.6. The set of real numbers $\mathbb{R}$ with the *Euclidean topology* τ_{Ecl}, i.e. the topology generated by the basis

$$\mathcal{F} = \{(a, b) \mid a, b \in \mathbb{R} \; \wedge a \leq b\}$$

is Polish. Indeed, the metric

$$d(x, y) = |x - y|$$

is complete and $\tau_d = \tau_{Ecl}$. Moreover, a countable dense set is $\mathbb{Q}$, the set of rational numbers.

Let us see now other examples of Polish spaces that are relevant to Automata Theory.

Firstly, fix some notation. Given a non empty countable set A, let A^n be the space of the functions of the form

$$s \colon \{0, \ldots, n-1\} \to A.$$

Such a function s can be represented as a word

$$s = (s(0), \ldots, s(n-1)) = s_0 s_1 \ldots s_{n-1}$$

over the alphabet A. If $s = s_0 s_1 \ldots s_{n-1}$ then we say that n is the *length* of s, and we denote it by $lh(s)$. The empty word is denoted by ε. We set $lh(\varepsilon) = 0$ and $A^0 = \{\varepsilon\}$. By $A^{\leq n}$ we denote the set of words over A of length at most n, i.e.

$$A^{\leq n} = A^0 \cup A^1 \cup \ldots \cup A^n.$$

We denote by A^* the set of all the finite words over A:

$$A^* = \bigcup_{n \in \omega} A^n.$$

By A^ω we denote the set of infinite words over the alphabet A. Formally the elements of this space are functions of the form

$$\alpha : \omega \to A,$$

where, as usual, ω is the first infinite ordinal and it is identified with the set of natural numbers. Such a function can be represented as an infinite sequence

$$(\alpha(0), \alpha(1), \alpha(2), \ldots) = \alpha_0 \alpha_1 \alpha_2 \ldots.$$

Finally, we set

$$A^\infty = A^* \cup A^\omega.$$

If $\alpha \in A^\infty$ and $n \in \omega$, we set $\alpha{\restriction}n = (\alpha_0, \ldots, \alpha_{n-1}) \in A^n$ (if α is finite this definition makes sense only if $n \leq lh(\alpha)$). We say that $s \in A^*$ is an *initial segment* or a *prefix* of $\alpha \in A^\infty$ if $s = \alpha{\restriction}n$ for some integer n; in symbols, $s \preceq \alpha$. We write $s \prec \alpha$ if $s \preceq \alpha$ but $s \neq \alpha$. The *concatenation* of $s, t \in A^*$, where $s = s_0 \ldots s_n$ and $t = t_0 \ldots t_m$, is the word

$$s^\frown t = st = s_0 \ldots s_n t_0 \ldots t_m.$$

We can also consider the concatenation $s^\frown\alpha$ of a finite word s and an infinite word α defined in the obvious way:

$$s^\frown\alpha = s_0 s_1 s_2 \ldots s_{lh(s)-1} \alpha_0 \alpha_1 \ldots$$

The set A^ω can be endowed with the topology generated by$\{N_s \mid s \in A^*\}$, where

$$N_s = \{\alpha \in 2^\omega \mid s \prec \alpha\}.$$

It is easy to check that each N_s is also closed, hence clopen. This topology is called the *prefix topology* and we denote it by τ_{pref}.

Definition 1.7. Let (X, τ) be a topological space. X is a *zero-dimensional* topological space if there exists a basis B for τ such that B contains just clopen sets.

Fact 1.8. *The topological space (A^ω, τ_{pref}) is an uncountable zero-dimensional Polish space.*

Proof. Consider the metric obtained by setting, for distinct $\alpha, \beta \in A^\omega$,

$$d(\alpha, \beta) = \frac{1}{2^n},$$

where n is the minimum index such that $\alpha(n) \neq \beta(n)$. The metric d is complete and the topology induced by d is τ_{pref}. Moreover, τ_{pref} is separable. Indeed, fix a symbol $c \in A$: then the set

$$S = \{s^\frown ccccccccccccc \ldots \mid s \in A^*\}$$

is countable and dense. □

The *Baire space* is the space of the form A^ω, where $A = \omega$. It is denoted by ω^ω. The *Cantor space* is the space of the form A^ω, where $A = 2 = \{0, 1\}$. It is denoted by 2^ω.

Descriptive Set Theory is essentially concerned with the study of certain classes of definable subsets of Polish spaces, usually stratified in suitable hierarchies. Some of these topological hierarchies are introduced in the next subsections.

1.1.2 The Borel hierarchy

The *Borel* subsets of X, denoted by $\mathcal{BOR}(X)$, are those belonging to the smallest σ-algebra[1] that contains all the open subsets of X. Borel sets can be stratified in a hierarchy of complexity as follows:

[1] We recall that a *σ-algebra* on a set X is a subset of $\mathcal{P}(X)$ closed under complementation and countable union.

Definition 1.9. Let (X, τ) be a topological space and let ω_1 be the first uncountable ordinal. We define, by a transfinite recursion on $1 \leq \xi < \omega_1$, the following classes:

- $\mathbf{\Sigma}^0_1(X) = \{A \subseteq X \mid A \text{ is open}\}$.
- $\mathbf{\Pi}^0_\xi(X) = \{A \mid A^c \in \mathbf{\Sigma}^0_\xi(X)\}$, where A^c is the complement of A;
- $\mathbf{\Sigma}^0_\xi(X) = \{\bigcup_n A_n \mid A_n \in \mathbf{\Pi}^0_{\xi_n}(X),\ 1 \leq \xi_n < \xi,\ n \in \omega\}$.

Moreover, for $1 \leq \xi \leq \omega_1$ we define

$$\mathbf{\Delta}^0_\xi(X) = \mathbf{\Sigma}^0_\xi(X) \cap \mathbf{\Pi}^0_\xi(X).$$

It can easily be shown that

$$\mathcal{BOR}(X) = \bigcup_{\xi \in \omega_1} \mathbf{\Sigma}^0_\xi(X) = \bigcup_{\xi \in \omega_1} \mathbf{\Pi}^0_\xi(X) = \bigcup_{\xi \in \omega_1} \mathbf{\Delta}^0_\xi(X).$$

If α is the minimum index such that a set A belongs to $\mathbf{\Sigma}^0_\alpha \cup \mathbf{\Pi}^0_\alpha$ then α is called the *Borel rank* of A. When the space X is clear from the context, we write $\mathbf{\Sigma}^0_\xi$, $\mathbf{\Pi}^0_\xi$ and $\mathbf{\Delta}^0_\xi$ instead of $\mathbf{\Sigma}^0_\xi(X)$, $\mathbf{\Pi}^0_\xi(X)$ and $\mathbf{\Delta}^0_\xi(X)$, respectively.

This definition does not guarantee that the classes $\mathbf{\Sigma}^0_\xi$ and $\mathbf{\Pi}^0_\xi$ are increasing and they form a hierarchy of length ω_1. E.g. in a discrete space $(X, \mathcal{P}(X))$ the hierarchy collapses at the first level. If X is Polish and it is countable then the hierarchy collapses at the second level. But any uncountable Polish space yields a non-collapsing hierarchy.

Fact 1.10. *Let (X, τ) be any uncountable Polish space. Then the Borel hierarchy over X does not collapse up to ω_1: if $\alpha < \beta < \omega_1$ then*

$$\mathbf{\Sigma}^0_\alpha, \mathbf{\Pi}^0_\alpha \subset \mathbf{\Delta}^0_\beta.$$

In particular the Borel hierarchy of the Cantor space 2^ω is a proper hierarchy. This thesis we will focus on its first two levels.

1.1.3 The Projective hierarchy

Beyond Borel sets we find the hierarchy of *projective sets*. The class of projective sets of a topological space X, denoted by $\mathcal{PROJ}(X)$, stratifies itself in a hierarchy of length ω, starting from the *analytic sets*.

Definition 1.11. A subset of X is *analytic* if it is the projection on X of a Borel subset of the space $X \times Y$, where Y is Polish and $X \times Y$ is endowed with the product topology.

Now that we have the first level, we can give the definition of the whole hierarchy.

Definition 1.12. Let X be a topological space. By induction on $n \in \omega$ we define:

1. $\mathbf{\Sigma}^1_1(X) = \{A \subseteq X \mid A \text{ is analytic}\}$.
2. $\mathbf{\Pi}^1_n(X) = \{A \subseteq X \mid A^c \in \mathbf{\Sigma}^1_n(X)\}$.
3. $\mathbf{\Sigma}^1_{n+1}(X) = \{A \subseteq X \mid$ there is a Polish space Y and there exists a set $B \in \mathbf{\Pi}^1_n(X \times Y)$ such that A is the projection on X of $B\}$.
4. $\mathbf{\Delta}^1_n(X) = \mathbf{\Pi}^1_n(X) \cap \mathbf{\Sigma}^1_n(X)$.

Finally:

$$\mathcal{PROJ}(X) = \bigcup_{n\in\omega} \mathbf{\Sigma}^1_n(X) = \bigcup_{n\in\omega} \mathbf{\Pi}^1_n(X) = \bigcup_{n\in\omega} \mathbf{\Delta}^1_n(X).$$

Also in this case, we can suppress from the notation the reference to X when there is no danger of confusion. Sets in $\mathbf{\Pi}^1_1$ are called *co-analytic*.

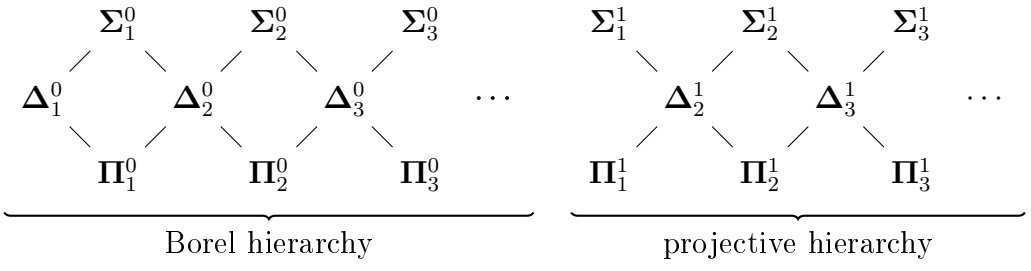


Figure 1.1: The Borel and projective hierarchies.

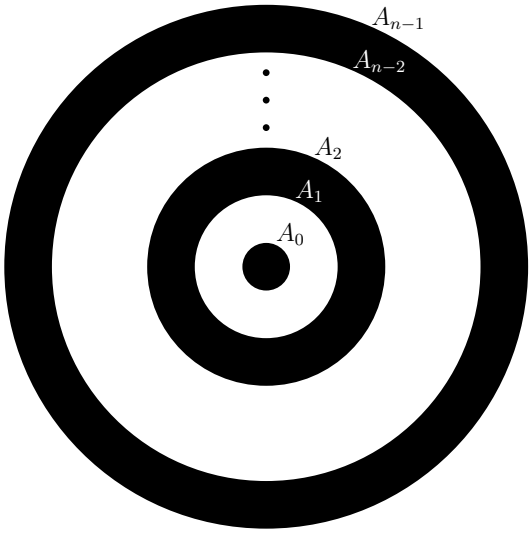


Figure 1.2: The set $A = A_0 \cup (A_2 \setminus A_1) \cup \ldots \cup (A_{n-1} \setminus A_{n-1})$ (A is the union of the black rings) with $A_0 \subseteq A_1 \subseteq A_2 \subseteq \ldots \subseteq A_{n-2} \subseteq A_{n-1}$ and $A_i \in \mathbf{\Sigma}^0_\xi$ for every i.

1.1.4 The difference hierarchy

The Borel hierarchy is refined by the so-called *difference hierarchy*.

Every ordinal θ can be uniquely written as $\lambda+n$, where λ is a limit ordinal and $n \in \omega$. So we call θ *even* (resp. *odd*) if $\theta = \lambda + n$ and n is even (resp. odd).

Definition 1.13. Let X be a topological space and let $A \subseteq X$. Let θ be any non null ordinal. We say that A is a θ-*difference of* $\mathbf{\Sigma}^0_m$ sets, and we denote it by $A \in \mathcal{D}_\theta\big(\mathbf{\Sigma}^0_m\big)$, if there is a θ-sequence of sets $A_0, A_1, \ldots, A_\eta, \cdots \in \mathbf{\Sigma}^0_m$, with $\eta < \theta$, such that:

$$x \in A \iff \text{the minimum } \eta < \theta \text{ such that } x \in A_\eta, \text{ has parity opposite to that of } \theta.$$

In particular, for any integer n, $A \in \mathcal{D}_n\big(\mathbf{\Sigma}^0_m\big)$ if and only if

$$A = A_0 \cup (A_2 \setminus A_1) \cup \ldots \cup (A_{n-1} \setminus A_{n-2})$$

if n is odd, whereas

$$A = (A_1 \setminus A_0) \cup \ldots \cup (A_{n-1} \setminus A_{n-2})$$

if n is even, with $A_0, \ldots, A_{n-1}$ belonging to $\mathbf{\Sigma}^0_m$.

Remark 1.14. *Without loss of generality, in Definition 1.13 we can suppose that $A_i \subseteq A_j$ if $i < j$. Indeed, it is enough to replace A_i with $\bigcup_{j \leq i} A_j$.*

Therefore, a set in $\mathcal{D}_n\big(\mathbf{\Sigma}^0_\xi\big)$, with n odd, looks as depicted in Figure 1.2.

Theorem 1.15 (Hausdorff, Kuratowski, see [Kec95, Theorem 22.27, page 176])**.** *In Polish spaces and for any $1 \leq \xi < \omega_1$ we have that*

$$\mathbf{\Delta}^0_{\xi+1} = \bigcup_{1 \leq \theta < \omega_1} \mathcal{D}_\theta\big(\mathbf{\Sigma}^0_\xi\big).$$

Hence the difference hierarchy is a good refinement of the Borel hierarchy: between two levels of the Borel hierarchy there are ω_1-many different new levels of the difference hierarchy.

1.2 Determinacy

Let X be a non empty set and let $A \subseteq X^\omega$. We associate with A the infinite game $G(X, A)$, called the *Gale-Stewart game on X and A*, defined in the following way:

I	a_0		a_2	$\ldots$
II		a_1		a_3 $\ldots$

In this game Player I chooses an element $a_0 \in X$, then Player II chooses $a_1 \in X$, and so on for infinitely many turns. Player I wins if and only if the infinite word $(a_0, a_1, \ldots)$ belongs to A. A is called the *payoff set* of the game $G(X, A)$.

Definition 1.16. A *strategy* for Player I is a map $\sigma\colon X^* \to X$. Player I *follows* the strategy σ in the game $G(X, A)$ if he plays $a_0 = \sigma(\varepsilon)$ at the first move, $a_2 = \sigma(a_0, a_1)$ at the second move (where a_1 is the answer of Player II), and so on. The strategy σ is *winning* for I in $G(X, A)$ if the infinite word $(a_0, a_1, \ldots)$ belongs to A whenever Player I follows σ. All these definitions are given for Player II in the dual way.

We say that a game $G(X, A)$, or just the set A, is *determined* if one of the two players has a winning strategy.

Theorem 1.17 (Gale-Stewart Theorem, see [Kec95, Theorem 20.1, page 138])**.** *Let A be a closed or open subset of X^ω (endowed with the prefix topology). Then the game $G(X, A)$ is determined.*

Remark 1.18. *Gale-Stewart Theorem is provable in* ZFC *but not in* ZF*. Indeed, it turns out (see [Kec95, Exercise 20.3, page 139]) that in* ZF *Gale-Stewart Theorem is equivalent to the Axiom of Choice* AC*. However, this fact does not create any problem to us because in this thesis we work in* ZFC*, so any use of the Axiom of Choice will be allowed.*

Gale-Stewart Theorem was extended by Martin to every Borel set.

Theorem 1.19 (Martin Theorem, see [Kec95, Theorem 20.5, page 140])**.** *Let A be a Borel subset of X^ω. Then the game $G(X, A)$ is determined.*

Remark 1.20. *Determinacy of Borel sets is the maximum amount of determinacy provable inside* ZFC*. Determinacy of classes outside the Borel hierarchy is equivalent (in* ZFC*) to some large cardinal hypothesis (see [Jec02]).*

1.3 The Wadge hierarchy of the Cantor space

We already saw that the difference hierarchy refines the Borel hierarchy. Now we further refine the Borel hierarchy by the use of simple tools such as continuous functions.

Definition 1.21. Let X, Y be two topological spaces. We say that a set $A \subseteq X$ *continuously reduces to* $B \subseteq Y$ if there is a continuous function $f\colon X \to Y$ such that $f^{-1}(B) = \{x \in X \mid f(x) \in B\} = A$ (i.e. $x \in A \Leftrightarrow f(x) \in B$ for every $x \in X$).

Definition 1.22. Let Γ be a topological class (e.g. $\Gamma = \mathbf{\Sigma}^0_\alpha$) closed under continuous pre-images (i.e. if B is a subset of a topological space Y such that $B \in \Gamma(Y)$ and $f\colon X \to Y$ is a continuous function from a topological space X to Y, then $f^{-1}(B) \in \Gamma(X)$). Then Γ is called a *boldface pointclass*.

Proposition 1.23. *The levels of the Borel hierarchy and of the difference hierarchy are boldface pointclasses.*

Now we present the tools required to build the Wadge hierarchy of a topological space and then we will focus on the specific case of the Wadge hierarchy of the Cantor space 2^ω.

Definition 1.24. Let X and Y be two topological spaces and let $A \subseteq X$ and $B \subseteq Y$. We say that A is *Wadge reducible to* B, and we denote it by $A \leq_{\mathrm{W}} B$, if there exists a continuous reduction of A to B. We say that A is *Wadge equivalent to* B, in symbols $A \equiv_{\mathrm{W}} B$, if $A \leq_{\mathrm{W}} B$ and $B \leq_{\mathrm{W}} A$. Finally, we write $A <_{\mathrm{W}} B$ if $A \leq_{\mathrm{W}} B$ but $B \leq_{\mathrm{W}} A$ does not hold.

Definition 1.25. Let Γ be a class of sets in Polish spaces. If Y is a Polish space, we call $A \subseteq Y$ Γ-*hard* if $B \leq_{\mathrm{W}} A$ for any $B \in \Gamma(X)$, where X is a zero-dimensional Polish space. Moreover, if $A \in \Gamma(X)$, we call A Γ-*complete*.

Remark 1.26. *Notice that if A is Γ-hard and $A \leq_{\mathrm{W}} B$ then B is Γ-hard as well.*

Fact 1.27. $\equiv_{\mathrm{W}}$ *is an equivalence relation.*

If we fix a space X and we restrict the ordering induced by $\leq_{\mathrm{W}}$ to the sets of X, we obtain the *Wadge hierarchy*[2] of X. If A is a subset of X, then by $[A]_{\mathrm{W}}$ we denote its Wadge degree:

$$[A]_{\mathrm{W}} = \{B \subseteq X \mid B \equiv_{\mathrm{W}} A\}.$$

Even tough the Wadge hierarchy can be defined for any topological space, its shape for a generic space can be very hard to describe (for example the Wadge hierarchy of many non zero-dimensional topological spaces, including the space of real numbers $\mathbb{R}$, is very complicated: see for instance [RS14] and [RSS15]). We focus on the Cantor space 2^ω and we will see that in this space the Wadge hierarchy has a quite simple shape inside the Borel hierarchy (while outside it the structure of the hierarchy depends on the axioms that we assume). We refer the reader to [AC13] for a description of the structure of the general Wadge hierarchy for 2^ω and the missing proofs omitted here.

Remark 1.28. Here we consider the Cantor space 2^ω and restrict the ordering induced by $\leq_{\mathrm{W}}$ to *Borel* sets of 2^ω. One reason is that what happens beyond the Borel hierarchy is not completely clear and on the other hand not connected to the purpose of this thesis (since we work with very low levels of the hierarchy). Moreover, dealing with Borel sets does not create any

[2]The term "hierarchy" is motivated by the fact that the first space where the relation $\equiv_{\mathrm{W}}$ has been studied was the Baire space ω^ω and on it we have a smooth structure that is useful to classify the topological complexity of sets. In other spaces where the partial ordering is very complicated this term is a bit of an abuse of terminology.

problem relative to the axiomatic framework where we work: indeed, here we will mention theorems relative to particular classes of sets whose proofs require that determinacy holds for those classes. This means that working in ZFC all the theorems of this section (related to Borel sets) are provable by Martin Theorem without any extra axiom, in particular no large cardinal hypothesis is needed (cf. Remark 1.20). The higher the levels we go to are, the more we need determinacy. To define the general Wadge hierarchy we should make use of the Axiom of Determinacy AD, the axiom stating that every game is determined. But it turns out that AD is not consistent with the Axiom of Choice AC (see chapter 33 of [Jec02]) and since in this thesis we work in ZFC, we cannot use AD.

Theorem 1.29 (Wadge's Lemma, see [Kec95, Theorem 21.14, page 156])**.** *For any $A, B \in \mathcal{BOR}(2^\omega)$ it holds:*

$$A \leq_{\mathrm{W}} B \quad \vee \quad B^c \leq_{\mathrm{W}} A.$$

Theorem 1.30 (Wadge, Martin-Monk[3], see [Kec95, Theorem 21.15, page 158])**.** *The ordering $\leq_{\mathrm{W}}$ among the Borel sets of 2^ω is well-founded.*

Definition 1.31. A set which is Wadge reducible to its complement is said *self-dual*, otherwise it is said *non self-dual*.

We will see that every non trivial clopen set of 2^ω is self-dual. Since the notion of self-duality is invariant under $\equiv_{\mathrm{W}}$, we can speak of self-dual and non self-dual Wadge degrees. If $[A]_{\mathrm{W}}$ is a non self-dual Wadge degree, then we say that the pair $\{[A]_{\mathrm{W}}, [A^c]_{\mathrm{W}}\}$ is a *non self-dual pair*.

Corollary 1.32. *The antichains in the Wadge degrees have length at most 2 and are of the form*

$$\big\{[A]_{\mathrm{W}}, [A^c]_{\mathrm{W}}\big\},$$

with A non self-dual.

Fact 1.33. In the Cantor space, $[2^\omega]_{\mathrm{W}} = \{2^\omega\}$ and $[\emptyset]_{\mathrm{W}} = \{\emptyset\}$ are the two bottoms Wadge degrees and they clearly form a non self-dual pair. Then we have the Wadge degree formed by any clopen set different from 2^ω and $\emptyset$ and this is a self-dual Wadge degree. Indeed, if A is clopen and B is a non trivial

[3]Actually Wadge proved the theorem for the Borel subsets of 2^ω, then the general version, that uses suitable determinacy hypothesis, is due to Martin and Monk.

subset of 2^ω, we can consider the function $f\colon 2^\omega \to 2^\omega$ such that $f(z) = x$ if $z \in A$ and $f(z) = y$ if $z \in A^c$, where x is an element of B and y of B^c. This function is a reduction of A to B. Then the hierarchy continues with an alternation of non self-dual pairs and self-dual Wadge degrees.

Remark 1.34. Notice that technically every Wadge degree does not contain the elements contained in the previous classes of the hierarchy. For example the Wadge degree $[C]_{\mathrm{W}}$, where C is a clopen set different from 2^ω and $\emptyset$, contains all the clopen sets except the whole space 2^ω and the empty set $\emptyset$. But we can define the *Wadge class* of a set as the union of its Wadge degree with all its predecessors in the hierarchy. For example the Wadge class $\mathbf{\Delta}^0_1$ is obtained by taking the union of the Wadge degree $\mathbf{\Delta}^0_1 \setminus \{\emptyset, 2^\omega\}$ with its predecessors $\{\emptyset\}$ and $\{2^\omega\}$. It is clear that the two hierarchies, the one of Wadge degrees and the one of Wadge classes, are isomorphic as orders. In the pictures of this section we show the hierarchy of the Wadge classes because they are more intuitive and easier to describe.

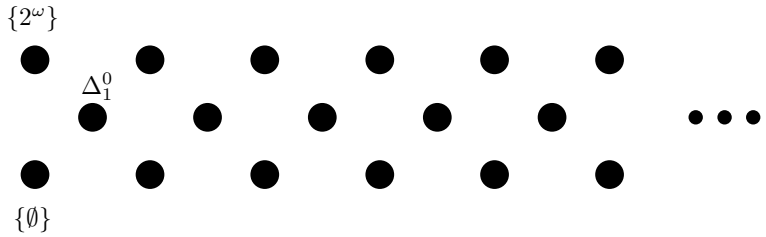


Figure 1.3: the Wadge hierarchy of 2^ω. The black dots represent the Wadge classes.

We can assign an ordinal to any level of the hierarchy. This ordinal is the *Wadge rank* of a Wadge degree (or of the corresponding Wadge class). For technical reasons, we choose to give the same Wadge rank to three Wadge degrees, namely to the elements of a non self-dual pair and of its subsequent self-dual Wadge degree, and we start our enumeration at 1. So, the two bottom Wadge degrees $\{\emptyset\}$ and $\{2^\omega\}$, together with the Wadge degree $\mathbf{\Delta}^0_1 \setminus \{\emptyset, 2^\omega\}$ have Wadge rank 1, the three successive Wadge degree have Wadge rank 2, and so on.[4] Formally, we identify in one single level three Wadge

[4] In literature there are other possible enumerations, for example it is quite common to start the enumeration with 0 instead of 1. The technical reasons for our choice will be clear in the next chapter.

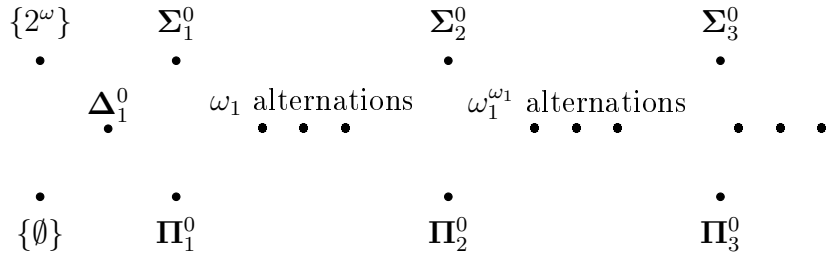


Figure 1.4: An initial fragment of the Wadge hierarchy of 2^ω.

degree (or Wadge classes), i.e. the elements of a non self-dual pair and its subsequent self-dual Wadge degree. At that point we have a well-founded linear order and the enumeration is standard. All the limit levels of the Wadge hierarchy on 2^ω consist of a non self-dual pair.

Among the non self-dual Wadge classes we find the classes $\mathbf{\Sigma}^0_n$ and $\mathbf{\Pi}^0_n$. The classes $\mathbf{\Sigma}^0_1$ and $\mathbf{\Pi}^0_1$ are located immediately after the Wadge class $\mathbf{\Delta}^0_1$, so their Wadge rank is 2. Then, between the non self-dual pair

$$\{\mathbf{\Sigma}^0_n, \mathbf{\Pi}^0_n\}$$

and the successive

$$\{\mathbf{\Sigma}^0_{n+1}, \mathbf{\Pi}^0_{n+1}\}$$

there are $\omega_1^{\omega_1^{\cdot^{\cdot^{\cdot^{\omega_1}}}}}$ (n times) Wadge classes. In particular, there are ω_1 Wadge classes between the pair

$$\{\mathbf{\Sigma}^0_1, \mathbf{\Pi}^0_1\}$$

and

$$\{\mathbf{\Sigma}^0_2, \mathbf{\Pi}^0_2\}.$$

Hence, the Wadge rank of the Wadge classes $\mathbf{\Sigma}^0_2$ and $\mathbf{\Pi}^0_2$ is ω_1, while the Borel class $\mathbf{\Delta}^0_2$ contains ω_1 different levels of the Wadge hierarchy. A more precise presentation of the Wadge hierarchy is depicted in Figure 1.4.

Now we focus on the segment that we will study in this thesis, that is the initial segment from the beginning of the hierarchy until the Wadge classes $\mathbf{\Sigma}^0_2$ and $\mathbf{\Pi}^0_2$. To characterise all the Wadge classes inside this portion of hierarchy we make use of the difference hierarchy.

Fact 1.35. *Every class $\mathcal{D}_\theta(\mathbf{\Sigma}^0_m)$ corresponds to a Wadge class of a non self-dual pair.*

So the Wadge hierarchy is a refinement of the difference hierarchy. We know that between the first and the second level of the Borel hierarchy we find ω_1 different levels of the difference hierarchy, but also ω_1 different Wadge classes and in fact the two hierarchies essentially coincide. Indeed, if we define

$$\check{\mathcal{D}}_n(\mathbf{\Sigma}^0_1) = \{A \subseteq 2^\omega \mid A^c \in \mathcal{D}_n(\mathbf{\Sigma}^0_1)\}$$

we can obtain a complete picture of the first segment of the Wadge hierarchy of the Cantor space 2^ω:

$$\begin{array}{ccccccccc}
\{2^\omega\} & & \Sigma^0_1 & & \mathcal{D}_2(\Sigma^0_1) & & \mathcal{D}_3(\Sigma^0_1) & & \\
& \Delta^0_1 & & \mathcal{D}_2(\Sigma^0_1) \cap \check{\mathcal{D}}_2(\Sigma^0_1) & & \mathcal{D}_3(\Sigma^0_1) \cap \check{\mathcal{D}}_3(\Sigma^0_1) & & \bullet\ \bullet\ \bullet & \\
\{\emptyset\} & & \Pi^0_1 & & \check{\mathcal{D}}_2(\Sigma^0_1) & & \check{\mathcal{D}}_3(\Sigma^0_1) & &
\end{array}$$

Figure 1.5: The first fragment of the Wadge hierarchy.

Hence, in the Cantor space the difference hierarchy is an important tool to understand the Wadge hierarchy, especially in the initial part, while beyond $\mathbf{\Sigma}^0_2$ the difference hierarchy skips a lot of Wadge degrees.

1.4 Automata Theory

In this section we define automata for infinite words. Among the possible acceptance conditions we choose the *parity acceptance condition*. For other formalisms and missing proofs the reader can consult, for instance, [Eil74, Skr16, Skr16, PP04, GTW02].

Let A be a finite *alphabet*, i.e. a finite set of symbols, and consider the set A^ω of infinite words over A.

Definition 1.36. A subset $L \subseteq A^\omega$ is called a *word language*.[5]

[5] Actually we should distinguish between *word languages* and *ω-word languages* (as well as *tree languages* and *ω-tree languages*): the first ones are languages of finite words, the second ones of infinite words. Since in this thesis we talk just about infinite words (and infinite trees), we will avoid the prefix ω-.

To render the presentation more intuitive, we can fix once and for all the alphabet. So, by sake of simplicity, if it is not explicitly specified, A is always supposed to be $A = \{a, b\}$.

Definition 1.37. A *(word) parity non-deterministic automaton* is a tuple

$$\mathcal{A} = \langle A, Q^{\mathcal{A}}, q_{\mathrm{I}}^{\mathcal{A}}, \Delta^{\mathcal{A}}, \Omega^{\mathcal{A}} \rangle,$$

where A is the alphabet, $Q^{\mathcal{A}}$ is a finite set consisting of the *states* of the automaton (the *state set*), $q_{\mathrm{I}}^{\mathcal{A}} \in Q^{\mathcal{A}}$ and is called the *initial state* of the automaton, $\Delta^{\mathcal{A}} \subseteq Q^{\mathcal{A}} \times A \times Q^{\mathcal{A}}$ and it is called the *transition relation* and finally $\Omega^{\mathcal{A}}$ is a function

$$\Omega^{\mathcal{A}} \colon Q^{\mathcal{A}} \to \omega$$

that assigns a natural number to every state: for $q \in Q^{\mathcal{A}}$ the number $\Omega^{\mathcal{A}}(q)$ is called the *priority* of q.

By sake of readability, unless it creates some ambiguity we will always drop the superscript $\mathcal{A}$.

Definition 1.38. A *run* of $\mathcal{A}$ over a word $\sigma \in A^\omega$ is a word $\rho \in Q^\omega$ such that:

- $\rho(0) = q_{\mathrm{I}}$.
- For any $i \in \omega$ every transition $\big(\rho(i), \sigma(i), \rho(i+1)\big)$ belongs to Δ.

A run is *successful* or *accepting* if the *parity condition* holds: the highest priority met infinitely many times in the states of ρ is even, i.e.

$$\limsup_{n \to \infty} \Omega(\rho(n)) \text{ is even.}$$

Definition 1.39. A word $\sigma \in A^\omega$ is *accepted* by $\mathcal{A}$ if there exists a successful run of $\mathcal{A}$ over σ. The *language recognised by* $\mathcal{A}$ is the set of all the words accepted by $\mathcal{A}$:

$$\mathrm{L}(\mathcal{A}) = \{\sigma \in A^\omega \mid \text{ there exists a run } \rho \text{ of } \mathcal{A} \text{ over } \sigma \text{ that is successful}\}.$$

Definition 1.40. Let $L \subseteq A^\omega$ be a word language. L is *regular* if there exists a word parity non-deterministic automaton $\mathcal{A}$ such that $\mathrm{L}(\mathcal{A}) = L$.

Example 1.41. The following languages are regular:

1. $L = \{\sigma \in A^\omega \mid \sigma \text{ has finitely many } a\}$.
2. $L = \{\sigma \in A^\omega \mid \sigma \text{ has exactly one } a\}$.
3. $L = \{\sigma \in A^\omega \mid \sigma \text{ at some point has three } a \text{ in a row}\}$.

Definition 1.42. A *(word) parity deterministic automaton* $\mathcal{A}$ is a word parity non-deterministic automaton

$$\mathcal{A} = \langle A, Q, q_{\mathrm{I}}, \Delta, \Omega \rangle,$$

where the relation Δ is functional, i.e. if (q, a, q') is an element of Δ, then Δ does not contain any other transition of the form (q, a, q''). In this case we write δ instead of Δ and we treat δ as a function $\delta : Q \times A \to Q$.

If $\mathcal{A}$ is a parity deterministic automaton, for any word $\sigma \in A^\omega$ we have only one run of $\mathcal{A}$ over σ, that is the word $\rho \in Q^\omega$ such that $\rho(0) = q_{\mathrm{I}}$ and if $\rho(i) = q$ and $\sigma(i) = c$, then $\rho(i+1) = \delta(q, c)$. As before, a run is *successful* if the parity condition holds and the *language recognised by* $\mathcal{A}$, (still denoted by $\mathrm{L}(A)$), is the set of all the words σ for which the unique run of $\mathcal{A}$ over σ is successful.

Definition 1.43. Two automata $\mathcal{A}$ and $\mathcal{B}$ are *equivalent* if $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})$. The definition can be extended to classes of automata: two classes are equivalent if the collection of languages that they recognise is the same.

Theorem 1.44. *Let $L \subseteq A^\omega$. The following conditions are equivalent:*

1. *L is recognised by a word parity non-deterministic automaton.*
2. *L is recognised by a word parity deterministic automaton.*

Moreover, given a parity non-deterministic automaton $\mathcal{A}$, we can effectively construct a parity deterministic automaton $\mathcal{B}$ that is equivalent to $\mathcal{A}$.

So non-determinism does not add anything to the expressive power of word automata. We will see that the situation is completely different for trees.

Regular word languages are closed under Boolean operations.

Fact 1.45. *Let L_1 and L_2 be two regular word languages. Then the following properties hold:*

1. *L_1^c is regular, where $L_1^c = A^\omega \setminus L$ we denote the complement of L_1.*
2. *The union $L_1 \cup L_2$ and the intersection $L_1 \cap L_2$ are regular.*

Moreover, the construction of automata recognising these languages is effective.

1.5 Monadic Second Order Logic

Monadic Second Order Logic, denoted by the acronym MSO, is one of the first formalism to be studied among the ones capturing regular languages. This logic was born in the 60's and among the first questions related to it there were the ones about decidability of MSO and its fragments. These questions represented a strong motivation to introduce automata: being equivalent to MSO or to certain fragments of it, they allowed to easily solve some problems related to decidability.

The presentation we give now follows [GTW02]. We take for granted the main notions coming from Model Theory. A standard textbook containing such basic notions is for example [Kun08] (in the first chapter).

Monadic Second Order Logic is a fragment of Second Order Logic. We have two kinds of quantification: first order quantification over elements and second order quantification over sets of elements. In general in Second Order Logic we have that the second order quantification ranges over n-ary relations of elements, but in MSO we allow only unary relations, i.e. sets of elements. To distinguish between the two quantifications we use lower-case letters $x, y, z, \ldots$ for first order variables and upper-case letters $P, Q, R, S, \ldots$ for second order variables.

Example 1.46. Fix the language $\mathcal{L} = \{E\}$ where E is a binary relation. Consider the formula

$$\forall P\big(\forall x \forall y((P(x) \wedge E(x,y)) \rightarrow P(y)).$$

The above formula says that P (that is a set of elements) is closed under E. Notice that E is a binary relation, but it is not under the scope of any quantifier but it has to be interpreted by a structure for the language $\mathcal{L}$. So the formula in this example is a formula of MSO.

In general a language is a set that contains three different kinds of objects: constants, functions and relations. When we work with the fragment of MSO for words all the languages are *relational*, i.e. they contain just first order relations. In particular, fixed an alphabet A, we consider languages of the form:

$$\mathcal{L} = \{S_0\} \cup \{P_c \mid c \in A\}$$

consisting of a unary relational symbol for any letter of the alphabet A and an extra binary relational symbol S_0. Such a language is called a *word vocabulary*.

Now we present a way to use an infinite word σ as a structure for a word vocabulary. The domain of the structure is the set of natural numbers $\mathbb{N}$. S_0 is always interpreted as the successor relation, i.e.

$$S_0 = \{(m, m+1) \mid m \in \mathbb{N}\}.$$

Finally, a relation P_c is interpreted as the subset of $\mathbb{N}$ that contains all the natural numbers n such that $\sigma(n) = c$:

$$P_c = \{n \in \mathbb{N} \mid \sigma(n) = c\}.$$

In this way, we have an interpretation for any symbol of the alphabet. Notice that the infinite word σ actually interprets only the relations P_c, while the interpretation of S_0 is always the same. Hence, we identify a word σ with such structure we have defined. If φ is a sentence (i.e. a formula without free variables), we write $\sigma \vDash \varphi$ to say that the structure induced by σ satisfies φ.

Definition 1.47. The fragment of MSO where all the languages are word vocabularies and the structures for these languages are infinite words is denoted by $S1S$.

Definition 1.48. Let φ be a sentence (i.e. a formula without free variables) of $S1S$. The language *captured* by φ is the set of all the words σ that satisfy φ:

$$\mathrm{L}(\varphi) = \{\sigma \in A^\omega \mid \sigma \vDash \varphi\}.$$

Theorem 1.49 (see [GTW02], pages 214-220.)**.** *Let L be a word language. Then the following are equivalent:*

1. *There exists a word parity non-deterministic automaton $\mathcal{A}$ such that $\mathrm{L}(\mathcal{A}) = L$ (i.e. L is regular).*

2. *There exists a sentence φ in $S1S$ such that* $\mathrm{L}(\varphi) = L$.

Moreover, given an automaton $\mathcal{A}$ we can effectively construct a formula φ such that $\mathrm{L}(\varphi) = \mathrm{L}(\mathcal{A})$, *and vice-versa.*

Example 1.50. Consider the language

$$\mathcal{L} = \{S_0, P_a, P_b\}$$

and the formula φ

$$0 \in P_a \wedge S_0(0) \in P_b \wedge S_0(S_0(0)) \in P_a.$$

Then φ defines the clopen set

$$\mathrm{L}(\varphi) = N_s = \{\sigma \in A^\omega \mid s \prec \alpha\},$$

where $s = aba$.

Theorem 1.49 was proved by Büchi in 1962 and it was a milestone in the context of interplay between Automata Theory and Logic. The original work by Büchi is [Büc62].

1.6 The concept of regularity

In the previous sections we gave a formal definition of *regular* language by making use of MSO and parity automata. But why $S1S$ is the "right" logic to define the concept of *regular* language? The crucial point is that we need second order quantification to have a logic that expresses the basic properties of words, therefore we really need to go to Second Order Logic and once we have second order quantification we take a suitably weak logic that uses it.

Moreover, it turns out that the languages captured by parity automata and MSO are not related to the choice of this particular computational model, but many formalisms capture exactly the same class of languages. Among the first classes of automata to be originally introduced we find *Büchi* automata: in this case the state set is split into *final* states and not final states and a successful run visits final states infinitely many times. In *Muller* and *Rabin* automata the set of states visited infinitely many times during a successful run has to belong to a given collection of set of states. All these classes of

automata are equivalent for words. In the next theorem we list several other equivalent formalisms that capture regular languages. We can not present all the definitions that appear in the theorem (all the notions, however, can be found for example in [Eil74, Skr16, Skr16, PP04, GTW02, PP04, AN01, Rab72]), but we want to stress that regular languages form a very strong class. This class is *the* right class: it is captured by many formalisms and it is also stable under a lot of operations.

Theorem 1.51. *Let $L \subseteq A^\omega$ be a word language. Then the following sentences are equivalent:*

1. *L is recognisable by a word parity non-deterministic automaton.*
2. *L is recognisable by a word parity deterministic automaton.*
3. *L is definable in Monadic Second Order Logic for words.*
4. *L is definable in Weak Monadic Second Order Logic for words.*
5. *L is defined by an ω-regular expression.*
6. *L is recognisable by a word non-deterministic Büchi autamaton.*
7. *L is recognisable by a word non-deterministic Rabin automaton.*
8. *L is recognisable by a word deterministic Rabin automaton.*
9. *L is recognisable by a word non-deterministic Muller automaton.*
10. *L is recognisable by a word deterministic Muller automaton.*
11. *L is recognisable by a word parity alternating automaton.*
12. *L is recognisable by a word weak-alternating automaton.*
13. *L is definable in μ-calculus.*
14. *L is recognisable by a word guidable automaton.*

1.7 Descriptive Set Theory and Automata for words

Now we can see the connections between Descriptive Set Theory and Automata Theory in the context of word languages.

The theorem that, to the best of our knowledge, constitutes the first historical connection between Descriptive Set Theory and Automata Theory was proved by Landweber in 1969 ([Lan69]).

Theorem 1.52 (Landweber, 1969)**.** *Let $L \subseteq A^\omega$. If L is regular, then L is a $\mathbf{\Delta}^0_3$ subset of the Polish space A^ω.*

Hence, regular word languages occupy very low levels of the Borel hierarchy of the space A^ω. But we can go deeper and we can see what the connections between regularity and the Wadge hierarchy are.

Remark 1.53. A natural question that one could ask is wheter the Wadge hierarchy is a suitable hierarchy for word languages: the Wadge hierarchy is a topological hierarchy, so it could fail to reflect the structure of regular word languages. Surprisingly this is not the case. Topological hierarchies interfered with formal languages with the Klaus Wagner's work [Wag79] (for a more moder presentation see [CP94] and [CP97]). In this work Wagner defined a natural hierarchy based on the concepts of *chain* and *superchain* of regular word languages: it turned out that this hierarchy is exactly the Wadge hierarchy, and this incredible coincidence encouraged further investigation for other spaces, such as the space of trees.

Fact 1.54. *Let A and B be two finite sets. Then A^ω is homemorphic to B^ω.*

Proof. A^ω and B^ω are perfect, compact, metrizable and zero-dimensional, so they are homemorphic (see [Kec95], page 35). □

Corollary 1.55. *The Cantor space 2^ω and A^ω - the space of infinite words over a finite alphabet A - are homeomorphic. Hence, the Wadge hierarchy of A^ω is the same as the one of 2^ω.*

Theorem 1.56. *The following properties hold:*

1. *The Wadge rank of the Wadge degrees of any regular word language has the form*
$$\omega_1^k n_k + \ldots + \omega_1 n_1 + n_0,$$
where $n_0, \ldots, n_k, k \in \omega$.

2. *Languages recognised by word parity deterministic automata that use* k *priorities correspond to Wadge degrees with Wadge rank less than* ω_1^k.

3. *Given a regular language* L*, there is an effective algorithm that establishes its precise position in the Wadge hierarchy.*

It is interesting to observe that the priorities used by parity deterministic automata, that *a priori* are not connected to topology at all, have a very strong connection with the Wadge hierarchy. The last theorem sums up the main results of some works (for example [Wag79, Wil93, CPP07]) that completed the research for words. We will see in the next chapter that for trees the state of the art is much more lacking.

Chapter 2

Tree languages

In this chapter we generalise to infinite trees the topological notions and the automata seen for words, and we present the state of the art for this new setup.

2.1 Trees

As we will formally see below, in this thesis we will talk about *labelled* trees, i.e. trees where every node has a label from a finite alphabet A. By sake of readability, we do not use the adjective *labelled* and we will talk just about *trees*. In this thesis, by sake of simplicity, we focus on trees with binary branching, where the two *directions* are *left* $\mathtt{L}$ and *right* $\mathtt{R}$. The theory would be the same if we had more (finitely) directions. A *partial tree* over an alphabet A is a partial function

$$t\colon \mathrm{dom}(t) \to A$$

with a non empty prefix-closed domain $\mathrm{dom}(t) \subseteq \{\mathtt{L},\mathtt{R}\}^*$ (i.e. if $s \in t$ and $s' \prec s$ then $s' \in t$). The elements of $\mathrm{dom}(t)$ are called *nodes*. A node $u \in \mathrm{dom}(t)$ is an *internal node* if both $u^\frown\mathtt{L}$ and $u^\frown\mathtt{R}$ belong to $\mathrm{dom}(t)$; it is an *unary node* if exactly one of $u^\frown\mathtt{L}$ and $u^\frown\mathtt{R}$ belongs to $\mathrm{dom}(t)$, and it is a *leaf* if none of $u^\frown\mathtt{L}$ and $u^\frown\mathtt{R}$ belongs to $\mathrm{dom}(t)$. For the sake of readability, we write $u \in t$ to denote that $u \in \mathrm{dom}(t)$ is a node of t. The empty sequence ε belongs to every partial tree and is called *root* of the tree. A *branch* of a partial tree t is a word π (over the alphabet $\{\mathtt{L},\mathtt{R}\}$) such that $\pi{\restriction}n \in t$ for every $n \leq lh(\pi)$ if π is finite (resp. for any $n \in \omega$ if π is infinite). An infinite

branch of a partial tree t is called a *path* of t. A node u is on a branch (finite or infinite) π if it is a prefix of π, i.e. if $u \preceq \pi$. If p and t are partial trees, we say that p is a *prefix* of t, and we denote it by $p \subseteq t$, if $\mathrm{dom}(p) \subseteq \mathrm{dom}(t)$ and $p(u) = t(u)$ for every $u \in \mathrm{dom}(p)$. We write $p \subset t$ if $p \subseteq t$ but $p \neq t$. Finally, if t is a partial tree and u is a node of t, by $t.u$ we indicate the partial tree t *truncated* in u:

$$t.u(w) = t(uw)$$

for any w such that $uw \in \mathrm{dom}(t)$.

A *tree* over A is a partial tree t with $\mathrm{dom}(t) = \{\mathtt{L}, \mathtt{R}\}^*$ (i.e. a *complete* tree). The set of all trees over an alphabet A is denoted Tr_A:

$$\mathrm{Tr}_A \stackrel{\mathrm{def}}{=} \{t \mid t\colon \{\mathtt{L}, \mathtt{R}\}^* \to A\}.$$

A subset $L \subseteq \mathrm{Tr}_A$ is a *tree language*.

2.2 Topology on trees

The prefix topology seen for words can be easily generalised to trees: in this case we endow Tr_A with the topology whose basis is the set

$$B = \{N_p \mid p \text{ is a finite partial tree over } A\}$$

where N_p is the set of trees of which p is a prefix:

$$N_p = \{t \in \mathrm{Tr}_A \mid p \subset t\}.$$

Even in this case, the sets N_p are clopen. Notice that restricting the attention to sets N_p with $\mathrm{dom}(p) = \{\mathtt{L}, \mathtt{R}\}^{\leq n}$ still gives a basis for the same topology. We call again this topology the *prefix topology* and we denote it by τ_{pref}. The topology τ_{pref} is metrizable, for instance by the metric

$$d_{\mathrm{pref}}(t_1, t_2) = \frac{1}{2^n}$$

where n is the minimum length of a node u such that $t_1(u) \neq t_2(u)$. The open balls of d_{pref} are exactly the basic clopen sets N_p, with p such that $\mathrm{dom}(p) = \{\mathtt{L}, \mathtt{R}\}^{\leq n}$.

Yet, in this thesis we will also consider a different metric, denoted by λ and called *discounted distance*, that also generates τ_{pref} but has less intuitive family of open balls. We will understand the importance of this metric in the fourth chapter, when we talk about optimal strategy trees (see Definition 4.55). To define λ first we have to define the function dist: fix some enumeration $u_1, u_2, \dots$ of all the nodes in $\{\mathtt{L}, \mathtt{R}\}^*$, given two trees t_1 and t_2, for any node u we define $\mathsf{dist}(t_1(u), t_2(u)) = 0$ if $t_1(u) = t_2(u)$, 1 in the other case. At this point we define

$$\lambda(t_1, t_2) = \sum_{n \geq 1} \frac{1}{2^n} \mathsf{dist}\big(t_1(u_n), t_2(u_n)\big).$$

Fact 2.1. *Regardless of the enumeration $u_1, u_2, \dots$, the prefix and discounted distances yield the same topology.*

Proof. It is enough to observe that τ_{pref} is exactly the product topology obtained by starting from the discrete topology and the discounted distance is exactly the product metric. □

Fact 2.2. *The Cantor space 2^ω and the space Tr_A are homeomorphic, so they have the same topological hierarchies (Borel, Wadge, etc).*

Proof. Arguing as in Fact 1.54, the result follows from the fact that the topological space Tr_A is perfect, compact, metrizable and zero-dimensional. An explicit homeomorphism can be given as follows.

We identify the sets $\{\mathtt{L}, \mathtt{R}\}^*$ and ω with any bijection h (e.g. set $h(\varepsilon) = 0$, $h(\mathtt{L}) = 1$, $h(\mathtt{R}) = 2$, $h(\mathtt{LL}) = 3$, and so on in the obvious way). Now we define the function

$$f \colon 2^\omega \to \mathrm{Tr}_A$$

that maps a word $\sigma \in 2^\omega$ to the tree t such that for any node u $t(u) = \sigma(h(u))$. □

Corollary 2.3. *The space Tr_A with the topology τ_{pref} is a Polish space.*

2.3 Regularity for trees

Now we generalise to trees the notion of regularity seen for words.

Definition 2.4. A *(tree) parity non-deterministic automaton* $\mathcal{A}$ is a tuple

$$\mathcal{A} = \langle A, Q^{\mathcal{A}}, q_{\mathrm{I}}^{\mathcal{A}}, \Delta^{\mathcal{A}}, \Omega^{\mathcal{A}} \rangle,$$

where A is the alphabet, $Q^{\mathcal{A}}$ is the *state set*, $q_{\mathrm{I}}^{\mathcal{A}}$ is the *initial state*, $\Omega^{\mathcal{A}}$ is a function $\Omega^{\mathcal{A}} \colon Q^{\mathcal{A}} \to \omega$ that assigns a *priority* to every state of the automaton and $\Delta^{\mathcal{A}}$ is the *transition relation* which is a subset of

$$Q^{\mathcal{A}} \times A \times Q^{\mathcal{A}} \times Q^{\mathcal{A}}.$$

If the automaton is clear from the context then we omit the superscript $\mathcal{A}$.

Definition 2.5. A *run* of a tree parity non-deterministic automaton over a tree $t \in \mathrm{Tr}_A$ is a map $\rho \colon \{\mathtt{L}, \mathtt{R}\}^* \to Q$ (i.e. it is a Q-labelled tree) such that

1. $\rho(\varepsilon) = q_{\mathrm{I}}$.
2. $\big(\rho(u), t(u), \rho(u\mathtt{L}), \rho(u\mathtt{R})\big) \in \Delta$ for any $u \in \{\mathtt{L}, \mathtt{R}\}^*$.

A run ρ is *successful* or *accepting* if for every path π of t the sequence of states of ρ on π satisfies the *parity condition*, which means that the highest priority repeated infinitely often is even, i.e. for any path $\pi \in \{\mathtt{L}, \mathtt{R}\}^\omega$ of t:

$$\limsup_{u \in \pi} \ \Omega(\rho(u)) \text{ is even.}$$

A tree t is *accepted* by a tree parity non-deterministic automaton $\mathcal{A}$ if there exists a run ρ of $\mathcal{A}$ over t that is successful. The *language recognised* by an automaton A is the set of all trees accepted by $\mathcal{A}$ and it is denoted by $\mathrm{L}(A)$. As for words, two automata (or two classes of automata) are *equivalent* if they recognise the same language (or the same collection of languages).

Definition 2.6. A tree language L is *regular* if there exists a tree parity non-deterministic automaton $\mathcal{A}$ such that $\mathrm{L}(\mathcal{A}) = L$.

Regular tree languages are also closed under Boolean operations.

Fact 2.7. *Let L_1 and L_2 be two regular tree languages. Then L_1^c, $L_1 \cup L_2$ and $L_1 \cap L_2$ are regular. Moreover, the construction of automata for these languages is effective.*

Definition 2.8. Similarly to words, a (tree) parity *deterministic* automaton $\mathcal{A}$ is a parity non-deterministic automaton where the relation $\Delta^{\mathcal{A}}$ is functional. Also in this case we write $\delta^{\mathcal{A}}$ instead of $\Delta^{\mathcal{A}}$ and we treat $\delta^{\mathcal{A}}$ as a function.

So, if $\mathcal{A} = \langle A, Q^{\mathcal{A}}, q_{\mathrm{I}}^{\mathcal{A}}, \delta^{\mathcal{A}}, \Omega^{\mathcal{A}} \rangle$ is a parity deterministic automaton, where

$$\delta^{\mathcal{A}} \colon Q^{\mathcal{A}} \times A \to Q^{\mathcal{A}} \times Q^{\mathcal{A}}$$

and t is a tree, there is just one run of $\mathcal{A}$ over t. A language L is *deterministic* if it is recognised by a parity deterministic automaton. In between the notion of determinism and non-determinism there is the notion of *unambiguity*: a parity non-deterministic automaton $\mathcal{A}$ is *unambiguous* if for any tree $t \in \mathrm{L}(\mathcal{A})$ there is exactly one accepting run of $\mathcal{A}$ over t. A language is *unambiguous* if there is an unambiguous automaton that recognises it. We will see in the next chapter that every clopen set of Tr_A is an unambiguous language (see Fact 3.2).

Notice that the definition of tree parity non-deterministic automata is the most natural generalisation to trees of the definition we gave for words. So, to obtain the notion of regularity for trees we have to generalise all the notions that appear in Theorem 1.51 and we have to take the stronger formalisms that we obtain. Not all the models in Theorem 1.51 remain equivalent each other once we generalise them to trees. For example in the context of trees it turns out that for trees parity deterministic automata are much weaker than parity non-deterministic ones.

Theorem 2.9. *The underlined formalisms have the same expressive power and they capture regular tree languages, whereas the other ones are strictly weaker (i.e. they recognise less tree languages).*

1. *Tree parity non-deterministic automata.*
2. *Tree parity deterministic automata.*
3. MSO *for trees.*
4. WMSO *for trees.*
5. *Tree ω-regular expressions.*
6. *Tree non-deterministic Büchi autamata.*
7. *Tree non-deterministic Rabin automata.*
8. *Tree deterministic Rabin automata.*

9. *Tree non-deterministic Muller automata.*
10. *Tree deterministic Muller automata.*
11. *Tree parity alternating automata.*
12. *Tree weak-alternating automata.*
13. *μ-calculus.*
14. *Tree guidable automata.*

In the next sections we will consider some of the formalisms cited in Theorem 2.9 and we will prove some of these equivalences. For the missing definitions and equivalences the reader can consult for example [Eil74, Skr16, Skr16, PP04, GTW02, AN01, Rab72]).

2.4 Monadic Second Order Logic for trees

In this section we generalise to trees the logic $S1S$.

Consider languages of the form:

$$\mathcal{L} = \{S_{\mathtt{L}}, S_{\mathtt{R}}\} \cup \{P_c \mid c \in A\},$$

where A is our alphabet, $S_{\mathtt{L}}$ and $S_{\mathtt{R}}$ are two binary relational symbols and any P_c is an unary relational symbol. Such a language is called a *tree vocabulary*. A tree $t \in \mathrm{Tr}_A$ can be seen as a structure for a tree vocabulary: the domain of the structure is $\{\mathtt{L}, \mathtt{R}\}^*$ and $S_{\mathtt{L}}$ and $S_{\mathtt{R}}$ are interpreted as follows:

$$S_{\mathtt{L}} = \{(x, y) \mid x \in \{\mathtt{L}, \mathtt{R}\}^* \ \wedge \ y = x^\frown\mathtt{L}\},$$

$$S_{\mathtt{R}} = \{(x, y) \mid x \in \{\mathtt{L}, \mathtt{R}\}^* \ \wedge \ y = x^\frown\mathtt{R}\}.$$

Finally, in t the interpretation of the symbol P_c is:

$$P_c = \{x \in \{\mathtt{L}, \mathtt{R}\}^* \mid t(x) = c\}.$$

Similarly to words, the role of t is to interpret all the unary symbols P_c, while $S_{\mathtt{L}}$ and $S_{\mathtt{R}}$ always have the same interpretation.

Definition 2.10. $S2S$ is the fragment of MSO where all the languages are tree vocabularies and the structures are elements of Tr_A.

If we take a sentence φ of $S2S$, we call *language captured by* φ the set

$$\mathrm{L}(\varphi) \stackrel{\text{def}}{=} \{t \in \mathrm{Tr}_A \mid t \vDash \varphi\}.$$

Theorem 2.11. *The following conditions are equivalent for a language L:*

1. *There exists a tree parity non-deterministic automaton $\mathcal{A}$ such that $\mathrm{L}(\mathcal{A}) = L$.*

2. *There exists a sentence φ in $S2S$ such that*

$$\mathrm{L}(\varphi) = L.$$

Moreover, given an automaton $\mathcal{A}$ we can effectively construct a formula φ such that $\mathrm{L}(\varphi) = \mathrm{L}(\mathcal{A})$, and vice-versa.

The last theorem was proved by Rabin in 1969. The original work is [Rab69] and in which Rabin proved the equivalence between Rabin automata and $S2S$. For a more modern proof, see [GTW02, pages 214-220].

Now we define a weaker fragment of MSO that is important for the theory of regular tree languages.

Definition 2.12. *Weak Monadic Second Order Logic* is the fragment of MSO, denoted by the acronym WMSO, where every second order quantification is restricted to finite sets. So, if $\mathcal{L}$ is a language and $\mathcal{K}$ is a structure for $\mathcal{L}$ whose domain is K, we have:

1. $\mathcal{K} \vDash \exists R\varphi$ if there exists a finite subset $B \subseteq K$ such that $\mathcal{K} \vDash \varphi(B)$.

2. $\mathcal{K} \vDash \forall R\varphi$ if for any finite subset $B \subseteq K$, $\mathcal{K} \vDash \varphi(B)$.

We define $WS1S$ the fragment of WMSO for infinite words and $WS2S$ the fragment of WMSO for infinite trees. The expressive power of $WS1S$ is the same as $S1S$ (as stated in Theorem 1.51), but we will see that $WS2S$ is strictly weaker than $S2S$.

2.5 Parity Alternating Automata

Now we introduce another formalism that captures regular languages based on the concept of *alternation*. The concept of *alternation* applied to logical theories was introduced by Chandra in [CKS81], and in [MS87] Muller and Schupp extended this idea to automata with the definition of *alternating automata*. This class of automata fits topological hierarchies in a proper way: there is a connection between priorities used by some subclasses of alternating automata and Borel rank of the languages recognised by them.

Definition 2.13. A *(tree) parity alternating automaton* $\mathcal{A}$ is a tuple

$$\mathcal{A} = \langle A, Q^{\mathcal{A}}, q_{\mathrm{I}}^{\mathcal{A}}, \delta^{\mathcal{A}}, \Omega^{\mathcal{A}} \rangle,$$

where A is the alphabet, $Q^{\mathcal{A}}$, $q_{\mathrm{I}}^{\mathcal{A}}$ and $\Omega^{\mathcal{A}}$ are defined as usual and $\delta^{\mathcal{A}}$ is a function

$$\delta^{\mathcal{A}} \colon Q^{\mathcal{A}} \times A \to \mathcal{L}(\{\mathtt{L}, \mathtt{R}\} \times Q^{\mathcal{A}}),$$

where $\mathcal{L}(\{\mathtt{L}, \mathtt{R}\} \times Q^{\mathcal{A}})$ denotes the free distributive lattice generated by all the possible pairs $(d, q) \in \{\mathtt{L}, \mathtt{R}\} \times Q^{\mathcal{A}}$. So $\delta^{\mathcal{A}}$ is a function that assigns to each pair (q, a), where $q \in Q^{\mathcal{A}}$ and $a \in A$, a finite positive Boolean combination of pairs (d, q'), with $d \in \{\mathtt{L}, \mathtt{R}\}$ and $q' \in Q^{\mathcal{A}}$.

If the automaton is clear from the context then we omit the superscript $\mathcal{A}$. Moreover, for technical reasons we require our automata to be *complete*, meaning that for every state $q \in Q$ and letter $a \in A$, $\delta^{\mathcal{A}}(q, a) \neq \emptyset$.

Using distributivity, we can always suppose that every formula in $\mathcal{L}(\{\mathtt{L}, \mathtt{R}\} \times Q^{\mathcal{A}})$ is written in disjunctive normal form, i.e. is a formula of the form

$$\bigvee_{i=1}^{n} \Big(\bigwedge_{k=1}^{m_i} L_{i,k} \Big).$$

The acceptance condition defined originally (like in [MS87] and [MS95]) was given in terms of accepting runs. We now give a more modern condition based on *games*.

A tree parity alternating automaton $\mathcal{A}$ induces, for every tree $t \in \mathrm{Tr}_A$, a parity game $G(t)$ called the *acceptance game* of $\mathcal{A}$ on t. The positions of this game are of the form (u, ϕ) where $u \in \{\mathtt{L}, \mathtt{R}\}^*$ is a node of the full binary tree and ϕ is either an element of $\mathcal{L}(\{\mathtt{L}, \mathtt{R}\} \times Q)$ or a state of $\mathcal{A}$. The initial position is $(\varepsilon, q_{\mathrm{I}})$.

- A position of the form $(u, \phi \vee \psi)$ is controlled by Player $\Diamond$, called *Diamond*, and he can choose to move either to (u, ϕ) or (u, ψ).[1]
- Dually, a position of the form $(u, \phi \wedge \psi)$ is controlled by Player $\Box$, called *Box*, and he can choose to move either to (u, ϕ) or (u, ψ).
- Positions of the form $\big(u, (d, q)\big)$ are deterministic, the only available move from them is to the position $\big(ud, q\big)$.
- Positions of the form (u, q) are also deterministic, the only available move from them is to the position $\big(u, \delta^{\mathcal{A}}(q, t(u))\big)$.

We say that a state q is *visited* during a play of $G(t)$ if at some point the two players reach a position of the form (u, q). An infinite play π of $G(t)$ is winning for $\Diamond$ if the sequence of states visited during the play satisfies the parity condition. The automaton $\mathcal{A}$ *accepts* a tree $t \in \mathrm{Tr}_A$ if $\Diamond$ has a winning strategy in the game $G(t)$.

It is useful to represent an alternating automaton through a finite graph. To do so we give another equivalent definition of alternating automata.

Definition 2.14. A *(tree) parity alternating automaton* is a tuple

$$\mathcal{A} = \langle A, Q_{\Diamond}, Q_{\Box}, q_{\mathrm{I}}, \Delta, \Omega \rangle$$

where the state set of the automaton Q is partitioned into two disjoint sets $Q_{\Diamond}$ and $Q_{\Box}$ and Δ is a subset of

$$Q \times A \times \{\varepsilon, \mathtt{L}, \mathtt{R}\} \times Q.$$

For technical reasons we always assume that our automata are *complete*. i.e. for every $q \in Q$ and $a \in A$ there is at least one transition

$$(q, a, d, q') \in \Delta.$$

Following this definition, we can represent a parity alternating automaton as a finite directed graph and we give the respective acceptance condition by using the graph. The nodes are the states $Q_{\Diamond} \cup Q_{\Box}$. If $q \in Q_i$ then the owner

[1] We can always suppose that a formula in disjunctive normal form can be written as $\phi \vee \psi$ (even if we had more than two elements) by using parenthesis in a proper way. The same for the connective $\wedge$ and formulas in conjunctive normal form.

of q is i, with $i \in \{\Diamond, \Box\}$. Graphically we represent all the states in $Q_\Diamond$ with diamonds and all the states in $Q_\Box$ with squares. Then, if $(q, a, d, q') \in \Delta$, in the graph we add an edge from the state q to the state q' and we label this transition with a, d. The initial state is indicated by an incoming edge that is not labelled. Finally, we write the priority of a state inside it. Such an example can be found in Figure 2.1. In the picture there is a parity alternating automaton with two states: the initial state is the one with priority 0 and it belongs to $\Diamond$, and the other state is with priority 1 and it belongs to $\Box$, while the transitions are represented through the edges and their labels.

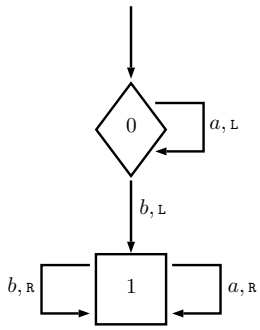


Figure 2.1: A parity alternating automaton represented as a graph.

In order to define the acceptance condition, if $t \in \mathrm{Tr}_A$ we define the game $Alt(t)$ on the graph of an alternating automaton. It is an infinite game played by $\Diamond$ and $\Box$. The positions of the game are elements of $\{\mathtt{L}, \mathtt{R}\}^* \times Q$. The initial position is $(\varepsilon, q_\mathrm{I})$. If (u, q) is the current position of a play, it is the turn of the owner of q: if $q \in Q_\Diamond$ then $\Diamond$ has to move, otherwise $\Box$. The owner of q can move from (u, q) to $(u^\frown d, q')$ if in the graph there is a transition from q to q' labelled by $t(u), d$. An infinite play π of $Alt(t)$ is winning for $\Diamond$ if the sequence of states visited during the play satisfies the parity condition. The automaton $\mathcal{A}$ *accepts* a tree $t \in \mathrm{Tr}_A$ if $\Diamond$ has a winning strategy in the game $Alt(t)$.

Fact 2.15 (Folklore)**.** *The two classes of automata defined in Definition 2.13 and Definition 2.14 with the respective acceptance conditions are equivalent.*

Sketch of the proof. Formulas of the form $\varphi \wedge \psi$ can be thought of as a state of $\Box$ and formulas of the form $\varphi \vee \psi$ as a state of $\Diamond$. Following this idea,

we can easily turn an automaton $\mathcal{A} = \langle A, Q^{\mathcal{A}}, q_{\mathrm{I}}^{\mathcal{A}}, \delta^{\mathcal{A}}, \Omega^{\mathcal{A}} \rangle$ into a graph that represents an alternating automaton and vice-versa. □

Proposition 2.16. *The following properties are equivalent for a given language L.*

1. *L is recognised by a tree parity non-deterministic automaton (Definition 2.4).*

2. *L is recognised by a tree parity alternating automaton.*

Moreover, given a tree parity non-deterministic automaton $\mathcal{A}$ that recognises L we can effectively construct a tree parity alternating automaton $\mathcal{B}$ equivalent to $\mathcal{A}$, and vice-versa.

For the implication $2 \Rightarrow 1$, see [MSS86]. Now we give an alternative proof of the implication $1 \Rightarrow 2$ by using the graph of a parity alternating automaton. Before of the formal proof, we describe the idea.

Let $\mathcal{A} = \langle A, Q, q_{\mathrm{I}}, \Delta, \Omega \rangle$ be a parity non-deterministic automaton. We want to build a parity alternating automaton $\mathcal{B}$ equivalent to $\mathcal{A}$. The idea of the procedure is to build an automaton $\mathcal{B}$ in which there is a copy of every state in Q, these copies belong to $\Diamond$ and they have the same priority as in $\mathcal{A}$; from these states, that are the only states belonging to $\Diamond$, some transitions labelled with c, ε (where $c \in A$) start and they end in a state of $\Box$. Thank to these transition essentially $\Diamond$ chooses the run among all the possible runs in $\mathcal{A}$. All the other states belong to $\Box$ and $\Box$ chooses, during any move, whether going left or right along the path that he wants to visit. So the idea is: $\Diamond$ chooses the run and $\Box$ chooses the path.

Let us consider an example. Suppose, for simplicity, that in Δ there are two different transitions δ_1 and δ_2 that start with (q_{I}, c) (i.e. $(q_{\mathrm{I}}, c) \prec \delta_1, \delta_2$). Let $\delta_1 = (q_{\mathrm{I}}, c, q_1, q_2)$ and $\delta_2 = (q_{\mathrm{I}}, c, q_3, q_4)$. In the automaton $\mathcal{B}$ we will have the situation depicted in Figure 2.2. Now we see the formal proof.

Proof of the implication $1 \Rightarrow 2$ of Proposition 2.16. We want to formalise the idea described in informal way above. Let $\mathcal{A} = \langle A, Q^{\mathcal{A}}, q_{\mathrm{I}}^{\mathcal{A}}, \Delta^{\mathcal{A}}, \Omega^{\mathcal{A}} \rangle$ be a parity non-deterministic automaton. We want to build a parity alternating automaton $\mathcal{B}$ equivalent to $\mathcal{A}$. We do so through the following procedure:

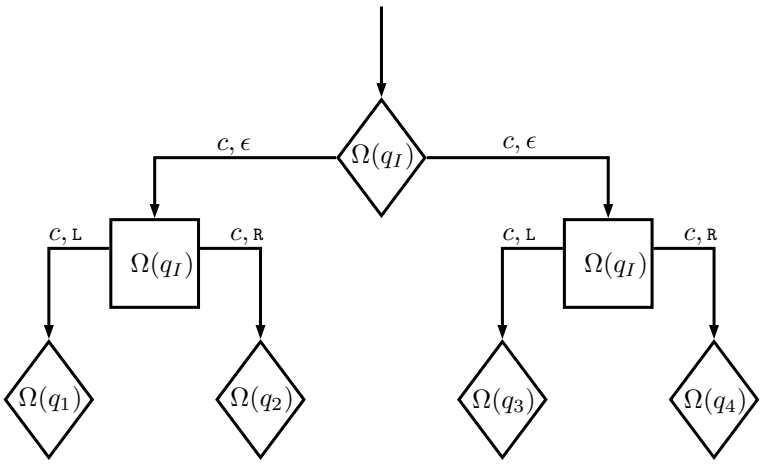


Figure 2.2: The states of $\Diamond$ are copy of the states q_{I}, q_1, q_2, q_3 and q_4.

- For every state $q \in Q^{\mathcal{A}}$ we add in the graph of $\mathcal{B}$ a state $\bar{q}$ with the same priority as q. These states all belong to $\Diamond$. We set $\bar{q}_{\mathrm{I}}$ as the initial state of $\mathcal{B}$.

- For every $\delta = (q, a, q_1, q_2) \in \Delta^{\mathcal{A}}$ we add a new state $\bar{q}_\delta$ that belongs to $\Box$. The priority of this new state is the same as q (that in turn is the same as $\bar{q}$). Moreover, we add an edge from $\bar{q}$ to $\bar{q}_\delta$ labelled by a, ε. Finally we add an edge from $\bar{q}_\delta$ to $\bar{q}_1$ labelled by $a, \mathtt{L}$, and another from $\bar{q}_\delta$ to $\bar{q}_2$ labelled by $a, \mathtt{R}$.

We have to check that $\mathcal{A}$ and $\mathcal{B}$ are equivalent.

- Suppose that a tree t is accepted by $\mathcal{A}$. Then there exists a run ρ that witnesses it. The run ρ provides a winning strategy for $\Diamond$ in $Alt(t)$ in the following way: suppose in $Alt(t)$ we have reached the position $(u, \bar{q})$ (where $\bar{q}$ is a state belonging to $\Diamond$) and suppose that $\rho(u) = q$, $t(u) = a$, $\rho(u^\frown\mathtt{L}) = q_1$ and $\rho(u^\frown\mathtt{R}) = q_2$. This means that the transition $\delta = (q, a, q_1, q_2) \in \Delta^{\mathcal{A}}$. Then $\Diamond$ has to move to $\bar{q}_\delta$. It is immediate to check that this strategy is winning for $\Diamond$.

- Now, suppose that $\Diamond$ has a winning strategy σ in $Alt(t)$. If we use σ to face every possible path chosen by $\Box$, we can construct a run ρ that is successful. We define ρ in the following way: as usual $\rho(\varepsilon) = q_{\mathrm{I}}$. Suppose that $\rho(u) = q$ and $t(u) = a$. We see what move $\Diamond$ does in the

position $(u, \bar{q})$ if he follows σ. Suppose that $\Diamond$ would move to $\bar{q}_\delta$ and $\delta = (q, a, q_1, q_2)$. Then we set $\rho(u^\frown{}_{\mathtt{L}}) = q_1$ and $\rho(u^\frown{}_{\mathtt{R}}) = q_2$. It is easy to check that ρ is a successful run.

Hence, the transformation described above is correct and it makes the correspondence between successful runs and winning strategies explicit. □

Corollary 2.17. *The following properties hold:*

1. *If L is recognised by a parity non-deterministic automaton $\mathcal{A}$ then there exists a parity alternating automaton that recognises L and uses the same priorities as $\mathcal{A}$. In particular parity alternating automata do not use more priorities than parity non-deterministic ones.*

2. *If L is unambiguous then there exists an alternating automaton $\mathcal{A}$ that recognises L such that $\Diamond$ has just one winning strategy for any input t that is accepted.*

Proof. Both the properties directly follow from the construction in the proof of Fact 2.16. □

Remark 2.18. The advantage of using alternating automata, instead of parity non-deterministic ones, is that the operation of complement does not require any extra state. Indeed, given the finite graph that represents an automaton $\mathcal{A}$, to obtain an automaton $\mathcal{A}^c$ that recognises $\mathrm{L}(\mathcal{A})^c$ is enough to swap the owner of the states and to arrange the priorities in the following way: if the priorities used by $\mathcal{A}$ are $\{i, \ldots, k\}$ the ones used by $\mathcal{A}^c$ are $\{i-1, \ldots, k-1\}$ (i.e. we decrease any priority by one) if i is odd, $\{i+1, \ldots, k+1\}$ if i is even. An example can be found in Figure 2.3.

Definition 2.19. A *(tree) weak-alternating automaton* is a parity alternating automaton such that in any transition from a state to another the priorities cannot decrease.

For example the automaton in Figure 2.1 is a weak-alternating automaton. The ones in Figure 2.3 are alternating, but not weak-alternating.

Weak-alternating automata are strictly weaker than alternating automata. In particular it holds:

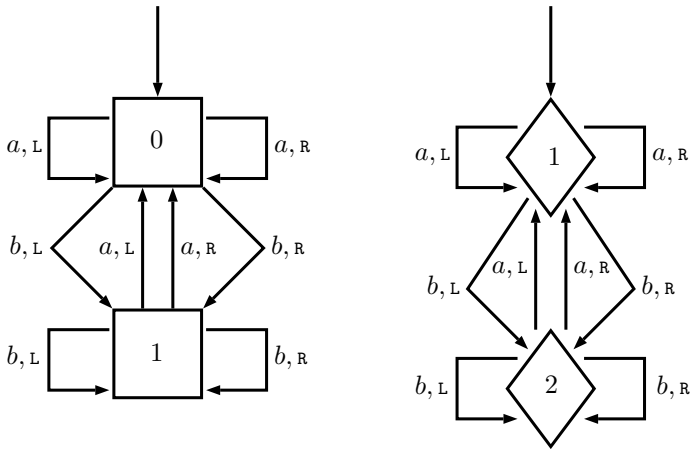


Figure 2.3: Two parity alternating automata: the first one recognises the language $L = \{t \in \mathrm{Tr}_A \mid \text{every path of } t \text{ has infinitely many } a\}$. The second one recognises L^c.

Theorem 2.20 ([MSS86])**.** *The following conditions are equivalent for a tree language $L \subseteq \mathrm{Tr}_A$:*

1. *There exists a weak-alternating automaton $\mathcal{A}$ such that $\mathrm{L}(\mathcal{A}) = L$.*

2. *There exists a sentence φ in $WS2S$ such that*

$$\mathrm{L}(\varphi) = L.$$

Moreover, given a weak-alternating automaton $\mathcal{A}$ we can effectively construct a formula $\varphi \in WS2S$ such that $\mathrm{L}(\varphi) = \mathrm{L}(\mathcal{A})$ and vice-versa.

So we have the following situation:

$$\begin{array}{ccc} S2S & \equiv & \text{Alternating} \\ \cup & & \cup \\ WS2S & \equiv & \text{Weak-Alternating} \end{array}$$

2.6 Guidable automata

Guidable automata were introduced in [CL08] and they are another formalism that captures regular languages.

Definition 2.21. Let $\mathcal{A}$ be a parity non-deterministic automaton. By $Run^{\mathcal{A}}$ we denote the set of all the possible runs of the automaton $\mathcal{A}$.

Notice that $Run^{\mathcal{A}}$ is a closed subset of Tr_Q, where Q is the state set of $\mathcal{A}$. If $\mathcal{A}$ and $\mathcal{B}$ are two parity non-deterministic automata, we say that a function $\tau : Run^{\mathcal{B}} \to Run^{\mathcal{A}}$ is *top-down deterministic* if it is a Lipschitz function with constant 1. In particular this means that if $\rho_1, \rho_2 \in Run^{\mathcal{B}}$ and $\rho_1(u) = \rho_2(u)$ for any node $u \prec u'$, then $\tau(\rho_1)(u') = \tau(\rho_2)(u')$.

Definition 2.22. A *(tree) guidable automaton* $\mathcal{A}$ is a particular tree parity non-deterministic automaton $\mathcal{A} = \langle A, Q^{\mathcal{A}}, q_1^{\mathcal{A}}, \Delta^{\mathcal{A}}, \Omega^{\mathcal{A}} \rangle$ for which for every parity non-deterministic automaton $\mathcal{B}$ such that $\mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{A})$ there exists a top-down deterministic function

$$\tau : Run^{\mathcal{B}} \to Run^{\mathcal{A}}$$

with the following property:

$\forall t \, \forall \rho^{\mathcal{B}}$ run of $\mathcal{B}$ over t, if $\rho^{\mathcal{B}}$ is successful then $\tau(\rho^{\mathcal{B}})$ is a successful run over t.

The idea is that a guidable automaton $\mathcal{A}$ is able to simulate the behaviour of any other automaton that recognises a regular subset of $\mathrm{L}(\mathcal{A})$. Moreover, the fact that the function τ is top-down deterministic is crucial: it means that a guidable automaton throws away the redundant non-determinism. We could say that an automaton that recognises L is guidable if it is "the least non-deterministic" parity automaton that recognises L. Quite surprisingly, guidable automata have the same power as parity non-deterministic automata:

Theorem 2.23. *The following are equivalent for a language L:*

1. *L is recognised by a parity non-deterministic automaton.*
2. *L is recognised by a guidable automaton.*

Moreover, given a parity non-deterministic automaton $\mathcal{A}$ we can effectively construct a guidable automaton $\mathcal{B}$ equivalent to $\mathcal{A}$.

The only non trivial direction is $1 \Rightarrow 2$, see [CL08].

The definition of guidable automaton looks very handy, even if it is not still clear if the notion of *guidability* is so useful to face problems related to automata in easier ways. In the next chapter we will show an application of the notion of guidability to the topological theory of regular tree languages.

2.7 Weak non-deterministic automata

We conclude with the definition of another class of automata that is strictly weaker than parity non-deterministic automata.

Definition 2.24. A *(tree) weak non-deterministic automaton* is a parity non-deterministic automaton $\mathcal{A} = \langle A, Q, q_{\mathrm{I}}, \Delta, \Omega \rangle$ such that for any transition $(q, a, q', q'') \in \Delta$ we have that

$$\Omega(q) \leq \Omega(q'), \Omega(q'').$$

To the best of our knowledge, this class has not been deeply investigated. These automata are weaker than general parity non-deterministic automata, but it is not clear what the class of languages captured by this class of automata is. We will show in this thesis that they are enough to capture every regular tree language in the first levels of the Borel hierarchy of Tr_A.

2.8 Current state of the art

The research related to the interplay between Automata Theory and Descriptive Set Theory is threefold:

Direction 1: Study which levels of topological hierarchies are inhabited by regular tree languages.

Direction 2: Relationship between topological classes and amount of priorities used by given classes of automata.

Direction 3: given a regular tree language L, decide if L belongs to Γ, where Γ is a given topological class.

The first direction is not trivial, since we have countably many automata but the topological hierarchies we have considered (Borel, Wadge, etc) have uncountably many levels. For the second direction we have to find the right formalism that fits the topological structure of a language and weak-alternating automata are a reasonable candidate for this purpose. Regarding the third direction, most of the known effective characterisations speak about languages of either words or finite trees. The case of regular languages of infinite trees seems to be much difficult, mainly because of the inherent non-determinism needed to recognise these languages. Thus, the known examples of effective characterisations are usually limited either to simple classes of sets (e.g. open sets) or to restricted classes of languages (e.g. recognised by deterministic automata).

By Theorem 1.56, all three points are completely solved for regular word languages. For some particular subclasses of regular tree languages these problems have been solved as well. For instance, by [NW03] and [Mur08a] we have a complete solution for deterministic languages. In [Hum12], [MS16] and [DFH15] there are partial solutions for unambiguous languages. Some important results for Büchi automata have been obtained in [MS16], [SW16] and [Urb00].

In this thesis we work with general regular tree languages, without restrictions on our class of automata. The state of the art for general regular tree languages is quite chaotic and there are several problems.

By looking at the definition of a language accepted by a parity non-deterministic automaton and by using the fact that regular tree languages are closed under complementation we easily obtain the following:

Proposition 2.25 (see [PP04], Chapter X, Section 5)**.** *If a tree language $L \subseteq \mathrm{Tr}_A$ is regular then it is in the class $\mathbf{\Delta}^1_2$ of the space Tr_A.*

This upper bound is rough and it has been greatly refined in [LFS15]. Let us now concentrate on what happens within the Borel hierarchy.

Proposition 2.26 ([Sku93])**.** *For any n, there is a regular tree language W that is $\mathbf{\Sigma}^0_n$-complete and is definable in $WS2S$ (or, equivalently, it is recognised by a weak-alternating automaton). Dually for $\mathbf{\Pi}^0_n$.*

It is still an open problem if there are Borel regular tree languages more complicated in topological terms than the ones found by Skurczyński.

Problem 1. Given an ordinal $\alpha \geq \omega$, is there any Borel regular tree language that is $\mathbf{\Sigma}^0_\alpha$-complete?

A partial solution of Problem 1 is in [SW16] where Skrzypczak and Walukiewicz have proved that the answer is negative for Büchi languages (they proved that if L is a regular tree language recognised by a Büchi automaton, then either L is Borel with finite Borel rank or it is $\mathbf{\Sigma}^1_1$, so non Borel) and this result could suggest that the answer to Problem 1 is always negative. Moreover, it is not clear if $WS2S$ is enough to capture every Borel regular tree language.

Problem 2. Every Borel regular tree language is definable in $WS2S$.

As far as the Wadge hierarchy is concerned, in [DM07] Jacques Duparc and Filip Murlak proved the following result:

Theorem 2.27. *Let L_1 and L_2 be two regular tree languages with Wadge rank respectively α_1 and α_2. Then we can effectively construct a regular language $L_1 \oplus L_2$ with Wadge rank $\alpha_1 + \alpha_2$ and a regular language $L_2 \otimes \omega$ with Wadge rank $\alpha_2\omega$.*

Operations on sets corresponding to ordinal sum and product by ω were already known in literature (see e.g. [Wes78] and [Dup01]), but the important contribution of this theorem is that the languages $L_1 \oplus L_2$ and $L_2 \otimes \omega$ are regular.

Sketch of the proof of Theorem 2.27. Let L_1 and L_2 be regular tree languages. Let $\mathcal{B}$ and $\mathcal{A}$ be two parity alternating automata that recognise L_1 and L_2 respectively and let α_1 and α_2 be the two Wadge rank of L_1 and L_2. We define the language $L_1 \oplus L_2$ of all the trees such that:

1. Either for any n $t(\mathtt{RRL}^n) = a$ and $t.(\mathtt{RL}) \in \mathrm{L}(\mathcal{A})$,
2. Or $\mathtt{RL}^m$ is the first node of the form $\mathtt{RL}^n$ labelled with b and either $t(\mathtt{RL}^m\mathtt{L}) = a$ and $t.(\mathtt{RL}^m\mathtt{LL}) \in \mathrm{L}(\mathcal{B})$ or $t(\mathtt{RL}^m\mathtt{L}) = b$ and $t.(\mathtt{RL}^n\mathtt{LL}) \in \mathrm{L}(\mathcal{B})^c$.

It is known ([Dup01]) that $L_1 \oplus L_2$ has Wadge rank $\alpha_1 + \alpha_2$. It is easy to check that this language is recognised by the alternating automaton in Figure 2.4.

Now we define the language $L_2 \otimes \omega$ of all the trees such that:

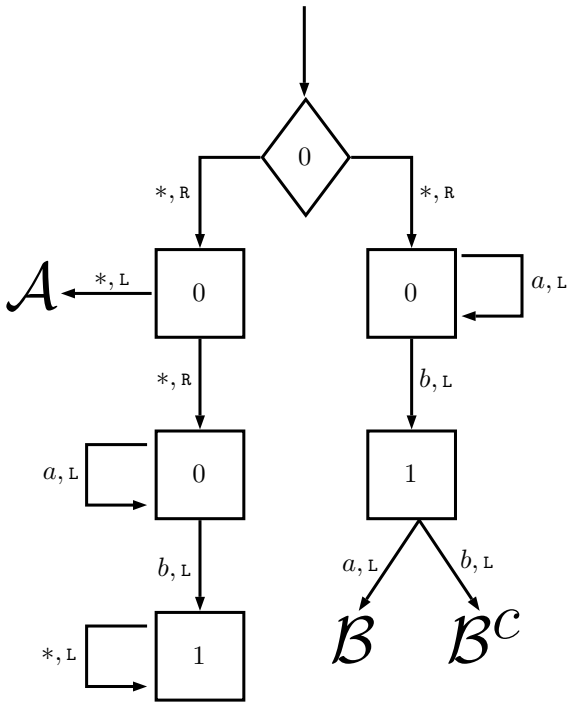


Figure 2.4: The alternating automaton recognising $L_1 \oplus L_2$. The symbol $*$ can be both a and b (it is not relevant for the definition of the language). When an edge ends in an automaton it means that it ends in the initial state of that automaton and then the graph continues with the graph of the corresponding automaton.

1. Either they satisfy the following conditions for some $0 < i \leq k$:
   - $\mathtt{R}^k$ is the first node on the rightmost path labelled by b.
   - i is minimal such that for all $i < j \leq k$, $t(\mathtt{R}^j\mathtt{L}^n) = a$ for any n.
   - $\mathtt{R}^i\mathtt{L}^m$ is the first node labelled by b of the form $\mathtt{R}^i\mathtt{L}^n$.
   - either $t(\mathtt{R}^i\mathtt{L}^m\mathtt{L}) = a$ and $t.(\mathtt{R}^i\mathtt{L}^m\mathtt{L}\mathtt{L}) \in \mathrm{L}(\mathcal{A})$ or $t(\mathtt{R}^i\mathtt{L}^m\mathtt{L}) = b$ and $t.(\mathtt{R}^i\mathtt{L}^m\mathtt{L}\mathtt{L}) \in \mathrm{L}(\mathcal{A})^c$.
2. Or there is no b on the rightmost path starting in the root.

This language has wadge rank $\alpha_2\omega$ (see [Dup01]). It is easy to check that this language is recognised by the automaton in Figure 2.5. □

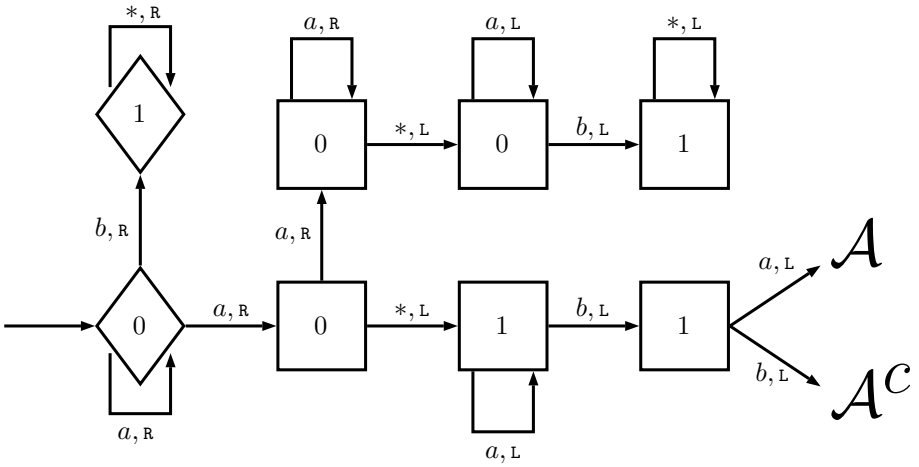


Figure 2.5: The alternating automaton recognising $L_2 \otimes \omega$.

In particular, within the first block of the Wadge hierarchy we can conclude that there exist regular tree languages in every Wadge degree with Wadge rank $\alpha < \omega^\omega$. A reasonable question is whether this ordinal is the upper bound inside $\mathbf{\Delta}^0_2$.

Conjecture 1 (Duparc)**.** There are no regular tree languages in the Wadge degrees with Wadge rank α when

$$\omega^\omega \leq \alpha < \omega_1.$$

Let us focus now on Direction 2.

Definition 2.28. An automaton (parity or alternating) that uses priorities inside the set $\{i, i+1, \ldots, j\}$ is said an (i,j)-automaton. Without loss of generality we can suppose that i is always equal to 0 or 1. We denote the *dual* of the index (i,j) by $\overline{(i,j)}$: the dual of $(0,k)$ is $(1,k+1)$ and vice-versa. We define a partial order of indexes through the formula:

$$(i,j) < (i',j') \Leftrightarrow j - i < j' - i'.$$

In this thesis we will focus on the indexes of weak-alternating automata (Definition 2.19). If $\mathcal{A}$ is a weak-alternating automaton then the pair (i,j) is called the *weak index* of $\mathcal{A}$. The weak index is strongly connected to the Borel rank of a language. Indeed it holds:

Proposition 2.29 ([DM07]). *If L is recognised by a weak-alternating $(1, n+1)$-automaton then $L \in \mathbf{\Sigma}^0_n$ (dually for $(0, n)$-automata and $\mathbf{\Pi}^0_n$).*

Problem 3. If a regular language is in $\mathbf{\Sigma}^0_n$, is it recognised by a weak-alternating $(1, n+1)$-automaton (dually for $\mathbf{\Pi}^0_n$)?

Apparently the general feeling in the community of researchers working on this problem is that the answer to Problem 3 is positive. If this scenario was true we would obtain the following equality:

$$\text{Weak index} \quad = \quad \text{Borel rank.}$$

To the best of our knowledge the problem has not been solved for any class. In this thesis we prove it for the first two levels of the Borel hierarchy.

Summing up all these facts, the optimal situation would be:

Conjecture 2. Every Borel regular tree language has finite Borel rank and it is captured by $WS2S$. Moreover, if it is in $\mathbf{\Sigma}^0_n$ ($\mathbf{\Pi}^0_n$) then there exists a weak-alternating $(1, n+1)$-automaton (respectively $(0, n)$-automaton) that recognises it.

However, there is no strong evidence that Conjecture 2 holds.

Chapter 3

The first level of the Borel hierarchy

In this chapter we give a characterisation of the first level of the Borel hierarchy of Tr_A in terms of decidability and weak index.

3.1 Clopen sets

We start with clopen sets.

Proposition 3.1 (Folklore)**.** *Let $L \subseteq \mathrm{Tr}_A$. L is clopen if and only if L is a finite union of basic clopen sets, i.e. there exist $n \in \omega$ and $p_1, \ldots, p_n$ finite partial trees such that*

$$L = \bigcup_{i=1}^{n} N_{p_i}.$$

Proof. The direction $\Rightarrow$ easily follows by compactness of the space Tr_A, while the direction $\Leftarrow$ is trivial. □

Fact 3.2. *Every clopen set $C \subseteq \mathrm{Tr}_A$ is regular and for every clopen set C there exist a weak non-deterministic $(0,1)$-automaton and a weak non-deterministic $(1,2)$-automaton that recognise C.*

Proof. Firstly consider a basic clopen set N_p. We build a weak non-deterministic $(1,2)$-automaton $\mathcal{A}$ that recognises N_p. We set

$$Q^{\mathcal{A}} = \{q_u \mid u \in p\} \cup \{\top, \bot\},$$

and set q_ε as initial state of $\mathcal{A}$. If u is a node of p, we add in the relation $\Delta^{\mathcal{A}}$ the transition $(q_u, p(u), q_{u^\frown \mathtt{L}}, q_{u^\frown \mathtt{R}})$ if u is not a leaf of p, otherwise we add the transition $(q_u, p(u), \top, \top)$. Then we add all the transitions of the form $(q_u, a, \bot, \bot)$ whenever $a \neq p(u)$ and finally, for every symbol of the alphabet c we add $(\top, c, \top, \top)$ and $(\bot, c, \bot, \bot)$ to $\Delta^{\mathcal{A}}$. We give priority 1 to every state except $\top$, which we give priority 2 to. It is immediate to check that $\mathcal{A}$ recognises N_p. By Fact 3.1 we can easily conclude the proof: if C is clopen we write it as

$$C = \bigcup_{i \in \{1,\dots,n\}} N_{p_i}$$

where we can suppose that this representation is minimal, i.e. there are no p_j and p_k, with $j, k \in \{1, \dots, n\}$, such that $p_j \subset p_k$. In this case we set

$$Q^{\mathcal{A}} = \{q_\varepsilon\} \cup \{q_{u_i} \mid u \in p_i, u \neq \varepsilon\} \cup \{\bot, \top\}$$

and we follow the same idea seen earlier for basic clopen set. The proof for the index $(0, 1)$ is dual. □

Remark 3.3. *Notice that the automata that we have built to recognise N_p are deterministic, whereas the automata built to recognise a generic clopen set C are unambiguous.*

Remark 3.4. *We can also notice that it is easy to write a formula in S2S that captures a clopen set C. If C is a basic clopen set N_p then the formula*

$$\bigwedge_{u \in p} u \in P_{p(u)}$$

captures exactly N_p. If C is a finite union of basic clopen sets then we take the disjunction of the formulas that capture the basic clopen sets of its representation.

3.2 Closed and open sets

Now we prove decidability and the equivalence between weak index and Borel rank for the classes $\mathbf{\Pi}^0_1$ and $\mathbf{\Sigma}^0_1$. First of all we recall two topological notions that we will use in this chapter.

Definition 3.5. Let X be a topological space and let $B \subseteq X$. The *topological closure* of B is the smallest closed set that contains B and it is obtained as the intersection of all the closed sets that contain B. Dually, the *interior* of B is the biggest open set contained in B, obtained as the union of every open set contained in B.

Lemma 3.6 (folklore)**.** *Let $L \subseteq \mathrm{Tr}_A$ be a tree language. It is decidable if L belongs to:*

1. *The Borel class $\mathbf{\Delta}_1^0$.*
2. *The Borel class $\mathbf{\Sigma}_1^0$.*
3. *The Borel class $\mathbf{\Pi}_1^0$.*

Proof. It is enough to prove the third point and we automatically have all of the three points, since decidability is a property that is closed under complementation and intersection. If L is a regular language and it is closed, then we have a formula in $S2S$ that defines the topological closure of L. Indeed, the closure of L is formed by all the trees t such that every finite prefix of t can be extended to some trees in L. Hence, we can build a parity non-deterministic automaton for L and another one for the closure of L. Since it is decidable if two different automata recognise the same language (it is the same as the difference between them is the empty set), our proof is complete. □

Corollary 3.7. *It is decidable if L belongs to the Wadge degrees:*

1. $\mathbf{\Delta}_1^0 \setminus \{\mathrm{Tr}_A, \emptyset\}$.
2. $\mathbf{\Sigma}_1^0 \setminus \mathbf{\Delta}_1^0$.
3. $\mathbf{\Pi}_1^0 \setminus \mathbf{\Delta}_1^0$.

Now move on to the weak index.

Definition 3.8. Let $\mathcal{A} = \langle A, Q, q_{\mathrm{I}}, \Delta, \Omega \rangle$ be a parity non-deterministic automaton and let $q \in Q$. By $\mathrm{L}_q(\mathcal{A})$ we denote the language that $\mathcal{A}$ would recognise if the initial state was q. A state q is called *all-rejecting* if $\mathrm{L}_q(\mathcal{A}) = \emptyset$ and it is called *all-accepting* if $\mathrm{L}_q(\mathcal{A}) = \mathrm{Tr}_A$.

Remark 3.9. *Notice that if q is all-rejecting in $\mathcal{A}$ and (q, a, q_1, q_2) is a transition of the automaton, then at least one of q_1 and q_2 has to be all-rejecting as well. The same remark can not be done for all-accepting states: it is not true that if q is all-accepting and (q, a, q_1, q_2) is a transition then at least one of q_1 and q_2 has to be all-accepting.*

Let $\mathcal{A}$ be a parity non-deterministic automaton. We define $\mathcal{A}_{\mathrm{TC}}$ the automaton obtained from $\mathcal{A}$ by doing the following operations:

1. We take the all-rejecting states in $\mathcal{A}$ and we replace them with a single state $\perp$ and we replace all the transitions (q, a, q_1, q_2) where at least one of q_1 or q_2 is all-rejecting with $(q, a, \perp, \perp)$,
2. We add the transitions $(\perp, c, \perp, \perp)$ for any symbol c in the alphabet,
3. We give priority 1 to $\perp$ and 0 to all the other states.

Proposition 3.10. *The automaton $\mathcal{A}_{TC}$ recognises the topological closure of* $\mathrm{L}(\mathcal{A})$.

Proof. The automaton $\mathcal{A}_{\mathrm{TC}}$ is closed by Fact 2.16 and Fact 2.29. It is immediate to show that $\mathrm{L}(\mathcal{A}) \subseteq \mathrm{L}(\mathcal{A}_{\mathrm{TC}})$, so the topological closure of $\mathrm{L}(\mathcal{A})$ is contained in $\mathrm{L}(\mathcal{A}_{\mathrm{TC}})$ because $\mathrm{L}(\mathcal{A}_{\mathrm{TC}})$ is closed. Now, take a tree t in $\mathrm{L}(\mathcal{A}_{\mathrm{TC}})$. By definition of $\mathcal{A}_{\mathrm{TC}}$, there is a run of $\mathcal{A}$ over t that does not contain any all-rejecting state, which means that every finite prefix p of t can be extended to a tree that belongs to $\mathrm{L}(\mathcal{A})$, and this means that t belongs to the topological closure of $\mathrm{L}(\mathcal{A})$. □

Proposition 3.11. *Let L be a regular language and let $\mathcal{A}$ be a parity non-deterministic automaton that recognises L. The following properties are equivalent:*

1. $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{A}_{TC}) = L$.
2. *L is recognised by a weak-alternating $(0, 1)$-automaton.*
3. *L is closed.*

Proof. The implication $1 \Rightarrow 2$ directly follows from Fact 2.16. The implication $2 \Rightarrow 3$ follows from Fact 2.29. The implication $3 \Rightarrow 1$ follows from Fact 3.10. □

So we obtain the following:

Theorem 3.12. *The weak index and the Borel rank coincide for the first level of the Borel hierarchy:*

- *If L is regular and closed then there exists a weak-alternating $(0,1)$-automaton that recognises L.*
- *If L is regular and open then there exists a weak-alternating $(1,2)$-automaton that recognises L.*

Proof. It follows from Fact 3.14, Fact 2.16 and Remark 2.18. □

Notice that we obtain the equivalence

$$\text{Weak-alternating } (1,2)\text{-automata} \equiv \text{open sets}$$

just by using the standard procedure that we can consider for the complementation of alternating automata (see Remark 2.18). But *a priori* it is also possible to have a direct argument for open sets, without using anything related to closed sets. Yet, it is interesting that to obtain the equality between weak index and Borel rank starting from open sets we need the notion of guidable automaton.

Let $\mathcal{A}$ be a parity non-deterministic automaton. We define $\mathcal{A}_{\mathrm{INT}}$ the automaton obtained from $\mathcal{A}$ by doing the following operations:

1. we take the all-accepting states in $\mathcal{A}$ and we replace them with a single state $\top$ and we replace them with $\top$ in every transition where they appear.
2. We add the transitions $(\top, c, \top, \top)$ for any symbol $c \in A$.
3. We give priority 2 to $\top$ and 1 to all the other states.

Proposition 3.13. *If $\mathcal{A}$ is a guidable automaton, then the automaton $\mathcal{A}_{INT}$ recognises the interior of* $\mathrm{L}(\mathcal{A})$.

Proof. The automaton $\mathcal{A}_{\mathrm{INT}}$ is open by Fact 2.29. Moreover it is easy to check that $\mathrm{L}(\mathcal{A}_{\mathrm{INT}})$ is contained in $\mathrm{L}(\mathcal{A})$, so it is contained in the interior of $\mathrm{L}(\mathcal{A})$. We write the interior of $\mathrm{L}(\mathcal{A})$ as

$$\bigcup_{s \in I} N_s.$$

Fix one of this N_s. By Fact 3.2 we know that N_s is regular and it is recognised by a weak non-deterministic automaton $\mathcal{D}$ where every state has priority 1 except an all-accepting state with priority 2 (see the proof of Fact 3.2). Since $\mathcal{A}$ is guidable we have a top-down deterministic function

$$\tau : Run^{\mathcal{D}} \to Run^{\mathcal{A}}$$

with the following property:

$\forall t\ \forall \rho^{\mathcal{D}}$ run of $\mathcal{D}$ over t, if $\rho^{\mathcal{D}}$ is successful, then $\tau(\rho^{\mathcal{D}})$ is successful over t.

For any node u that corresponds to a leaf of s the states of the form $\tau(\rho)(u^\frown d)$ have to be all-accepting, with $d \in \{\mathtt{L}, \mathtt{R}\}$. Indeed, if there was a state $\tau(\rho)(u^\frown d)$ that is not all-accepting, we can take a tree

$$S \in L_{\tau(\rho)(u^\frown d)}(\mathcal{A})^c$$

and consider the tree t' obtained by plugging S in the node $u^\frown d$. The run ρ would be a successful run of $\mathcal{D}$ over t' but $\tau(\rho)$ would not be successful for t', and this is a contradiction. Hence, if a tree is in the interior of $\mathrm{L}(\mathcal{A})$ (i.e. it starts with a prefix $s \in I$) then there is a run of $\mathcal{A}$ over t that ends with the all-accepting state in every node u for which there exists $u' \in s$ such that $u' \prec u$, and therefore it is accepted by $\mathcal{A}_{\mathrm{INT}}$. □

Hence, we obtain:

Proposition 3.14. *Let L be a regular language and let $\mathcal{A}$ be a guidable automaton that recognises L. The following properties are equivalent:*

1. $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{A}_{INT}) = L$.
2. *L is recognised by a weak-alternating $(1,2)$-automaton.*
3. *L is open.*

We stress that to obtain the equivalence between Borel rank and weak index for open sets the hypothesis that $\mathcal{A}$ is guidable has been crucial. Let us see an example where $\mathcal{A}_{\mathrm{INT}}$ does not recognise the interior of $\mathrm{L}(\mathcal{A})$.

Example 3.15. Consider the language

$$L = \{t \in \mathrm{Tr}_A \mid t(\varepsilon) = a\}.$$

L is clopen, so it is regular. We build a parity non-deterministic automaton $\mathcal{A}$ that recognises L. Let $\mathcal{B}$ be a parity non-deterministic automaton such that $\mathrm{L}(\mathcal{B})$ is analytic and suppose that $\mathcal{B}$ does not have any all-accepting state and let $\mathcal{B}^c$ another parity non-deterministic automaton that recognises $\mathrm{L}(\mathcal{B})^c$ (that is a co-analytic set) and suppose that $\mathcal{B}^c$ does not have any all-accepting state as well (like in Figure 2.3). Now we build $\mathcal{A}$. We define the state set $Q^{\mathcal{A}}$ in the following way:

$$Q^{\mathcal{A}} = q_{\mathrm{I}}^{\mathcal{A}} \cup Q^{\mathcal{B}} \cup Q^{\mathcal{B}^c} \cup \{\top, \bot\}$$

and we set $q_{\mathrm{I}}^{\mathcal{A}}$ as initial state. Then we build $\Delta^{\mathcal{A}}$ by adding the transitions:

1. the tuple $(q_{\mathrm{I}}^{\mathcal{A}}, a, q_{\mathrm{I}}^{\mathcal{B}}, \top)$,
2. the tuple $(q_{\mathrm{I}}^{\mathcal{A}}, a, q_{\mathrm{I}}^{\mathcal{B}^c}, \top)$,
3. the transitions $(q_{\mathrm{I}}^{\mathcal{A}}, c, \bot, \bot)$ for any symbol $c \neq a$,
4. the usual transitions for $\top$ and $\bot$ ($\top$ is all-accepting and $\bot$ is all-rejecting),
5. all the transitions inside $\Delta^{\mathcal{B}}$ and $\Delta^{\mathcal{B}^c}$.

We give $q_{\mathrm{I}}^{\mathcal{A}}$ priority 0 and the states in $Q^{\mathcal{A}}$ and $Q^{\mathcal{B}}$ the same priorities they have in $\mathcal{B}$ and $\mathcal{B}^c$. It is immediate to verify that $\mathrm{L}(\mathcal{A}) = L$. Moreover, $\mathcal{A}_{\mathrm{TC}}$ is a $(0,1)$-automaton and it recognises L, whereas $\mathcal{A}_{\mathrm{INT}}$ is a $(1,2)$-automaton but it recognises the empty language $\emptyset$.

Chapter 4

Between the first and the second level

We now climb the Wadge hierarchy and we study the Wadge classes between the non self-dual pairs $\{\mathbf{\Pi}^0_1, \mathbf{\Sigma}^0_1\}$ and $\{\mathbf{\Pi}^0_2, \mathbf{\Sigma}^0_2\}$.

In this chapter we prove some results related to Direction 3 listed in Section 2.8, i.e. results that are concerned with *decidability*: we prove that it is decidable if a given regular tree language L belongs to Γ, for some topological Wadge class Γ placed between the pair $\{\mathbf{\Pi}^0_1, \mathbf{\Sigma}^0_1\}$ and the pair $\{\mathbf{\Pi}^0_2, \mathbf{\Sigma}^0_2\}$. More precisely, defining a quite sophisticated algebraic approach, we prove that it is decidable if a given regular tree language L belongs to

1. Any Wadge degree (or Wadge class) with Wadge rank below ω, that essentially correspond to the finite levels of the difference hierarchy of the topological space Tr_A.

2. The union of all the Wadge degrees with Wadge rank $< \omega$, that corresponds to the class of all the Boolean combinations of open sets.

3. The Borel class $\mathbf{\Delta}^0_2$, i.e. the first class of the second level of the Borel hierarchy, that correspond to the union of all Wadge degrees with Wadge rank $< \omega_1$.

Actually the second point was already proved in [BP12] and the third one was stated in [FM14], but there are some differences with respect to the original works. Comparing to [BP12], here we provide more detailed and

polished proofs, more pictures, and certain minor glitches corrected; in particular newly added Lemma 4.88 provides an explicit characterisation of the edges in the strategy graph G_L (defined in Section 4.9.1). Regarding [FM14], unfortunately the logical structure of this work is not sound. Here we repair the gap by providing a direct proof of existence of certain edges in the strategy graph G_L. For that, we introduce a new concept of an *essential node* (see Definition 4.100) and a combinatorial lemma about sections of matrices, see Lemma 4.105.

Since in this chapter we will use it, we recall the statement of the so-called *Pigeon-hole Principle*. Pigeon-hole Principle is the combinatorial principle that states that if we have $n+k$ (with $k > 0$) objects placed in n containers, then at least one container has to contain at least two objects. In this chapter we also use the infinite version of the principle: if we have infinite objects placed in finitely many containers, at least one container has to contain infinite objects.

4.1 The game for finite Wadge ranks

In this first section of the chapter we define a game that we will use to obtain results of decidability of Wadge degrees with Wadge ranks below ω. This game is played by two players, named Alternator and Constrainer and it is a finite duration game. We will work in the space Tr_A, but *a priori* this game can be defined in any topological space and the characterisation that it gives holds in general. In the case of trees, yet, it is possible to decide which player wins the game if we play on regular languages and this fact will be crucial to state results about decidability.

Let us describe the game. Let X be a topological space, $U_0 \subseteq X$ open and non empty, and let $X_1, \ldots, X_n$ be arbitrary subsets of X. We define the game

$$\mathcal{H}_{U_0}(X_1, \ldots, X_n)$$

played by Constrainer (choosing open sets of X) and Alternator (choosing points of X). The game will last for n rounds, a round i for $1, \ldots, n$ of the game is played as follows:

1. Alternator chooses a point $x_i \in U_{i-1} \cap X_i$. If there is no such point x_i,

the game $\mathcal{H}_{U_0}(X_1, \ldots, X_n)$ is interrupted and Constrainer wins immediately.

2. Constrainer chooses an open set $U_i \subseteq U_{i-1}$ that contains x_i and the next round is played.

If Alternator manages to survive n rounds then he wins, otherwise Constrainer wins. Since the duration of the game is finite, it is determined. Actually we can say more:

Fact 4.1. *The game $\mathcal{H}_U(X_1, \ldots, X_n)$ is positionally determined: there are two positional strategies (σ_C for Constrainer and σ_A for Alternator) defined in all the positions of the game, such that each position of the game is either:*

- *winning for Constrainer and σ_C is winning from that position,*
- *winning for Alternator and σ_A is winning from that position.*

A special variant of the game, when $U_0 = X$ is the whole space is denoted $\mathcal{H}(X_1, \ldots, X_n)$. Now we prove some properties of this game.

Let (X, τ) be a topological space, $U \subseteq X$ open non empty, and let $X_1, \ldots, X_n$ be subsets of X. Consider the game $\mathcal{H}_U(X_1, \ldots, X_n)$. In this framework we can represent a position of a play through a tuple

$$\langle U_0, x_1, U_1, x_2, U_2, \ldots, x_i, U_i \rangle,$$

where $x_1, \ldots, x_i \in X$ and $U_0, U_1, \ldots, U_i \in \tau$ with $U_0 = U$. A *strategy for Constrainer* in the game $\mathcal{H}_U(X_1, \ldots, X_n)$ is a function

$$\sigma \colon X^{\leq n} \to \tau.$$

If s is a word belonging to $X^{\leq n}$ compatible with the game $\mathcal{H}$ and with σ, then $\sigma(s)$ is the open set played by Constrainer in the position

$$\langle s_0, \sigma(s_0), s_1, \sigma(s_0 s_1), \ldots, s_{lh(s)-1}, \sigma(s) \rangle.$$

If s does not represent a position compatible with the game $\mathcal{H}$ and with σ (for example because the second letter of s is not an element of X_2 or it is not an element of $\sigma(s_0)$), then we set $\sigma(s) = \emptyset$ by convention. As usual, a strategy σ for Constrainer is *winning* if Constrainer wins every play where he follows σ.

In a specular way we could define strategies for Alternator, but we will not use them in the thesis. The first property we prove is Refinement Lemma, that states that if the sets $X_1, \ldots, X_n$ are split into finitely many parts each and Alternator wins $\mathcal{H}(X_1, \ldots, X_n)$ then he can win for some choice of parts of $X_1, \ldots, X_n$.

Lemma 4.2 (Refinement Lemma)**.** *Let $X_1, \ldots, X_n$ be subsets of a topological space X. For $i \in \{1, \ldots, n\}$, let $\mathcal{Y}_i$ a finite family of sets partitioning X_i. For any non empty open $U \subseteq X$, if Alternator wins*

$$\mathcal{H}_U(X_1, \ldots, X_n)$$

then there exist $Y_1 \in \mathcal{Y}_1, \ldots, Y_n \in \mathcal{Y}_n$ such that Alternator wins

$$\mathcal{H}_U(Y_1, \ldots, Y_n).$$

Proof. We prove the theorem by induction on n. The induction base is immediate, because Alternator always wins when $n = 0$ and he wins when $n = 1$ if and only if $X_1 \neq \emptyset$. Now prove the induction step. Consider the first move by Alternator, where he chooses a point $x \in U$. This point necessarily belongs to some $Y_1 \in \mathcal{Y}_1$. For $i \in \omega$, let U_i be the open ball around x of radius $\frac{1}{i}$. By the definition of the game, we know that Alternator wins

$$H_{U_i}(X_1, \ldots, X_n)$$

for every i. By the inductive assumption, we know that for every i there exist $Y_2^{(i)} \in \mathcal{Y}_2, \ldots, Y_n^{(i)} \in \mathcal{Y}_n$ such that Alternator wins

$$H_{U_i}(Y_2^{(i)}, \ldots, Y_n^{(i)}).$$

By Pigeon-hole Principle, there must be some $Y_2, \ldots, Y_n$ such that

$$(Y_2, \ldots, Y_n) = (Y_2^{(i)}, \ldots, Y_n^{(i)})$$

holds for infinitely many i. Since the game $\mathcal{H}_V(Y_2, \ldots, Y_n)$ grows more difficult for Alternator as the open set V becomes smaller, and since every open set V that contains x contains some U_i, we conclude that Alternator wins

$$\mathcal{H}_V(Y_2, \ldots, Y_n)$$

for every V that contains x. By viewing V as a response of Constrainer to Alternator's move $x \in Y_1$, we conclude that Alternator wins the game

$$\mathcal{H}_U(Y_1, \ldots, Y_n).$$

The proof is complete. □

Now let Y be a subset of our topological space X and consider a particular case of the game where the sets $X_1, \ldots, X_n$ alternate between Y and its complement, i.e. we consider $\mathcal{H}(X_1, \ldots, X_n)$ where X_i is Y if i is odd, Y^c otherwise. We denote by $\mathcal{H}^{\in,\notin}(Y,n)$ that game and by $\mathcal{H}_U^{\in,\notin}(Y,n)$ the variant relativised to a non empty open set $U \subseteq X$.

Example 4.3. Consider the game where the topological space X is the space of real numbers $\mathbb{R}$ and $Y = \mathbb{Q}$, i.e. the rational numbers. Then for every n, Alternator wins the game $\mathcal{H}^{\in,\notin}(Y,n)$.

Remark 4.4. *Notice that if Alternator wants to survive in $\mathcal{H}^{\in,\notin}(Y,n)$ as long as possible, he has to avoid to play points in the interior of Y and the interior of Y^c. For example, if at the first round Alternator plays x belonging to the interior of Y then Constrainer can play (a subset of) the interior of Y and Alternator loses because he cannot go outside Y any more.*

Example 4.5. In the real numbers $\mathbb{R}$, let Y be the complement of $\{\frac{1}{n} \in \mathbb{R} \mid n \in \omega\}$. Alternator wins $\mathcal{H}^{\in,\notin}(Y,3)$. Indeed, in the first round Alternator can play $0 \in Y$. In the second round, Alternator plays $\frac{1}{n} \notin Y$ for some large n depending on Constrainer's move. In the third round, Alternator plays $\frac{1}{n} + \varepsilon \in Y$, for some small ε depending on Constrainer's move. Moreover, Constrainer wins (Y,n) for $n \geq 4$.

Remark 4.6. *Let σ_1 and σ_2 be two strategies for Constrainer such that $\sigma_1(s) \subseteq \sigma_2(s)$ for any finite word $s \in X^{\leq n}$. If σ_2 is winning for Constrainer then σ_1 is winning for Constrainer too.*

Lemma 4.7. *Choose some basis B for the topology of the topological space X. If Constrainer has a winning strategy in $\mathcal{H}^{\in,\notin}(Y,n)$ then he has a winning strategy which uses only basic open sets from B.*

Proof. Using the Axiom of Choice AC we can define a function f that to every pair (U,x), where U is an open set and $x \in U$, gives a basic open set $V \in B$ such that $x \in V$ and $V \subseteq U$. Let σ be a winning strategy for Constrainer. We can define another winning strategy $\bar{\sigma}$ that takes sets always from B: given a finite word s of length i, we define $\bar{\sigma}(s) = f\big(\sigma(s), s_i\big)$. Since σ was winning for Constrainer, by Remark 4.6 also $\bar{\sigma}$ is winning. □

Now we are ready to give a characterisation of $\mathcal{D}_n\big(\mathbf{\Sigma}^0_1\big)$ sets in terms of the game $\mathcal{H}$.

Theorem 4.8. *Let X be a topological space and let $Y_0 \subseteq X$. The following conditions are equivalent:*

1. *Y_0 belongs to $\mathcal{D}_{n_0}(\mathbf{\Sigma}^0_1)$.*
2. *Constrainer has a winning strategy in $\mathcal{H}^{\in,\notin}(Y_0, n_0+1)$.*

Proof. We have to prove both directions separately.

Implication $1 \Rightarrow 2$. Suppose that n_0 is odd and let

$$Y = A_0 \cup (A_2 \setminus A_1) \cup \ldots \cup (A_{n_0-1} \setminus A_{n_0-2}), \tag{4.1}$$

where $A_0 \subseteq A_1 \subseteq A_2 \subseteq \ldots \subseteq A_{n_0-2} \subseteq A_{n_0-1}$ and every A_i is open for $0 \leq i \leq n_0 - 1$. Then, a winning strategy for Constrainer in $\mathcal{H}^{\in,\notin}(Y_0, n_0)$ is to play A_{n_0-1} as the first move, A_{n_0-2} as the second move, and so on. Equation (4.1) implies that this is a valid strategy of Constrainer. The n_0-th move will be A_0, and since $A_0 \subseteq Y_0$, at that point Alternator loses at the (n_0+1)-th round. The case of n_0 even is completely dual, in that case $A_0 \subseteq Y_0^c$.

Implication $2 \Rightarrow 1$. Consider a non empty open set $U \subseteq X$, an arbitrary set $Y \subseteq X$, and $n \geq 0$. We define $\mathrm{Str}(U, Y, n)$ the set of winning strategies of Constrainer in $\mathcal{H}^{\in,\notin}_U(Y, n)$. By the hypothesis $\mathrm{Str}(X, Y_0, n_0+1)$ is non empty. We want to select a particular winning strategy $\bar{\sigma}(U, Y, n)$ for Constrainer in such a way that in every turn Constrainer plays the biggest possible open set. Given a sequence $s \in X^{\leq n}$ we define:

$$\bar{\sigma}(U, Y, n)(s) = \bigcup_{\sigma \in \mathrm{Str}(U,Y,n)} \sigma(s).$$

Notice that, since $\bar{\sigma}(U, Y, n)$ selects in any turn the biggest possible open set, it can not happen that s is compatible with $\sigma \in \mathrm{Str}(U, Y, n)$ but it is not compatible with $\bar{\sigma}(U, Y, n)$. We will now prove the following claim.

Claim 4.9. *If $\mathrm{Str}(U, Y, n)$ is non empty then the strategy $\bar{\sigma}(U, Y, n)$ is winning for Constrainer, i.e. $\bar{\sigma}(U, Y, n) \in \mathrm{Str}(U, Y, n)$.*

Proof. By induction on the duration of the game n. For $n = 1$ the claim is easy, as Constrainer wins $\mathcal{H}_U(Y, 1)$ if and only if $Y \cap U = \emptyset$ and moreover, if $Y \cap U = \emptyset$ then $\bar{\sigma}(U, Y, n)$ is winning.

Now suppose that for n the claim holds and prove the claim for $n+1$. Assume for the sake of contradiction that $\bar{\sigma}(U, Y, n)$ is not winning for Constrainer in $\mathcal{H}_U^{\in,\notin}(Y, n+1)$. It means that there is a play

$$\langle x_1, U_1, x_2, U_2, \ldots, x_n, U_n \rangle$$

where Constrainer follows $\bar{\sigma}(U, Y, n)$ and Alternator can survive selecting a point x_{n+1} inside U_n that obeys the rules of the game (i.e. $x_{n+1} \in Y^c$ if n is even, otherwise it belongs to Y).

Consider a point $x \in U_1$. Since U_1 is the union of all $\sigma(x_1)$ for $\sigma \in \mathrm{Str}(U, Y, n)$ it means that there exists $\sigma \in \mathrm{Str}(U, Y, n)$ such that $x \in \sigma(x_1) \subseteq U_1$. However, as $\sigma \in \mathrm{Str}(U, Y, n{+}1)$, Constrainer has a winning strategy from a position $\langle x_1, \sigma(x_1), x \rangle$. Thus, by Remark 4.1 we know that Constrainer has a winning strategy from $\langle x_1, U_1, x \rangle$. Therefore, Constrainer wins the game $\mathcal{H}_{U_1}^{\in,\notin}(Y^c, n)$. By inductive hypothesis, Constrainer wins

$$\mathcal{H}_{U_1}^{\in,\notin}(Y^{\mathrm{c}}, n)$$

following $\bar{\sigma}(U_1, Y^{\mathrm{c}}, n)$. But it is a contradiction, since we supposed that $\bar{\sigma}(U, Y, n{+}1)$ was not winning for Constrainer. This completes the proof of the claim. □

Now suppose n_0 is odd (the proof for n_0 even is specular). We want to write Y_0 as

$$Y_0 = A_0 \cup (A_2 \setminus A_1) \cup \ldots \cup (A_{n_0-1} \setminus A_{n_0-2}),$$

where $A_0 \subseteq A_1 \subseteq \ldots \subseteq A_{n_0-1}$ are open sets. We define, for $i \in \{0, 1, \ldots, n_0 - 1\}$, the sets:

$$A_i \stackrel{\text{def}}{=} \bigcup_{s \in X^{n_0-i}} \bar{\sigma}(X, Y_0, n_0{+}1)(s).$$

Remember that we have established that if s is not a position compatible with the game and a strategy σ then $\sigma(s) = \emptyset$, so we can allow s to range among the whole set X^{n_0-i} for any i. Every A_i is open because it is a union of open sets and we claim that

$$Y_0 = A_0 \cup (A_2 \setminus A_1) \cup \ldots \cup (A_{n_0-1} \setminus A_{n_0-2}).$$

Notice that if an element of Y_0 belongs to A_{n_0-2i} for some i then $x \in A_{n_0-(2i+1)}$. For example if $x \in Y_0$ belongs to A_{n_0-2} then it means that x

can be played as the third move of Alternator (the third move has to be inside Y_0), so $x \in A_{n_0-3}$ as well. So it can not happen that an element $x \in Y_0$ belongs to A_i, with i an odd number less than n_0, and $x \notin A_{i-1}$.

Now, call C the difference $A_0 \cup (A_2 \setminus A_1) \cup \ldots \cup (A_{n_0-1} \setminus A_{n_0-2})$. We are ready to prove both the inclusions.

$Y \subseteq C$. Suppose $x \in Y_0$. In this case x can be the first move of Alternator, so $x \in A_{n_0-1}$ (by definition A_{n_0-1} contains at least all the possible first moves of Alternator). If $x \notin A_{n_0-2}$ we are done, otherwise x has to belong to A_{n_0-3} as well. So either at some point there is an index i such that x belongs to the bracket $(A_i \setminus A_{i-1})$, or, if such an i does not exist, we obtain that $x \in A_0$. In both case x belongs to the difference C.

$C \subseteq Y$. Now suppose that x belongs to

$$A_0 \cup (A_2 \setminus A_1) \cup \ldots \cup (A_{n_0-1} \setminus A_{n_0-2}).$$

In this case either $x \in A_0$ or there is an i such that $x \in (A_i \setminus A_{i-1})$. In the first case we are done because A_0 has to be entirely contained in Y_0 (otherwise $\bar{\sigma}$ would not be winning). If $x \in (A_i \setminus A_{i-1})$ then x has to belong to Y_0: indeed, if $x \in Y_0^c$ but it belongs to A_i, it means that x can be played as $(i{+}1)$th move of Alternator and so x would have to belong to A_{i-1} as well.

So we conclude that

$$Y = A_0 \cup (A_2 \setminus A_1) \cup \ldots \cup (A_{n_0-1} \setminus A_{n_0-2}).$$

Specular proof if n_0 is even.

The proof is complete. □

Corollary 4.10. *The following conditions are equivalent for a set Y:*

1. *Y belongs to $\bigcup_{n\in\omega} \mathcal{D}_n(\mathbf{\Sigma}^0_1)$.*
2. *Constrainer wins the game $\mathcal{H}^{\in,\notin}(Y, n)$ for all but finitely many n.*

Proof. It follows from Theorem 4.8. □

4.2 Decidability of finite levels of the Wadge hierarchy

In this section we prove that, given a natural number n, it is decidable if a regular language L is an n-difference of open sets. To obtain that, we use the game $\mathcal{H}^{\in,\notin}(Y,n)$ in the topological space Tr_A. So, first of all let us revisit the game in terms of trees. We recall that a basis of τ_{pref} is given by the clopen sets N_p, with p a finite partial tree. So, thanks to Lemma 4.7 we can suppose that Constrainer plays finite prefixes of the trees played by Alternator, and Alternator has to extend the finite prefixes played by Constrainer.

Example 4.11. Consider the language

$$L = \{t \in \mathrm{Tr}_A \mid \text{infinitely many } a \text{ appear in } t\}.$$

L is regular and it is easy to check that Alternator can win the game $\mathcal{H}^{\in,\notin}(L,n)$ for every $n \in \omega$. This is because every finite prefix can be extended to a tree with finitely many a or to a tree with infinitely many a. So L is not a Boolean combination of open sets (it is known, indeed, that L is a $\mathbf{\Pi}^0_2$ set but it is not a $\mathbf{\Sigma}^0_2$ set).

Lemma 4.12. *Given regular tree languages $L_1, \ldots, L_n$, one can decide who wins $\mathcal{H}(L_1, \ldots, L_n)$. In particular, given L and n, one can decide who wins $\mathcal{H}^{\in,\notin}(L,n)$.*

Proof. We prove the statement for two regular languages L_1, L_2. It is easy to generalise to n regular languages. The sentences "Alternator wins the game $\mathcal{H}(L_1,L_2)$" and "Constrainer wins the game $\mathcal{H}(L_1,L_2)$" can be effectively formalised in $S2S$. For instance the sentence

"Alternator wins the game $\mathcal{H}(L_1,L_2)$"

is:

there exists a tree $t_1 \in L_1$ such that for any finite prefix p of T_1

there exists a tree $t_2 \in L_2$ that extends p.

In similar way we can write the sentence that says that Constrainer wins for $n > 2$. □

So we obtain:

Corollary 4.13. *It is decidable, given a regular tree language L and $n \in \omega$, whether L is an n-difference of open sets.*

Proof. It directly follows from Theorem 4.8 and Lemma 4.12. □

Hence, since Wadge degrees with Wadge ranks below ω are formed by Boolean combinations of the levels of the difference hierarchy, we easily obtain:

Theorem 4.14. *Given a regular language L and a Wadge degree $[A]_W$ with Wadge rank less than ω, it is decidable if L belongs to $[A]_W$.*

Now our next goal is decidability of the class $\bigcup_{n\in\omega} \mathcal{D}_n(\mathbf{\Sigma}^0_1)$. Notice that Corollary 4.13 gives us a semi-algorithm for deciding if a regular language is in $\bigcup_{n\in\omega} \mathcal{D}_n(\mathbf{\Sigma}^0_1)$. Indeed, for $n = 1, 2, \ldots$ we can use Corollary 4.13 to decide if $L \in \mathcal{D}_n(\mathbf{\Sigma}^0_1)$. If for some n it is the fact then $L \in \bigcup_{n\in\omega} \mathcal{D}_n(\mathbf{\Sigma}^0_1)$ and the algorithm terminates. Otherwise the algorithm does not terminate. In Sections 4.3, 4.4, 4.5, 4.6, 4.7, and 4.8 we will develop tools and provide an algorithm that always terminates and solves the above decision problem.

4.3 The algebra on trees

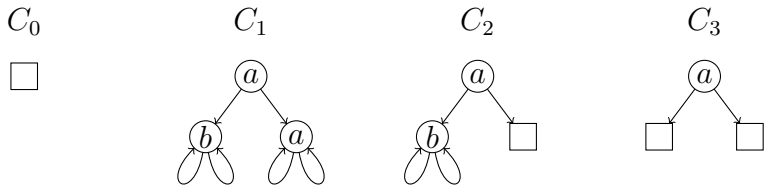
The algebraic approach we define in this section is based on the so-called *Myhill-Nerode equivalence* that allows to distinguish trees based on their behaviour when put into certain *contexts*.

Definition 4.15. A *multicontext* over an alphabet A is a partial tree t over $A \sqcup \{\Box\}$ where $\Box \notin A$ such that: t does not contain any unary nodes and a node of t is a leaf if and only if it is labelled $\Box$. A *port* of a multicontext C is any node of t that is labelled $\Box$ (i.e. any leaf of t).

The number of ports is called the *arity* of the multicontext. *A priori* a multicontext may have infinitely many ports and in that case the arity is ∞. A multicontext with exactly one port is called a *context*. Given a multicontext C and a valuation η which maps ports of C to trees in Tr_A, we write $C[\eta]$ for the tree obtained by replacing each port u by the tree $\eta(u)$. The tree $C[\eta]$ is said to *extend* the multicontext C. If L is a set of trees and C is a multicontext then by $C[L]$ we denote the set of trees obtained by plugging

trees of L in the ports of C in all the possible ways. The set of all trees extending a multicontext C is denoted by $C[*]$. If C is a multicontext, possibly with infinitely many ports, and t is a tree, we denote by $C[t]$ the tree obtained by putting t in every port of C.

Example 4.16. The multicontext C_0 consists of only one node — the root. It is called the *trivial context* and denoted $\Box$. C_1 is a tree, it has no ports, and $C_1[*]$ is $\{C_1\}$. The multicontext C_2 is a context and if we complete C_2 with a tree we obtain a tree where the root label is a and the left subtree of the root is labelled with only letters b. Finally, C_3 is a finite multicontext and $C_3[*] = \{t \mid t(\varepsilon) = a\}$.



Now we focus on contexts, i.e. multicontexts with exactly one port. We write Cntx_A for the set of all non trivial contexts over A. Given two contexts C, D we write $C \cdot D$ for the context obtained by replacing the port of C with D. Moreover, if $C \neq \Box$ (i.e. C is not trivial) then we write C^∞ for the infinite tree

$$C \cdot C \cdot C \cdot C \cdots$$

Remark 4.17. *It is easy to verify that* $\cdot$ *is* associative, *therefore* $\big(\mathsf{Cntx}_A \sqcup \{\Box\}, \cdot\big)$ *is an infinite monoid, with the trivial context* $\Box$ *as the neutral element.*

Now we define the two Myhill-Nerode equivalence relations: one for trees and one for contexts. These equivalence relations depend on a fixed language $L \subseteq \mathrm{Tr}_A$.

Definition 4.18. In the Myhill-Nerode equivalence for trees, we say that two trees t and t' are *L-equivalent* if

$$C[t] \in L \iff C[t'] \in L \quad \text{for every multicontext } C.$$

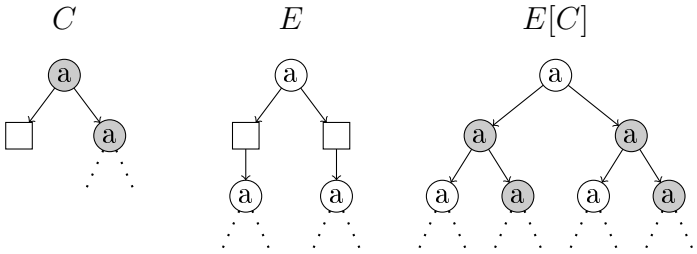
Example 4.19. Consider the language $L = \{t \in \mathrm{Tr}_A \mid t(\varepsilon) = a\}$. In this case we have just two equivalence classes that are L and the complement L^c. The context that establishes if a tree belongs to L or L^c is the trivial context $\Box$.

To give a similar definition for contexts, we use a variant of multicontexts where the ports can be substituted by contexts and not trees.

Definition 4.20. A *context environment* over an alphabet A is a partial tree labelled by $A \sqcup \{\Box\}$ such that: t has no leaves and a node of t is unary if and only if it is labelled $\Box$. A *port* of a context environment is any node of t that is labelled $\Box$ (i.e. any unary node of t).

Given a context environment E and a non trivial context C, we write $E[C]$ for the tree obtained by substituting C for every port of E.

Example 4.21. C is a context, E is a context environment and $E[C]$ is a tree.



Definition 4.22. We define two non trivial contexts C and C' to be *L-equivalent* if

$$E[C] \in L \iff E[C'] \in L \quad \text{for every context environment } E.$$

We denote by H_L the set of the equivalence classes of trees with respect to L, the elements of H_L are called *tree types*. Similarly we denote by V_L the set of the equivalence classes of non trivial contexts with respect to L, the elements of V_L are called *context types*. By 1_L we denote an artificial context type that does not belong to V_L and corresponds to the trivial context $\Box$.

Definition 4.23. We define the *syntactic morphism* of L, denoted by α_L, as the two-sorted function

$$\alpha_L \colon (\mathrm{Tr}_A, \mathsf{Cntx}_A) \to (H_L, V_L),$$

that maps a tree $t \in \mathrm{Tr}_A$ into its L-equivalence class $\alpha_L(t) \in H_L$ (analogously for $C \in \mathsf{Cntx}_A$ we have $\alpha_L(C) \in V_L$).

The following fact follows from standard methods of constructing algebra from automata, see for instance [BI09] where algebras with structure richer than here are used.

Fact 4.24. *If L is a regular language then both H_L and V_L are finite. Moreover, given a parity non-deterministic automaton $\mathcal{A}$ recognising L, one can compute in* EXPTIME *a syntactic algebra (H_L, V_L) for L where both H_L and V_L have at most exponentially many elements in the number of states of $\mathcal{A}$.*

Remark 4.25. *Finiteness of H_L and V_L is necessary but not sufficient for regularity, for instance both H_L and V_L are finite for any language defined in* MSO+U *(see [Boj04]).*

As expressed by the following lemma, the natural operations on contexts and trees respect the L-equivalence. Therefore the lemma implies that α_L imposes a two-sorted algebraic structure on (H_L, V_L).

Lemma 4.26. *The following operations respect L-equivalence.*

1. *For every multicontext D, the operation $t \to D[t]$.*
2. *For every context environment E, the operation $C \to E[C]$.*
3. *The composition of contexts $(C_1, C_2) \to C_1 \cdot C_2$.*
4. *Substituting a tree in the port of a context $(C, t) \to C \cdot t$.*
5. *Infinite iteration of a non trivial context $C \to C^\infty$.*
6. *For every symbol $c \in A$, the operations $t \mapsto c(t, \Box)$ and $t \mapsto c(\Box, t)$, that produce new contexts with roots labelled c and t plugged as one of the subtrees under the root.*

Proof. We prove all the six items.

Items 1 and 2 These items follow directly from the definition of L-equivalence.

Item 3 Let C_1, C_2, D_1, D_2 be contexts such that C_1 is L-equivalent with D_1 and C_2 is L-equivalent with D_2. Let E be some context environment, we prove:

$$E[C_1 \cdot C_2] \in L \Leftrightarrow E[D_1 \cdot D_2] \in L.$$

We construct from E, C_1 and E, D_2 respectively two new context environments E_C and E_D such that

$$E_C[C_2] = E[C_1 \cdot C_2] \text{ and } E_D[D_1] = E[D_1 \cdot D_2].$$

Using the L-equivalence, we have:

$$\begin{array}{rcl} E_D[C_1] \in L & \Leftrightarrow & E_D[D_1] \in L \\ E_C[C_2] \in L & \Leftrightarrow & E_C[D_2] \in L \end{array}$$

Note that by the definition of E_C and E_D we have $E_C[D_2] = E_D[C_1]$. It follows that

$$E[C_1 \cdot C_2] \in L \Leftrightarrow E[D_1 \cdot D_2] \in L.$$

Indeed

$$E[C_1 \cdot C_2] \in L \Leftrightarrow E_C[C_2] \in L \Leftrightarrow E_C[D_2] \in L \Leftrightarrow E_D[C_1] \in L \Leftrightarrow E_D[D_1] \in L.$$

Therefore, $C_1 \cdot C_2$ and $D_1 \cdot D_2$ are L-equivalent.

Item 4 Let C, C' be L-equivalent contexts and t, t' be L-equivalent trees. We want to prove that $C[t]$ and $C'[t']$ are two L-equivalent trees. Let D be a generic multicontext. We prove that

$$D[C[t]] \in L \Leftrightarrow D[C'[t']] \in L.$$

Let D' be a multicontext such that $D'[t] = D[C[t]]$ and E a context environment such that $E[C'] = D[C'[t']]$. Using the L-equivalence we have:

$$\begin{array}{rcl} D'[t] \in L & \Leftrightarrow & D'[t'] \in L \\ E[C] \in L & \Leftrightarrow & E[C'] \in L \end{array}$$

By definition of E, D' we have $D'[t'] = E[C]$. It follows that

$$D[C[t]] \in L \Leftrightarrow D[C'[t']] \in L.$$

Therefore, $C[t]$ and $C'[t']$ are L-equivalent.

Item 5 Let C, C' be L-equivalent non trivial contexts. We want to prove that C^∞ and C'^∞ are two L-equivalent trees. Let D be some multicontext, we prove that

$$D[C^\infty] \in L \Leftrightarrow D[C'^\infty] \in L.$$

Consider a context environment E constructed from D by replacing each port of D with an infinite chain of ports. Using L-equivalence of C and C' we get:

$$E[C] \in L \Leftrightarrow E[C'] \in L$$

By definition of E we have $E[C] = D[C^\infty]$ and $E[C'] = D[C'^\infty]$. It follows that

$$D[C^\infty] \in L \Leftrightarrow D[C'^\infty] \in L.$$

Therefore, C^∞ and C'^∞ are L-equivalent.

Item 6 We only do the proof for $t \mapsto c(t, \Box)$, the other operations is handled symmetrically. Let t, t' be L-equivalent trees. Let E be some context environment, we prove that

$$E[c(t, \Box)] \in L \Leftrightarrow E[c(t', \Box)] \in L.$$

By inserting c into the ports of E we construct a multicontext C such that for all trees s, $C[s] = E[c(s, \Box)]$. Using L-equivalence of t and t' we get:

$$C[t] \in L \Leftrightarrow C[t'] \in L$$

Therefore, $c(t, \Box)$ and $c(t', \Box)$ are L-equivalent.

The proof is complete. □

Corollary 4.27. *The syntactic morphism α_L introduces the following algebraic structure on (H_L, V_L):*

- *composition of context types $V_L \ni u, v \longmapsto u \cdot v \in V_L$,*
- *action of V_L on H_L i.e. $V_L \times H_L \ni (u, h) \longmapsto u \cdot h \in H_L$,*
- *infinite composition $V_L \ni u \longmapsto u^\infty \in H_L$,*
- *creation of contexts $H_L \ni h \longmapsto c(\Box, h), c(h, \Box) \in V_L$ for $c \in A$.*

Moreover, $(V_L, \cdot)$ *is a semigroup acting over* H_L *via* $\cdot$*,* $\big(V_L \sqcup \{1_L\}, \cdot\big)$ *is a monoid, and* (H_L, V_L) *satisfy the axioms of a Wilke algebra [Wil93].*

The following fact follows from a general algebraic considerations, see [Idz12, Lemma 24 on page 42]; it can also be proved directly, by minimality of the syntactic algebra.

Fact 4.28. *The syntactic algebra* (H_L, V_L) *is* faithful *with respect to* V_L*: consider two elements* $v, v' \in V_L$ *and assume that for all* $h \in H_L$ *we have* $vh = v'h$ *and for all* $u \in V_L$ *we have* $(vu)^\infty = (v'u)^\infty$*; then* $v = v'$*.*

The next fact is considered folklore, see for instance [PP04, Proposition 1.11 in Annex A on page 442].

Fact 4.29. *Given any finite semigroup* V*, there is a number* $\sharp_V$ *(denoted just* $\sharp$ *if* V *is known from the context) such that for each element* v *of* V *the element* $v^\sharp$ *is an idempotent, i.e.* $v^\sharp = v^\sharp v^\sharp$*.*

4.3.1 Quotients

Similarly as in the case of finite words, the syntactic algebra induces a natural notion of a quotient of a language: given a multicontext D with n holes and a language K of trees, by $D^{-1}(K)$ we denote the set of tuples $(t_1, \ldots, t_n)$ such that the valuation η mapping the ith port of D into t_i satisfies

$$D[\eta] \in K.$$

Notice that if D is a context then $D^{-1}(K)$ is a set of trees. Moreover, the fact whether $D[t] \in K$ depends only on $\alpha_L(D) \in V_L \sqcup \{1_L\}$ and $\alpha_L(t) \in H_L$. Therefore, it makes sense to write $v^{-1}(K)$ for $v \in V_L \sqcup \{1_L\}$. Notice that from the definition

$$(vu)^{-1}(K) = u^{-1}\big(v^{-1}(K)\big). \tag{4.2}$$

4.3.2 The game on types

Now we want to extend the definition of the game $\mathcal{H}$ to sets of contexts. Recall that contexts are defined as a special case of partial trees, with an additional port label that appears in exactly one leaf. Hence, there is a natural notion of product topology on contexts and therefore we can talk about open sets of contexts. This yields the definition of a game $\mathcal{V}(K_1, \ldots, K_n)$

for a sequence $K_1, \ldots, K_n$ of context languages (i.e. subsets of $\mathsf{Cntx}_A \sqcup \{\Box\}$), which is played by Alternator and Constrainer. The game is played in n rounds. Round $i = 1$ is special: Alternator chooses a context $C_1 \in K_1$. Let u be the port of the context C_1. This port will stay fixed for the rest of the game; all contexts produced by Alternator will have their port in the node u. Next, Constrainer chooses a finite prefix D_1 of C_1, which has one of its leaves in the node u.

A subsequent round $i \in \{2, \ldots, n\}$ is played as follows. Let D_{i-1} be the finite partial tree over $A \sqcup \{\Box\}$ chosen by Constrainer in the previous round with a leaf in the node u.

- Alternator provides a context C_i, which extends D_{i-1}, belongs to K_i, and has its port in the node u. If there is no such context, the game is interrupted and Constrainer wins immediately.
- Constrainer chooses a finite prefix D_i of C_i and which has a leaf in the node u.

If Alternator manages to survive n rounds then he wins. Recall that by the definition of the syntactic morphism, a tree type $h \in H_L$ is actually equal to the set of trees $\alpha_L^{-1}(h)$, similarly for a context type $v \in V_L$. Therefore, it makes sense to talk about the games $\mathcal{H}(h_1, \ldots, h_n)$, and $\mathcal{V}(v_1, \ldots, v_n)$ for sequences of types. Using these games we define the following sets of sequences of types:

Definition 4.30. We define two sets:

$$\mathcal{H}_L = \{(h_1, \ldots, h_n) \in (H_L)^* \mid \text{ Alternatonr wins } \mathcal{H}(h_1, \ldots, h_n)\},$$
$$\mathcal{V}_L = \{(v_1, \ldots, v_n) \in (V_L)^* \mid \text{ Alternatonr wins } \mathcal{V}(v_1, \ldots, v_n)\}.$$

A comment on notation is in order here. The sets $\mathcal{H}_L$ and $\mathcal{V}_L$ contain words, over alphabets H_L and V_L, respectively. Usually when dealing with words, one omits the brackets and commas and writes abc instead of (a, b, c). When the alphabet is V_L this leads to ambiguity, since the expression vwu can be interpreted as a word with a single letter obtained by multiplying the three context types v, w and u, or a three-letter word over the alphabet V_L. These two interpretations should not be confused, so we write $(v_1 \ldots, v_n)$ for n-letter words over the alphabet V_L. For the sake of uniformity, we also write $(h_1, \ldots, h_n)$ for n-letter words over the alphabet H_L, although there is no risk of ambiguity here.

Remark 4.31. *Since we have to deal with a lot of symbols, we sum up the notations introduced until now:*

- Tr_A *is the set of all the full binary trees over the finite alphabet* A*.*
- H_L *is the set of all the equivalence classes of Myhill-Nerode relation for trees. The elements of* H_L *are called tree types.*
- Cntx_A *is the set of contexts over the alphabet* A*.*
- V_L *is the set of the equivalence classes of Myhill-Nerode relation for contexts. The elements of* V_L *are called context types.*
- $\mathcal{H}_L$ *is the set of words* $(h_1, \dots, h_n) \in (H_L)^*$ *such that Alternator wins the game* $\mathcal{H}(h_1, \dots, h_n)$*.*
- $\mathcal{V}_L$ *is the set of words* $(v_1, \dots, v_n) \in (V_L)^*$ *such that Alternator wins the game* $\mathcal{V}(v_1, \dots, v_n)$*.*

It turns out that the sets of words $\mathcal{H}_L$ and $\mathcal{V}_L$ have specific structure. We say that a word u is a *subword* of w if u can be obtained from w by removing some letters. It is clear from the definitions of the games $\mathcal{H}$ and $\mathcal{V}$ that both sets $\mathcal{H}_L$ and $\mathcal{V}_L$ are closed under removing letters. Therefore, Higman's Lemma implies that both these sets are regular.

Lemma 4.32 (Higman's Lemma, see [Hig52])**.** *The set of finite words* A^* *over a finite alphabet* A *with the subword ordering is a well-quasi order: there is no infinite antichain nor an infinite descending chain.*

Corollary 4.33. *If* $L \subseteq A^*$ *is closed under removing letters then* L *is regular.*

Proof. Consider $L \subseteq A^*$ that is closed under removing letters. Then L forms a downward-closed set with respect to the subword relation. Thus, Higman's Lemma implies that there exists a finite set of words $w_1, \dots, w_N$ such that w does not belong to L if and only if one of $w_1, \dots, w_N$ is a subword of w. Such a condition is a regular condition. □

Proposition 4.34. *Both* $\mathcal{H}_L$ *and* $\mathcal{V}_L$ *are regular languages of finite words.*

Proof. Both languages are closed under removing letters and therefore Corollary 4.33 applies. □

The above fact is amusing, but useless, because it does not say how to compute automata for $\mathcal{H}_L$ and $\mathcal{V}_L$ as a function of a representation of the language L.

Lemma 4.35. *The following properties hold:*

1. $(h_1, \ldots, h_n) \in \mathcal{H}_L$ *implies* $(C[h_1], \ldots, C[h_n]) \in \mathcal{H}_L$.
2. $(v_1, \ldots, v_n) \in \mathcal{V}_L$ *implies* $(E[v_1], \ldots, E[v_n]) \in \mathcal{H}_L$.
3. $(v_1, \ldots, v_n), (w_1, \ldots, w_n) \in \mathcal{V}_L$ *implies* $(v_1 w_1, \ldots, v_n w_n) \in \mathcal{V}_L$.
4. $(v_1, \ldots, v_n) \in \mathcal{V}_L, (h_1, \ldots, h_n) \in \mathcal{H}_L$ *implies* $(v_1 h_1, \ldots, v_n h_n) \in \mathcal{H}_L$.
5. $(v_1, \ldots, v_n) \in \mathcal{V}_L$ *implies* $(v_1^\infty, \ldots, v_n^\infty) \in \mathcal{H}_L$.
6. $(h_1, \ldots, h_n) \in \mathcal{H}_L$ *implies* $(c[\Box, h_1], \ldots, c[\Box, h_n]) \in \mathcal{V}_L$.
7. $(h_1, \ldots, h_n) \in \mathcal{H}_L$ *implies* $(c[h_1, \Box], \ldots, c[h_n, \Box]) \in \mathcal{V}_L$.

Proof. All properties are proved by composing strategies, we prove the first one. All other properties are proved similarly. Assume that $(h_1, \ldots, h_n) \in \mathcal{H}_L$, and consider some multicontext C (possibly with infinitely many ports). For all i let L_i be the set of trees that are Alternator's first move in some winning strategy for $\mathcal{H}(h_i, \ldots, h_n)$. Note that since $(h_1, \ldots, h_n) \in \mathcal{H}_L$, L_i is non empty for all i.

Claim 4.36. *For all $i \leq n$, for all trees t obtained by plugging trees of L_i in the ports of C, Alternator has a winning strategy in $\mathcal{H}(C[h_i], \ldots, C[h_n])$ such that the tree chosen in round 1 is t.*

Proof. We proceed by induction on i. For $i = n$ this is obvious. Assume the result holds for i and we prove it for $i - 1$. Let p be a prefix of t, for all subtree s plugged into a port of C, p yields some (possibly empty) prefix p_s of s. Since $s \in L_{i-1}$, p_s can be completed into a tree $s' \in L_i$. It follows that p can be completed into t' obtained by plugging trees of L_i in the ports of C. By induction hypothesis, Alternator has winning strategy in $\mathcal{H}(C[h_i], \ldots, C[h_n])$ such the tree chosen in round 1 is t'. Finally we conclude that Alternator has winning strategy in $\mathcal{H}(C[h_{i-1}], \ldots, C[h_n])$ such the tree chosen in round 1 is t. □

So our result follows from the Claim just proved. □

Definition 4.37. We define the *alternation* of a finite word to be its length, after iteratively eliminating letters that are identical to their predecessors. We say that a set of words has *unbounded alternation* if it contains words with arbitrarily large alternation. In the other case we say that a set has *bounded alternation*. A word is *alternating* if every two consecutive letters are distinct.

For example the alternation of *abaabbb* is 4. The notion of *alternation* gives us another characterization of the class $\bigcup_{n\in\omega} \mathcal{D}_n\big(\mathbf{\Sigma}^0_1\big)$.

Lemma 4.38. *For a regular language L of infinite trees, the following conditions are equivalent:*

1. *Alternator wins the game $\mathcal{H}^{\in,\notin}(L,n)$ for any n.*
2. *The set $\mathcal{H}_L$ has unbounded alternation.*

Proof. We have to prove both the implications.

$1 \Rightarrow 2$. We show that if Alternator wins the game $\mathcal{H}^{\in,\notin}(L,n)$, then $\mathcal{H}_L$ contains a word of length n where every two consecutive letters are different. Suppose that Alternator wins $\mathcal{H}^{\in,\notin}(L,n)$. Both L and L^c can be partitioned into tree types. By Lemma 4.2, Alternator wins $\mathcal{H}(h_1,\ldots,h_n)$ for some sequence of types, such that h_i is included in L or its complement, depending on the parity of i. In particular, the consecutive types are different.

$2 \Rightarrow 1$. Suppose that $\mathcal{H}_L$ has unbounded alternation. It is easy to check that $\mathcal{H}_L$ is closed under removing letters, so there must be some $g,h \in H_L$ such that $\mathcal{H}_L$ contains all the words

$$(g,h),(g,h,g,h),(g,h,g,h,g,h),\ldots$$

Since g and h are different elements of the syntactic algebra, it follows that there must be some multicontext C such that the tree type $C[g]$ is contained in L, while the tree type $C[h]$ is disjoint with L. By the first item of Lemma 4.35, we can conclude that $\mathcal{H}_L$ contains all the words

$$\big(C[g],C[h]\big),\big(C[g],C[h],C[g],C[h]\big),\big(C[g],C[h],C[g],C[h],C[g],C[h]\big),\ldots$$

It follows that Alternator can alternate arbitrarily long between the language L and its complement.

The proof is complete. □

Lemma 4.39. *If $\mathcal{V}_L$ has unbounded alternation then so does $\mathcal{H}_L$.*

Proof. Assume that $\mathcal{V}_L$ has unbounded alternation. Take $n > 0$, we will find a sequence in $\mathcal{H}_L$ of alternation at least n. Let

$$N \stackrel{\text{def}}{=} 2 \cdot n \cdot |V_L|^2$$

and let $(v_1, \ldots, v_N)$ be an alternating sequence in $\mathcal{V}_L$ of alternation N — such a sequence exists by the assumption and the fact that the language $\mathcal{V}_L$ is closed under removing letters. By Pigeon-hole Principle, there exist two types $v \neq v' \in V_L$ such that the word $(v, v')^n$ can be obtained from $(v_1, \ldots, v_N)$ by removing letters. Thus, $(v, v')^n \in \mathcal{V}_L$. By the fact that $v \neq v'$ and Fact 4.28 we know that one of the following two cases holds.

The first case is that there exists $h \in H_L$ such that $vh \neq v'h$. Clearly $(h, h, \ldots, h) \in \mathcal{H}_L$. By Item 4 from Lemma 4.35 we know that in that case $(vh, v'h, vh, v'h, \ldots, v'h) \in \mathcal{H}_L$ but as $vh \neq v'h$ the alternation of that sequence is at least n.

The second case is that there exists $u \in V_L$ such that $(vu)^\infty \neq (v'u)^\infty$. Similarly as before, Items 3 and 5 of Lemma 4.35 imply that both the sequences $(vu, v'u, \ldots, v'u)$ and $\big((vu)^\infty, (v'u)^\infty\big)^n$ belong to $\mathcal{V}_L$ and $\mathcal{H}_L$ respectively. As $(vu)^\infty \neq (v'u)^\infty$, the alternation of that element of $\mathcal{H}_L$ is at least n and the proof is concluded. □

4.4 Effective characterisation of $\mathsf{BC}(\mathbf{\Sigma}_1^0)$

In this section we state the crucial result of the chapter, providing an effective characterisation of the class $\bigcup_{n\in\omega} \mathcal{D}_n\big(\mathbf{\Sigma}_1^0\big)$. Notice that this class coincides with the class of all the Boolean combinations of open sets, denoted by $\mathsf{BC}(\mathbf{\Sigma}_1^0)$. Indeed, the difference between two open sets A and B can be written as a Boolean combination of open sets:

$$A \setminus B = (A \cap B)^c \cap A.$$

Vice-versa, the intersection and the complement can be written as differences of open sets: the complement of A^c is the difference between the whole space

X (that is always open) and A, and the intersection of two open sets A and B is

$$A \cap B = [(A \cup B) \setminus A] \setminus B,$$

(notice that $A \cup B$ is open, being a union of open sets). Hence, we have:

$$\bigcup_{n \in \omega} \mathcal{D}_n(\mathbf{\Sigma}^0_1) = \mathsf{BC}(\mathbf{\Sigma}^0_1).$$

We stress this equality because the class

$$\mathsf{BC}(\mathbf{\Sigma}^0_1)$$

is more intuitive than

$$\bigcup_{n \in \omega} \mathcal{D}_n(\mathbf{\Sigma}^0_1),$$

so from now on we can talk about Boolean combinations of open sets, instead of differences and our goal is to prove decidability of $\mathsf{BC}(\mathbf{\Sigma}^0_1)$. We have already given characterisations of $\mathsf{BC}(\mathbf{\Sigma}^0_1)$ that are not effective. Now we add a final characterisation, which uses identities and is effective.

Theorem 4.40. *For a regular language L of infinite trees, the following conditions are equivalent.*

1. *L is a Boolean combination of open sets, i.e. $L \in \mathsf{BC}(\mathbf{\Sigma}^0_1)$.*
2. *Constrainer wins the game $\mathcal{H}^{\in,\notin}(L, n)$ for all but finitely many n.*
3. *The set $\mathcal{H}_L$ has bounded alternation.*
4. *The following identities are satisfied in the algebra (H_L, V_L):*

$$u^\sharp u w^\sharp = u^\sharp v w^\sharp = u^\sharp w w^\sharp \qquad \text{if } (u, v, w) \in \mathcal{V}_L \text{ or } (w, v, u) \in \mathcal{V}_L \tag{4.3}$$

$$(u_2 w_2^\sharp v)^\sharp u_1 w_1^\infty = (u_2 w_2^\sharp v)^\infty \qquad \text{if } v \in V_L \text{ and } (u_1, u_2), (w_1, w_2) \in \mathcal{V}_L \tag{4.4}$$

Proof. We have already proved the implications $1 \Leftrightarrow 2 \Leftrightarrow 3$ respectively in Lemma 4.38 and Corollary 4.10. It remains to prove $3 \Leftrightarrow 4$. The direction $3 \Rightarrow 4$ is not hard and we prove it in the next section, whereas the direction $4 \Rightarrow 3$ forms the technical core of the chapter. □

Corollary 4.41. *The problem whether a regular language L belongs to $\mathsf{BC}(\mathbf{\Sigma}_1^0)$ is* EXPTIME*, when the language L is given as a parity non-deterministic automaton $\mathcal{A}$ recognising L.*

Proof. Given a representation of L, one can compute the algebra (H_L, V_L) in EXPTIME, see Fact 4.24. Then, verifying the equations from condition 4 of Theorem 4.40 can be done in time polynomial in the size of the algebra (which is exponential in the number of states of the given automaton for L). □

4.5 The implication $3 \Rightarrow 4$

In this section we prove the implication $3 \Rightarrow 4$ of Theorem 4.40.

Proposition 4.42. *If the set $\mathcal{H}_L$ has bounded alternation then the identities* (4.3) *and* (4.4) *of Theorem 4.40 are satisfied.*

We prove the contrapositive: if one of the identities (4.3) or (4.4) is violated then $\mathcal{H}_L$ has unbounded alternation.

The case when (4.3) **is violated.** The assumption that (4.3) is violated says that there are $u, v, w \in V_L$ such that

$$(u, v, w) \in \mathcal{V}_L \qquad \text{or} \qquad (w, v, u) \in \mathcal{V}_L,$$

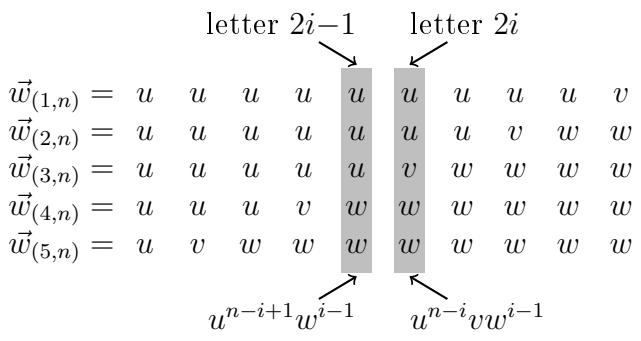
but the three context types $u^\sharp uw^\sharp$, $u^\sharp vw^\sharp$, and $u^\sharp ww^\sharp$ are not all equal. If the three context types are not equal then the second one must be different from either the first one or the third one. We only do the proof for the case when $(u, v, w) \in \mathcal{V}_L$ and when $u^\sharp uw^\sharp \neq u^\sharp vw^\sharp$; the other cases are entirely dual. For $n \geq 0$ and $i \in \{1, \ldots, n\}$, define

$$\vec{w}_{(i,n)} \stackrel{\text{def}}{=} \big(\overbrace{u, u, \ldots, u}^{2(n-i)+1},\ v,\ \overbrace{w, w, \ldots, w}^{2(i-1)}\big) \quad \in (V_L)^{2n}.$$

This word is obtained from (u, v, w) by duplicating some letters, and therefore it belongs to $\mathcal{V}_L$. For a given n, consider the words

$$\vec{w}_{(1,n)}, \ldots, \vec{w}_{(n,n)} \in \mathcal{V}_L.$$

These are n words of length $2n$. Let us multiply all these words coordinate-wise, yielding a word $\vec{w}_n$, also of length $2n$, which is depicted in the following picture:



As $\vec{w}_n$ is obtained by a coordinate-wise multiplication of the rows of the above matrix, and all these rows belong to $\mathcal{V}_L$, Lemma 4.35 implies that also $\vec{w}_n \in \mathcal{V}_L$.

Recall that $\sharp$ is a number dependant on V_L such that for every $z \in V_L$ we know that $z^\sharp$ is an idempotent. Choose some k, and take $n = k \cdot \sharp + 1$, and $i \in \{\sharp{+}1, 2\sharp{+}1, \ldots, (k-1)\cdot\sharp{+}1\}$. Consider the letters $2i-1$ and $2i$ in the word $\vec{w}_n$, which are

$$u^{n-i+1}w^{i-1} = u^\sharp u w^\sharp \qquad u^{n-i}vw^{i-1} = u^\sharp v w^\sharp.$$

By the assumption, these letters are different, and therefore the word $\vec{w}$ has alternation at least k. Because k was chosen arbitrarily, it follows that $\mathcal{V}_L$ has unbounded alternation. Lemma 4.39 implies that in that case also $\mathcal{H}_L$ has unbounded alternation.

The case when (4.4) **is violated.** The assumption that (4.4) is violated says that $\mathcal{V}_L$ contains pairs (u_1, u_2) and (w_1, w_2) such that for some $v \in V_L$,

$$e^\infty \neq e u_1 w_1^\infty \qquad \text{for } e \stackrel{\text{def}}{=} (u_2 w_2^\sharp v)^\sharp.$$

Let $h_1 = e^\infty$ and $h_2 = e u_1 w_1^\infty$, by the above assumption we know that $h_1 \neq h_2$. It turns out that a violation of Equation (4.4) has even stronger consequences than a violation of Equation (4.3), as expressed by the following claim.

Claim 4.43. *If Equation* (4.4) *is violated then Alternator wins the game* $\mathcal{H}^\infty(h_1, h_2, h_1, \ldots)$.

The above claim implies in particular that for every n the sequence

$$\big(h_1, h_2\big)^n$$

belongs to $\mathcal{H}_L$, and thus $\mathcal{H}_L$ has unbounded alternation.

Proof. Let C_{u_1}, C_{u_2} be contexts of types u_1, u_2 that witness that

$$(u_1, u_2), (w_1, w_2) \in \mathcal{V}$$

i.e. they are contexts played by Alternator in the first rounds of the respective games. Let C_{u_2}, C_{w_2}, and C_v be any three contexts of types u_2, w_2, and v respectively. Let $C_e \stackrel{\text{def}}{=} \big(C_{u_2} C_{w_2}^{\sharp} C_v\big)^{\sharp}$. The type of the context C_e is e.

The strategy of Alternator starts with the tree $t_1 = \big(C_e\big)^{\infty}$. We will demonstrate how it works for the first three rounds of the game, the rest is analogous.

Consider a prefix p_1 of the tree t_1 that is fixed by Constrainer. It must be the case that p_1 is a prefix of $\big(C_e\big)^{n_1}$ for some n_1. Thus, Alternator can now provide the tree

$$t_2 = \big(C_e\big)^{n_1} \cdot C_{u_1} \cdot C_{w_1}^{\infty}.$$

Now Constrainer chooses a prefix p_2 of the above tree, in fact p_2 is a prefix of $\big(C_e\big)^{n_1} \cdot C_{u_1} \cdot C_{w_1}^{\sharp \cdot n_2}$ for some n_2. Thus, by the assumptions on C_{u_1} and C_{u_2}, Alternator is able to provide a tree of the form

$$t_2 = \big(C_e\big)^{n_1} \cdot \big(D_{u_2} \cdot D_{w_2}^{\sharp \cdot n_2} \cdot C_v\big) \cdot \big(C_e\big)^{\infty},$$

where the contexts D_{u_2} and D_{w_2} depend on the prefix p_2 and have types u_2 and w_2 respectively. Now we proceed inductively as before, because any prefix p_3 of t_2 must be a prefix of $\big(C_e\big)^{n_1} \cdot \big(D_{u_2} \cdot D_{w_2}^{\sharp \cdot n_2} \cdot C_v\big) \cdot \big(C_e\big)^{n_3}$.

Notice that by the choice of e, the type of the context $D_{u_2} \cdot D_{w_2}^{\sharp \cdot n_2} \cdot C_v$ is e. Therefore, the trees t_i for odd i have type $e^{\infty} = h_1$ and the trees t_i for even i have type $eu_1 w_1^{\infty} = h_2$. □

Since (H_L, V_L) is a syntactic algebra, every two distinct types $h_1 \neq h_2$ can be distinguished with respect to the language, see Definition 4.18. Thus, by using the above strategy under some fixed multicontext C, we obtain the following corollary.

Corollary 4.44. *If Equation* (4.4) *is violated then Alternator wins* $\mathcal{H}^{\infty}(L)$.

4.6 The implication $4 \Rightarrow 3$ — case distinction

We now move to the implication $4 \Rightarrow 3$. To prove it we need to introduce a crucial concept witnessing large alternation of the set $\mathcal{H}_L$. The objects witnessing that will be called *strategy trees* and *locally optimal strategy trees*. Using these objects we will split the proof of that implication into two separate cases. We deal with these cases in Section 4.7 and Section 4.8.

4.6.1 Strategy trees

First, a *type-labelled tree* is a tree such that the nodes are labelled with tree types, i.e. it is a tree over the alphabet H_L.

Definition 4.45. If t is a tree over the alphabet A, the *type-labelled tree induced by* t is the type-labelled tree σ where the label of a node u is the tree type of the subtree $t.u$, i.e. $\sigma(u) = \alpha_L(t.u)$. In particular $\sigma(\varepsilon) = \alpha_L(t)$.

Definition 4.46. Let σ be a type-labelled tree and let t be a tree over A. We say that σ is *locally consistent* with t if for every node $u \in \{\mathtt{L}, \mathtt{R}\}^*$, whose label in t is a, we have that $\sigma(u)$ is the type obtained by applying the letter a to the pair of types $\sigma(u\mathtt{L})$ and $\sigma(u\mathtt{R})$, that is

$$\sigma(u) = a\big(\sigma(u\mathtt{L}), \Box\big) \cdot \sigma(u\mathtt{R}). \tag{4.5}$$

In other words, if we take a tree t_1 inside $\sigma(u\mathtt{L})$ and a tree t_2 inside $\sigma(u\mathtt{R})$ and we plug these two trees as the left and the right children of a, then the type of the result is $\sigma(u)$.

Remark 4.47. *The type-labelled tree induced by t is locally consistent with t.*

Example 4.48. Consider the language

$$L = \{t \in \mathrm{Tr}_A \mid t \text{ contains at least one } b\}$$

over the alphabet $A = \{a, b\}$. H_L contains two tree types that (treated as sets of trees) are L and L^c. Now consider a type-labelled tree σ where every node is labelled with L. It is easy to check that σ is locally consistent with every tree $t \in \mathrm{Tr}_A$, even with the tree that contains only letters a.

Fact 4.49. *The set of pairs*

$$\big\{(t,\sigma) \in \mathrm{Tr}_A \times \mathrm{Tr}_{H_L} \mid \sigma \text{ is locally consistent with } t\big\}$$

is closed in the product topology.

Proof. As Condition (4.5) speaks about finitely many values of σ and t, it corresponds to a clopen set of pairs. A type-labelled tree σ is locally consistent with a tree t if they obey the local consistency conditions from (4.5) in all the vertices. Thus, the above defined set of pairs is an intersection of a family of clopen sets. □

Lemma 4.50. *Let $(t_n)_{n\in\mathbb{N}}$ be a sequence of trees that converges to t_* and let $(\sigma_n)_{n\in\mathbb{N}}$ be a sequence of type-labelled trees that converges to σ_*. If σ_n is locally consistent with t_n for every n then σ_* is locally consistent with t_*.*

Proof. Follows directly from Fact 4.49. □

Now we are ready to define strategy trees, an important concept used in the rest of the chapter.

Definition 4.51. A *strategy tree* is a tuple $\sigma = (t, \sigma_1, \ldots, \sigma_n)$ where:

1. t is a tree over A, called the *support* of σ.
2. σ_1 is the type-labelled tree induced by t.
3. The type-labelled trees $\sigma_2, \ldots, \sigma_n$ are locally consistent with t.
4. For each node u of t, the sequence $\big(\sigma_2(u), \ldots, \sigma_n(u)\big)$ belongs to $\mathcal{H}_L$.

Notice that the fourth condition of Definition 4.51 does not mention the type-labelled tree σ_1. By the definition, σ can be interpreted as a single tree over the alphabet $A \times H_L^n$.

Intuitively speaking, a strategy tree represents a special kind of strategy for Alternator. In the first round, Alternator plays the support t of σ. However, Alternator also declares all the types that will appear in the nodes of t as the game progresses. More specifically, he declares that for every node u of t and round $k \in \{2, \ldots, n\}$, he has a strategy so that for the tree played in the round k, the subtree in the node u has type $\sigma_k(u)$.

Alternations. The number n is called the *duration* of a strategy tree $\sigma = (t, \sigma_1, \ldots, \sigma_n)$. We define the *root sequence* of a strategy tree to be the sequence of root labels of $\sigma_1, \ldots, \sigma_n$. If the duration is n, the root sequence is in H_L^n. We define the *root alternation* of a strategy tree σ to be the alternation of its root sequence. We define the *limit alternation* of σ to be the maximal number ℓ such that infinitely many subtrees of σ have root alternation at least ℓ. This means that if the limit alternation of σ is ℓ then there exist infinitely many nodes u such that their root sequence (i.e. the root sequence of the strategy tree obtained by truncating σ in the node u) has alternation at least ℓ.

Context zones. A *context zone* is a set X of nodes for which there exists a node u, called the *root* of X, and a node y, called the *port* of X, such that $u \prec y$ and X contains the nodes that are in the subtree of u, but not in the subtree of y:

$$X = \big\{x \in \{\mathtt{L}, \mathtt{R}\}^* \mid u \preceq x \wedge y \not\preceq x\big\}.$$

We say that context zones $X_1, \ldots, X_n$ are *consecutive* if for each $i \in \{1, \ldots, n-1\}$, the port of X_i is the root of X_{i+1}. The union of consecutive context zones is a context zone.

Consider a tree t, a type-labelled tree σ, and a context zone X. X can be seen as a context inside t taking into account the types of the subtrees of t as declared in σ. This is achieved by defining a value $\mathsf{val}(t, \sigma, X) \in V_L$. The definition of $\mathsf{val}(t, \sigma, X)$ is inductive on the number of nodes v in X such that $v \preceq y$, where y is the port of X. If there is only one such node then the port y of X is a child of its root u. Let a be the label of u in t and $v \in X$ be the other child of u. We set $\mathsf{val}(t, \sigma, X)$ as $a(\Box, \sigma(v))$ if v is the right child and $a(\sigma(v), \Box))$ if v is the left child. Otherwise, let u be the root of X, y its port and z the child of u such that $z \preceq y$. Let X_1 be the context zone with root u and port z and X_2 the context zone of root z and port y. We define

$$\mathsf{val}(t, \sigma, X) \stackrel{\text{def}}{=} \mathsf{val}(t, \sigma, X_1) \cdot \mathsf{val}(t, \sigma, X_2).$$

Figure 4.1 illustrates an example of a tree and a context zone with

$$\mathsf{val}(t, \sigma, X) = b(\Box, h_1) \cdot c(h_2, \Box) \in V_L.$$

In other words, $\mathsf{val}(t, \sigma, X)$ is the value of the context $C \stackrel{\text{def}}{=} t{\restriction}X$ when we assume that the subtrees of t aside of the branch leading to the port of C have types as declared by σ.

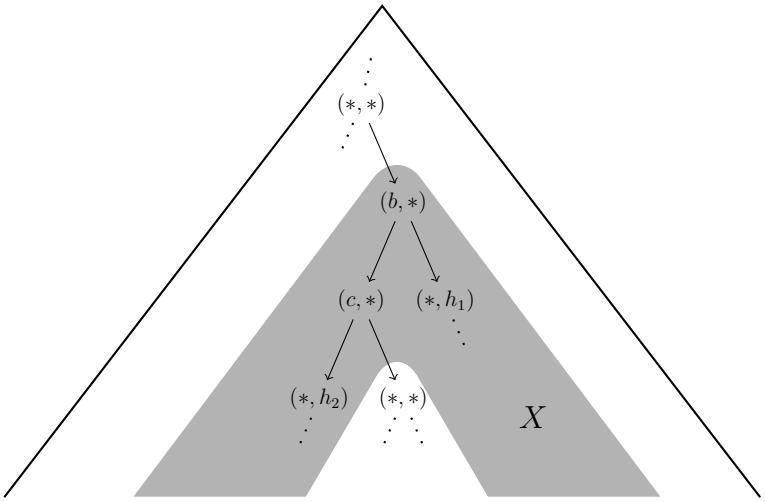


Figure 4.1: An illustration to the value $\mathsf{val}(t, \sigma, X)$. The gray area is the set X, a label (a, h) of a node u represents the values of $t(u) = a$ and $\sigma(u) = h$ respectively. The symbol $*$ indicates values that are irrelevant for the value of the set X.

Now assume that $\sigma = (t, \sigma_1, \ldots, \sigma_n)$ is a strategy tree. Let X be a context zone of σ (we can see σ as a single tree). For a round $i \in \{1, \ldots, n\}$, we define the type

$$\mathsf{val}(\sigma, X, i) \stackrel{\text{def}}{=} \mathsf{val}(t, \sigma_i, X) \in V_L.$$

Fact 4.52. *Let $\sigma = (t, \sigma_1, \ldots, \sigma_n)$ be a strategy tree and X be a context zone. Then*

$$\big(\mathsf{val}(\sigma, X, 1), \ldots, \mathsf{val}(\sigma, X, n)\big) \in \mathcal{V}_L.$$

Proof. This is by construction and Items 3, 6 and 7 of Lemma 4.35. □

4.6.2 Existence of strategy trees

We will now prove that whenever Alternator has a strategy in $\mathcal{H}$ then this fact is witnessed by a strategy tree. This fact is expressed by the following two lemmas. For inductive reasons these lemmas are parametrised by a finite multicontext C. The games $\mathcal{H}_C(h_1, \ldots, h_n)$ and $\mathcal{V}_C(h_1, \ldots, h_n)$ are defined as the games $\mathcal{H}_U(h_1, \ldots, h_n)$ and $\mathcal{V}_U(h_1, \ldots, h_n)$ respectively, where U is the open set of all trees (resp. contexts) that can be obtained from C, i.e. $C[*]$.

Lemma 4.53. *Let C be a finite multicontext for strategy trees. Consider a sequence of valuations $\eta_1, \ldots, \eta_n$ that map ports of C to H_L. If*

$$(\eta_1(u), \ldots, \eta_n(u)) \in \mathcal{H}_L$$

for every port u, then Alternator wins the game

$$\mathcal{H}_C(C[\eta_1], \ldots, C[\eta_n]).$$

Proof. For every port u of C, Alternator has a winning strategy for the game

$$\mathcal{H}(\eta_1(u), \ldots, \eta_n(u)).$$

We describe a winning strategy for Alternator in $\mathcal{H}_C(C[\eta_1], \ldots, C[\eta_n])$. Alternator starts with a tree obtained by plugging the initial tree for his winning strategy on $\mathcal{H}(\eta_1(u), \ldots, \eta_n(u))$ in every port u of C. Then every prefix of this tree yields a prefix for each subtree plugged into a port of C and Alternator can then use his strategies for the game $\mathcal{H}(\eta_1(u), \ldots, \eta_n(u))$ to answer. □

Lemma 4.54. *For a sequence $(h_1, \ldots, h_n) \in H_L^*$ and a finite multicontext C the following conditions are equivalent:*

1. *Alternator wins $\mathcal{H}_C(h_1, \ldots, h_n)$.*
2. *There is a strategy tree whose support extends C, and whose root sequence is $(h_1, \ldots, h_n)$.*

Proof. We prove the lemma by induction on n. The base case when $n = 0$ or $n = 1$, is trivial. We do the induction step. Let us begin with the easier implication from 2 to 1. Suppose that

$$\sigma \equiv (t, \sigma_1, \ldots, \sigma_n)$$

is a strategy tree as in item 2. Alternator's strategy is as follows. In the first round, he plays the tree t, which has type h_1. Suppose Constrainer chooses a prefix D of t. Consider the valuations $\eta_2, \ldots, \eta_n$ that map ports of D to H_L with $\eta_i(u) = \sigma_i(u)$ for any port u. For every $i \in \{2, \ldots, n\}$, the root label of σ_i is the same as $D[\eta_i]$, because σ_i is locally consistent with t and D is a prefix of t. By Lemma 4.53, Alternator wins the game

$$\mathcal{H}_D(D[\eta_2], \ldots, D[\eta_n]).$$

This shows that for every choice of D, Alternator has a winning strategy in the remaining part of the game.

Now we move to the more difficult implication from 1 to 2.

Suppose that Alternator wins $\mathcal{H}_C(h_1, \ldots, h_n)$. Let t be the tree of type h_1 that Alternator plays in the first round. This tree has prefix C. Also, for every finite prefix D of t, Alternator wins $\mathcal{H}_D(h_2, \ldots, h_n)$. By the induction assumption, for every finite prefix D of t, there is a strategy tree

$$\sigma_D = (t_D, \sigma_{D_2}, \ldots, \sigma_{D_n})$$

such that t_D has prefix D, and the root sequence of σ_D is $(h_2, \ldots, h_n)$.

A sequence of finite multicontexts $(D_i)_{i\in\omega}$ is said to converge to t if all of the multicontexts are prefixes of t, and for every $j \in \omega$, only finitely many multicontexts have some port at depth at most j. By compactness, there is an infinite sequence of finite multicontexts $(D_i)_{i\in\omega}$ which converges to the tree t and such that all of the sequences

$$(t_{D_i})_{i\in\omega} \qquad (\sigma_{D_i 2})_{i\in\omega} \qquad \ldots \qquad (\sigma_{D_i n})_{i\in\omega}$$

are convergent. Let the limits of these sequences be

$$t_* \qquad \sigma_{*2} \qquad \ldots \qquad \sigma_{*n}.$$

Because the sequence $(D_i)_{i\in\omega}$ converges to t, it follows that $t_* = t$. For each D, the type trees

$$(\sigma_{D2}, \ldots, \sigma_{Dn})$$

are locally consistent with t_D. Therefore, by Lemma 4.50 it follows that the limits $\sigma_{*2}, \ldots, \sigma_{*n}$ are locally consistent with t. Finally, define σ_{*1} to be the unique type tree that is globally consistent with t. We have just proved that

$$(\sigma_{*1}, \ldots, \sigma_{*n})$$

is a strategy tree. Because root values are preserved under limits, the root value of this strategy tree is the desired $(h_1, \ldots, h_n)$. □

4.6.3 Locally optimal strategy trees

To make the structure of a strategy tree more rigid, we will introduce a notion of a *locally optimal strategy tree*. In such a strategy tree, whenever $\sigma_i(u) \neq \sigma_{i+1}(u)$, there is some concrete reason for that fact.

Definition 4.55. Consider a strategy tree $(t, \sigma_1, \ldots, \sigma_n)$ and let $v \in V_L \sqcup \{1_L\}$. The strategy tree is called *locally optimal for* v if for every $i \in \{2, \ldots, n\}$ and for every type-labelled tree σ', if σ' is locally consistent with t and $v \cdot \sigma'(\varepsilon) = v \cdot \sigma_i(\varepsilon)$, then

$$\lambda(\sigma_{i-1}, \sigma_i) \leq \lambda(\sigma_{i-1}, \sigma'),$$

where λ is the discounted distance (defined in Section 2.2).

If σ is optimal for the context type 1_L of the trivial context $\Box$ then we just say that σ is *locally optimal*.

Fact 4.56. *If σ is locally optimal for $u \cdot v$ then σ is locally optimal for v as well.*

Lemma 4.57. *Consider a strategy tree σ with root sequence $(h_1, \ldots, h_n)$. Let $v \in V_L \sqcup \{1_L\}$ be a context type. Then there exists a locally optimal strategy tree σ' for v with root sequence $(h'_1, \ldots, h'_n)$ such that*

$$v \cdot (h_1, \ldots, h_n) = v \cdot (h'_1, \ldots, h'_n). \tag{4.6}$$

Proof. Take a strategy tree $\sigma = (t, \sigma_1, \ldots, \sigma_n)$. Consider the set Σ_2 of type-labelled trees σ'_2 that are:

- locally consistent with t,
- if h (resp. h') is the root value of σ_2 (resp. σ'_2) then $v \cdot h = v \cdot h'$.

Fact 4.49 implies that Σ_2 is a closed set. As a closed subset of the compact space of all trees, Σ_2 is compact. It follows that some elements of Σ_2 minimise the discounted distance with respect to σ_1. We choose such an element as the new σ_2 and we iterate this mechanism to build a locally optimal strategy tree $\bar{\sigma}$. This tree satisfies condition (4.6). If $v \neq 1_L$ then the root sequence of the new strategy tree might be different than the original root sequence. However, if $v = 1_L$ then the root sequence is not modified. $\Box$

Corollary 4.58. *If $(h_1, h_2, \ldots, h_n) \in \mathcal{H}_L$ then there exists a locally optimal strategy tree σ with root sequence $(h_1, \ldots, h_n)$.*

Using the above properties, without loss of generality we can restrict our attention to locally optimal strategy trees. One of the direct consequences of working with such strategy trees is expressed by the following lemma.

Lemma 4.59. *If σ is a locally optimal strategy tree, $u \preceq y$ two nodes of σ then for all $i = 1, \ldots, n$ we have*

$$\sigma_i(y) \neq \sigma_{i+1}(y) \Rightarrow \sigma_i(u) \neq \sigma_{i+1}(u).$$

Proof. Assume that $\sigma_i(y) \neq \sigma_{i+1}(y)$ and $\sigma_i(u) = \sigma_{i+1}(u)$. We show that this contradicts local optimality. Consider the type-labelled tree σ'_{i+1} defined as follows:

- for every z such that $u \preceq z$, $\sigma'_{i+1}(z) = \sigma_i(z)$,
- for all other nodes z, $\sigma'_{i+1}(z) = \sigma_{i+1}(z)$.

By the construction, σ'_{i+1} is locally consistent with t (this is by the definition for all nodes $z \neq u$, and because $\sigma_i(u) = \sigma_{i+1}(u)$). Note that by the definition, for all nodes z:

$$\mathsf{dist}(\sigma_i(z), \sigma'_{i+1}(z)) \leq \mathsf{dist}(\sigma_i(z), \sigma_{i+1}(z))$$

Moreover, $\mathsf{dist}(\sigma_i(y), \sigma'_{i+1}(y)) = 0$ and $\mathsf{dist}(\sigma_i(y), \sigma_{i+1}(y)) = 1$. Combining all this we obtain

- $\lambda(\sigma_i, \sigma'_{i+1}) < \lambda(\sigma_i, \sigma_{i+1})$.
- $\mathsf{val}(\sigma'_{i+1}) = \mathsf{val}(\sigma_{i+1})$.

This contradicts local optimality of σ. □

Corollary 4.60. *If σ is a locally optimal strategy tree and $u \preceq y$ are two nodes of σ then the root alternation of $\sigma.u$ is not smaller than the root alternation of $\sigma.y$.*

4.6.4 Case distinction

We can now split the proof of the implication $4 \Rightarrow 3$ of Theorem 4.40 into two subcases, as discussed below. Then we will treat these cases separately, as expressed by Propositions 4.61 and 4.62.

Assume for the sake of contradiction that Condition 3 of Theorem 4.40 is violated, what means that $\mathcal{H}_L$ has unbounded alternation. Thus, by Corollary 4.58 we know that the root alternation of the set of all locally optimal strategy trees for L (denoted Σ_L) is unbounded. Now consider the following two dual subcases:

(C1) Σ_L has unbounded root alternation and there exists a subset $\Sigma' \subseteq \Sigma_L$ that has unbounded root alternation but bounded limit alternation.

(C2) Σ_L has unbounded root alternation and every subset $\Sigma' \subseteq \Sigma_L$ with unbounded root alternation has also unbounded limit alternation.

The following two propositions consider these cases separately:

Proposition 4.61. *Assuming (C1), Equation* (4.3) *is violated.*

Proposition 4.62. *Assuming (C2), Equation* (4.4) *is violated.*

The proofs of these propositions are given in Sections 4.7 and 4.8. Notice that since either (C1) or (C2) must hold, both these propositions complete the proof of the implication $4 \Rightarrow 3$ of Theorem 4.40.

Roughly speaking, Case (C1) corresponds to the situation in which high alternation of $\mathcal{H}_L$ is achieved by modifications in the strategy trees that are spread inside their structure. On the opposite, Case (C2) corresponds to the situation in which the high alternation occurs on some infinite branch of the considered strategy trees.

4.7 Case (C1) — limit alternation is bounded

In this section we assume that $\Sigma' \subseteq \Sigma_L$ is a set of locally optimal strategy trees such that the root alternation of Σ' is unbounded but the limit alternation of Σ' is bounded. Under that assumption we prove that Equation (4.3) is violated. We do this in two steps, using a new object called *strategy matrix*

as an intermediary. Strategy matrices represent special strategies for Alternator in the game on tree types. In our first step, we show that if the root alternation of Σ' is unbounded but the limit alternation of Σ' is bounded then there exist special strategy matrices of arbitrarily large size. Finally we show that the existence of sufficiently large special strategy matrices violates Equation (4.3).

Definition 4.63. A *strategy matrix* is a rectangular matrix with entries from V_L such that every row belongs to $\mathcal{V}_L$. The *value* of a column of a strategy matrix is the value in V_L obtained by multiplying the entries in that column in V_L in the top-to-bottom order.

Definition 4.64. A strategy matrix M is called *parity alternating* if for some $n \in \omega$ it has $2n$ columns and n rows, and one of the following conditions holds (see Figure 4.2):

a) For every $i \in \{1, \ldots, n\}$,
   - Columns $2i-1$ and $2i$ have the same entries in all rows except for row i.
   - The values of columns $2i-1$ and $2i$ are different.

b) Condition a) holds when the order of columns is reversed.

If the case *a*) holds then the matrix is called *top-down* and if the case *b*) holds then it is called *bottom-up*. The set of parity alternating matrices with n rows and $2n$ columns is denoted by $\mathbb{P}_n$.

Figure 4.2 depicts a top-down parity alternating strategy matrix in $\mathbb{P}_6$. The picture presents columns $2i-1$ and $2i$ for $i = 3$, the entries of these columns agree everywhere except the third row, where the values x and y are distinct. The differences in all the other pairs of columns are indicated by gray rectangles, their values are not written down for the sake of clarity. It is important that the definition of a parity alternating strategy matrix requires the total values of the respective columns to differ, not only the entries in gray rectangles.

4.7.1 Constructing strategy matrices

As we have already said, we want to prove that under our assumptions we can build arbitrary large strategy matrices: the goal that we aim now, is to prove

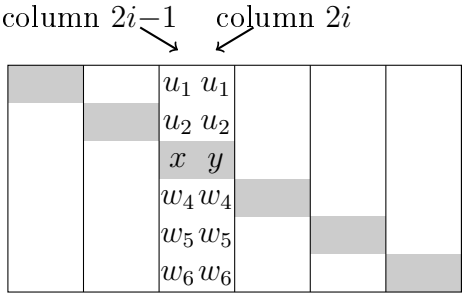


Figure 4.2: A matrix in $\mathbb{P}_6$.

that if Σ' has unbounded root alternation and bounded limit alternation, then $\mathbb{P}_n$ is non empty for every $n \in \omega$.

Fix some arbitrary $n \in \omega$. We want to construct a strategy matrix $M \in \mathbb{P}_n$. Let ℓ be the maximal limit alternation among the strategy trees in Σ'. Choose a strategy tree $\sigma \in \Sigma'$ with root alternation at least $\ell \cdot 2^{n^2}$ and let m be the duration of σ, i.e. $\sigma = (t, \sigma_1, \ldots, \sigma_m)$.

During the proof we will use a known combinatorial fact that we recall now.

Theorem 4.65 (Erdös-Szekeres Theorem, see [ES35])**.** *For given $r, s \in \mathbb{N}$, any sequence of length at least $(r-1)(s-1)+1$ contains a monotonically increasing subsequence of length r or a monotonically decreasing subsequence of length s.*

Lemma 4.66. *There are consecutive contexts zones $X_1, \ldots, X_n$ in σ and a sequence of rounds $i_1, \ldots, i_n \in \{1, \ldots, m-1\}$ such that the sequence of rounds is either strictly increasing or strictly decreasing, and*

$$\mathsf{val}(\sigma, i_j, X_j) \neq \mathsf{val}(\sigma, i_j{+}1, X_j) \qquad \text{for every } j \in \{1, \ldots, n\}$$

Proof. For a round $i \in \{1, \ldots, m-1\}$, define:

$$\mathsf{change}_i(\sigma) = \{x \in \{\mathtt{L}, \mathtt{R}\}^* \mid \sigma_i(x) \neq \sigma_{i+1}(x)\}.$$

Now we prove three different claims. Once these claims are proved, we easily finish the proof of the lemma.

Claim 1 Let X be a context zone, and let π be an infinite path that passes through the root of X, but not through the port. Then

$$\pi \subseteq \mathsf{change}_i(\sigma) \quad \Longrightarrow \quad \mathsf{val}(\sigma, i, X) \neq \mathsf{val}(\sigma, i{+}1, X)$$

holds for every round $i \in \{1, \ldots, m-1\}$.

Proof of Claim 1. This is by local optimality of σ. Assume that $\pi \subseteq \mathsf{change}_i(\sigma)$ and $\mathsf{val}(\sigma, i, X) = \mathsf{val}(\sigma, i+1, X)$ for some round i. We construct a type-labelled tree σ'_{i+1} that is closer to σ_i than σ_{i+1} regarding the discounted distance λ and with the same root value as σ_{i+1}, contradicting local optimality.

Consider the type-labelled tree σ'_{i+1} defined as follows:

- For nodes $x \in X$ $\sigma'_{i+1}(x) = \sigma_i(x)$. Hence, for $x \in X$

$$\mathsf{dist}(\sigma_i(x), \sigma'_{i+1}(x)) = 0 \leq \mathsf{dist}(\sigma_i(x), \sigma_{i+1}(x))$$

- For other nodes y we define $\sigma'_{i+1}(y) = \sigma_{i+1}(y)$. Hence

$$\mathsf{dist}(\sigma_i(y), \sigma'_{i+1}(y)) = \mathsf{dist}(\sigma_i(y), \sigma_{i+1}(y)).$$

Because $\pi \subseteq \mathsf{change}_i(\sigma)$, there exists at least one node $x \in X$ such that s

$$\mathsf{dist}(\sigma_i(x), \sigma_{i+1}(x)) = 1.$$

It follows that:

$$\lambda(\sigma_i, \sigma'_{i+1}) < \lambda(\sigma_i, \sigma_{i+1})$$

Moreover since $\mathsf{val}(\sigma, i, X) = \mathsf{val}(\sigma, i+1, X)$, we have $\sigma'_{i+1}(\varepsilon) = \sigma_{i+1}(\varepsilon)$. This contradicts local optimality of σ.

The proof of Claim 1 is complete. □

Claim 2 There is a set Π of at least 2^{n^2} distinct infinite paths, and a function $\mathsf{when}\colon \Pi \to \{1,\ldots,m-1\}$ such that for every $\pi \in \Pi$

$$\pi \subseteq \mathsf{change}_{\mathsf{when}(\pi)}(\sigma).$$

Proof of Claim 2. By consistency of a strategy tree, every node in $\mathsf{change}_j(\sigma)$ has at least one child in $\mathsf{change}_j(\sigma)$. Therefore, each of the non empty sets $\mathsf{change}_j(\sigma)$ contains at least one infinite path. By the assumption on the root alternation there are at least $\ell \cdot 2^{n^2}$ rounds j where $\mathsf{change}_j(\sigma)$ is non empty. By the assumption on limit alternation, an infinite path can be contained in sets $\mathsf{change}_j(\sigma)$ for at most ℓ different values of j. This proves the statement of Claim 2. □

Claim 3 Let Π be a set of 2^k distinct infinite paths in a binary tree. There exist consecutive context zones $X_1,\ldots,X_k$ and paths $\pi_1,\ldots,\pi_k \in \Pi$ such that for every $i \in \{1,\ldots,k\}$, the path π_i passes through the ports of the context zones $X_1,\ldots,X_{i-1}$, but not through the port of X_i.

Proof of Claim 3. The proof is by induction on k. The induction base of $k=1$ is obvious.

Now assume that the thesis holds for k and consider a set Π of 2^{k+1} paths.

Let u be the deepest node in the tree that belongs to all paths of Π (u exists since the root belongs to all paths of Π). Let $\Pi_{\mathtt{L}}$ (respectively, $\Pi_{\mathtt{R}}$) be those paths in Π that pass through the left child of u (respectively, the right child of u). One of the sets $\Pi_{\mathtt{L}}$ or $\Pi_{\mathtt{R}}$ must have at least half of the paths, i.e. at least 2^k paths. By symmetry, assume that $\Pi_{\mathtt{L}}$ has at least 2^k paths and let y be the left child of u. We apply the induction hypothesis to $\Pi_{\mathtt{L}}$ and obtain paths $\pi_2,\ldots,\pi_{k+1} \in \Pi_{\mathtt{L}}$ and consecutive context zones $X_2,\ldots,X_{k+1}$ such that for every $i \in \{2,\ldots,k+1\}$, path π_i passes through the ports of contexts $X_2,\ldots,X_{i-1}$, but not through the port of X_i.

We slightly modify X_2 by setting y as its root. Note that this does not affect the properties of the paths $\pi_2,\ldots,\pi_{k+1} \in \Pi_{\mathtt{L}}$. Now, we define X_1 as the context zone with the root u and the root of X_2 as the port. Let π_1 be some arbitrary path in $\Pi_{\mathtt{R}}$. By the definition, $X_1,\ldots,X_{k+1}$ are consecutive context zones. Moreover, the paths $\pi_2,\ldots,\pi_{k+1}$ are paths of $\Pi_{\mathtt{L}}$ and therefore pass through y, i.e. the port of $X_{\mathtt{L}}$. Finally, by the definition π_1 passes through

the right child of u and therefore not through the port of X_1. The proof of the third claim is complete. □

Now we can easily complete the proof of Lemma 4.66. Let Π and the function when be as defined in Claim 2. Apply Claim 3 to Π, yielding a sequence of paths, $\tau_1, \ldots, \tau_{n^2}$, and a sequence of context zones, $Y_1, \ldots, Y_{n^2}$, such that for every $i \in \{1, \ldots, n^2\}$, the path τ_i passes through the ports of context zones $Y_1, \ldots, Y_{i-1}$, but not through the port of Y_i. Consider the sequence

$$\mathsf{when}(\tau_1), \ldots, \mathsf{when}(\tau_{n^2}) \in \{1, \ldots, m-1\}.$$

By Erdös-Szekeres Theorem, we can find a sequence of indexes

$$j_1 < \ldots < j_n \in \{1, \ldots, n^2\}$$

such that the sequence

$$\mathsf{when}(\tau_{j_1}), \ldots, \mathsf{when}(\tau_{j_n})$$

is either increasing or decreasing. For $i \in \{1, \ldots, n\}$, define π_i to be τ_{j_i} and X_i to be the union of the context zones

$$Y_{j_i} \cup \ldots \cup Y_{j_{i+1}-1}.$$

By the construction, we know that the path π_i passes through the ports of the context zones $X_1, \ldots, X_{i-1}$, but not through the port of the context zone X_i. By Claim 1 we know that

$$\mathsf{val}\big(\sigma, \mathsf{when}(\pi_i), X_i\big) \neq \mathsf{val}\big(\sigma, \mathsf{when}(\pi_i) + 1, X_i\big).$$

Therefore, our proof is complete if we define $i_1, \ldots, i_n$ to be

$$\mathsf{when}(\pi_1), \ldots, \mathsf{when}(\pi_n).$$

This concludes the proof of Lemma 4.66. □

Definition 4.67. Let M be a matrix and consider a row j and a column i of M which is not the first column. We define a new column, denoted by

$$\text{almostcopy}_i^{\neq j}(M),$$

as follows: $\text{almostcopy}_i^{\neq j}(M)$ is equal to column i of M in all rows except for row j, where it is equal to column $i-1$ of M.

Definition 4.68. Let σ be a strategy tree of duration m and let $X_1, \ldots, X_n$ be consecutive context zones. We define a matrix with n rows and m columns as follows (for $i = 1, \ldots, m$ and $j = 1, \ldots, n$):

$$\mathrm{matrix}\big(\sigma, X_1, \ldots, X_n\big)\,[i,j] \stackrel{\mathrm{def}}{=} \mathsf{val}(\sigma, i, X_j).$$

Note that it follows from Fact 4.52 that every row of $\mathrm{matrix}(\sigma, X_1, \ldots, X_n)$ belongs to $\mathcal{V}_L$ and therefore it is a strategy matrix.

Lemma 4.69. *Let N be a strategy matrix defined by*

$$N = \mathrm{matrix}(\sigma, X_1, \ldots, X_n),$$

for some locally optimal strategy tree σ and consecutive context zones $X_1, \ldots, X_n$. Let j be a row and i a column, which is not the first column. If columns i and $i-1$ in N have different entries in row j then the value of the column

$$\mathrm{almostcopy}_i^{\neq j}(N)$$

is different from the value of column i in N.

Proof. This is a consequence of local optimality of σ. The proof is the same as Claim 2 in the proof of Lemma 4.66. □

Now we are ready to prove the first intermediary step: constructing a matrix in $\mathbb{P}_n$.

Proposition 4.70. *Under our assumptions about the strategy tree σ we can construct a matrix $M \in \mathbb{P}_n$.*

Proof. Let $X_1, \ldots, X_n$ and $i_1, \ldots, i_n$ be as in Lemma 4.66. Consider the strategy matrix $N = \mathrm{matrix}(\sigma, X_1, \ldots, X_n)$.

By Lemma 4.66 we know that for every $j \in \{1, \ldots, n\}$ the entries in row j are different in columns i_j and $i_j + 1$.

Suppose first that the sequence $i_1, \ldots, i_n$ is strictly increasing. Define a new matrix M, which has n rows and $2n$ columns as follows.

- For $j \in \{1, \ldots, n\}$, column $2j-1$ of M is $\mathrm{almostcopy}_{i_j+1}^{\neq j}(N)$.
- For $j \in \{1, \ldots, n\}$, column $2j$ of M is column $i_j + 1$ of N.

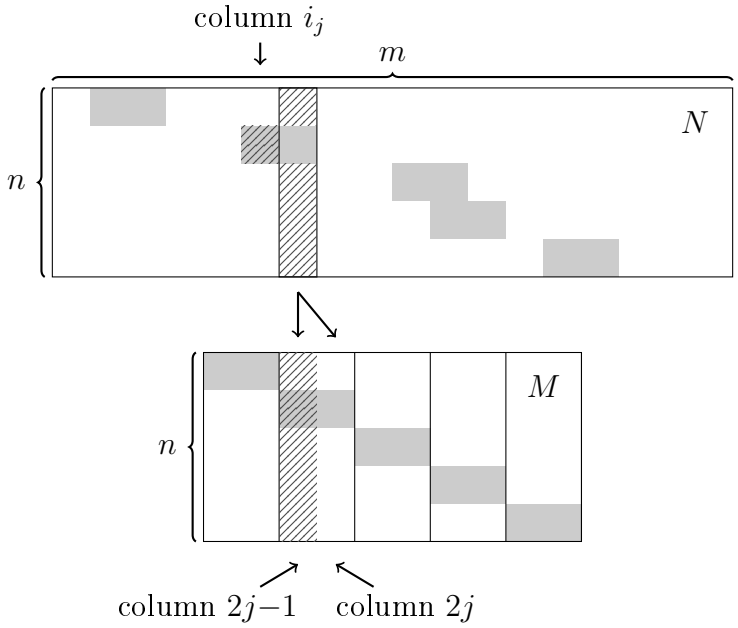


Figure 4.3: Construction of the matrix M from N. Column $2j-1$ of M equals almostcopy of column i_j+1 of N (i.e. the dashed part), while column $2j$ just equals column i_j+1 of N.

Figure 4.3 depicts the process of constructing the matrix M. Gray regions in the matrix N indicate pairs of distinct values in a row j of the consecutive columns i_j and i_j+1. We assume that the sequence i_j is strictly increasing (the other case leads to a bottom-up parity alternating strategy matrix).

Now we show that $M \in \mathbb{P}_n$. The dimensions of the matrix M are correct: it has n rows and $2n$ columns. We now show that M is a strategy matrix, which means that each row belongs to $\mathcal{V}_L$. For $j \in \{1,\ldots,n\}$, let us see how row j of M

$$M[j,1],\ldots,M[j,2n]$$

depends on row j of N

$$N[j,1],\ldots,N[j,m].$$

By reading the definition of M, we see that the dependency is

- When $k \neq j$, then $M[j,2k-1] = M[j,2k] = N[j,i_k+1]$.

- When $k = j$, then $M[j, 2k-1] = N[j, i_k]$ and $M[j, 2k] = N[j, i_k + 1]$.

It follows that row j of M is obtained from row j of N by eliminating some letters and duplicating some other letters. Since $\mathcal{V}_L$ is closed under eliminating and duplicating letters, and since N was a strategy matrix, it follows that also M is a strategy matrix.

By Lemma 4.69, for every $j \in \{1, \ldots, n\}$, the values of columns $2j-1$ and $2j$ in M are different. By the construction, columns $2j-1$ and $2j$ in M have the same entries, except for row j. So M is parity alternating.

When the sequence $i_1, \ldots, i_n$ is strictly decreasing, the matrix M is defined like for a strictly increasing sequence, except that the columns of M are filled in not from left to right, but from right to left. Formally speaking:

- For $j \in \{1, \ldots, n\}$, column $2(n-j+1)-1$ of M is $\mathrm{almostcopy}^{\neq j}_{i_j+1}(N)$.
- For $j \in \{1, \ldots, n\}$, column $2(n-j+1)$ of M is column $i_j + 1$ of N.

The proof that M belongs to $\mathbb{P}_n$ is the same as above. □

4.7.2 From strategy matrices to violation of (4.3)

So now we can do the second intermediary step. Let us assume that $\mathbb{P}_n$ is non empty for any n. By the symmetry we can assume that for every n there exists a top-down parity alternating matrix of size n (the other case is handled symmetrically): under this assumption we prove that Equation (4.3) is violated. To do that, we need to start from a parity alternating matrix of a large size, and by consecutive simplifications, construct a small matrix from the the violation can be easily extracted.

Take a strategy matrix M. Our aim is define a matrix obtained form M by *merging* two consecutive rows i and $i+1$ of M. Let

$$(v_{1,1}, \ldots, v_{1,n}), \ldots, (v_{k,1}, \ldots, v_{k,n}) \in V_L^n.$$

be the rows in M. The merge operation removes rows

$$(v_{i,1}, \ldots, v_{i,n}) \quad \text{and} \quad (v_{(i+1),1}, \ldots, v_{(i+1),n})$$

and replaces them by the row

$$(v_{i,1} \cdot v_{(i+1),1}, \ldots, v_{i,n} \cdot v_{(i+1),n}),$$

which is the product of the two removed rows in the monoid V_L^n. Clearly, the result of the merging operation is also a strategy matrix with the same values of all the columns.

Now we define four *safe rules*, i.e. rules that preserve parity alternating strategy matrices.

Lemma 4.71. *For every $M \in \mathbb{P}_n$ that is top-down and $i \in \{1, \ldots, n\}$, applying the following rules yields a parity alternating strategy matrix in $\mathbb{P}_{n-1}$:*

- *remove columns $2i-1$ and $2i$ and then merge row i with $i{+}1$;*
- *remove columns $2i-1$ and $2i$ and then merge row $i{-}1$ with i;*
- *remove the first two columns and remove the first row;*
- *remove the last two columns and remove the last row.*

The same holds for a bottom-up parity alternating matrices but then we count the columns in the reversed order.

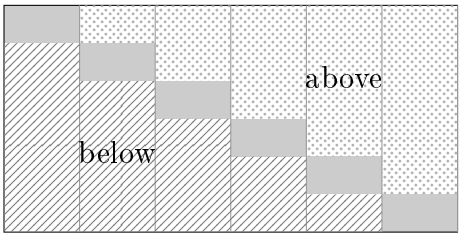
Proof. Immediate by the definition of the rules. □

Notice that $ux \neq uy$ implies that $x \neq y$, but $uvx \neq uvy$ does not imply that $ux \neq uy$, therefore we cannot safely remove columns $2i-1$ and $2i$ and remove row i except for $i = 1$ or $i = n$.

Consider a parity alternating strategy matrix $M \in \mathbb{P}_n$ and two indices $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, 2n\}$. We say that the entry $M[i, j]$ is *above diagonal* (resp. *below diagonal*) if:

- if M is top-down then $2i$ must be smaller than j (resp. $2i$ must be grater than $j + 1$),
- if M is bottom-up parity alternating then $2(n-i)$ must be greater than $j - 1$ (resp. $2(n - i)$ must be smaller than $j - 2$).

The following picture depicts the regions above and below the diagonal of a top-down parity alternating matrix in $\mathbb{P}_6$.



Definition 4.72. A parity alternating strategy matrix M is upper (resp. lower) *idempotent* if there exists an idempotent $e \in V_L$ such that every entry above (resp. below) the diagonal of M equals e. M is called just *idempotent* if it is both upper and lower idempotent (possibly with two distinct idempotents $e, e' \in V_L$).

Fact 4.73. *The safe rules preserve the fact that a given parity alternating matrix is upper (resp. lower) idempotent.*

Lemma 4.74. *For each $m \in \mathbb{N}$ there is some $n \in \mathbb{N}$ such that for any matrix in $\mathbb{P}_n$ there is a sequence of safe rules that yields a matrix $N \in \mathbb{P}_m$ that is upper (resp. lower) idempotent.*

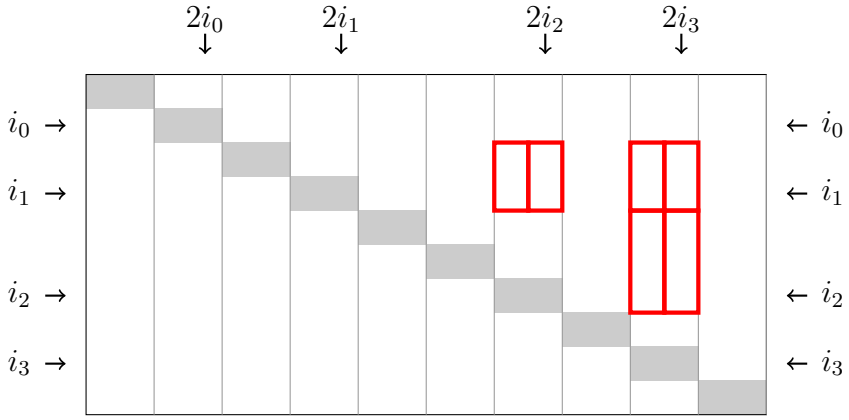
Proof. By the symmetry we will only deal with the upper idempotent case. The proof uses Ramsey Theorem for hypergraphs with edges of size 3. This theorem says that for every $m \in \mathbb{N}$ there exists a number $f(m)$ such that for any complete hypergraph with edges coloured over V_L, there exists a complete sub-hypergraph of size m in which all edges share the same colour. We choose $n = f(m+1)$.

Fix a matrix $M \in \mathbb{P}_n$. Again by the symmetry we assume that M is a top-down parity alternating matrix. Consider the hypergraph where the nodes are $\{1, \ldots, n\}$ and an edge $\{i < j < k\}$ is coloured by the value obtained by multiplying, in the monoid V_L, the entries that appear in rows $i+1, \ldots, j$ of column $2k$. By the choice of n, we can apply Ramsey Theorem to this colouring and get a subset of size $m+1$:

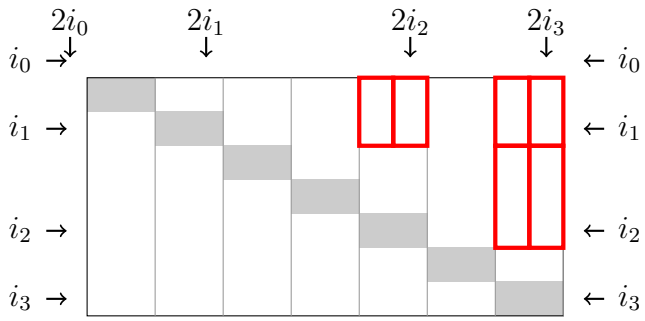
$$I = \{i_0 < i_1 < \ldots < i_m\} \subseteq \{1, \ldots, n\}$$

such that all the hyperedges on I have the same colour, say $e \in V_L$. This colour must be an idempotent, i.e. it must satisfy $e = ee$. This situation is illustrated below, with a matrix in $\mathbb{P}_{10}$. The multiplication in V_L of the entries in the regions marked by red rectangles always gives the idempotent

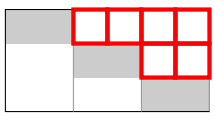
e. The regions come in pairs, for columns $2i_\ell$ and $2i_\ell - 1$ but as the matrix is parity alternating, the entries in these columns agree (except row i_ℓ).



Now we apply a sequence of safe rules. First, we remove the first i_0 rows and first $2i_0$ columns. Then we remove the last $n - i_m$ rows and last $2(n - i_m)$ columns. We assume that the indices of rows and columns are shifted accordingly to the operations we perform; i.e. now $i_\ell := i_\ell - i_0$ for $\ell = 0, \ldots, m$. After these operations, the matrix gets the following shape.



Now, for $\ell = 1, 2, \ldots, m$ we remove columns $2i_{\ell-1} + 1, \ldots, 2i_\ell - 2$ and merge rows $i_{\ell-1} + 1, \ldots, i_\ell$ into one row (with value e in columns $2i_{\ell+1}, \ldots, 2i_m$). This way, we get the following matrix, which is upper idempotent.



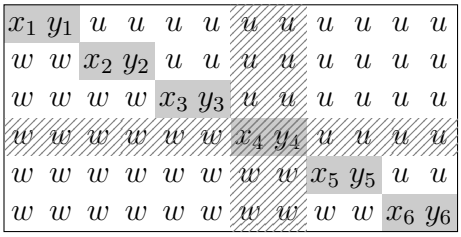
□

Corollary 4.75. *For each $m \in \mathbb{N}$ there is some $n \in \mathbb{N}$ such that for any matrix in $\mathbb{P}_n$ there is a sequence of safe rules that yields a matrix $N \in \mathbb{P}_m$ that is idempotent.*

Proof. Since the safe rules preserve the fact that a matrix is upper (resp. lower) idempotent, we can apply Lemma 4.74 twice, once to get an upper idempotent matrix and then to make it also lower idempotent. □

Fact 4.76. *If M is an idempotent parity alternating matrix, the the operation that removes columns $2i$ and $2i-1$ and removes row i preserves the fact that the matrix is idempotent parity alternating.*

Proof. If $i = 1$ or $i = n$ then the above operation is safe. For $1 < i < n$ we use the fact that the matrix is idempotent, so the values of columns $2i-1$ and $2i$ depend only on the unique entry in that column which is neither above nor below the diagonal. The following picture depicts an idempotent parity alternating matrix in $\mathbb{P}_6$ where the upper idempotent is u and the lower idempotent is w. Removing row 4 and columns 7 and 8 of that matrix preserves the fact that the matrix is idempotent parity alternating.



□

4.8 Case (C2) — limit alternation is unbounded

In this section we assume that Σ_L has unbounded root alternation and every subset $\Sigma' \subseteq \Sigma_L$ with unbounded root alternation has also unbounded limit alternation. Under that assumption we need to prove that Equation (4.4) is violated.

We start by defining a notion of a *strategy graph* that represents very specific strategies of Alternation in the game on types. We then show that the above assumptions imply that the strategy graph needs to be *recursive*

— roughly speaking it contains complicated connected components. Then we finish the proof of Proposition 4.62 by showing that recursivity of the strategy graph gives rise to a violation of Equation (4.4).

4.8.1 Strategy graph

Given a language L, the *strategy graph of* L (denoted G_L) is defined as follows. The set of nodes of G_L is $\big(V_L \sqcup \{1_L\}\big) \times H_L$. There is an edge from a node (v, h) to a node (v', h') if there exist:

$$(u_1, u_2,), (w_1, w_2) \in \mathcal{V}_L, \ z \in V_L \sqcup \{1_L\}$$

such that

$$h = v u_1 w_1^{\infty} \quad \text{and} \quad v' = v u_2 w_2^{\sharp} z.$$

Notice that the above definition does not invoke h' (the value of h' matters for a successive edge from (v', h')).

The following lemma provides a less explicit definition of edges in G_L.

Lemma 4.77. *There exists an edge from (v, h) to (v', h') in G_L if and only if the following condition is satisfied:*

There exists a tree $t \in \mathrm{Tr}_A$ such that $v \cdot \alpha_L(t) = h$ and there exists an infinite path π of t such that if D is a finite prefix of t then D can be completed into a context C' with the port located on π such that $v \cdot \alpha_L(C') \cdot z = v'$ for some $z \in V_L \sqcup \{1_L\}$.

Proof. First assume that there exists an edge from (v, h) to (v', h') in G_L with $(u_1, u_2), (w_1, w_2), z$ witnessing that. Let C_u, C_w, C_z be contexts of types u_1, w_1, z respectively. Take $t \stackrel{\text{def}}{=} C_u \cdot C_w^{\infty}$. In that case $v \cdot \alpha_L(t) = h$ and let π be the infinite path that contains all the ports of the contexts $C_u \cdot C_w^k$ for $k = 0, 1, \ldots$

Consider a finite prefix D of t. Let D' be an extension of D that is also finite, contains exactly one port, and D' is a prefix of $C_u \cdot C_w^{(\sharp \cdot k)}$ for some k.

By the definition, D' can be written as

$$D = D_0 \cdot D_1 \cdot \ldots D_{\sharp \cdot k},$$

where $D_0, \ldots, D_{\sharp \cdot k}$ are finite prefixes of contexts (i.e. each of them has exactly one port); and moreover D_0 is a prefix of C_u and all $D_1, \ldots, D_{\sharp \cdot k}$ are prefixes of C_w. By the assumption that $(u_1, u_2), (w_1, w_2) \in \mathcal{V}_L$ we know that we can extend all $D_0, D_1, \ldots, D_{\sharp \cdot k}$ into contexts $C_0, C_1, \ldots, C_{\sharp \cdot k}$ such that $\alpha_L(C_0) = u_2$ and $\alpha_L(C_i) = w_2$ for $i = 1, \ldots, \sharp \cdot k$. Take $C' = C_0 \cdot C_1 \cdot \ldots \cdot C_{\sharp \cdot k}$. Clearly the port of C' is located on π. Notice that

$$\alpha_L(C') = u_2 \cdot w_2^{\sharp \cdot k} = u_2 \cdot w_2^{\sharp}.$$

Therefore, $v \cdot \alpha_L(C') \cdot z = v'$ by the assumption that there is an edge from (v, h) to (v', h').

Now consider the more involved direction: we assume that (v, h), (v', h') satisfy the condition from Lemma 4.77 and we need to prove that there is an edge between them in G_L. This will be achieved by the Ramsey theorem. Consider a triple of nodes $x \prec y \prec \delta$ on π. Let $d = |\delta|$ be the depth of δ and let D_d be the finite prefix of t up to the depth d (i.e. D_d is t restricted to the subset $\{\mathtt{L}, \mathtt{R}\}^{\leq d}$ of its domain). Let us fix C' given from the assumption in Lemma 4.77.

Let u_1, u_2, w_1, w_2 be the α_L-types of the context zones:

- u_1 (resp. u_2) the type of the context zone rooted in ε with the port in x of t (resp. of C');
- w_1 (resp. w_2) the type of the context zone rooted in x with the port in y of t (resp. of C').

Let z be the type given by the assumption for C'.

Define $f(x, y, \delta)$ as the quintuple $(u_1, u_2, w_1, w_2, z) \in V_L^5$. Apply the Ramsey theorem for triples to f to get an infinite set X of nodes on π. We know that for all the triples from X the function f is constantly equal a fixed quintuple $(u_1, u_2, w_1, w_2, z) \in V_L^5$. It is easy to see that $\alpha_L(t) = u_1 \cdot w_1^{\infty}$. Thus, $h = v u_1 w_1^{\infty}$. Similarly, the Ramsey theorem guarantees that $w_2 \cdot w_2 = w_2$ and therefore $v' = v u_2 w_2^{\sharp} z$. Moreover, for fixed $x \prec y$ from X there are infinitely many $\delta \in X$ and therefore we know that (u_1, u_2) and (w_1, w_2) are both elements of $\mathcal{V}_L$. Thus, we have proved that there is an edge from (v, h) to (v', h') in G_L. □

Lemma 4.78. *The strategy graph G_L is transitive.*

Proof. Consider an edge between (v,h) and (v',h') witnessed by (u_1,u_2), (w_1,w_2), and z; and an edge between (v',h') and (v'',h'') witnessed by (u_1',u_2'), (w_1',w_2'), z'. Then $h = vu_1w_1^\infty$ and $v'' = vu_2w_2^\sharp\big(zu_2'(w_2')^\sharp z'\big)$. It means that there is an edge between (v,h) and (v'',h'') with $(u_1,u_2),(w_1,w_2) \in \mathcal{V}_L$, and $z'' \stackrel{\text{def}}{=} zu_2'(w_2')^\sharp z'$. □

Definition 4.79 (*Recursive strategy graphs*)**.** We say that a strategy graph G_L is *recursive* if there exists a strongly connected component of G_L that contains two nodes (v,h), (v',h') with $h \neq h'$.

Lemma 4.80. *If the strategy graph G_L is recursive then Equation* (4.4) *is violated.*

Proof. Assume contrarily that G_L is recursive but Equation (4.4) holds. Consider a pair of nodes (v,h) and (v',h') with $h \neq h'$ that witness recursivity of G_L. By Lemma 4.78 there must exist edges in G_L from (v,h) to (v',h') and back. Let (u_1,u_2), (w_1,w_2), z; and (u_1',u_2'), (w_1',w_2'), z' witness the existence of these edges respectively.

Apply Equation (4.4) to obtain that:

$$\big(u_2(w_2)^\sharp\ zu_2'(w_2')^\sharp z'\big)^\sharp\ u_1(w_1)^\infty = \big(u_2(w_2)^\sharp\ zu_2'(w_2')^\sharp z'\big)^\infty$$
$$\big(u_2'(w_2')^\sharp\ z'u_2(w_2)^\sharp z\big)^\sharp\ u_1'(w_1')^\infty = \big(u_2'(w_2')^\sharp\ z'u_2(w_2)^\sharp z\big)^\infty$$

Let $W = u_2(w_2)^\sharp z$ and $W' = u_2'(w_2')^\sharp z'$. Then by the assumptions on the edges between (v,h) and (v',h') we get that $vW = v'$ and $v'W' = v$. Moreover, the above equations get the form

$$\big(WW'\big)^\sharp u_1(w_1)^\infty = \big(WW'\big)^\infty$$
$$\big(W'W\big)^\sharp u_2(w_2)^\infty = \big(W'W\big)^\infty$$

And therefore by using the values of h, h' and multiplying these equations by v and v' respectively we get:

$$h = v\cdot u_1(w_1)^\infty = v\cdot\big(WW'\big)^\sharp\ u_1(w_1)^\infty = v\cdot\big(WW'\big)^\infty$$
$$h' = v'\cdot u_1'(w_1')^\infty = v'\cdot\big(W'W\big)^\sharp\ u_1'(w_1')^\infty = v'\cdot\big(W'W\big)^\infty$$

Now since $h = v\cdot\big(WW'\big)^\infty = vW\cdot\big(W'W\big)^\infty = v'\cdot\big(W'W\big)^\infty = h'$ we obtain the contradiction as we assumed that $h \neq h'$. □

4.8.2 Constructing a path in G_L

We will now use the assumption that (C2) holds to construct an infinite path in G_L such that every two consecutive nodes on that path (v, h), (v', h') satisfy $h \neq h'$. Since G_L is finite, such an infinite path witnesses that some strongly connected components of G_L contains such two nodes and therefore G_L is recursive. By Lemma 4.80 it means a violation of Equation (4.4) and the proof of Proposition 4.62 is finished.

The construction of the path will be inductive, preserving an invariant that the last node constructed on the path is *alternating*: a node (v, h) in G_L is *alternating* if $v \cdot \mathcal{H}_L$ contains words that begin with h and have arbitrarily high alternation.

Lemma 4.81. *G_L contains at least one alternating node.*

Proof. This is because $\mathcal{H}_L$ has unbounded alternation. Therefore, by Pigeonhole Principle there exists some $h \in H_L$ such that $\mathcal{H}_L$ contains words that begin with h that have arbitrarily high alternation. By the definition, this means that the node $(1_L, h)$ is alternating in G_L. □

The inductive step of the construction will be based on the following lemma.

Lemma 4.82. *If (v, h) is an alternating node of G_L and (C2) holds then there exists an edge in G_L from (v, h) to (v', h') such that $h \neq h'$ and (v', h') is also alternating.*

The rest of the section is devoted to a proof of Lemma 4.82. Let $\Sigma_{(v,h)}$ be the set of all strategy trees σ (with root sequences of the form $(h_1, \ldots, h_n)$) such that:

- σ is locally optimal for v (see Definition 4.55),
- $v \cdot h_1 = h$,
- The sequence $(vh_1, \ldots, vh_n)$ is alternating (i.e. every two consecutive values in the sequence are distinct, see Definition 4.37).

Fact 4.83. *The set $\Sigma_{(v,h)}$ is a subset of Σ_L that has unbounded root alternation and by (C2) also unbounded limit alternation.*

Proof. Fact 4.56 implies that all strategy trees in $\Sigma_{(v,h)}$ belong also to Σ_L. By the fact that (v,h) is alternating and by Lemma 4.57 we know that $\Sigma_{(v,h)}$ has unbounded root alternation. Therefore, (C2) guarantees that $\Sigma_{(v,h)}$ has also unbounded limit alternation. □

Now observe that for each $g, g' \in H_L$ either $(g,g')^k \in \mathcal{H}_L$ for all $k \in \omega$, or there exists a unique $k_{g,g'} \in \omega$ such that $(g,g')^{k_{g,g'}} \in \mathcal{H}_L$ but $(g,g')^{1+k_{g,g'}} \notin \mathcal{H}_L$. Since the set H_L is finite, there exists a number K that is greater than all the defined numbers $k_{g,g'}$. Thus, the following fact holds.

Fact 4.84. *If for some pair $g, g' \in H_L$ we have $(g,g')^K \in \mathcal{H}_L$ then $(g,g')^k \in \mathcal{H}_L$ for all $k \in \omega$.*

Let $\ell \stackrel{\text{def}}{=} |H_L|^2 \cdot K$ and let $\sigma = (t, \sigma_1, \ldots, \sigma_n)$ be a strategy tree in $\Sigma_{(v,h)}$ that has limit alternation greater than ℓ.

We will now construct a path π in t on which the high limit alternation of σ is located. Let Z be the set of nodes z of t such that the root alternation of $\sigma.z$ is greater than ℓ. By Corollary 4.60 we know that Z is prefix closed and by the fact that the limit alternation of σ is greater than ℓ we know that Z is infinite. Therefore, by König lemma we know that Z contains an infinite path π.

Lemma 4.85. *There exists an infinite set $X \subseteq \pi$ and two sequences $(h_1, \ldots, h_n) \in H_L^*$ and $(v_1, \ldots, v_n) \in V_L^*$ such that for all nodes $x \in X$:*

- $\big(\sigma_1(x), \ldots, \sigma_n(x)\big) = (h_1, \ldots, h_n)$,
- $\big(\mathsf{val}(\sigma, X, 1), \ldots, \mathsf{val}(\sigma, X, n)\big) = (v_1, \ldots, v_n)$, *where X is the context zone with the root in ε and port in x.*

Moreover, $(h_1, \ldots, h_n) \in \mathcal{H}_L$ and $(v_1, \ldots, v_n) \in \mathcal{V}_L$.

Proof. The choice of X and the sequences $(h_1, \ldots, h_n) \in H_L^*$, $(v_1, \ldots, v_n) \in V_L^*$ is just by Pigeon-hole Principle. By the definition of a strategy tree we know that $(h_1, \ldots, h_n) \in \mathcal{H}_L$. By Fact 4.52 we know that also $(v_1, \ldots, v_n) \in \mathcal{V}_L$. □

Since the alternation of $(h_1, \ldots, h_n)$ is greater than $\ell = |H_L|^2 \cdot K$, we know that there exists a pair $g \neq g' \in H_L$ and a set of indices $I \subseteq \{1, \ldots, n-1\}$ such that $|I| \geq K$ and for all $i \in I$ we have $h_i = g$ and $h_{i+1} = g'$. Fix as i_0

the minimal element of I. By closure of $\mathcal{H}_L$ under subwords, we know that $(g, g')^K \in \mathcal{H}_L$. By Fact 4.84 it implies that

$$(g, g')^k \in \mathcal{H}_L \qquad \text{for all } k \in \omega. \tag{4.7}$$

Let $u = v_{i_0}$ and $u' = v_{i_0+1}$ where i_0 is the minimal element of I. We will finish the proof by showing the following three lemmas.

Lemma 4.86. *The values $vu'g$ and $vu'g'$ are distinct.*

Proof. Recall that $g = h_{i_0}$ and $g' = h_{i_0+1}$. Assume that the value $vu'g$ is equal to $vu'g'$. Consider a strategy tree σ' that equals σ except for the subtree $\sigma'_{i_0+1}.x$ where x is the $\preceq$-minimal element of X. Over that subtree, let σ'_{i_0+1} be equal to σ_{i_0}. Let $(h'_1, \ldots, h'_n)$ be the root sequence of σ'. We know that $v \cdot (h'_1, \ldots, h'_n) = (h_1, \ldots, h_n)$ and $\mathsf{dist}(\sigma_{i_0}, \sigma'_{i_0+1})$ is strictly smaller than $\mathsf{dist}(\sigma_{i_0}, \sigma_{i_0+1})$, what contradicts local minimality of σ for v. □

Lemma 4.87. *The nodes $(vu', vu'g)$ and $(vu', vu'g')$ are both alternating in G_L.*

Proof. By (4.7) and Fact 4.35 we know that $(vu'g, vu'g')^k \in \mathcal{H}_L$ for all $k \in \omega$. Moreover, Lemma 4.86 says that $vu'g \neq vu'g'$. □

Now it remains to prove the following lemma.

Lemma 4.88. *There exist edges in G_L from (v, h) to both $(vu', vu'g)$ and $(vu', vu'g')$.*

Proof. Since the existence of an edge from (v, h) to (v', h') does not depend on h', it is enough to show an edge from (v, h) to $(vu', vu'g)$. This will be done using Lemma 4.77. Let σ' be the strategy tree obtained from σ by removing the first i_0 rounds: $\sigma' = (t, \sigma_{i_0+1}, \sigma_{i_0+2}, \ldots, \sigma_n)$ (i_0 is the minimal element of I).

Consider the tree t from the strategy tree σ and the path π. By the definition of σ we know that $v \cdot \alpha_L(t) = h$. Let D be a finite prefix of t. The strategy tree σ' witnesses that D can be extended into a context C' with a port located on π such that that $\alpha_L(C') = v_{i_0+1} = u'$. Therefore $v \cdot \alpha_L(C') \cdot 1_{V_L} = vu'$.

This way we have proved the property in Lemma 4.77 and therefore there must be an edge in G_L from (v, h) to $(vu', vu'g)$. □

Thus, Lemma 4.86 implies that at least one of the nodes $(vu', vu'g)$ and $(vu', vu'g')$ has the second coordinate distinct than h, Lemma 4.87 implies that this node is alternating, and Lemma 4.88 implies that G_L contains an edge from (v, h) to that node. Thus, the proof of Lemma 4.82 is concluded.

Lemma 4.89. *If $\mathbb{P}_n$ is non empty for arbitrarily big n then there are $u, w, x, y \in V_L$ such that u and w are idempotents and $\mathbb{P}_4$ contains the matrix*

x	y	u	u	u	u	u	u
w	w	x	y	u	u	u	u
w	w	w	w	x	y	u	u
w	w	w	w	w	w	x	y

Proof. By the hypothesis, we can apply Corollary 4.75 for $m = 4 * |V_L|^2$. Let N be the resulting matrix. Each row of that matrix is of the form $w \cdots w \ \cdot x_i \cdot y_i \cdot \ u \cdots u$. Thus, there exists a pair (x, y) that appears as (x_i, y_i) in at least 4 rows. Using Fact 4.76 we can remove all the remaining rows (and the corresponding pairs of columns) from N and the result has the above form. □

We can conclude the proof of Proposition 4.61 by showing that under our assumptions Equation (4.3) is violated.

Let M be the matrix described in Lemma 4.89. Let $\{u_1, \ldots, u_8\}$ be the values of all the columns in M. The matrix is in $\mathbb{P}_4$ so $u_3 \neq u_4$. Because u and w are idempotent,

$$u_3 = uxww = uxw = uuxw = u_5$$

For the same reason, $u_4 = u_6$. This is depicted in the picture below.

x	y	u	u	u	u	u	u
w	w	x	y	u	u	u	u
w	w	w	w	x	y	u	u
w	w	w	w	w	w	x	y
↑	↑	↑	↑				
u_3	u_4	u_3	u_4				

Since u_3 and u_4 are different, at least one of them is different than uw. Without loss of generality suppose that $u_3 \neq uw$. Because each row of the matrix belongs to $\mathcal{V}_L$ and $\mathcal{V}_L$ is closed under removing letters, it follows that

$$(w, x, u) \in \mathcal{V}_L.$$

This means that we have a violation of Equation (4.3), which requires that

$$uw = u^\omega \cdot u \cdot w^\omega = u^\omega \cdot x \cdot w^\omega = u_3.$$

This concludes the proof of Proposition 4.61.

4.9 The class $\mathbf{\Delta}^0_2$

This final section is devoted to the effective characterisation of the Borel class $\mathbf{\Delta}^0_2$ (that corresponds to the union of the first ω_1 Wadge degrees). Decidability of the class $\mathbf{\Delta}^0_2$ can be actually obtained as a direct corollary of [CMS17] (that we present in the next chapter of the thesis): in this work the authors proved that it is decidable if a regular tree language is in the Borel class $\mathbf{\Pi}^0_2$; since regular languages are closed under complement and $\mathbf{\Delta}^0_2 = \mathbf{\Pi}^0_2 \cap \mathbf{\Sigma}^0_2$ we automatically obtain that it is decidable if a regular tree language is in $\mathbf{\Delta}^0_2$. However, here we show that the algebraic approach developed in the previous sections covers the case of the Borel class $\mathbf{\Delta}^0_2$ as well.

We split this section into three subsections. In the first one we give a non effective characterisation of the class $\mathbf{\Delta}^0_2$ and in the other ones we present the effective characterisation.

4.9.1 The infinite variant of the game

We denote by $\mathcal{H}^\infty(L)$ the infinite variant of $\mathcal{H}^{\in,\notin}(L,n)$: $\mathcal{H}^\infty(L)$ is the infinite duration game played the same way as $\mathcal{H}^{\in,\notin}(L,n)$ but the winning condition for Alternator is that he has to survive for infinitely many turns. By $\mathcal{H}^\infty_U(L)$ we denote the relativised game.

Remark 4.90. *The game $\mathcal{H}^\infty(L)$ is determined: every play where Constrainer wins is finite. Therefore the winning condition for Constrainer is an open condition, while the winning condition for Alternator is a closed condition. Hence, the game is determined by Gale-Stewart Theorem.*

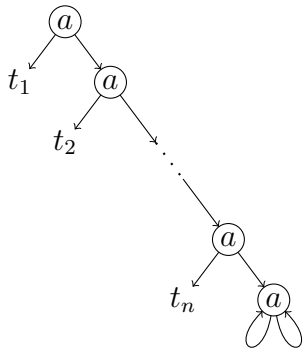
Remark 4.91. *In the same fashion as in the case of the finite game, without loss of generality we can assume that Constrainer in his strategies uses only basic open sets.*

The following fact follows directly from the definition of the two variants of the game.

Fact 4.92. *If Alternator wins $\mathcal{H}^{\infty}(L)$ then he wins $\mathcal{H}^{\in,\notin}(L,n)$ for any n.*

Proposition 4.93. *The converse to Fact 4.92 is not true.*

Proof. We have to exhibit a counterexample. To do that it is convenient to work with an alphabet with three different symbols, so let A be the alphabet $\{a,b,c\}$. In general, if $t_1,\ldots,t_n$ are tree, we denote by $[t_1,\ldots,t_n]$ the tree



Now we set L the language of all the trees of the form $[t_1,\ldots,t_n]$ where every t_i, for $i \in \{1,\ldots,n\}$, is

1. Either a tree with every node labelled by a,
2. Or a tree that ends with c everywhere, i.e. a tree for which there exists a finite prefix p of t_i such that $t_i(u) = c$ for any node $u \notin p$.

If t_i respects the first condition we say that t_i is a *first case tree*, if it respects the second condition we say it is a *second case tree*. We first prove that Alternator wins $\mathcal{H}^{\in,\notin}(L,n)$ for any n. Fix a natural number n, we provide a winning strategy for Alternator for the game $\mathcal{H}^{\in,\notin}(L,2n)$. We define the following sets of trees:

- For $i \in \{1,\ldots,n\}$ define L_i the set of trees $[t_1,\ldots,t_n]$ such that the trees $t_1,\ldots,t_{i-1}$ are second case trees and the trees $t_i,\ldots,t_n$ are first case trees. It is clear that $L_i \subseteq L$ for any i.
- We define $\bar{L}_i$ as L_i except that the tree t_i contains both a and b nodes, but no c nodes. Obviously $\bar{L}_i$ is disjoint from L.

Every prefix of a tree in L_i can be completed into a tree in $\bar{L}_i$ and every prefix of a tree in $\bar{L}_i$ can be completed into a tree in L_{i+1}. It follows that Alternator wins the game

$$\mathcal{H}(L_1, \bar{L}_1, L_2, \bar{L}_2, \ldots, L_n, \bar{L}_n)$$

and therefore also Alternator wins $\mathcal{H}^{\in,\notin}(L, 2n)$.

Now we prove that Alternator loses $\mathcal{H}^{\infty}(L)$. Consider the tree played by Alternator in the first round. Since this tree belongs to L, it must be of the form $[t_1, \ldots, t_n]$ with $t_1, \ldots, t_n$ either first case trees or second case trees. Let C_1 be a finite prefix of this tree which contains the node $\mathfrak{r}^n$. Constrainer uses a strategy, which preserves the following properties:

1. All the prefixes played by Constrainer extend the prefix C_1. Consequently, all trees played by Alternator are of the form $[s_1, \ldots, s_n]$. Indeed, if Alternator modified something under the node $\mathfrak{r}^n$, Constrainer would immediately win by extending C_1 in a proper way. Hence, all the modifications done by Alternator are relative to the trees $t_1, \ldots, t_n$ and therefore for $k \in \{1, \ldots, n\}$ it is meaningful to talk about the k-th coordinate of the tree played by Alternator in a round which refers to the tree s_k.

2. Suppose that Alternator plays a tree $[s_1, \ldots, s_n]$ in some round i. Let C_i be a finite prefix of this tree such that for every coordinate $k \in \{1, \ldots, n\}$ we have:

   - If s_k is a second case tree, then C_i contains a prefix of s_k such that under that prefix every node is labelled by c.
   - If s_k contains some b, then C_i contains some b in the subtree s_k.

   In the next round Constrainer chooses C_i. Consequently, if i, j are rounds with $i < j$ and $k \in \{1, \ldots, n\}$ then

   - If the k-th coordinate of Alternator's tree in round i is a second case tree then also the k-th coordinate of Alternator's tree in round j has to be a second case tree.
   - If the k-th coordinate of Alternator's tree in round i contains a b, then also the k-th coordinate of Alternator's tree in round j contains a b.

So, in an odd-numbered round, Alternator's tree belongs to the language and therefore all the coordinates with a b are second case trees. In an even-numbered round, Alternator's tree is outside the language. Therefore, when going from an odd-numbered round to the next even-numbered round, Alternator must change some coordinate from a first case tree without b to a tree with b. It follows that the number coordinate's with b increases in each even-numbered round. Since this can happen at most n times, Alternator must lose after at most $2n$ rounds.

The proof is complete. □

Now we can give a non effective characterisation of the class $\mathbf{\Delta}^0_2$ for topological spaces that are completely metrizable (for the levels $\mathcal{D}_n(\mathbf{\Sigma}^0_1)$ we gave a characterisation that holds in general for any topological space, but here we are forced to require complete metrizability).

Proposition 4.94. *Let X be a completely metrizable topological space and let Y be a subset of X. Then the following propositions are equivalent:*

1. *Constrainer wins $\mathcal{H}^\infty(Y)$.*
2. *$Y \in \mathbf{\Delta}^0_2(X)$.*

Proof. The proof is very similar to the analysis of other games of this kind, for instance Banach–Mazur game, see [Kec95, Section 8.H].

Implication $1 \Rightarrow 2$. Assume that Constrainer has a winning strategy σ in $\mathcal{H}^\infty(Y)$. By Remark 4.91 we can assume that σ plays only basic open sets. Notice that σ (seen as a tree) is well-founded because the strategy is winning and therefore it admits no infinite play. We will prove by induction on the structure of σ that if $\langle U_0, x_1, U_1, \ldots, U_{i-1}\rangle$ is a position compatible with σ then $Y \cap U_{i-1} \in \mathbf{\Delta}^0_2$. Consider such a position $P = \langle U_0, x_1, U_1, \ldots, U_{i-1}\rangle$ and assume that the thesis holds for all the positions extending that one. If the position P is instantly winning for Constrainer (i.e. Alternator cannot play a single round from P) then, depending on parity of i, either $U_{i-1} \subseteq Y^c$ or $U_{i-1} \subseteq Y$. In both cases the inductive thesis holds. Now assume that P is not instantly winning for Constrainer. By the symmetry lets assume that i is odd, i.e. Alternator is forced to play $x_i \in U_{i-1} \cap Y$. Let $(B_x)_{x \in U_{i-1} \cap Y}$ be the indexed family of basic open sets B_x played by σ as a response to Alternator

playing x. By the inductive assumption we know that for each $x \in U_{i-1} \cap Y$ we have $Y \cap B_x \in \mathbf{\Delta}^0_2$.

By the definition of the family B_x we know that

$$Y \cap U_{i-1} = \bigcup_{x \in U_{i-1} \cap Y} \big(B_x \cap Y\big),$$

where the union is in fact countable since there is only countably many basic open sets in X. As every set taken in the union is $\mathbf{\Sigma}^0_2$ (in fact $\mathbf{\Delta}^0_2$) we know that $Y \cap U_{i-1}$ is $\mathbf{\Sigma}^0_2$. Dually

$$Y^c \cap U_{i-1} = \left(U_{i-1} \setminus \bigcup_{x \in U_{i-1} \cap Y} B_x \right) \cup \bigcup_{x \in U_{i-1} \cap Y} \big(B_x \cap Y^c\big),$$

which again is a $\mathbf{\Sigma}^0_2$ presentation of $Y^c \cap U_{i-1}$.

Thus, the above induction implies that $Y \cap U_0 = Y \cap X = Y$ is $\mathbf{\Delta}^0_2$.

Implication $2 \Rightarrow 1$**.** We need to prove that if $Y \in \mathbf{\Delta}^0_2$ then Constrainer wins $\mathcal{H}^\infty(Y)$. Indeed, if Y is in $\mathbf{\Delta}^0_2$ we can write Y and its complement as

$$Y = \bigcap_{j \in \omega} A_j \quad \text{and} \quad Y^c = \bigcap_{j \in \omega} B_j,$$

where all the sets A_i and B_i are open. Now we can describe a winning strategy for Constrainer in $\mathcal{H}^\infty(Y)$. Suppose $i = 1, 2, \ldots$ is the round we are playing and i is odd (resp. i is even). Let $j = \lfloor \frac{i-1}{2} \rfloor$. Assume that U_{i-1} is the open set that was played last ($U_0 = X$) and let x_i be the point played by Alternator in the current round. By the definition of the game, if i is odd then $x_i \in Y$ and otherwise $x_i \in Y^c$. Let Constrainer play U_i such that $\overline{U_i} \subseteq U_{i-1}$; $U_i \subseteq A_j$ (resp. $U_i \subseteq B_j$); and the diameter of U_i is smaller than 2^{-i}. Such a set exists because $x_i \in U_{i-1} \cap A_j$ (resp. $x_i \in U_{i-1} \cap B_j$).

Clearly it is a valid strategy of Constrainer. Consider an infinite play consistent with this strategy. Since X is Polish and the sets U_i are of decreasing diameter with $\overline{U_i} \subseteq U_{i-1}$, there must exists $x \in \bigcap_{n \in \omega} U_i$. But by the construction of U_i, such x must belong to both $\bigcap_{j \in \omega} A_j = Y$ and $\bigcap_{j \in \omega} B_j = Y^c$, a contradiction. Thus, each play consistent with the above strategy is finite and therefore winning for Constrainer. $\square$

Corollary 4.95. *Let X be a completely metrizable topological space and Y be a subset of X. Then $Y \in \mathbf{\Delta}_2^0 \setminus \mathsf{BC}(\mathbf{\Sigma}_1^0)$ if and only if Alternator wins $\mathcal{H}^{\in,\notin}(Y,n)$ for any n but he loses $\mathcal{H}^{\infty}(Y)$.*

4.9.2 Effective characterisation of $\mathbf{\Delta}_2^0$

What we will prove is the following:

Theorem 4.96. *A regular tree language L belongs to $\mathbf{\Delta}_2^0$ if and only if its syntactic algebra satisfies Equation* (4.4) *from Theorem 4.40.*

This theorem is the main result of [FM14]; unfortunately the proof provided there is wrong:

- Proposition 4 in [FM14] cites Lemma G.2 from [BP12] in a wrong way. The logical claim in Proposition F.2 is of the form: if there is a set Σ of unbounded root alternation then there is a set Σ' of unbounded limit alternation[1]. Lemma G.2 says that if Proposition F.2 is violated then the strategy graph is recursive. Logically, it takes the form: if there is a set Σ of unbounded root alternation but no set Σ' has unbounded limit alternation then the graph is recursive. The way Proposition 4 summarises that statement is: if there exists a set Σ of unbounded limit alternation then the strategy graph is recursive. This statement does not follow from [BP12].

- The proof of Theorem 1 in the implication (2) $\Rightarrow$ (1), shows how to construct, given an infinite strategy tree s^{∞}, a family of strategy trees of unbounded limit alternation. The first step of the proof is to construct a family $\mathcal{G}$ of strategy trees of finite duration but unbounded root alternation. Then, an invalid application of a compactness argument (to find the set X) shows that $\mathcal{G}$ has in fact unbounded limit alternation. Such a set X does not need to exist, it can be the case that the limit alternation of s is 2 but for each pair $k < k' \leq j$ there are infinitely many nodes n in s such that $\sigma_k(n) \neq \sigma_{k'}(n)$ — it is enough that the nodes for distinct values k, k' lie on distinct infinite branches.

The proof we provide here follows the logical structure of [FM14], with the following differences:

[1] Unbounded limit alternation implies unbounded root alternation.

- Instead of the wrong reference used in Proposition 4 of [FM14] we show recursivity of G_L using a new Lemma 4.88 that characterises existence of edges in G_L.
- Instead of the statement about existence of the set X from the proof of Theorem 1 we provide here a direct construction (Lemma 4.104) showing that a winning strategy of Alternator in $\mathcal{H}^\infty(L)$ must take a specific structure that fits the characterisation from Lemma 4.104.

As observed in [FM14], the techniques used to characterise $\mathsf{BC}(\mathbf{\Sigma}^0_1)$ provide a big part of a proof of Theorem 4.96. First, Proposition 4.94 says that L is $\mathbf{\Delta}^0_2$ if and only if Constrainer wins $\mathcal{H}^\infty(L)$. Corollary 4.44 says that if Equation (4.4) is violated then Alternator wins $\mathcal{H}^\infty(L)$. Thus, the "only if" part of Theorem 4.96 follows.

On the other hand, Lemma 4.80 says that if the strategy graph G_L is recursive then Equation (4.4) is violated. It means that the only remaining statement to prove Theorem 4.96 is expressed by the following proposition.

Proposition 4.97. *If Alternator wins $\mathcal{H}^\infty(L)$ then the strategy graph G_L is recursive.*

The rest of this section is devoted to a proof of the above proposition. During the proof we will inductively construct an infinite path $\big(v_n, h_n\big)_{n\in\omega}$ in the strategy graph such that the sequence h_n alternates between L and L^c. The invariant of our construction is expressed by the following definition, using the notions of quotients from Subsection 4.3.1.

Definition 4.98. Consider a node (v, h) of the strategy graph G_L. We say that (v, h) is *prolongable* if there exists a winning strategy σ of Alternator in $\mathcal{H}^\infty\big(v^{-1}(L)\big)$ or in $\mathcal{H}^\infty\big(v^{-1}(L)^c\big)$ such that if t is the tree played by σ in the first round then $v \cdot \alpha_L(t) = h$.

Let us fix a strategy σ of Alternator in $\mathcal{H}^\infty(L)$ and let t_0 be the tree played by σ in the first round. Let $v_0 = 1_L$ and $h_0 = \alpha_L(t_0)$. Directly from the definition we know that (v_0, h_0) is prolongable. Therefore, to prove Proposition 4.97 it is enough to prove the following inductive lemma.

Lemma 4.99. *If (v, h) is prolongable then there exists a node (v', h') in the strategy graph G_L such that $h' \neq h$, (v', h') is prolongable, and there is an edge from (v, h) to (v', h').*

Let us fix a node (v, h) that is prolongable, as witnessed by a strategy σ and a tree t. By the symmetry we can assume that σ is winning in $\mathcal{H}^\infty\big(v^{-1}(L)^c\big)$. We will now analyse the structure of the strategy σ in such a way to extract a witness of an edge in G_L as characterised by Lemma 4.77. For that we need to find an infinite path π of t. This will be achieved by finding a sequence of *essential nodes* in t.

For $d > 0$ a *d-prefix* of a tree t is the prefix $p \stackrel{\text{def}}{=} t{\restriction}\{\mathtt{L},\mathtt{R}\}^{<d}$, i.e. the prefix of t containing all the nodes at depths smaller than d. For instance, the 1-prefix of t consists of the root of t only.

Definition 4.100. Consider a node $u \in \{\mathtt{L},\mathtt{R}\}^d$ for $d > 0$. Let p be the d-prefix of t. We say that u is *essential* if there exists a context C of a type $w \in V_L$, such that the port of C is in u; C extends p; and we have $\big(vw\big)^{-1}(L) \notin \mathbf{\Delta}_2^0$. The last condition implies that Alternator has a winning strategy in the game[2] $\mathcal{H}^\infty\big(\big(vw\big)^{-1}(L)\big)$. Let t' be a tree played by one of such winning strategies of Alternator and let $g = \alpha_L(t')$. Using the above notions we say that u is (w, g)-*essential*.

Fact 4.101. *If C is a context then the function $t \mapsto C[t]$ is continuous. Therefore, if $w^{-1}(K) \notin \mathbf{\Delta}_2^0$ then also $K \notin \mathbf{\Delta}_2^0$. In particular, using (4.2) we obtain that for $v, w \in V_L \sqcup \{1_L\}$ if $(vw)^{-1}(L) \notin \mathbf{\Delta}_2^0$ then also $v^{-1}(L) \notin \mathbf{\Delta}_2^0$. It means that if $\varepsilon \prec u \preceq u'$ and u' is essential then also u is essential.*

The crucial step towards the proof of Lemma 4.99 is expressed by the following claim.

Claim 4.102. *Using the above notions, there are infinitely many essential nodes in t.*

A proof of this claim is given in Subsection 4.9.3. Now we show how the claim implies Lemma 4.99. Since there are infinitely many essential nodes, and a prefix of an essential node is also essential (see Fact 4.101), there exists a path π such that for all $\varepsilon \prec u \prec \pi$ the node u is essential. Notice that each of these nodes u comes with at least one pair $(w_u, g_u) \in V_L \times H_L$ such that u is (w_u, g_u)-essential, see Definition 4.100. Let (w, g) be a pair that equals

[2] We have assumed that σ is winning in $\mathcal{H}^\infty\big(v^{-1}(L)^c\big)$ and therefore we have no complement here; in the dual case σ is winning in $\mathcal{H}^\infty\big(v^{-1}(L)\big)$ and here we consider a winning strategy in $\mathcal{H}^\infty\big(\big(vw\big)^{-1}(L)^c\big)$.

(w_u, g_u) for infinitely many u. Let $v' = vw$ and $h' = vwg$. By the choice of h and h' we know that $h \subseteq L^c$ and $h' \subseteq L$ and therefore $h' \neq h$. Also, directly from the definition of a (w, g)-essential node we know that the node (v', h') is prolongable.

Therefore, to conclude Lemma 4.99 it is enough to show that G_L contains an edge from (v, h) to (v', h'). For that we will check the condition from Lemma 4.88. Consider a finite prefix D of t. There must exist an essential node $u \prec \pi$ such that $u \in \{\mathtt{L}, \mathtt{R}\}^d$ for some $d > 0$; $(w_u, g_u) = (w, g)$; and D is a prefix of the d-prefix of t. Since u is essential, D can be completed into a context C with the port located in u, such that $\alpha_L(C) = w$. Thus, $v \cdot \alpha_L(C) \cdot 1_L = v \cdot w = v'$. This fulfils the requirements of Lemma 4.88 and therefore G_L contains an edge from (v, h) to (v', h').

4.9.3 Proof of Claim 4.102

Consider a number $d > 0$ and let p be the d-prefix of t. We will find an essential node $u \in \{\mathtt{L}, \mathtt{R}\}^d$. Let D be the multicontext obtained from p by making all the nodes in $\{\mathtt{L}, \mathtt{R}\}^d$ ports; let $u_1, \ldots, u_N$ be the list of these ports ($N = 2^d$); and $U \stackrel{\text{def}}{=} D[*]$ be the basic open set defined by p. Finally, put

$$M \stackrel{\text{def}}{=} D^{-1}\big(v^{-1}(L)\big) \subseteq \mathrm{Tr}_A^N.$$

Remark 4.103. *The fact whether $(t_1, \ldots, t_N) \in M$ depends only on the types $\big(\alpha_L(t_1), \ldots, \alpha_L(t_N)\big)$.*

Since (v, h) is prolongable and U is a valid response of Constrainer to Alternator playing t in $\mathcal{H}^\infty\big(v^{-1}(L)^c\big)$, we know that Alternator has a winning strategy in $\mathcal{H}^\infty_U\big(v^{-1}(L)\big)$. This game is equivalent to the game $\mathcal{H}^\infty\big(M\big)$ played in the topological space Tr_A^N. Therefore, by Proposition 4.94 we know that $M \notin \mathbf{\Delta}_2^0$.

Lemma 4.104. *Using the above notation, if $M \subseteq \mathrm{Tr}_A^N$ is not $\mathbf{\Delta}_2^0$ then there exists a port u_i of D and trees $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_N$ such that for the context*

$$C \stackrel{def}{=} D[t_1, \ldots, t_{i-1}, \Box, t_{i+1}, \ldots, t_N],$$

we have $C^{-1}\big(v^{-1}(L)\big) \notin \mathbf{\Delta}_2^0$.

Notice that if we put $w = \alpha_L(C)$ then the last condition in the above lemma says that $\big(vw\big)^{-1}(L) \notin \mathbf{\Delta}_2^0$ and therefore the context C witnesses that u_i is essential. Thus, to conclude Claim 4.102 it is enough to show the above lemma.

Notice that the set $C^{-1}\big(v^{-1}(L)\big)$ from Lemma 4.104 can be equivalently defined as

$$\big\{t \in \mathrm{Tr}_A \mid (t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_N) \in M\big\}.$$

We will call sets of that form *sections* of M. Due to Remark 4.103, taking $h_j = \alpha_L(t_j)$ for $j = 1, \ldots, N$, $j \neq i$, we can equivalently define the above section as

$$\begin{aligned}\big(h_1, \ldots, h_{i-1}, \Box, h_{i+1}, \ldots, h_N\big) &\stackrel{\text{def}}{=} \\ &\big\{t \in \mathrm{Tr}_A \mid h_1 \times \ldots \times h_{i-1} \times \{t\} \times h_{i+1} \times \ldots \times h_N \subseteq M\big\}. \qquad (4.8)\end{aligned}$$

Thus, Lemma 4.104 can be equivalently stated as.

Lemma 4.105. *If $M \notin \mathbf{\Delta}_2^0$ then there exists a section of M that is not $\mathbf{\Delta}_2^0$.*

In general such a property does not hold, however it holds for M due to the fact that M has only finitely many distinct sections, see Remark 4.103.

Notice that the notation from (4.8) naturally extends to formulae with more than one $\Box$. Such sets are called *multi-sections*. The number of holes in a multi-section is called its *dimension*, i.e. a multi-section of dimension n is a subset Tr_A^n. There is one multi-section of the maximal dimension N: $\big(\Box, \ldots, \Box\big) = M$.

Claim 4.106. *Each multi-section can be obtained from sections by taking finite unions, finite intersections, and products.*

Proof. The proof will be inductive on the dimension of the given multi-section. For the sake of simplicity of notation we even consider multi-sections of dimension zero, i.e. expressions of the form $(h_1, \ldots, h_N)$ with no holes. Formally such an expression is either the empty set, or a set containing the empty tuple () (depending on whether $h_1 \times \ldots \times h_N \subseteq M$ or not).

A multi-section of dimension 1 is just a section, so the claim follows. Consider a multi-section of the form (this form is generic up to rearranging the coordinates):

$$N = \big(\Box, \Box, \ldots, \Box, h_{i+1}, h_{i+2}, \ldots, h_N\big).$$

We will prove that N can be represented as in Claim 4.106.

Consider a tuple $h_1, \ldots, h_{i-1}$. If $\big(h_1, \ldots, h_{i-1}, \square, h_{i+1}, \ldots, h_N\big) = \emptyset$ then let $S(h_1, \ldots, h_{i-1}) = \emptyset$ and otherwise let $S(h_1, \ldots, h_{i-1})$ be

$$\bigcap_{h_i \subseteq \big(h_1, \ldots, h_{i-1}, \square, h_{i+1}, \ldots, h_N\big)} \big(\square, \ldots, \square, h_i, \ldots, h_N\big) \times \big(h_1, \ldots, h_{i-1}, \square, h_{i+1}, \ldots, h_N\big).$$

Now consider the set K defined as

$$\bigcup_{h_1, \ldots, h_{i-1}} S(h_1, \ldots, h_{i-1}).$$

By the inductive assumption the set K can be obtained from sections by finite unions, finite intersections, and products. It remains to prove that $K = N$.

First consider a tuple of types $(g_1, g_2, \ldots, g_i)$ such that

$$g_1 \times \ldots \times g_i \subseteq N. \tag{4.9}$$

We will show that this product is contained in $S(g_1, \ldots, g_{i-1})$. First, by (4.9) we know that $g_i \subseteq \big(g_1, \ldots, g_{i-1}, \square, h_{i+1}, \ldots, h_N\big)$ and therefore the later set is not empty. Consider any value $h_i \subseteq \big(g_1, \ldots, g_{i-1}, \square, h_{i+1}, \ldots, h_N\big)$. By the assumption about the considered values h_i we know that

$$g_1 \times \ldots \times g_{i-1} \subseteq \big(\square, \ldots, \square, h_i, \ldots, h_N\big).$$

Therefore, $g_1 \times \ldots \times g_i$ is contained in the product

$$\big(\square, \ldots, \square, h_i, \ldots, h_N\big) \times \big(h_1, \ldots, h_{i-1}, \square, h_{i+1}, \ldots, h_N\big).$$

Now consider the other implication: take a tuple of types $(g_1, g_2, \ldots, g_i)$ such that

$$g_1 \times \ldots \times g_i \subseteq S(h_1, \ldots, h_{i-1}), \tag{4.10}$$

for some choice of types $h_1, \ldots, h_{i-1}$. It means that there must exist $h_i \subseteq \big(h_1, \ldots, h_{i-1}, \square, h_{i+1}, \ldots, h_N\big)$, because the later set cannot be empty. Therefore, $g_i \subseteq \big(h_1, \ldots, h_{i-1}, \square, h_{i+1}, \ldots, h_N\big)$ and it means that g_i is among the values h_i over which we take the intersection. Thus

$$g_1 \times \ldots \times g_i \subseteq \big(\square, \ldots, \square, g_i, \ldots, h_N\big) \times \big(h_1, \ldots, h_{i-1}, \square, h_{i+1}, \ldots, h_N\big).$$

In particular $g_1 \times \ldots \times g_{i-1} \subseteq \big(\Box, \ldots, \Box, g_i, \ldots, h_N\big)$ what implies that

$$g_1 \times \ldots \times g_i \subseteq N.$$

This concludes the proof that $N = K$ and thus the claim is proved. □

Proof of Lemma 4.105. Assume contrarily that all the sections of M are $\mathbf{\Delta}_2^0$. By Claim 4.106 the multi-section $(\Box, \ldots, \Box) = M$ can be obtained from the sections of M using finite unions, finite intersections, and products. As all these operations preserve $\mathbf{\Delta}_2^0$ sets, the set M is also $\mathbf{\Delta}_2^0$. This contradicts the assumption. □

4.10 Conclusions

We conclude this chapter with some considerations on the results proved here.

Limitations In general there is no proper algebraic framework for analysing all regular languages of infinite trees. The available algebras are either too weak [BI09, BS13], too strong [Boj10], or not closed under homomorphic images (i.e. contain a hidden existential quantifier) [Blu11]. Therefore, effective characterisations based on algebraic approach are limited either to certain subclasses of languages as the input, or to simple classes that are being characterised.

In our work the input is not restricted (i.e. the characterisation works for all regular languages as the input), but the characterised classes are low in the Borel hierarchy. To climb higher in the Borel hierarchy one should consider some class of algebras with richer structure; unfortunately there is no natural candidate for such a class at the moment. The known characterisations of classes higher in the hierarchy[3] [SW16, CMS17] are based directly on games instead of using algebras.

Further work There are still natural classes of languages within $\mathbf{\Delta}_2^0$ that await characterisations:

[3]There are other characterisations where the input is restricted to languages with limited use of non-determinism, see [Mur08b, FMS16].

- **Equations for finite levels.** We have seen that the finite levels of the difference hierarchy can be characterised using sentences of MSO. It remains open whether these finite levels correspond to equations (or their ordered variants) in the syntactic algebra of the language.

- **Transfinite levels.** We still do not know what the upper bound of levels inhabited by regular tree language inside the class $\mathbf{\Delta}_2^0$ is (see Conjecture 1). But actually not only we do not know how many transfinite levels of the difference hierarchy (and so Wadge hierarchy) are occupied, but there is no effective characterisation of any transfinite level known. Thus, one can ask for instance how to verify if the Wadge degree η of a regular tree language verifies $\omega \leq \eta < \omega \cdot 2$.

- **Higher in the hierarchy.** The operations of the algebras used here seem to be suited exactly to the classes up to $\mathbf{\Delta}_2^0$. However, one might imagine enriching their algebraic structure just a bit in such a way to cover one more level of the Borel hierarchy. This motivates the following problem.

  Problem 4. Is there a natural algebraic structure extending the algebras given in this work which allow an equational characterisation of the Borel class $\mathbf{\Delta}_3^0$?

Chapter 5

Second level of the Borel hierarchy

We conclude our climbing reaching the second level of the Borel hierarchy of Tr_A. In this chapter we show an algorithm that establishes if a given regular tree language is in the Borel class $\mathbf{\Pi}^0_2$ and provides a solution of Problem 3 for the second level of the Borel hierarchy. These results are all contained in [CMS17].

Now we are going to use tree parity alternating automata. The definition we consider throughout the chapter is the first one we gave in Section 2.5, i.e. Definition 2.13 with the corresponding acceptance condition. We will not speak about the graph of a parity alternating automaton.

The main theorem we prove here is:

Theorem 5.1. *If L is a regular tree language then either:*

- *L can be recognised by a weak-alternating $(0,2)$-automaton and so $L \in \mathbf{\Pi}^0_2$,*
- *L cannot be recognised by a weak alternating $(0,2)$-automaton, $L \notin \mathbf{\Pi}^0_2$, and L is $\mathbf{\Sigma}^0_2$-hard.*

Moreover, it can be effectively decided which of the cases holds. If $L \in \mathbf{\Pi}^0_2$ then a weak-alternating $(0,2)$-automaton can effectively be constructed for L.

Let us see a short overview of the proof of Theorem 5.1.

Let L be a regular language and let us fix once and for all two parity non-deterministic automata $\mathcal{A}$ and $\mathcal{B}$ that recognise respectively: $\mathrm{L}(\mathcal{B}) = L$ is the given language and $\mathrm{L}(\mathcal{A}) = L^c$ is its complement. The proof will consist of the following steps:

- First we define a game $\mathcal{F}$ of infinite duration and perfect information. The game $\mathcal{F}$ is played by two players: Eve ($\exists$) and Adam ($\forall$). Player $\exists$ constructs a tree t together with three runs: one of the automaton $\mathcal{A}$ and two of the automaton $\mathcal{B}$, whose the second one is influenced by $\forall$ who can ask $\exists$ to *restart* whenever he wants. The crucial property of the game $\mathcal{F}$ is that it is played over a finite set of positions and the winning condition is regular.

- If $\exists$ wins $\mathcal{F}$ then her winning strategy can be used to prove that the language $\mathrm{L}(\mathcal{B})$ is actually $\mathbf{\Sigma}^0_2$-hard. In particular it cannot be recognised by a weak-alternating automaton of index $(0, 2)$. To prove this topological hardness we test the winning strategy of $\exists$ against a well-designed family of strategies of $\forall$. In terms of Descriptive Set Theory it can be seen as finding an embedding of the Cantor space 2^ω that intersects $\mathrm{L}(\mathcal{B})$ on rationals.

- If $\forall$ wins $\mathcal{F}$ then we use his finite-memory winning strategy to construct a finite approximation of the automaton $\mathcal{B}$ that is denoted $\mathcal{G}_{K_0}$. The construction ensures that $\mathcal{G}_{K_0}$ is a weak-alternating automaton of index $(0, 2)$ that recognises $\mathrm{L}(\mathcal{B})$.

5.1 The game $\mathcal{F}$

We start by defining the game $\mathcal{F}$ of infinite duration that is based on the tree parity non-deterministic automata

$$\mathcal{A} = \langle A, Q^{\mathcal{A}}, q_{\mathrm{I}}^{\mathcal{A}}, \Delta^{\mathcal{A}}, \Omega^{\mathcal{A}} \rangle \text{ and } \mathcal{B} = \langle A, Q^{\mathcal{B}}, q_{\mathrm{I}}^{\mathcal{B}}, \Delta^{\mathcal{B}}, \Omega^{\mathcal{B}} \rangle$$

for L^c and L respectively. The purpose of $\mathcal{F}$ is to satisfy the following two propositions.

Proposition 5.2. *If $\exists$ wins $\mathcal{F}$ then $\mathrm{L}(\mathcal{B})$ is $\mathbf{\Sigma}^0_2$-hard.*

Proposition 5.3. *If* $\forall$ *wins* $\mathcal{F}$ *then* $\mathrm{L}(\mathcal{B})$ *is recognised by a weak-alternating automaton of index* $(0,2)$*, so is in the class* $\mathbf{\Pi}^0_2$.

The above propositions give a complete characterisation of the topological complexity and the weak index of $\mathrm{L}(\mathcal{B})$.

Positions of $\mathcal{F}$. The positions of $\mathcal{F}$ are of the form $(p,q,s) \in Q^{\mathcal{A}} \times Q^{\mathcal{B}} \times Q^{\mathcal{B}}$ where:

- $p \in Q^{\mathcal{A}}$ is called an $\mathcal{A}$-*state*,
- $q \in Q^{\mathcal{B}}$ is called a $\mathcal{B}$-*state*,
- $s \in Q^{\mathcal{B}}$ is called an *active state*.

The initial position of $\mathcal{F}$ is $(q^{\mathcal{A}}_{\mathrm{I}}, q^{\mathcal{B}}_{\mathrm{I}}, q^{\mathcal{B}}_{\mathrm{I}})$.

Rounds of $\mathcal{F}$. Assume that a round of $\mathcal{F}$ starts in a position (p,q,s). The choices done by the players are as follows:

1. $\forall$ can choose to *restart* by letting $s' = q$ or to *stay* by keeping $s' = s$.
2. $\exists$ declares: (i) a letter $a \in A$; (ii) a transition $(p, a, p_{\mathtt{L}}, p_{\mathtt{R}}) \in \Delta^{\mathcal{A}}$ of $\mathcal{A}$; (iii) a transition $(q, a, q_{\mathtt{L}}, q_{\mathtt{R}}) \in \Delta^{\mathcal{B}}$ of $\mathcal{B}$; (iv) another transition $(s', a, s'_{\mathtt{L}}, s'_{\mathtt{R}}) \in \Delta^{\mathcal{B}}$ of $\mathcal{B}$.
3. $\forall$ responds by selecting a direction $d \in \{\mathtt{L}, \mathtt{R}\}$.

After such a round the game proceeds to the position (p_d, q_d, s'_d). Four example rounds of $\mathcal{F}$ are presented in Figure 5.1.

If π is a finite or infinite play of $\mathcal{F}$, a *trace* is a finite or infinite sequence of active states s in consecutive rounds in which $\forall$ has not *restarted*. Thus, those active states come from successive transitions of the automaton $\mathcal{B}$.

Winning condition of $\mathcal{F}$. Now we will define the winning condition for $\exists$ in $\mathcal{F}$. It will depend on a Boolean combination of the following three properties, speaking about the sequence of rounds that were played:

(WR) $\forall$ has *restarted* infinitely many times.

(WA) The sequence of $\mathcal{A}$-states p is accepting in $\mathcal{A}$.

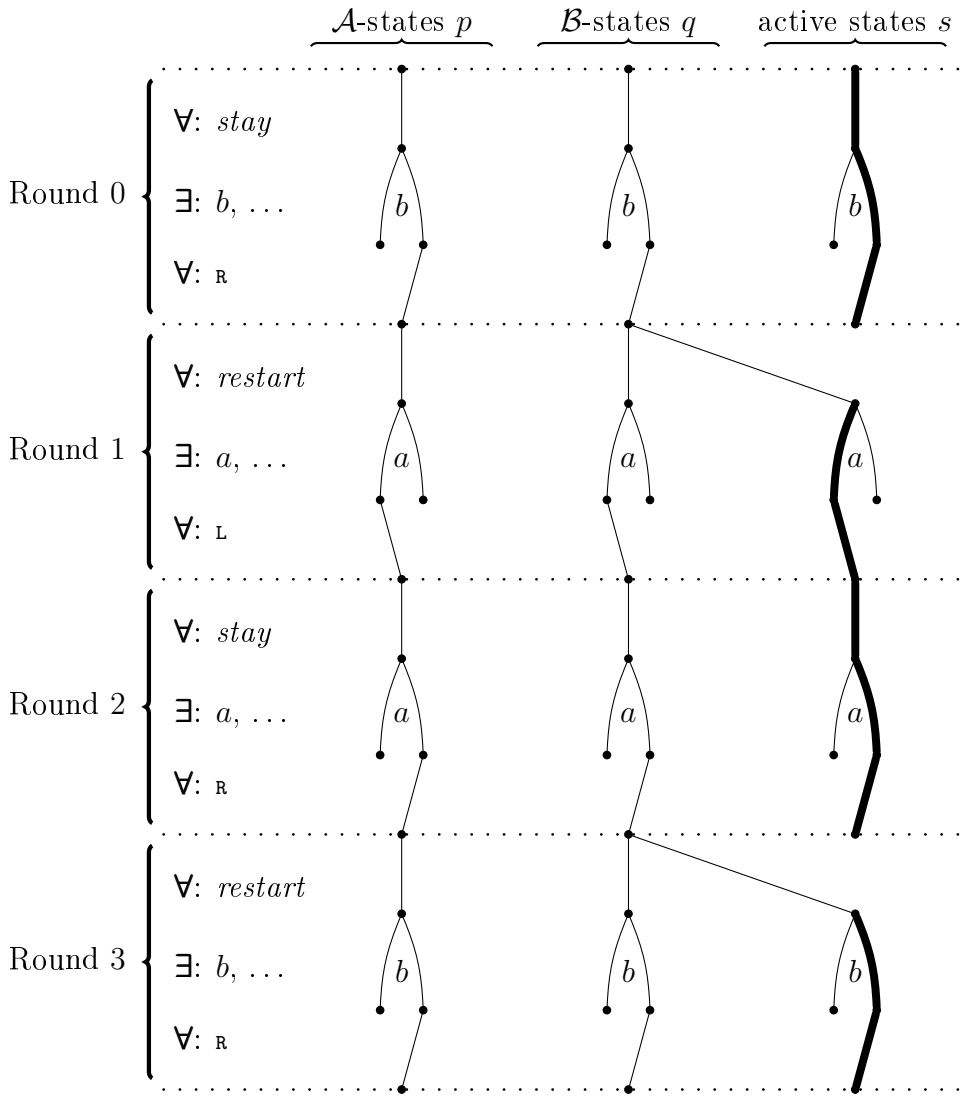


Figure 5.1: Four consecutive rounds of the game $\mathcal{F}$. The black dots are the states of the automata $\mathcal{A}$ and $\mathcal{B}$. Each round consists of three choices: first ∀ either *restarts* or *stays*, then ∃ provides a letter and three transitions (depicted by those Λ-shaped gadgets), finally ∀ chooses a direction. The three boldfaced paths are three traces formed by the active states: the first one lasts in Round 0; the second one in Rounds 1 and 2; the third one starts in Round 3.

(WB) The sequence of active states s is accepting in $\mathcal{B}$ (i.e. it satisfies the parity condition).

A play of $\mathcal{F}$ is winning for $\exists$ if it satisfies

$$\big((\mathrm{WR}) \wedge (\mathrm{WA})\big) \ \vee \ \big(\neg(\mathrm{WR}) \wedge (\mathrm{WB})\big). \tag{5.1}$$

In other words, there are two cases: If $\forall$ has restarted infinitely many times then $\exists$ wins if and only if the sequence of visited $\mathcal{A}$-states satisfies the parity condition. If $\forall$ has restarted only finitely many times then $\exists$ wins if and only if the sequence of visited active states satisfies the parity condition. Notice that *a priori* both (WA) and (WB) can happen simultaneously, because for example there may exist two runs $\rho_{\mathcal{A}}$ and $\rho_{\mathcal{B}}$ of the automata $\mathcal{A}$ and $\mathcal{B}$ that both satisfy the parity condition on some particular path.

By the definition, the winning condition of $\mathcal{F}$ is a regular property of sequences of rounds. Additionally, there are only finitely many positions of $\mathcal{F}$ and each round allows finitely many possible choices by the players. Therefore, we obtain the following fact.

Fact 5.4 ([BL69]). *The winner of $\mathcal{F}$ can be effectively found and he/she can win using a finite memory winning strategy.*

5.2 If ∃ wins

First we prove that if $\exists$ wins $\mathcal{F}$ then $\mathrm{L}(\mathcal{B})$ is $\mathbf{\Sigma}^0_2$-hard. Let $\sigma_\exists$ be her winning strategy. Let $C \subseteq \{0,1\}^\omega$ be the set of sequences containing only finitely many 1s. It is known that C is $\mathbf{\Sigma}^0_2$-complete (see [TL93]). We will construct a continuous reduction from C to $\mathrm{L}(\mathcal{B})$ and so we obtain that $\mathrm{L}(\mathcal{B})$ is $\mathbf{\Sigma}^0_2$-hard.

We will say that σ is a *quasi-strategy* of $\forall$ in $\mathcal{F}$ if σ specifies when to *restart* and leaves undecided the choice of directions d. Notice that if σ is a quasi-strategy of $\forall$ then we can construct a tree t consisting of the letters a played by $\sigma_\exists$ against σ: the letter $t(u)$ is the $(|u|+1)$th letter played by $\sigma_\exists$ against $\forall$ playing accordingly to σ and choosing successive directions of u.

To each sequence $\alpha \in \{0,1\}^\omega$ we will assign a quasi-strategy σ_α of $\forall$ in $\mathcal{F}$. Consider $\alpha \in \{0,1\}^\omega$ and an Mth round of $\mathcal{F}$ for $M = 0, 1, \ldots$

- If $\alpha(M) = 0$ then σ_α *stays* by keeping $s' = s$.

- If $\alpha(M) = 1$ then σ_α *restarts* by putting $s' = q$.

Let the tree t_α be the effect of confronting the strategy $\sigma_\exists$ against the quasi-strategy σ_α. Since the behaviour of the strategy σ_α in an Mth round of $\mathcal{F}$ depends only on the first M bits of α, the function $\alpha \mapsto t_\alpha$ is continuous. A routine verification (see below) shows that

$$\alpha \in C \iff t_\alpha \in \mathrm{L}(\mathcal{B}). \tag{5.2}$$

When $\alpha \in C$. First assume that $\alpha \in C$, i.e. that there are only finitely many 1s in α. Let M be the maximal number such that $\alpha(M-1) = 1$ (or $M = 0$ if there is no such M). Let $\rho^\mathcal{B}$ be the run of $\mathcal{B}$ defined as follows:

- For $|u| \leq M$ let $\rho^\mathcal{B}(u)$ be the $\mathcal{B}$-state q from the beginning of the $|u|$th round of the play consistent with $\sigma_\exists$ and σ_α in which the sequence of directions chosen by $\forall$ was u.

- For $|u| > M$ let $\rho^\mathcal{B}(u)$ be the active state s from the beginning of the $|u|$th round of the play consistent with $\sigma_\exists$ and σ_α in which the sequence of directions chosen by $\forall$ was u.

It is easy to see that $\rho^\mathcal{B}$ is in fact a run of $\mathcal{B}$ over t_α. It remains to see that it is successful. Consider an infinite branch of t_α. This branch corresponds to an infinite play of $\mathcal{F}$ consistent with $\sigma_\exists$ and the quasi-strategy σ_α. Since in all the rounds after the Mth one, $\forall$ has *stayed* by putting $s' = s$, the states of $\rho^\mathcal{B}$ form an infinite trace in that play. Therefore, the condition $\neg$(WR) holds. As the play is won by $\exists$, also (WB) must hold. It means that the trace must be accepting in $\mathcal{B}$, thus the run $\rho^\mathcal{B}$ is accepting on our branch. This way we have proved that $\rho^\mathcal{B}$ is accepting and $t_\alpha \in \mathrm{L}(\mathcal{B})$.

When $\alpha \notin C$. Now assume that $\alpha \notin C$, i.e. that there are infinitely many 1s in α. Our aim is to prove that the run $\rho^\mathcal{A}$ formed by the $\mathcal{A}$-states p played by $\exists$ in all the plays consistent with $\sigma_\exists$ and σ_α is accepting. Consider an infinite branch of t_α and the corresponding play π of $\mathcal{F}$. Since infinitely many times $\forall$ has *restarted*, this play satisfies (WR). As the play is won by $\exists$, also (WA) must hold. It means that the sequence of $\mathcal{A}$-states p must be accepting in $\mathcal{A}$. Thus, we have proved that $t_\alpha \in \mathrm{L}(\mathcal{A})$ and therefore $t_\alpha \notin \mathrm{L}(\mathcal{B})$. This concludes the proof of $\mathbf{\Sigma}^0_2$-hardness of $\mathrm{L}(\mathcal{B})$.

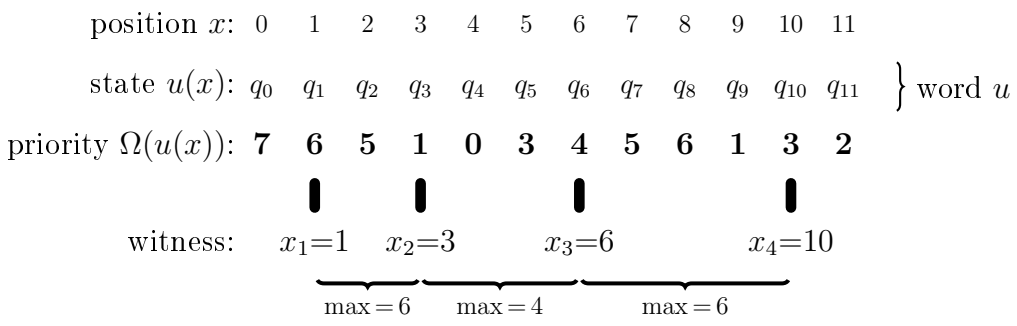


Figure 5.2: A word u that is 4-accepting. The sequence x_1, x_2, x_3, x_4 witnesses that. If we loop u between the positions 1 and 6, we get the sequence $q_0, q_1, \ldots, q_6, q_1, q_2, \ldots, q_6, q_1 \ldots$ that satisfies the parity condition.

5.3 If ∀ wins

Now we prove that if ∀ wins $\mathcal{F}$ then $\mathrm{L}(\mathcal{B})$ can be recognised by a weak-alternating automaton of index $(0, 2)$. Since the winning condition of $\mathcal{F}$ is regular, we can assume that ∀ wins using a strategy $\sigma_\forall$ based on a finite-memory structure M. Our aim is to construct an automaton recognising $\mathrm{L}(\mathcal{B})$.

K-accepting runs. We start by defining a notion of K-accepting sequences, that are sequences of states of $\mathcal{B}$ that are similar to accepting ones. We will show that the strategy $\sigma_\forall$ must avoid such sequences.

Let u be a finite or infinite sequence of states of $\mathcal{B}$. Consider a number $K \in \omega$. We say that u is *K-accepting* if there exists a sequence of positions $0 \leq x_1 < x_2 < \ldots < x_K < |u|$ such that for every $n = 1, 2, \ldots, K-1$ we have:

$$\max\left\{\Omega^{\mathcal{B}}\big(u(x)\big) \;\middle|\; x_n \leq x \leq x_{n+1}\right\} \text{ is even.} \tag{5.3}$$

In other words, for $n = 1, \ldots, K-1$, the maximal priority of states between positions x_n and x_{n+1} of u must be even. We call such a sequence of positions $(x_1, \ldots, x_K)$ a *witness* of K-acceptance, see Figure 5.2.

The above definition is constructed in such a way to guarantee the following properties:

(P1) If u is K-accepting then it contains K positions such that each cycle built using an interval between two of them gives us a sequence of states satisfying the parity condition.

(P2) For every K, the set of all finite words that are K-accepting is regular. It is not obvious how a regular expression for this language should look like, however, the definition of the property of being K-accepting is clearly MSO-definable, thus by the results of Rabin, Scott [RS59], and Trakhtenbrot [Tra62] (cf. e.g. [PP04]), we know that this language is regular.

(P3) If u is K-accepting then every word of the form uw is also K-accepting.

(P4) If $\alpha \in \left(Q^{\mathcal{B}}\right)^{\omega}$ satisfies the parity condition then for every $K \in \omega$ there exists a finite prefix of α that is K-accepting.

Lemma 5.5. *There exists a value $K_0 \in \omega$ such that if π is an infinite play of $\mathcal{F}$ consistent with $\sigma_{\forall}$ then no trace of π is K_0-accepting.*

Proof. Let

$$K_0 \stackrel{\text{def}}{=} \left(\left|Q^{\mathcal{A}}\right| \times \left|Q^{\mathcal{B}}\right| \times \left|Q^{\mathcal{B}}\right|\right) \times \left|M\right| + 1$$

where inside the brackets is the number of positions of $\mathcal{F}$ and M is the memory structure of $\sigma_{\forall}$.

Assume for the sake of contradiction that there exists a play π that is consistent with $\sigma_{\forall}$ and contains a K_0-accepting trace. For a round number $x \in \omega$ during π let (v_x, m_x) be the configuration of the game at the moment when x rounds were played: $v_x = (p_x, q_x, s_x)$ for an $\mathcal{A}$-state p_x, $\mathcal{B}$-state q_x, and active state s_x; and m_x is the current memory value of $\sigma_{\forall}$.

By the assumption we know that for some $x < y < \omega$ the sequence of active states s_x, s_{x+1}, $\ldots$, s_y is a trace (i.e. there is no *restart* during these rounds) and it is K_0-accepting. Let $x \leq x_1, \ldots, x_{K_0} \leq y$ be a sequence of numbers of rounds that is a witness for the K_0-acceptance of this trace, see Equation (5.3).

Figure 5.3 provides an illustration for this construction. The upper picture presents a play π seen in the product of the game $\mathcal{F}$ and the memory structure M used by $\sigma_{\forall}$. Small dots mark positions before successive rounds of this play. The lower picture presents the play π in a chronological way. The boldfaced vertical snake-like shape is a trace that is 5-accepting; the rounded shapes indicate a witness of this fact: $x_1 = 2$, $x_2 = 3$, $x_3 = 6$, $x_4 = 8$, and $x_5 = 10$. Since 5 is bigger than the number of available pairs (v, m) we have a

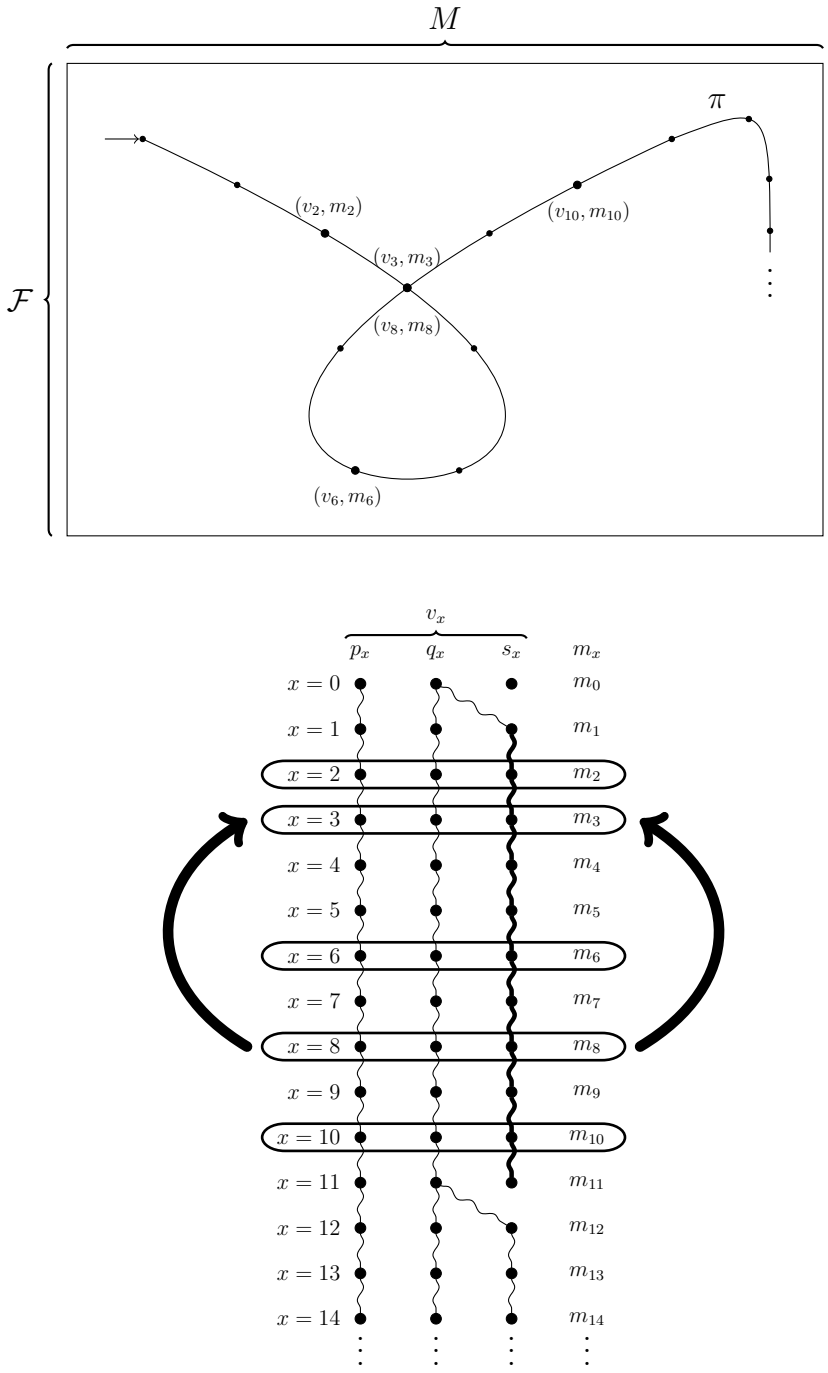


Figure 5.3: An illustration to the proof of Lemma 5.5.

repetition: $(v_3, m_3) = (v_8, m_8)$. This allows us to construct a new play π', by staying forever on the loop between the rounds 3 and 8. The play π' obtained this way contains an infinite trace that satisfies the parity condition.

By the choice of K_0 we know that for some $1 \leq n < n' \leq K_0$ we have: $v_{x_n} = v_{x_{n'}}$ and $m_{x_n} = m_{x_{n'}}$; i.e. there must be a repetition of the position of $\mathcal{F}$ and the memory of $\sigma_{\forall}$ among the positions witnessing K_0-acceptance of the trace.

Consider a play π' of $\mathcal{F}$ which starts as π for the first x_n rounds. Then π' follows the loop between the rounds $x_n + 1$ and $x_{n'}$. Notice that π' is in fact a play because $v_{x_n} = v_{x_{n'}}$. Since we have chosen the positions x_n, $x_{n'}$ from a trace, this loop does not contain a *restart*. Clearly π' is consistent with $\sigma_{\forall}$ because the memory values m_{x_n} and $m_{x_{n'}}$ are equal.

Because x_n and $x_{n'}$ are chosen from a witness of K_0-acceptance of the trace, Property (P1) implies that π' contains an infinite accepting trace. Therefore, the play π' satisfies $\big(\neg(\mathrm{WR}) \wedge (\mathrm{WB})\big)$ and thus is winning for $\exists$ in $\mathcal{F}$, what contradicts the assumption that $\sigma_{\forall}$ was a winning strategy of $\forall$. □

Construction of automata $\mathcal{G}_K$. Take a number $K \in \omega$. We will now define a weak-alternating automaton $\mathcal{G}_K$ of index $(0, 2)$. The language $\mathrm{L}(\mathcal{G}_K)$ will be an over-approximation of $\mathrm{L}(\mathcal{B})$. Later on we will prove that the strategy $\sigma_{\forall}$ witnesses the fact that $\mathrm{L}(\mathcal{B})$ actually equals $\mathrm{L}(\mathcal{G}_K)$ for some $K \in \omega$ (in fact for $K = K_0$ from Lemma 5.5).

The idea behind the automaton $\mathcal{G}_K$ is the following: $\mathcal{G}_K$ accepts a tree t if there exists a run $\rho_0^{\mathcal{B}}$ of $\mathcal{B}$ over t such that for every node u of the tree, it is possible to find another run of $\mathcal{B}$ over the subtree $t.u$ starting from the state $\rho_0^{\mathcal{B}}(u)$ that is K-accepting on every branch of this subtree.

Assume that $\mathcal{D}(K) = \langle Q^{\mathcal{B}}, Q^{\mathcal{D}}, q_{\mathrm{I}}^{\mathcal{D}}, \Delta^{\mathcal{D}}, F^{\mathcal{D}} \rangle$ is a parity deterministic automaton over finite words with the alphabet $Q^{\mathcal{B}}$ that recognises the language of K-accepting sequences of states of $\mathcal{B}$, see Property (P2).[1]

[1] Actually in this thesis we did not define automata for finite word languages. A parity deterministic automaton $\mathcal{D}(K) = \langle Q^{\mathcal{B}}, Q^{\mathcal{D}}, q_{\mathrm{I}}^{\mathcal{D}}, \Delta^{\mathcal{D}}, F^{\mathcal{D}} \rangle$ over finite words is defined the same way as for infinite words, except that $F^{\mathcal{D}}$ is a subset of $Q^{\mathcal{D}}$ of *final states*. A run of $\mathcal{D}$ over a finite word σ is a finite word over $Q^{\mathcal{D}}$ and it is successful if its last element belongs to $F^{\mathcal{D}}$ (i.e. the run has to end with a final state).

The states of $\mathcal{G}_K$ are of the form (q, τ) where $q \in Q^{\mathcal{B}}$ is a state of $\mathcal{B}$ and $\tau \in \{?\} \sqcup Q^{\mathcal{D}}$ is either $?$ or a state of $\mathcal{D}(K)$.

The initial state of $\mathcal{G}_K$ is $(q_{\mathrm{I}}^{\mathcal{B}}, ?)$. The transitions of $\mathcal{G}_K$ are built by the following rules. Given a state (q, τ) and a letter a, the successive state and direction are constructed in the following way (formally the following choices should be encoded as a finite positive Boolean combination of the consecutive directions and states).

1. If $\tau = ?$ then $\forall$ can choose to *start* by letting $\tau' = q_{\mathrm{I}}^{\mathcal{D}}$ or to *skip* by keeping $\tau' = ?$. If $\tau \in Q^{\mathcal{D}}$ then $\forall$ has no choice and in that case $\tau' = \tau$.
2. If $\tau' \in Q^{\mathcal{D}}$ then we let[2] $\tau'' = \Delta^{\mathcal{D}}(\tau, q)$, otherwise $\tau'' = \tau' = ?$ is unchanged.
3. $\exists$ proposes a transition of $\mathcal{B}$ of the form $\big(q, a, q_{\mathtt{L}}, q_{\mathtt{R}}\big)$.
4. $\forall$ chooses a direction $d \in \{\mathtt{L}, \mathtt{R}\}$.

After these choices are done, the automaton moves in the direction d to the state (q_d, τ''). Let the priority of a state (q, τ) of $\mathcal{G}_K$ be:

- (i) If $\tau = ?$ then the priority is 0.
- (ii) If $\tau \in Q^{\mathcal{D}} \setminus F^{\mathcal{D}}$ then the priority is 1.
- (iii) If $\tau \in F^{\mathcal{D}}$ then the priority is 2.

Notice that because of the structure of the transitions of $\mathcal{G}_K$, the above defined condition is a weak parity condition of index $(0, 2)$. It is important to notice that Property (P3) implies that once a state of priority 2 is reached then we never move to a state of priority 1.

Lemma 5.6. *For every $K \in \omega$ we have* $\mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{G}_K)$.

Proof. Take a tree $t \in \mathrm{L}(\mathcal{B})$ and let $\rho^{\mathcal{B}}$ be an accepting run of $\mathcal{B}$ on t. Then clearly $\exists$ can win the acceptance game $\mathcal{G}_K(t)$ by just playing consecutive transitions of $\rho^{\mathcal{B}}$. When $\forall$ chooses at some point to *start*, ultimately a state with $\tau \in F^{\mathcal{D}}$ will be reached because of Property (P4). Therefore, every play will be won by $\exists$. □

[2] We follow the transition of $\mathcal{D}$ from τ over q: $\tau \in Q^{\mathcal{D}}$ is a state of $\mathcal{D}$ and $q \in Q^{\mathcal{B}}$ is a letter read by $\mathcal{D}$.

Equivalence. We will now conclude the proof of Proposition 5.3 using the following lemma.

Lemma 5.7. *For K_0 from Lemma 5.5 we have* $\mathrm{L}(\mathcal{G}_{K_0}) = \mathrm{L}(\mathcal{B})$.

Proof. Assume contrarily that $\mathrm{L}(\mathcal{G}_{K_0}) \neq \mathrm{L}(\mathcal{B})$. Lemma 5.6 says that $\mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{G}_{K_0})$, so there must exists a tree $t \in \mathrm{L}(\mathcal{G}_{K_0}) \setminus \mathrm{L}(\mathcal{B})$. From that assumption we know that:

- there exists an accepting run $\rho^{\mathcal{A}}$ of $\mathcal{A}$ over t,
- $\exists$ has a winning strategy $\delta_\exists$ in the acceptance game $\mathcal{G}_{K_0}(t)$.

Our aim is to prove that $\exists$ can win in $\mathcal{F}$ against $\sigma_\forall$ by using a strategy $\sigma_\exists$ that is based on $\rho^{\mathcal{A}}$ and $\delta_\exists$. Let us define the strategy $\sigma_\exists$.

First, $\sigma_\exists$ plays the letters a and the transitions of $\mathcal{A}$ from the $\mathcal{A}$-states p according to the tree t and the run $\rho^{\mathcal{A}}$. This way we guarantee that every play of this strategy will satisfy (WA). Additionally $\sigma_\exists$ chooses the transitions of $\mathcal{B}$ from the $\mathcal{B}$-states q according to the strategy $\delta_\exists$ simulating the situation that $\forall$ has never *started*. Thus, at every moment of a play consistent with $\sigma_\exists$, there is a unique play of $\delta_\exists$ ending in a node u and a state of $\mathcal{G}_{K_0}$ of the form $(q, ?)$ with u being the sequence of directions played so-far by $\forall$ in $\mathcal{F}$ and q being the current $\mathcal{B}$-state. What remains is the choice of transitions of $\mathcal{B}$ from the active states s. For that, $\exists$ will keep track of a play of the acceptance game $\mathcal{G}_{K_0}(t)$ with $\tau \in Q^{\mathcal{D}}$. At the initial position of $\mathcal{F}$ the play is the one which begins by $\forall$ *starting* (i.e. $\tau = q_{\mathrm{I}}^{\mathcal{D}}$). Whenever $\forall$ *restarts* in $\mathcal{F}$, $\exists$ forgets about the previously tracked play of $\mathcal{G}_{K_0}(t)$ and begins to track the play that comes with the current $\mathcal{B}$-state q by simulating the situation in which $\forall$ has just *started* in $\mathcal{G}_{K_0}(t)$.

Consider the play π that is consistent with both $\sigma_\exists$ and $\sigma_\forall$. We need to prove that π is winning for $\exists$. As we have already observed, such a play satisfies (WA). We will prove that it also satisfies (WR) by proving the following claim. This concludes the proof of Lemma 5.7 by giving a contradiction: $\sigma_\forall$ is a winning strategy of $\forall$ but π is a play consistent with $\sigma_\forall$ that is winning for $\exists$.

Claim 5.8. *In the play π Player $\forall$ must have* restarted *infinitely many times.*

Proof. Assume contrarily that from some point on $\forall$ has not *restarted*. Thus, π contains an infinite trace on which $\exists$ has played successive transitions of $\mathcal{B}$ in the active states s according to her strategy $\delta_\exists$ in $\mathcal{G}_{K_0}(t)$. Since the strategy $\delta_\exists$ is winning, some prefix of the considered trace must be K_0-accepting. This gives a contradiction with Lemma 5.5. □

Now the proof of Theorem 5.1 is concluded. □

5.4 Weak non-deterministic automata

We conclude the thesis with this additional section dedicated to weak non-deterministic automata. In Chapter 3 we saw that for the indexes $(0,1)$ and $(1,2)$ weak non-deterministic automata recognise the same regular languages as the ones recognised by weak-alternating automata. These two classes of automata, indeed, recognise all the regular languages that are closed and open.

Now we prove the following additional result that may be considered folklore, although we have not found it in the literature. The construction is based on the standard de-alternation techniques together with the idea from [MH84].

Theorem 5.9. *If L is a language that can be recognised by a weak-alternating automaton $\mathcal{A}$ of index $(1,3)$ then L can be recognised by a weak non-deterministic automaton $\mathcal{B}$ of index $(1,3)$.*

This observation was important to properly define the game $\mathcal{F}$. However, somehow surprisingly, it does not play any role in the final proof of Theorem 5.1.

The idea of the proof is as follows. Given a tree $t \in \mathrm{Tr}_A$ the automaton $\mathcal{B}$ will guess a positional strategy of $\exists$ in the acceptance game $\mathcal{A}(t)$. Then it will verify that the guessed strategy is in fact winning. Therefore, it will track all the possible choices performed by $\forall$ along all the branches of the tree t. Thus, the set of states of $\mathcal{B}$ is the power set $\mathcal{P}(Q^{\mathcal{A}})$. Notice that the guessed strategy of $\exists$ is winning if for every branch of t the following conditions are satisfied:

- (i) No state of $\mathcal{A}$ of priority 3 is ever reached,

- (ii) Every play ultimately reaches a state of priority 2.

An easy application of König's lemma shows that the second condition above actually implies that at some point no state of priority 1 can belong to the set of reachable states of $\mathcal{A}$. This way, the automaton $\mathcal{B}$ can be seen as a naïve power set construction over $\mathcal{A}$. Such a construction can be performed for any parity alternating automaton (even not weak), however, in most of the cases the assignment of priorities to the states of the power set automaton is not correct. The crucial ingredient of this construction relies on the fact that the weak parity condition of index $(1,3)$ admits a correct priority assignment for the power set automaton.

Now we provide a formal proof of Theorem 5.9, showing that if $\mathcal{A}$ is a weak-alternating $(1,3)$-automaton then one can effectively construct an equivalent weak non-deterministic $(1,3)$-automaton.

Let $\mathcal{A} = \langle A, Q, q_{\mathrm{I}}, \Delta, \Omega \rangle$ be a weak-alternating automaton of index $(1,3)$ that recognises our language L. We recall that without loss of generality we can assume that Δ is given in a *disjunctive normal form*, i.e. for every $q \in Q$, $a \in A$ we have

$$\Delta(q,a) = \bigvee_{Y \in \mathcal{Y}_{q,a}} \bigwedge_{(d,q') \in Y} (d, q'), \tag{5.4}$$

for some family $\mathcal{Y}_{q,a} \subseteq \mathcal{P}(\{\mathtt{L}, \mathtt{R}\} \times Q)$. Notice that since our automata are complete, $\mathcal{Y}_{q,a}$ is non-empty and all its elements are non-empty.

We will now describe a weak non-deterministic automaton of index $(1,3)$

$$\mathcal{B} = \langle A, Q^{\mathcal{B}}, q_{\mathrm{I}}^{\mathcal{B}}, \Delta^{\mathcal{B}}, \Omega^{\mathcal{B}} \rangle$$

such that $\mathrm{L}(\mathcal{B}) = L$.

First, let $Q_{1,2} \subseteq Q$ be the set of states of $\mathcal{A}$ of priorities 1 and 2. Let us define the set of states

$$Q^{\mathcal{B}} \stackrel{\mathrm{def}}{=} \mathcal{P}(Q_{1,2}) \cup \{\bot\},$$

i.e. the power set of $Q_{1,2}$ together with the bottom state $\bot$. The initial state $q_{\mathrm{I}}^{\mathcal{B}} \stackrel{\mathrm{def}}{=} \{q_{\mathrm{I}}\}$ is the singleton of the initial state of $\mathcal{A}$ (without loss of generality we can assume that $q_{\mathrm{I}} \in Q_{1,2}$). Let the priority of a state $X \in Q^{\mathcal{B}}$ of $\mathcal{B}$ be defined as follows:

- if $X = \bot$ then $\Omega^{\mathcal{B}}(X) = 3$;
- otherwise $\Omega^{\mathcal{B}}(X) \stackrel{\text{def}}{=} 2$ if all the states in $X \subseteq Q_{1,2}$ have priority 2;
- otherwise $X \subseteq Q_{1,2}$ contains a state of priority 1 and $\Omega^{\mathcal{B}}(X) \stackrel{\text{def}}{=} 1$.

We now move to the definition of $\Delta^{\mathcal{B}}$.

First, for every letter $a \in A$ we have the transition $(\bot, a, \bot, \bot) \in \Delta^{\mathcal{B}}$.

Consider now a non-bottom state $X \subseteq Q_{1,2}$ of $\mathcal{B}$ and a letter $a \in A$. Intuitively, a transition of $\mathcal{B}$ from X over a is constructed in the following way:

a) We non-deterministically guess choices of $\exists$ in the acceptance game of $\mathcal{A}$ for all the states $q \in X$. Thus, for each $q \in X$ we guess an element $Y_q \in \mathcal{Y}_{q,a}$.

b) We simulate all the possible choices of $\forall$ by taking their union:

$$X' \stackrel{\text{def}}{=} \bigcup_{q \in X} Y_q \subseteq \{\mathtt{L}, \mathtt{R}\} \times Q^{\mathcal{A}}.$$

c) We split the obtained set into the two directions: $X'_d \stackrel{\text{def}}{=} X' \cap \big(\{d\} \times Q^{\mathcal{A}}\big)$ for $d = \mathtt{L}, \mathtt{R}$.

d) If a set X'_d contains a set of priority 3 then we substitute it with $\bot$.

e) This way we obtain a single transition $(X, a, X'_{\mathtt{L}}, X'_{\mathtt{R}})$ of $\mathcal{B}$.

f) We add to $\Delta^{\mathcal{B}}$ all such transitions for all the possible choices of $\big(Y_q\big)_{q \in X}$.

Now we will formally define $\Delta^{\mathcal{B}}$ for non-bottom states $X \neq \bot$. $\Delta^{\mathcal{B}}$ contains a transition $(X, a, X''_{\mathtt{L}}, X''_{\mathtt{R}})$ if there exists a function $\delta \colon X \to \mathcal{P}(Q^{\mathcal{A}})$ such that for every $q \in X$ we have

$$\delta(q) \in \mathcal{Y}_{q,a},$$

and additionally for

$$X' \stackrel{\text{def}}{=} \bigcup_{q \in X} \delta(q) \subseteq \{\mathtt{L}, \mathtt{R}\} \times Q^{\mathcal{A}},$$

$$X'_d \stackrel{\text{def}}{=} \{q' \mid (d, q) \in X'\} \subseteq Q^{\mathcal{A}} \qquad \text{for } d = \mathtt{L}, \mathtt{R},$$

we have $X''_d = X'_d$ if $X'_d \subseteq Q_{1,2}$ and $X''_d = \bot$ otherwise.

This way we have accomplished the definition of the automaton $\mathcal{B}$. Since the transitions of $\mathcal{A}$ are non-decreasing with respect to $\Omega^{\mathcal{A}}$ and $\mathcal{A}$ is complete, it easily implies that the transitions of $\mathcal{B}$ are also non-decreasing. Thus, $\mathcal{B}$ is a weak non-determinisic automaton of index $(1, 3)$. It remains to prove the following lemma.

Lemma 5.10. *The automaton $\mathcal{B}$ recognises the same language as $\mathcal{A}$.*

Proof. First note that if $\mathcal{B}$ accepts a tree t using a run ρ then this run indicates a strategy σ of $\exists$ in the game $\mathcal{A}(t)$. Since the run ρ is accepting, ultimately only 2 is a priority of states visited in the plays consistent with σ. Thus, σ is winning in $\mathcal{A}(t)$ and $t \in \mathrm{L}(\mathcal{A})$.

Now assume that $t \in \mathrm{L}(\mathcal{A})$ and take σ as a positional winning strategy of $\exists$ in $\mathcal{A}(t)$. We can inductively construct a run ρ of $\mathcal{B}$ over t that, given a state X in a position u, puts $\delta(q)$ as the choice of σ in the position $\big(u, \Delta(q, a)\big)$ of the game $\mathcal{A}(t)$, see (5.4). This way we preserve an invariant that if $\rho(u) = X$ with $q \in X$ then there exists a play of $\mathcal{A}(t)$ consistent with σ that reaches the position (u, q).

It remains to see that the run ρ is accepting. First assume that on some branch a state of priority 3 is reached. It means that the strategy σ reaches a state of priority 3 in the acceptance game $\mathcal{A}(t)$ what contradicts the fact that σ is winning. Now assume that on certain branch β of t the run ρ uses only states of the priority 1 (never reaching the priority 2). It means that for every $u \prec \beta$ there is a finite play of $\mathcal{A}(t)$ consistent with σ that reaches a position (u, q) with $\Omega^{\mathcal{A}}(q) = 1$. By applying König's lemma (or a compactness argument) we can find a single play consistent with σ that uses only states of priority 1 in $\mathcal{A}(t)$. This again contradicts the fact that σ is a winning strategy. Thus, the run ρ must satisfy the parity condition on every branch of t and therefore $t \in \mathrm{L}(\mathcal{B})$. □

Hence we conclude weak non-deterministic (i, j)-automata recognise the same languages as weak-alternating (i, j)-automata when (i, j) is equal to $(0, 1)$, $(1, 2)$ or $(1, 3)$. Moreover we can observe that this is the maximum for which the equality holds. Indeed, if we consider the tree language

$$L = \{t \in \mathrm{Tr}_A \mid \text{ the leftmost branch of } t \text{ contains infinitely many } a\},$$

we can notice that it is easy to build a weak-alternating $(0, 2)$-automaton that recognises L, but there is not a weak non-deterministic automaton that recognises L.

Bibliography

[AC13] Alessandro Andretta and Riccardo Camerlo. The descriptive set theory of the lebesgue density theorem. *Advances in Mathematics*, 234:1–42, 2013.

[AN01] André Arnold and Damian Niwiński. *Rudiments of mu-calculus*. Studies in Logic and the Foundations of Mathematics. Elsevier, 2001.

[BCPS18] Mikołaj Bojańczyk, Filippo Cavallari, Thomas Place, and Michał Skrzypczak. Effective characterisations of regular tree languages in low levels of wadge hierarchy. *Preprint*, 2018.

[BI09] Mikołaj Bojańczyk and Tomasz Idziaszek. Algebra for infinite forests with an application to the temporal logic EF. In *CONCUR*, pages 131–145, 2009.

[BL69] Julius Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

[Blu11] Achim Blumensath. Recognisability for algebras of infinite trees. *Theor. Comput. Sci.*, 412(29):3463–3486, 2011.

[Boj04] Mikołaj Bojańczyk. A bounding quantifier. In *CSL*, pages 41–55, 2004.

[Boj10] Mikołaj Bojańczyk. Algebra for trees. A draft version of a chapter that will appear in the AutomathA handbook, 2010.

[BP12] Mikołaj Bojańczyk and Thomas Place. Regular languages of infinite trees that are Boolean combinations of open sets. In *ICALP*, pages 104–115, 2012.

[BS13] Marcin Bilkowski and Michał Skrzypczak. Unambiguity and uniformization problems on infinite trees. In *CSL*, volume 23 of *LIPIcs*, pages 81–100, 2013.

[Büc62] Julius Richard Büchi. On a decision method in restricted second-order arithmetic. In *Proc. 1960 Int. Congr. for Logic, Methodology and Philosophy of Science*, pages 1–11, 1962.

[CKS81] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *J. ACM*, 28:114–133, 1981.

[CL08] Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In *ICALP (2)*, pages 398–409, 2008.

[CMS17] Filippo Cavallari, Henryk Michalewski, and Michał Skrzypczak. A characterisation of Pi^0_2 regular tree languages. In *MFCS*, pages 56:1–56:14, 2017.

[CP94] Olivier Carton and Dominique Perrin. Chains and superchains in ω-semigroups. *Semigroups, Automata and Languages*, pages 17–28, 1994.

[CP97] Olivier Carton and Dominique Perrin. The Wadge-Wagner hierarchy of ω-rational sets. *Automata, Languages and Programming*, 1256:17–35, 1997.

[CPP07] Olivier Carton, Dominique Perrin, and Jean-Éric Pin. Automata and semigroups recognizing infinite words. In *Logic and Automata, History and Perspectives*, pages 133–167, 2007.

[DFH15] Jacques Duparc, Kevin Fournier, and Szczepan Hummel. On unambiguous regular tree languages of index (0, 2). In *CSL*, pages 534–548, 2015.

[DM07] Jacques Duparc and Filip Murlak. On the topological complexity of weakly recognizable tree languages. *Fundamentals of computation theory*, 2007.

[Dup01] Jacques Duparc. Wadge hierarchy and Veblen hierarchy. Part I: Borel sets of finite rank. *The Journal of Symbolic Logic*, 66, 2001.

[Eil74] Samuel Eilenberg. *Automata, languages, and machines*. Pure and Applied Mathematics. Elsevier Science, 1974.

[ES35] Paul Erdös and George Szekeres. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935.

[FM14] Alessandro Facchini and Henryk Michalewski. Deciding the Borel complexity of regular tree languages. In *CiE 2014*, pages 163–172, 2014.

[FMS16] Alessandro Facchini, Filip Murlak, and Michał Skrzypczak. Index problems for game automata. *ACM Trans. Comput. Log.*, 17(4):24:1–24:38, 2016.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[Hig52] Graham Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, s3-2(1):326–336, 1952.

[Hum12] Szczepan Hummel. Unambiguous tree languages are topologically harder than deterministic ones. In *GandALF*, pages 247–260, 2012.

[Idz12] Tomasz Idziaszek. *Algebraic methods in the theory of infinite trees*. PhD thesis, University of Warsaw, 2012. unpublished.

[Jec02] Thomas Jech. *Set Theory*. Springer-Verlag, 2002.

[Kec95] Alexander Kechris. *Classical descriptive set theory*. Springer-Verlag, New York, 1995.

[Kun08] Kenneth Kunen. *The Foundations of Mathematics (Studies in Logic: Mathematical Logic and Foundations)*. College Publications, 2008.

[Lan69] L.H. Landweber. Decision problems for ω-automata. *Math. Syst. Theor*, 4:376–384, 1969.

[LFS15] D. Lecomte, Olivier Finkel, and Pierre Simonnet. An upper bound on the complexity of recognizable tree languages. *RAIRO-Theoretical Informatics and Applications*, 49:121–137, 2015.

[MH84] Satoru Miyano and Takeshi Hayashi. Alternating finite automata on omega-words. *Theor. Comput. Sci.*, 32:321–330, 1984.

[MS87] D.E. Muller and P. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.

[MS95] David E. Muller and Paul E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141(1&2):69–107, 1995.

[MS16] Henryk Michalewski and Michał Skrzypczak. Unambiguous Büchi is weak. In *Developments in Language Theory - 20th International Conference, DLT 2016, Montréal, Canada, July 25-28, 2016, Proceedings*, pages 319–331, 2016.

[MSS86] David E. Muller, Ahmed Saoudi, and Paul E. Schupp. Alternating automata. the weak monadic theory of the tree, and its complexity. In *ICALP*, volume 226 of *Lecture Notes in Computer Science*, pages 275–283, 1986.

[Mur08a] Filip Murlak. *Effective topological hierarchies of recognizable tree languages*. PhD thesis, University of Warsaw, 2008.

[Mur08b] Filip Murlak. The Wadge hierarchy of deterministic tree languages. *Logical Methods in Computer Science*, 4(4), 2008.

[NW03] Damian Niwiński and Igor Walukiewicz. A gap property of deterministic tree languages. *Theor. Comput. Sci.*, 1(303):215–231, 2003.

[Pap93] Christos H. Papadimitriou. *Computational Complexity*. Prentice Hall, 1993.

[PP04] Dominique Perrin and Jean-Éric Pin. *Infinite Words: Automata, Semigroups, Logic and Games*. Elsevier, 2004.

[Rab69] Michael Oser Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. of the American Math. Soc.*, 141:1–35, 1969.

[Rab72] Michael Oser Rabin. *Automata on Infinite Objects and Church's Problem*. American Mathematical Society, Boston, MA, USA, 1972.

[RS59] Michael Oser Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, April 1959.

[RS14] Luca Motto Ros and Philipp Schlicht. Lipschitz and uniformly continuous reducibilities on ultrametric polish spaces. *V. Brattka, H. Diener, and D. Spreen (Eds.), Logic, Computation, Hierarchies, Ontos Mathematical Logic*, 4:213–258, 2014.

[RSS15] Luca Motto Ros, Philipp Schlicht, and Victor Selivanov. Wadge-like reducibilities on arbitrary quasi-polish spaces. *Mathematical Structures in Computer Science*, 25:1705–1754, 2015.

[Skr16] Michał Skrzypczak. *Descriptive Set Theoretic Methods in Automata Theory - Decidability and Topological Complexity*, volume 9802 of *Lecture Notes in Computer Science*. Springer, 2016.

[Sku93] Jerzy Skurczyński. The Borel hierarchy is infinite in the class of regular sets of trees. *Theoretical Computer Science*, 112(2):413–418, 1993.

[Soa87] Robert Soare. *Recursively Enumerable Sets and Degrees. A Study of Computable Functions and Computably Generated Sets*. Perspectives in Mathematical Logic, 1987.

[SW16] Michał Skrzypczak and Igor Walukiewicz. Deciding the topological complexity of Büchi languages. In *ICALP (2)*, pages 99:1–99:13, 2016.

[TL93] Wolfgang Thomas and Helmut Lescow. Logical specifications of infinite computations. In *REX School/Symposium*, pages 583–621, 1993.

[Tra62] Boris A. Trakhtenbrot. Finite automata and the monadic predicate calculus. *Siberian Mathematical Journal*, 3(1):103–131, 1962.

[Urb00] Tomasz Fryderyk Urbański. On deciding if deterministic Rabin language is in Büchi class. In *ICALP*, pages 663–674, 2000.

[Wag79] Klaus Wagner. On ω-regular sets. *Information and Control*, 43(2):123–177, 1979.

[Wes78] Robert Van Wesep. Wadge degrees and descriptive set theory. *Seminaire Cabal 76-77, Lecture Notes in Mathematics*, 689, 1978.

[Wil93] Thomas Wilke. An algebraic theory for regular languages of finite and infinite words. *Int. J. Alg. Comput.*, 3:447–489, 1993.