Warsaw University
Faculty of Mathematics, Informatics and Mechanics

Filip Murlak

Effective Topological Hierarchies of Recognizable Tree Languages

PhD dissertation

Supervisor

dr hab. Damian Niwiński

Institute of Informatics
Warsaw University

January 2008

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

January 23, 2008 .....
date *Filip Murlak*

Supervisor's declaration:
the dissertation is ready to be reviewed

January 23, 2008 .....
date *dr hab. Damian Niwiński*

Abstract

In this study we investigate the topological complexity of recognizable languages of infinite trees. We are particularly interested in deciding topological properties of a language, based on the structure of an automaton that recognizes it.

We show that weak automata with index $(0, n)$ can only recognize languages from the Borel class Π^0_n and, dually, $(1, n+1)$-automata can only recognize Σ^0_n languages. For deterministic tree languages, we also prove the converse implication. In other words, we show that the Borel hierarchy and the weak index hierarchy coincide for deterministic tree languages. We provide an effective procedure to calculate the position of a given deterministic language in both hierarchies.

Then we move to the Wadge ordering of recognizable languages. We give an effective description of the Wadge ordering of deterministic tree languages. The obtained hierarchy has the height $\omega^{\omega \cdot 3} + 3$. We complete the results on deterministic languages by providing an algorithm computing the Wadge degree of a given deterministic language. Finally, we prove that the Wadge hierarchy restricted to weak languages has the height of at least ε_0, which is the least fixpoint of the exponentiation with the base ω.

Keywords

automata on infinite trees, Wadge hierarchy, Borel hierarchy, index hierarchy, decision problems

ACM Computing Classification System

F.4.3, F.1.1, F.1.3

AMS Mathematics Subject Classification

68Q45, 03D30, 03E15

Streszczenie

Niniejsze studium jest poświęcone topologicznej złożoności regularnych języków drzew nieskończonych. Szczególnie interesuje nas rozstrzyganie topologicznych własności języka na podstawie struktury rozpoznającego go automatu.

Pokazujemy, że słabe automaty o indeksie $(0, n)$ mogą rozpoznawać jedynie języki z klasy Σ^0_n i dualnie, słabe automaty o indeksie $(1, n+1)$ rozpoznają języki Π^0_n. Dla języków deterministycznych, tj. rozpoznawanych przez automaty deterministyczne, wykazujemy również odwrotną implikację. Uzyskujemy w ten sposób równość hierarchii borelowskiej i słabej hierarchii indeksu dla języków deterministycznych. Prezentujemy także procedurę obliczającą położenie zadanego języka deterministycznego w obu tych hierarchiach.

Następnie przechodzimy do analizy hierarchii Wadge'a dla regularnych języków drzew. Podajemy efektywny opis porządku Wadge'e na językach deterministycznych otrzymując hierarchię o wysokości $\omega^{\omega \cdot 3} + 3$. Prezentujemy algorytm obliczający stopień Wadge'a dla zadanego języka deterministycznego. Na koniec wykazujemy, że hierarchia Wadge'a dla języków rozpoznawanych przez słabe automaty ma wysokość co najmniej ε_0 (najmniejszy punkt stały potęgowania przy podstawie ω).

Słowa kluczowe

automaty na drzewach nieskończonych, hierarchia Wadge'a, hierarchia borelowska, hierarchia indeksu, problemy decyzyjne

ACM Computing Classification System

F.1.1, F.1.3, F.4.3

AMS Mathematics Subject Classification

68Q45, 03D30, 03E15

Acknowledgements

First and foremost I thank Damian Niwiński. He has been the best supervisor I could wish for. He talked me into taking on this topic, so perfectly combining my favourite areas: automata theory, infinite combinatorics and topology. By sending me to summer schools and conferences, he let me see people doing science for life – working hard and having fun out of it; this was an excellent motivation. He always had time to read and think over my writings, never accepting anything dull, ugly, or unclear. Often to my great discontent, he made me rewrite this thesis again and again, while the arguments were slowly becoming precise, adequate, and readable. He showed me what it really meant to understand, and how much too fast I was to believe I did. He pointed out for me this subtle point of balance between being humble and arrogant towards mathematics.

The second person who had immense influence on this thesis is Jacques Duparc from the University of Lausanne. We obtained the lower bound for weakly recognizable languages in collaboration, and also the other results on the Wadge degree rely entirely on what he taught me. Jacques has a rare ability to explain extremely complex mathematical tools in a way that makes them instantly fit your brain like a long used hammer fits the hand of a carpenter.

I am very grateful to Igor Walukiewicz from Université Bordeaux 1 for the invaluable suggestions about the exposition of the Wadge hierarchy for deterministic languages. If the present Chapter 3 has any elegance, it is in great part due to Igor's help.

I am also indebted to the professors at the Warsaw University and the Polish Academy of Sciences. Zofia Adamowicz, Adam Krawczyk, Jerzy Tyszkiewicz, Paweł Urzyczyn, and Piotr Zakrzewski taught me logic and set theory; their lectures were always difficult enough to be inspiring, and easy enough to be fun. The courses on automata, logic, and semigroups, by Mikołaj Bojańczyk, and on verification, by Sławomir Lasota, brought me up to date with the most recent developments in the field.

Many thanks to Anna Osmańska-Zych from the Secretariat for guiding me through all the rules and regulations of the PhD studies. She is one of the most helpful people in the department.

It felt really good to have the support of my father, always believing in me, my mother, understanding the dilemmas of a would-be mathematician, my brothers, putting up with me far more patiently than I deserved, my

friends, making this world feel like home, and Ania, giving me reasons to work, or not to work, somehow always knowing exactly which was it that I actually needed.

During the preparation of the thesis I was supported by Polish government grants 4 T11C 042 25 and N206 005 31/0881. The results on the Wadge hierarchy of weakly recognizable languages were partially obtained during my visit at the University of Lausanne, Switzerland, financed by AutoMathA (ESF Short Visit Grant 1410).

To lay out the thesis I used a latex template written by my colleague, Jarosław Buczyński.

Contents

1 Measuring Hardness **11**

1.1 Automata for Verification 11
1.2 Topological Hardness 12
1.3 Questions We Answer 13
1.4 Open Problems 14

2 Index and Borel Rank **17**

2.1 Words, Trees, Languages 17
2.2 Automata 19
2.3 Index Hierarchies 20
2.4 Alternation 22
2.5 Weak Hierarchy 24
2.6 Topological Hierarchy 25
2.7 Weak Index vs. Borel Rank 27
2.8 Unravelling the Patterns 30
2.9 The Power of the Weak 32
2.10 Solution 36

3 Wadge Ordering **39**

3.1 Scenario 39
3.2 Games and Automata 41
3.3 Operations 45
3.4 Canonical Automata 47
3.5 Without Branching 50
3.6 The Use of Replication 54
3.7 Automata in Order 58
3.8 Patterns in Automata 63
3.9 Hard Automata 66

3.10 Closure Properties 71
3.11 Completeness 79
3.12 Algorithm 82

4 Wadge Degrees **85**
4.1 More on Wadge Hierarchy 85
4.2 Arithmetic 87
4.3 Calculating Degrees 89
4.4 Conciliatory World 96
4.5 A Lower Bound 99

Bibliography **105**

Index **109**

Chapter 1

Measuring Hardness

During over 40 years of studies into recognizable languages of infinite words, two measures of hardness have proved useful and inspiring: the index hierarchy, which reflects the combinatorial complexity of the recognizing automaton, and the Wagde hierarchy, which is a subtle refinement of the classical Borel/projective hierarchy. The remarkable relations between the two hierarchies were best reflected in Klaus Wagner's results, giving rise to what is now known as the Wagner hierarchy.

Taking the Wagner hierarchy as a paragon, we attempt to provide an equally complete and, hopefully, not less beautiful picture for recognizable languages of infinite trees. In this we succeed only partially.

1.1 Automata for Verification

Tree automata, designed by Michael O. Rabin as a tool to prove decidability of second order monadic logic of two successors, are today – together with μ-calculus – an important theoretic tool in modeling and verification of concurrent systems. In one of possible approaches, a tree represents a possible behaviour of an analysed system, and an automaton is a coded correctness condition. While the efficiency of verification methods depends on the simplicity of the correctness conditions, they often result redundant when modeling real systems. Therefore, algorithmic methods for simplifying the form in which an actual correctness condition is expressed would be welcome.

The most interesting measure of complexity of such a condition is the nesting depth of positive and negative constraints on the events occurring

infinitely often. The formalization of that criterion gives the notion of the index of an automaton. Usually, possible behaviours of the system are also specified by means of an automaton and verification reduces to the inclusion problem for automata, which can be further reduced to the emptiness problem. The complexity of the emptiness problem seems to depend mainly on the index of the automaton: known algorithms are exponential in the index but only polynomial in the number of states [4, 8, 32].

This motivates the investigation of the index of automata. The basic task here is the index problem, i. e., calculating the minimal index of an automaton recognizing a given language. So far, there have been presented procedures calculating the non-deterministic indices of regular path languages [13, 25], and, more generally, deterministically recognizable tree languages [27, 37]. The μ-calculus approach resulted in a procedure deciding if a given formula of modal μ-calculus is equivalent to a formula of modal logic [28].

1.2 Topological Hardness

The classical descriptive set theory classifies subsets of Polish spaces with respect to their simplest definitions in terms of projection, countable union and complementation of the basic open sets. The result of this classification is the Borel/projective hierarchy which can be used as an alternative measure of complexity of recognizable languages. The examples by Skurczyński [34] and the Gap Theorem by Niwiński and Walukiewicz [26] suggest that the index and the Borel rank of recognizable languages are closely related. Comparing these two measures of complexity may be interesting on its own, but it is even more so thanks to the applications in the verification theory.

Yet another objective of studying the topological hardness of recognizable languages is to compare different models of computation. What is more powerful: deterministic or weak alternating automata? It is known that there are deterministic languages that are not weakly recognizable and vice versa. How to compare, if not by inclusion? An even more exotic case: deterministic tree languages versus deterministic context free word languages. How to compare trees with words?

A promising tool is the Wagde ordering based on the existence of continuous reductions between subsets of Polish spaces. This ordering induces a hierarchy of languages that refines immensely the hierarchy of Borel and gives a very subtle measure of hardness. Using the Wadge hierarchy one

can compare any classes of languages of infinite trees and words by simply looking at the height of the Wadge hierarchy restricted to these particular classes.

One can finally be interested in an absolute measure of hardness. Again, the Wadge hierarchy is a reasonable candidate, since it puts languages into an almost linear order. As it is, the Wadge hierarchy provides a measure for Borel sets only, but assuming well justified set-theoretic axiom of determinacy, one can extend its applicability enough to cover all recognizable languages of trees or words. In this setting, one may ask how far a particular recognizing device can get. We have known a lot about word languages [6, 10, 33, 41], but almost nothing about tree languages. Which levels of the Wadge hierarchy are inhabited by deterministically recognizable tree languages? And what about weakly recognizable tree languages?

1.3 Questions We Answer

The principal scope of this thesis are deterministic languages, but we start and finish with weak languages, i. e., languages recognizable by weak alternating automata. In 1993 Skurczyński gave examples of Π^0_n and Σ^0_n-complete languages recognized by weak alternating automata with index $(0, n)$ and $(1, n+1)$ accordingly [34]. We complete this result by showing that weak $(0, n)$-automata can only recognize Π^0_n languages and, dually, $(1, n+1)$-automata can only recognize Σ^0_n languages (Chapter 2, Sect. 7).

Then we turn to deterministic languages. We continue the investigation of weak index hierarchy and Borel hierarchy and prove that the two hierarchies actually coincide for deterministic tree languages (Chapter 2, Sect. 8–10). The starting point for this line of research is the Gap Theorem by Niwiński and Walukiewicz [26], which implies that a deterministic language is either Π^0_3 and can be recognized by a weak alternating $(0, 3)$-automaton or is Π^1_1-complete and not weakly recognizable. The Gap Theorem actually gives an effective criterion for this dichotomy: a deterministic automaton recognizes a Π^0_3 language if and only if its transition graph does not contain a certain forbidden pattern. Inspired by this result, we provide analogous forbidden patterns for the remaining five Borel classes and thus prove decidability of the weak index hierarchy and the Borel hierarchy for deterministic languages.

Afterwords, we refine our analysis and give an effective description of the Wadge ordering of deterministic tree languages (Chapter 3). The obtained

hierarchy has the height $\omega^{\omega \cdot 3} + 3$, which should be compared with ω^ω for regular word languages (the Wagner hierarchy) [41], ω^{ω^2} for deterministic context-free word languages [6], $(\omega_1^{CK})^\omega$ for word languages recognized by deterministic Turing machines [33], or an unknown ordinal $\xi > \varepsilon_0$ for nondeterministic context-free word languages [10]. The key notion of our argument is an adaptation of the Wadge game to tree languages, redefined entirely in terms of automata. Using this tool we construct a hierarchy of canonical automata representing all Wadge degrees of deterministic tree languages, and give a procedure calculating for a given deterministic automaton a Wadge equivalent canonical automaton, thus finding its place in the Wadge ordering. The procedure runs within the time of finding the productive states of the automaton (the exact complexity of this problem is unknown, but not worse than exponential).

Further on, we investigate the levels represented by deterministic tree languages in the Wadge hierarchy of all Borel sets (Chapter 4, Sect. 1–3). We complete the results on deterministic languages by calculating the Wadge degrees of the canonical automata. The technique we use is based on the set-theoretical counterparts of ordinal sum, supremum, and multiplication by ω_1 of Wadge degrees, used previously by Jacques Duparc to describe the Wadge hierarchy of deterministic context free word languages [6].

Towards the end, we return to weak languages and show a lower bound for the height of the Wadge hierarchy for those (Chapter 4, Sect. 4–5). Instead of full trees we work with conciliatory trees (those can have both finite and infinite branches). We prove that the conciliatory analog of Wadge hierarchy embeds into the ordinary Wadge hierarchy, and that the embedding preserves weak recognizability. The conciliatory weak languages are closed by three set-theoretic operations corresponding to the sum, multiplication by ordinals $< \omega^\omega$ and pseudo-exponentiation with the base ω_1 of the conciliatory Wadge degrees. In consequence, the Wadge hierarchy restricted to weak languages has the height of at least ε_0, which is the least fixpoint of the exponentiation with the base ω. This should be contrasted with the aforementioned height of the hierarchy of deterministic tree languages, which is as low as $\omega^{\omega \cdot 3} + 3$.

1.4 Open Problems

Just like in classical complexity theory, the real challenge in automata theory is nondeterminism. The power it gives to tree automata makes them

extremely difficult to tackle. The Borel hierarchy and the Wadge hierarchy of nondeterministic languages are still big unknowns. So is the decidability of the index hierarchy, both nondeterministic and alternating.

In this study we only consider a very restricted version of nondeterminism provided by weak alternating automata. Being the intersection of Büchi and co-Büchi languages [19, 30], weakly recognizable languages form a rather small subclass of all regular tree languages. In fact, this class does not even contain all deterministic languages. On the other hand, it captures some real nondeterminism, as it contains a lot of languages that cannot be recognized by deterministic automata: Skurczyński's examples show that weakly recognizable languages can have any finite Borel rank [34], while deterministic languages are either Π^1_1-complete or are in Π^0_3 [26]. However, even in this simple case, the three most important hierarchies – index, Borel, and Wadge – are not properly understood. Among other problems, there are two conjectures formulated in this thesis: that the height of the Wadge hierarchy of weak languages is ε_0, and that the weak index hierarchy coincides with the Borel hierarchy in the class of weak languages.

Chapter 2

Index and Borel Rank

The investigation of topological complexity of regular word languages was started by Lawrence H. Landweber in late 1960s. In [14] he described an algorithm to calculate the Borel rank of a given language. Further research brought elementary decision procedures for the index hierarchy [13, 25].

For trees the index hierarchy is well understood only for the deterministic automata [25, 27, 37]. As for topological complexity, it goes much higher than that of regular word languages. Indeed, while all regular word languages are Δ^0_3 [14], deterministic automata of an index as low as $(0,1)$ can recognize Π^1_1-complete tree languages [23], and there are even weakly recognizable tree languages on each finite level of the Borel hierarchy [34]. On the other hand, on each level of the deterministic index hierarchy one can find languages which are Δ^0_3. In other words, for some Π^1_1-languages the index $(0,1)$ is enough, but there are Δ^0_3 languages that need an arbitrarily high index! It might seem that the index and the Borel rank measure an entirely different kind of complexity. Yet, we will show that they are deeply related, provided that a suitable version of index is considered.

2.1 Words, Trees, Languages

We use the symbol ω to denote the set of natural numbers $\{0, 1, 2, \ldots\}$. For an alphabet X, X^* is the set of finite words over X and X^ω is the set of infinite words over X. The *concatenation* of words $u \in X^*$ and $v \in X^* \cup X^\omega$ will be denoted by uv, and the empty word by ε. The concatenation is naturally generalized for infinite sequences of finite words $v_1 v_2 v_3 \ldots$. The

concatenation of sets $A \subseteq X^*$, $B \subseteq X^* \cup X^\omega$ is $AB = \{uv : u \in A, v \in B\}$.

A *tree* is any subset of ω^* closed under the prefix relation. An element of a tree is usually called a *node*. A *leaf* is any node of a tree which is not a (strict) prefix of some other node. A Σ-*labeled tree* (or a tree over Σ) is a function $t : \operatorname{dom} t \to \Sigma$ such that $\operatorname{dom} t$ is a tree. A full n-ary Σ-labeled tree is a function $t : \{0, 1, \ldots, n-1\}^* \mapsto \Sigma$. The symbol T_Σ will denote the set of full binary trees over Σ.

For any trees t, s and v, a node of t, the result of the *substitution* of v with s in t is a tree t' whose domain is the set $\{w \in \operatorname{dom} t\colon v \text{ is not a prefix of } w\} \cup v \operatorname{dom} s$ and

$$t'(u) = \begin{cases} s(u') & \text{if } u = vu' \text{ for some } u' \\ t(u) & \text{otherwise} \end{cases} .$$

Note that $t'(v) = s(\varepsilon)$.

The concatenation of tree languages A, B is a tree language AB consisting of all trees obtained from some $t \in A$ by substituting every leaf u of t with some tree $s_u \in B$. The concatenation of infinite sequence of tree languages is a natural generalization of the above. A more precise definition requires an auxiliary notion of a limit. Let $t_0, t_1, \ldots$ be a sequence of trees such that

- $\operatorname{dom} t_0 \subseteq \operatorname{dom} t_1 \subseteq \ldots$,
- $\forall v \in \bigcup_{m\in\omega} \operatorname{dom} t_m \;\; \exists n_v \;\; \forall n \geq n_v \;\; t_n(v) = t_{n_v}(v)$.

The *limit* $t = \lim t_n$ is defined as follows:

- $\operatorname{dom} t = \bigcup_{m\in\omega} \operatorname{dom} t_m$,
- $t(v) = t_{n_v}(v)$.

An *infinite concatenation* of tree languages $L_0 L_1 \ldots$ consists of the limits of all sequences $t_0, t_1, \ldots$ such that $t_0 \in L_0$ and $t_{n+1} \in \{t_n\} L_{n+1}$ for all n.

The concatenation of trees s, t is the only element of the concatenation $\{s\}\{t\}$. Similarly, the concatenation of infinite sequence of trees $t = t_1 t_2 t_3 \ldots$ is the only element of $\{t_1\}\{t_2\}\{t_3\}\ldots$.

For $v \in \operatorname{dom} t$ we define $t.v$ as a subtree of t rooted in v, i. e., $\operatorname{dom}(t.v) = \{u : vu \in \operatorname{dom} t\}$, $t.v(u) = t(vu)$.

From now on, if not stated otherwise, a "tree" will mean a full binary tree over some alphabet.

2.2 Automata

Out of a variety of acceptance conditions for automata on infinite structures, we choose the parity condition. A *nondeterministic parity automaton on words* can be presented as a tuple $A = \langle \Sigma, Q, \delta, q_0, \mathrm{rank} \rangle$, where Σ is a finite input alphabet, Q is the set of states, $\delta \subseteq Q \times \Sigma \times Q$ is the relation of transition and $q_0 \in Q$ is the initial state. The meaning of the function $\mathrm{rank} : Q \to \omega$ will be explained later. Instead of $(q, \sigma, q_1) \in \delta$ one usually writes $q \xrightarrow{\sigma} q_1$. A *run* of an automaton A on a word $w \in \Sigma^\omega$ is a word $\rho_w \in Q^\omega$ such that $\rho_w(0) = q_0$ and if $\rho_w(n) = q$, $\rho_w(n+1) = q_1$, and $w(n) = \sigma$, then $q \xrightarrow{\sigma} q_1$. A run ρ_w is *accepting* if the highest rank repeating infinitely often in ρ_w is even; otherwise ρ_w is *rejecting*. A word is *accepted* by A if there exists an accepting run on it. The language recognized by A, denoted $L(A)$ is the set of words accepted by A. An automaton is *deterministic* if its relation of transition is a *total* function $Q \times \Sigma \to Q$. Note that a deterministic automaton has a unique run (accepting or not) on every word. We call a language deterministic if it is recognized by a deterministic automaton.

A *nondeterministic automaton on trees* is a tuple $A = \langle \Sigma, Q, \delta, q_0, \mathrm{rank} \rangle$, the only difference being that $\delta \subseteq Q \times \Sigma \times Q \times Q$. Like before, $q \xrightarrow{\sigma} q_1, q_2$ means $(q, \sigma, q_1, q_2) \in \delta$. We write $q \xrightarrow{\sigma,0} q_1$ if there exists a state q_2 such that $q \xrightarrow{\sigma} q_1, q_2$. Similarly for $q \xrightarrow{\sigma,1} q_2$. A *run* of A on a tree $t \in T_\Sigma$ is a tree $\rho_t \in T_Q$ such that $\rho_t(\varepsilon) = q_0$ and if $\rho_t(v) = q$, $\rho_t(v0) = q_1$, $\rho_t(v1) = q_2$ and $t(v) = \sigma$, then $q \xrightarrow{\sigma} q_1, q_2$. A path π of the run ρ_t is *accepting* if the highest rank repeating infinitely often in π is even; otherwise π is *rejecting*. A run is called accepting if all its paths are accepting. If at least one of them is rejecting, so is the whole run. An automaton is called deterministic if its transition relation is a total function $Q \times \Sigma \to Q \times Q$.

By A_q we denote the automaton A with the initial state set to q. A state q is *all-accepting* if A_q accepts all trees, and *all-rejecting* if A_q rejects all trees. A state (a transition) is called *productive* if it is used in some accepting run. Observe that being productive is more than just not being all-rejecting. A state q is productive if and only if it is not all-rejecting and there is a path $q_0 \xrightarrow{\sigma_0,d_0} q_1 \xrightarrow{\sigma_1,d_1} \ldots \xrightarrow{\sigma_n,d_n} q$ such that $q_i \xrightarrow{\sigma_i,\bar{d}_i} q'_i$, $\bar{d}_i \neq d_i$, and q'_i is not all-rejecting for $i = 0, 1, \ldots, n$.

Without loss of generality we may assume that all states in A are productive save for one all-rejecting state $\perp$ and that all transitions are either productive or are of the form $q \xrightarrow{\sigma} \perp, \perp$. The reader should keep in mind that

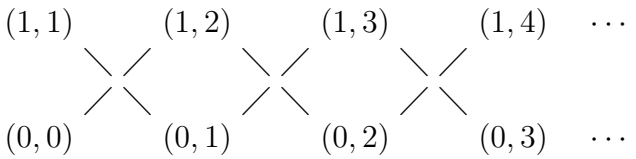


Figure 2.1: The Mostowski–Rabin index hierarchy.

this assumption has influence on the complexity of our algorithms. Transforming a given automaton into such a form of course needs calculating the productive states, which is equivalent to deciding a language's emptiness. The latter problem is known to be in NP ∩ co-NP, but it has no polynomial time solutions yet. Therefore, we can only claim that our algorithms are polynomial for the automata that underwent the above preprocessing. We will try to mention it whenever particularly important.

2.3 Index Hierarchies

The *Mostowski–Rabin index of an automaton* A is a pair

$$(\min \operatorname{rank} Q, \max \operatorname{rank} Q)\,.$$

An automaton with index (ι, κ) is often called a (ι, κ)-automaton. Scaling down the rank function if necessary, one may assume that $\min \operatorname{rank} Q$ is either 0 or 1. Thus, the indices are elements of $\{0,1\} \times \omega \setminus \{(1,0)\}$. For an index (ι, κ) we shall denote by $\overline{(\iota, \kappa)}$ *the dual index*, i. e., $\overline{(0, \kappa)} = (1, \kappa + 1)$, $\overline{(1, \kappa)} = (0, \kappa - 1)$. Let us define an ordering of indices with the following formula

$$(\iota, \kappa) < (\iota', \kappa') \text{ if and only if } \kappa - \iota < \kappa' - \iota'\,.$$

In other words, one index is smaller than another if and only if it uses less ranks. This means that dual indices are not comparable. *The Mostowski–Rabin index hierarchy* for a certain class of automata consists of ascending sets (levels) of languages recognized by (ι, κ)-automata (see Fig. 2.1).

The fundamental question about the hierarchy is the strictness, i. e., the existence of languages recognized by a (ι, κ)-automaton, but not by a $\overline{(\iota, \kappa)}$-automaton. The strictness of the hierarchy for deterministic automata follows easily from the strictness of the hierarchy for deterministic word automata [41]: if a word language L needs at least the index (ι, κ), so does

the language of trees that have a word from L on the leftmost branch. The index hierarchy for nondeterministic automata is also strict [24]. In fact, the languages showing the strictness may be chosen deterministic: one example is the family of the languages of trees over the alphabet $\{\iota, \iota+1, \ldots, \kappa\}$ satisfying the parity condition on each path.

The second important question one may ask about the index hierarchy is how to determine the exact position of a given language. This is known as the *index problem*.

Given a deterministic language, one may ask about its *deterministic index*, i. e., the exact position in the index hierarchy of deterministic automata (deterministic index hierarchy). This question can be answered effectively. Here we follow the method introduced by Niwiński and Walukiewicz [25].

A path in an automaton is a sequence of states and transitions:

$$p_0 \overset{\sigma_1,d_1}{\longrightarrow} p_1 \overset{\sigma_2,d_2}{\longrightarrow} \ldots \overset{\sigma_{n-1},d_{n-1}}{\longrightarrow} p_n \,.$$

A *loop* is a path starting and ending in the same state, $p_0 \longrightarrow p_1 \longrightarrow \ldots \longrightarrow p_0$. A loop is called *accepting* if $\max_i \operatorname{rank}(p_i)$ is even. Otherwise it is *rejecting*. A j-loop is a loop with the highest rank on it equal to j. A sequence of loops $\lambda_\iota, \lambda_{\iota+1}, \ldots, \lambda_\kappa$ in an automaton is called *an alternating chain* if the highest rank appearing on λ_i has the same parity as i and it is higher then the highest rank on λ_{i-1} for $i = \iota, \iota+1, \ldots, \kappa$. A (ι,κ)-*flower* is an alternating chain $\lambda_\iota, \lambda_{\iota+1}, \ldots, \lambda_\kappa$ such that all loops have a common state q.

Niwiński and Walukiewicz use flowers in their solution of the index problem for deterministic word automata.

Theorem 1 (Niwiński, Walukiewicz [25])**.** *A deterministic automaton on words is equivalent to a deterministic (ι,κ)-automaton iff it does not contain a $\overline{(\iota,\kappa)}$-flower.*

For a tree language L over Σ, let $\mathrm{Paths}(L) \subseteq (\Sigma \times \{0,1\})^\omega$ denote the language of generalized paths of L,

$$\mathrm{Paths}(L) = \{\langle(\sigma_1,d_1),(\sigma_2,d_2),\ldots\rangle \colon \exists_{t\in L}\ \forall_i\ t(d_1 d_2 \ldots d_{i-1}) = \sigma_i\}\ .$$

A deterministic tree automaton A, can be treated as a deterministic automaton recognizing $\mathrm{Paths}(L(A))$. Simply for $A = \langle Q, \Sigma, q_0, \delta, \mathrm{rank}\rangle$, take $\langle Q, \Sigma\times\{0,1\}, q_0, \delta', \mathrm{rank}\rangle$, where $(p,(\sigma,d),q) \in \delta' \iff (p,\sigma,d,q) \in \delta$. Conversely, given a deterministic word automaton recognizing $\mathrm{Paths}(L(A))$, one

may interpret it as a tree automaton, obtaining thus a deterministic automaton recognizing $L(A)$. Hence, applying Theorem 1 one gets the following result.

Proposition 1. *For a deterministic tree automaton A the language $L(A)$ is recognized by a deterministic (ι, κ)-automaton iff A does not contain a $\overline{(\iota, \kappa)}$-flower.*

For a deterministic language one may want to calculate its *nondeterministic index*, i.e. the position in the hierarchy of nondeterministic automata. This may be lower than the deterministic index, due to greater expressive power of nondeterministic automata. Consider for example the language $L_M^{0^\omega}$ consisting of trees whose leftmost paths are in a regular word language M. It can be recognized by a nondeterministic $(1,2)$-automaton, but its deterministic index is equal to the deterministic index of M, which can be arbitrarily high.

The problem transpired to be rather difficult and has only just been solved in [27]. Decidability of the general index problem for nondeterministic automata is one of the most important open questions in the field.

2.4 Alternation

Alternating automata were first introduced by Muller and Schupp [20]. The acceptance for that kind of automata is most conveniently defined by means of games. A *parity game* is a perfect information game of possibly infinite duration played by two players, Adam and Eve. We present it as a tuple $\langle V_\exists, V_\forall, E, v_0, \mathrm{rank}\rangle$, where $V_\exists$ and $V_\forall$ are disjoint sets of positions of Eve and Adam, $E \subseteq V \times V$ is the relation of possible moves, with $V = V_\exists \cup V_\forall$, $v_0 \in V$ is a designated initial position, and $\mathrm{rank} : V \to \omega$ is the ranking function.

The players start a play in the position v_0 and then move the token according to relation E (always to a successor of the current position), thus forming a path in the graph (V, E). The move is selected by Eve or Adam, depending on who is the owner of the current position. If a player cannot move, she/he looses. Otherwise, the result of the play is an infinite path in the graph, $v_0, v_1, v_2, \ldots$. Eve wins the play if the highest rank visited infinitely often is even, otherwise Adam wins.

A *positional* strategy for the player θ is a (partial) function $\sigma : V_\theta \to V$ satisfying $(v, \sigma(v)) \in E$ for all $v \in \mathrm{dom}\,\sigma$, which –intuitively – tells the

player where to go from a given position. A crucial property of parity games is the *positional determinacy*: in each parity game either Adam or Eve has a winning positional strategy [9, 18].

An *alternating automaton (on trees)* $A = \langle \Sigma, Q_\exists, Q_\forall, q_0, \delta, \mathrm{rank} \rangle$ is a modification of a nondeterministic automaton defined previously. The set of states Q is partitioned into existential states $Q_\exists$ and universal states $Q_\forall$, and $\delta \subseteq Q \times \Sigma \times \{0, 1, \varepsilon\} \times Q$.

An input tree t is accepted by an alternating automaton A iff Eve has a winning strategy in the parity game $\langle Q_\exists \times \{0,1\}^*, Q_\forall \times \{0,1\}^*, (q_0, \varepsilon), E, \mathrm{rank}' \rangle$, where $\mathrm{E} = \{((p, v), (q, vd)) \colon (p, t(v), d, q) \in \delta\}$ and $\mathrm{rank}'(q, v) = \mathrm{rank}(q)$.

The *computation tree* of A on an input tree t is obtained by unravelling the graph $\langle Q \times \{0,1\}^*, E \rangle$ from the vertex (q_0, ε) and labeling the node $(q_0, \varepsilon), (q_1, d_1), \ldots, (q_n, d_n)$ with q_n. The result of the parity game above only depends on the computation tree.

It is known that alternating automata have the same expressive power as nondeterministic ones [19], however they might use less ranks to recognize a language. Recall that deterministic languages can be arbitrarily high in the nondeterministic hierarchy. In the alternating hierarchy they are all on the second level, as the following folklore fact shows.

Proposition 2. *For deterministic languages the alternating index hierarchy collapses to* $(0, 1)$.

Proof. For a deterministic automaton A over the alphabet Σ, $L(A)^{\complement} = T_\Sigma \setminus L(A)$ can be recognized by a nondeterministic automaton B of index $(1, 2)$ that guesses a path rejected by A. To get an alternating $(0, 1)$-automaton recognizing $L(A)$ it is enough to interpret the automaton B as alternating and apply the usual complementation procedure: swap the existential and universal states, and decrease the ranks by 1. □

For all recognizable tree languages the alternating hierarchy is strict. Let $W_{(\iota,\kappa)}$ denote the set of trees t over the alphabet $\{\exists, \forall\} \times \{\iota, \iota+1, \ldots, \kappa\}$ such that Eve wins the parity game $G_t = \langle V_\exists, V_\forall, E, \varepsilon, \mathrm{rank} \rangle$, where $V_\theta = \{v \in \{0,1\}^* \colon t(v) = (\theta, n) \text{ for some } n\}$, $E = \{(v, vj) \colon v \in \{0,1\}^*,\ j \in \{0,1\}\}$, and $\mathrm{rank}(v) = n$ if $t(v) = (\theta, n)$ for some θ.

Theorem 2 (Bradfield [3])**.** *For all* (ι, κ), $W_{(\iota,\kappa)}$ *can be recognized by an alternating* (ι, κ)*-automaton, but not by an alternating* $\overline{(\iota, \kappa)}$*-automaton.*

An elegant topological proof of this fact based on the Banach Contraction Principle was given later by Arnold [1].

It is a bit surprising that $W_{(\iota,\kappa)}$ can actually be recognized by a *nondeterministic* (ι,κ)-automaton, just like the languages proving the strictness of the nondeterministic hierarchy could be recognized by deterministic automata.

2.5 Weak Hierarchy

In the previous section we have introduced the alternating hierarchy. For deterministic languages it collapsed at the second level. We will now restrict the power of alternating automata significantly and consider yet another hierarchy.

A *weak automaton* is an alternating automaton satisfying the condition

$$p \xrightarrow{\sigma,d} q \quad \Longrightarrow \quad \operatorname{rank} p \leq \operatorname{rank} q\,.$$

A more elegant definition of the class of weakly recognized languages is obtained by using *weak parity games* in the definition of acceptance by alternating automata. In those games Eve wins a play if the highest rank used at least once is even. For the purpose of the following lemma, let us call the first version *restricted alternating automata* and the second version, *weak automata*. Later, we will stick to the second definition.

Lemma 1. *For every L it holds that L is recognized by a restricted alternating (ι,κ)-automaton iff it is recognized by a weak (ι,κ)-automaton.*

Proof. Every restricted automaton can be transformed into an equivalent weak automaton by simply changing the acceptance condition to weak. Let us, then, concentrate on the converse implication.

Fix a weak automaton A using ranks (ι,κ). To construct a restricted automaton we will take one copy of A for each rank: $A^{(\iota)}, A^{(\iota+1)}, \ldots, A^{(\kappa)}$. By $q^{(i)}$ we will denote the counterpart of A's state q in $A^{(i)}$. Set $\operatorname{rank} q^{(i)} = i$. We want the number of the copy the computation is in to reflect the highest rank seen so far. To obtain that, we set the initial state of the new automaton to $q_0^{(\operatorname{rank} q_0)}$, and for each i and each transition $p \xrightarrow{\sigma,d} q$ in A we add a transition $p^{(i)} \xrightarrow{\sigma,d} q^{(\max(i,\operatorname{rank} q))}$. For each i and q, $q^{(i)}$ is universal iff q is universal. Checking the equivalence is straightforward. □

How weak are the weak automata? Compared to nondeterministic automata – very weak.

Theorem 3 (Rabin [31]; Muller, Saoudi, Schupp [19]). *A language L is weakly recognizable iff L and $L^{\complement}$ can be recognized by a nondeterministic $(1,2)$-automaton.*

However, in the course of this study we will see that they are not that weak in the realm of deterministic languages.

Just like for previous classes we can consider the index hierarchy of weak automata. The strictness of this hierarchy was established by Mostowski [17] via equivalence with the quantifier-alternation hierarchy for the weak monadic second order logic, whose strictness was proved by Thomas [35]. The *weak index problem*, i. e., computing the minimal weak index needed to recognize a given weak language, for the time being remains unsolved just like its previous versions.

In [21] we showed how to compute the *weak deterministic index* of a given deterministic language. The procedure is based on the method of difficult patterns used in Theorem 1 and Proposition 1. We need the simplest pattern exceeding the capability of weak deterministic (ι,κ)-automata. Just like in the case of the deterministic index, it seems natural to look for a generic pattern capturing all the power of $\overline{(\iota,\kappa)}$. Intuitively, we need to enforce the alternation of ranks provided by $\overline{(\iota,\kappa)}$. Let a *weak (ι,κ)-flower* be a sequence of loops $\lambda_\iota, \lambda_{\iota+1} \ldots, \lambda_\kappa$ such that λ_{j+1} is reachable from λ_j, and λ_j is accepting iff j is even.

Proposition 3 ([21]). *A deterministic automaton A is equivalent to a weak deterministic (ι,κ)-automaton iff it does not contain a weak $\overline{(\iota,\kappa)}$-flower.*

In this study we solve the weak index problem with input restricted to deterministic automata.

2.6 Topological Hierarchy

We start with a short recollection of elementary notions of descriptive set theory. For further information see [12].

Let 2^ω be the set of infinite binary sequences with a metric given by the formula

$$d(u,v) = \begin{cases} 2^{-\min\{i\in\omega\,:\,u_i\neq v_i\}} & \text{iff } u \neq v \\ 0 & \text{iff } u = v \end{cases}$$

and T_Σ be the set of infinite binary trees over Σ with a metric

$$d(s,t) = \begin{cases} 2^{-\min\{|x|\,:\,x\in\{0,1\}^*,\, s(x)\neq t(x)\}} & \text{iff } s \neq t \\ 0 & \text{iff } s = t \end{cases} .$$

Both 2^ω and T_Σ, with the topologies induced by the above metrics, are Polish spaces (complete metric spaces with countable dense subsets). In fact, both of them are homeomorphic to the Cantor discontinuum.

The class of Borel sets of a topological space X is the closure of the class of open sets of X by complementation and countable sums. Within this class one builds so called *Borel hierarchy*. The initial (finite) levels of the Borel hierarchy are defined as follows:

$\Sigma^0_1(X)$ – open subsets of X,

$\Pi^0_k(X)$ – complements of sets from $\Sigma^0_k(X)$,

$\Sigma^0_{k+1}(X)$ – countable sums of sets from $\Pi^0_k(X)$.

For example, $\Pi^0_1(X)$ are closed sets, $\Sigma^0_2(X)$ are F_σ sets, and $\Pi^0_2(X)$ are G_δ sets. By convention, $\Pi^0_0(X) = \{X\}$ and $\Sigma^0_0(X) = \{\emptyset\}$.

Even more general classes of sets from the *projective hierarchy*. We will need only its lowest level:

$\Sigma^1_1(X)$ – *analytical* subsets of X, i. e., projections of Borel subsets of X^2 with product topology,

$\Pi^1_1(X)$ – complements of sets from $\Sigma^1_1(X)$.

Whenever the space X is determined by the context, we will skip it in the notation above and write simply Σ^0_1, Π^0_1, and so on.

Let $\varphi : X \to Y$ be a continuous map of topological spaces. One says that φ is a *reduction* of $A \subseteq X$ to $B \subseteq Y$, if $\forall_{x\in X}\ x \in A \leftrightarrow \varphi(x) \in B$. Note that if B is in a certain class of the above hierarchies, so is A. For any class $\mathcal{C}$ a set B is *$\mathcal{C}$-hard*, if for any set $A \in \mathcal{C}$ there exists a reduction of A to B. The topological hierarchy is strict for Polish spaces, so if a set is $\mathcal{C}$-hard, it cannot be in any lower class. If a $\mathcal{C}$-hard set B is also an element of $\mathcal{C}$, then it is *$\mathcal{C}$-complete*.

We end this section with three examples which will turn out useful later.

EXAMPLE 1. Let $L_a^{0^*1^\omega} \subseteq T_{\{a,b\}}$ be the set of trees which have at least one a on every path from the set 0^*1^ω. First, observe that $L_a^{0^*1^\omega} = \bigcap_{n<\omega} L_a^{0^n1^\omega}$,

where $L_a^{0^n1^\omega}$ denotes the set of trees which have at least one a on the path 0^n1^ω. Since these languages are open, $L_a^{0^*1^\omega}$ is Π_2^0.

Suppose that it is also a Σ_2^0 set. Let $L_a^{0^*1^\omega} = \bigcup_{n\in\omega} F_n$, F_n is closed for all n. We claim that for every n there exists m_n such that in *every* tree from F_n the letter a occurs in some node 0^n1^m with $m < m_n$. If there was no such number, then we could find a sequence t_k of trees having no letters a in the nodes 0^n1^m for $m < l_k$, where $l_1 < l_2 < l_3 < \ldots$. As $T_{\{a,b\}}$ is compact, there exists a subsequence t_{k_i} convergent in F_n. However the limit of t_{k_i} cannot be in F_n for it has no letter a on the path 0^n1^ω. Now, consider a tree t with a in nodes $0^n1^{m_n+1}$ and b in other nodes. Clearly, $t \in L_a^{0^*1^\omega}$, but $t \notin \bigcup_{n\in\omega} F_n$. This way we have shown that $L_a^{0^*1^\omega} \notin \Sigma_2^0$.

It is well-known that any set in $\Pi_n^0 \setminus \Sigma_n^0$ is Π_n^0-complete. In consequence, $L_a^{0^*1^\omega}$ is Π_2^0-complete. •

EXAMPLE 2. Let $Q = (a^*b)^*a^\omega$. Following the method above one proves that $Q^{\complement} = (a^*b)^\omega \in \Pi_2^0 \setminus \Sigma_2^0$. Hence, $Q \in \Sigma_2^0 \setminus \Pi_2^0$. In particular, Q is Σ_2^0-complete. •

EXAMPLE 3. Let $L_Q^{0^*1^\omega}$ denote the language of trees such that infinite word on the rightmost path from every node of the form 0^* belongs to the language Q defined above. One easily writes $L_Q^{0^*1^\omega}$ as a countable intersection of Σ_2^0-sets. Hence, $L_Q^{0^*1^\omega} \in \Pi_3^0$. This time we will prove directly that it is Π_3^0-complete, and therefore it is not in Σ_3^0. Let us take any $M = \bigcap_{i<\omega} X_i$ with X_i in Σ_2^0. Since Q is Σ_2^0-complete, for each i there exists f_i reducing X_i to Q. One easily defines a continuous reduction of M to $L_Q^{0^*1^\omega}$ assigning to each t a tree having the word $f_i(t)$ on the path 0^i1^ω for all i, and as in all the other nodes. •

2.7 Weak Index vs. Borel Rank

We start the discussion of the relations between the index of a weak automaton and the Borel rank of the language it recognizes by recalling Skurczyński's results. Let us define a sequence of languages:

- $L_{(0,1)} = \{t\}$, where $t \in T_{\{a,b\}}$ is the tree with no b's,
- $L_{(1,n+1)} = L_{(0,n)}^{\complement}$ for $n \geq 1$,
- $L_{(0,n)} = \{t \in T_{\{a,b\}} : \ \forall_k\, t.0^k1 \in L_{(1,n)}\}$ for $n \geq 1$.

Theorem 4 (Skurczyński [34]). *For each $n \geq 1$,*

- *$L_{(0,n)}$ is a Π^0_n-complete language recognized by a weak $(0,n)$-automaton,*
- *$L_{(1,n+1)}$ is a Σ^0_n-complete language recognized by a weak $(1,n+1)$-automaton.*

We will now show that this construction is as efficient as it can be: ranks $(0,n)$ are necessary to recognize any Π^0_n-hard language (if it can be weakly recognized at all).

We will actually prove a bit stronger result. We will consider *weak game languages* $W^\flat_{(\iota,\kappa)}$, to which all languages recognized by weak (ι,κ)-automata can be reduced, and show show that $W^\flat_{(0,n)} \in \Pi^0_n$ and $W^\flat_{(1,n+1)} \in \Sigma^0_n$ (by Skurczyński's results, they are hard for these classes). The languages $W^\flat_{(\iota,\kappa)}$ are natural weak counterparts of strong game languages $W_{(\iota,\kappa)}$ which prove the strictness of the alternating index hierarchy. Lately Arnold and Niwiński proved that the strong game languages also form a strict hierarchy with respect to continuous reductions, but they are all non-Borel [2].

Fix a natural number N. Let $\mathcal{T}_{(\iota,\kappa)}$ denote the set of all N-ary trees over the alphabet $\{\exists,\forall\} \times \{\iota, \iota+1, \ldots, \kappa\}$ with the usual topology. For $\iota = 0, 1$ and $\kappa \geq \iota$ let $W^\flat_{(\iota,\kappa)} \subseteq \mathcal{T}_{(\iota,\kappa)}$ be the set of trees t such that Eve has a winning strategy in the *weak* parity game G_t (see page 23).

Theorem 5. *For all n, $W^\flat_{(0,n)} \in \Pi^0_n(\mathcal{T}_{(\iota,\kappa)})$ and $W^\flat_{(1,n+1)} \in \Sigma^0_n(\mathcal{T}_{(\iota,\kappa)})$.*

Proof. We will proceed by induction on n. For $n = 0$ the claim is obvious: $W^\flat_{(0,0)} = \mathcal{T}_{(0,0)}(\mathcal{T}_{(0,0)}) \in \Pi^0_0$, $W^\flat_{(1,1)} = \emptyset \in \Sigma^0_0(\mathcal{T}_{(1,1)})$.

Take $n > 0$. For each $t \in W^\flat_{(1,n+1)}$ there exists a strategy σ for Eve, such that it guarantees that the play reaches a node with the rank greater or equal to 2. By König lemma, this must happen in a bounded number of moves. Basing on this observation we will provide a Σ^0_n presentation of $W_{(1,n+1)}$.

Let *k-antichain* be a subset of the nodes on the level k. Let $\mathcal{A}$ denote the set of all possible k-antichains for all $k < \omega$. Obviously this set is countable. For a k-antichain A let W_A denote the set of trees such that there exists a strategy for Eve that guarantees visiting a node with the rank ≥ 2 during the initial k moves and reaching a node from A. This set is a clopen. We have a presentation

$$W^\flat_{(1,n+1)} = \bigcup_{A \in \mathcal{A}} \left(W_A \cap \bigcap_{v \in A} \left\{ t : t'.v \in W^\flat_{(0,n-1)} \right\} \right),$$

where t' is obtained from t by decreasing all the ranks by 2 (if the result is -1, take 0). The claim follows by induction hypothesis and the continuity of $t \mapsto t'$ and $t \mapsto t.v$.

Now, it remains to see that $W^\flat_{(0,n)} \in \Pi^0_n(\mathcal{T}_{(0,n)})$. For this, note that

$$W^\flat_{(0,n)} = \left\{ t : t'' \in (W_{(1,n+1)})^{\complement} \right\} ,$$

where t'' is obtained from t by swapping $\exists$ and $\forall$, and increasing ranks by 1. The claim follows by the continuity of $t \mapsto t''$. □

As a corollary we get the promised improvement of Skurczyński's result.

Corollary 1. *Let A be a weak alternating automaton.*

1. *If A is a $(0,n)$-automaton, $L(A) \in \Pi^0_n$.*
2. *If A is a $(1,n+1)$-automaton, $L(A) \in \Sigma^0_n$.*

Proof. Let A be a (ι,κ)-automaton. For sufficiently large N we may assume without loss of generality that the computation trees of the automaton are N-ary. For an input tree t consider a tree t' obtained from A's computation tree on t by replacing each label $q \in Q_\theta$ with $(\theta, \mathrm{rank}(q))$. The mapping $t \to t'$ is a continuous reduction of $L(A)$ to $W^\flat_{(\iota,\kappa)}$. Hence, the claim follows from the theorem above. □

In fact the corollary follows also from Mostowski's theorem on equivalence of weak automata and weak monadic second order logic on trees [17]. The present proof of Theorem 5 is actually just a repetition of Mostowski's proof in the setting of the Borel hierarchy. An entirely different proof can be found in [7].

We conjecture that the converse implication is also true: a weakly recognizable Π^0_n-language can be recognized by a weak $(0,n)$-automaton (and dually for Σ^0_n).

Conjecture 1. *For weakly recognizable languages the weak index hierarchy and the Borel hierarchy coincide.*

Later in this chapter we will see that the conjecture holds true when restricted to deterministic languages.

The results described in this section give yet another argument to one of the opposing parties in the everlasting dispute between the big-endians and the little-endians of game theory. Had we defined a play to be winning for

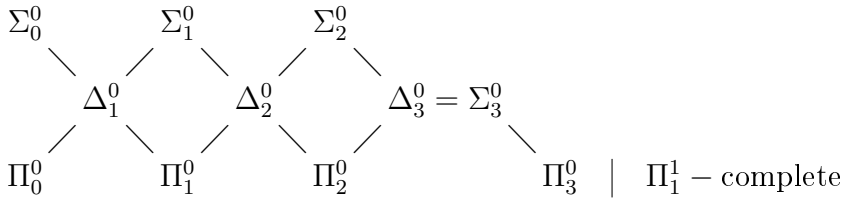


Figure 2.2: The Borel hierarchy for deterministic tree languages.

Eve if the *lowest* rank was even, the correspondence between the indices and Borel classes would be rather ugly.

2.8 Unravelling the Patterns

In this section we begin the investigation of the topological complexity of deterministic languages. We concentrate on the *Borel rank problem*: for a given language compute its exact position in the Borel/projective hierarchy.

In 2002 Niwiński and Walukiewicz discovered a surprising dichotomy in the family of deterministic languages: a deterministic language has either a very low Borel rank or it is not Borel at all (see Fig. 2.2). We say that an automaton A admits a *split* if there are two loops $p \xrightarrow{\sigma,0} p_0 \longrightarrow \ldots \longrightarrow p$ and $p \xrightarrow{\sigma,1} p_1 \longrightarrow \ldots \longrightarrow p$ such that the highest ranks occurring on them are of different parity and the higher one is odd.

Theorem 6 (Niwiński, Walukiewicz [26])**.** *For a deterministic automaton A, $L(A)$ is on the level Π_3^0 of the Borel hierarchy iff A does not admit split; otherwise $L(A)$ is Π_1^1-complete (hence non-Borel).*

Hence, the Borel hierarchy of deterministic languages collapses on the third level. Below Π_3^0 the hierarchy is strict, as follows from the examples of hard languages in Sect. 2.6.

An important tool used in the proof of the Gap Theorem is the technique of difficult patterns. In the topological setting the general recipe goes like this: for given class identify a pattern that can be unravelled to a language complete for this class; if an automaton does not contain the pattern, then $L(A)$ should be in the dual class.

Formalising the unravelling technique will require a few definitions. A *segment* of a tree t between u and uv is the restriction of the function $t.u$ to the set $\mathrm{dom}\,(t.u) \setminus v\{0,1\}^+$. A *partial run* of a deterministic automaton

A is a segment of any run of A. A partial run ρ *realizes* a finite path π in the automaton if it is a segment of an accepting run $\tilde{\rho}$ between two nodes x and y such that $\tilde{\rho}$ agrees with π between x and y. More precisely, if $\pi = p_0 \xrightarrow{\sigma_1,d_1} \dots \xrightarrow{\sigma_m,d_m} p_m$, then $y = xd_1d_1 \dots d_m$, $\tilde{\rho}(x) = p_0$, and $\tilde{\rho}(xd_1 \dots d_i) = p_i$ for $i = 1, \dots, m$. Note that, since ρ is a segment of an accepting run, all its infinite paths are accepting. A tree segment f *realizes* a path π if the corresponding partial run ρ_f realizes π.

Observe that if f_1 realises π_1, f_2 realises π_2, and the last state of π_1 is the same as the first state of π_2, then $f_1 f_2$ realises $\pi_1\pi_2$. A similar property holds for infinite concatenations.

By unravelling a pattern we will understand choosing for each ingredient of a pattern – a path or a loop – one tree fragment that realises it and constructing reductions that only use trees that are concatenations of these tree fragments. Such a concatenation realises some infinite paths in the pattern. Whether it is accepted or rejected by an automaton depends only on those paths, since all the others are accepting. Obviously, elements of the reduced language should be sent to trees realising accepting paths, and the elements of the complement to those realising rejecting paths. In the reductions one node of the input tree (one position of the input word) will correspond to one tree fragment in the resulting concatenation.

In the proof of the Gap Theorem, the split pattern is unravelled into the language of trees having only finitely many 1's on each path. This language is Π^1_1-complete (via a reduction of the set of well-founded trees). We are going to show how other patterns can be unravelled into languages hard for the remaining classes from the above hierarchy. But first, let us introduce one of the most important technical notions of this study. A state p is *replicated* by a loop $q_1 \xrightarrow{\sigma,d_0} q_2 \longrightarrow \dots \longrightarrow q_1$ if there exist a path $q_1 \xrightarrow{\sigma,d_1} q_2' \longrightarrow \dots \longrightarrow p$ such that $d_0 \neq d_1$. We will say that a flower is replicated by a loop λ if it contains a state replicated by λ. The phenomenon of replication is the main difference between trees and words. We will use it constantly to construct hard languages that have no counterparts among word languages. Some of them are listed in the proposition below.

Proposition 4. *Let A be a deterministic automaton.*

1. *If A contains a weak $(1,2)$-flower, $L(A)$ is Σ^0_1-hard.*

2. *If A contains a weak $(0,1)$-flower, $L(A)$ is Π^0_1-hard.*

3. *If A contains a $(0,1)$-flower, $L(A)$ is Σ^0_2-hard.*

4. *If A contains a $(1,2)$-flower or a weak $(1,2)$-flower replicated by an accepting loop, $L(A)$ is Π^0_2-hard.*

5. *If A contains a $(0,1)$-flower replicated by an accepting loop, $L(A)$ is Π^0_3-hard.*

Proof. (1) Let λ_2 be an accepting loop reachable from a rejecting loop λ_1. Let g_1 realize a path from the initial state q_0 to some $q_1 \in \lambda_1$, g_2 realize a path from q_1 to some $q_2 \in \lambda_2$, and f_1, f_2 realize loops λ_1 (from q_1 to q_1), λ_2 (from q_2 to q_2) respectively. Consider $t_n = g_1(f_1)^n g_2(f_2)^\omega$ and $t = g_1(f_1)^\omega$. Clearly, $t_n \in L(A)$ and $t_n \to t$ when $n \to \infty$, but $t \notin L(A)$. Hence $L(A)$ is not closed. Since any non-closed set is hard for Σ^0_1, the claim follows.

(2) This is proved analogously.

(3) Let λ_0 and λ_1 be the loops forming a $(0,1)$-flower, accepting and rejecting respectively. Let q be a state lying on both loops. Let f_0, f_1 be tree segments realizing λ_0 and λ_1 respectively (both from q to q). Consider a map $\varphi : \{0,1\}^\omega \to T_\Sigma$ defined by the formula

$$\varphi(x_0 x_1 x_2 \ldots) = f f_{x_0} f_{x_1} f_{x_2} \cdots ,$$

where f is a tree segment realizing a path from the initial state q_0 to q. The map φ is a continuous reduction of $(0^*1)^*0^\omega$ to $L(A)$. By Example 2 (page 27) $L(A)$ is Σ^0_2-hard.

(4) Unravelling a $(1,2)$-flower gives a reduction of $(1^*2)^\omega$ to $L(A)$, and unravelling a weak $(1,2)$-flower replicated by an accepting loop gives a reduction of $L_a^{0^*1^\omega}$ to $L(A)$. By Examples 1 and 2, in either case $L(A)$ is Π^0_2-hard.

(5) Once again, unravel the pattern to obtain a reduction of Π^0_3-complete language $L_Q^{0^*1^\omega}$ from Example 3 to $L(A)$. □

2.9 The Power of the Weak

In this section we will turn to the weak recognizability of deterministic languages. The Gap Theorem by Niwiński and Walukiewicz has also an index version.

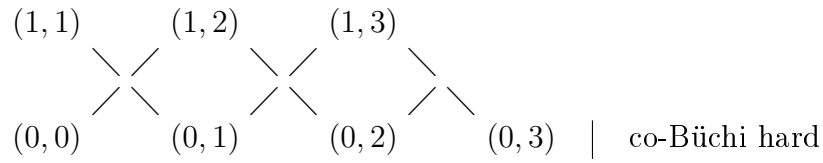


Figure 2.3: The weak index hierarchy for deterministic tree languages.

Theorem 7 (Niwiński, Walukiewicz [26])**.** *For a deterministic automaton A, $L(A)$ can be recognized by a weak $(0,3)$-automaton*[1] *iff A does not admit a split; otherwise $L(A)$ cannot be recognized by a Büchi automaton (hence it cannot be recognized by a weak automaton).*

The weak index hierarchy of deterministic languages is presented on Fig. 2.3.

Observe that the characterisation given in the Gap Theorem is effective: it can be checked (in polynomial time, if the productive states are given explicitly – see page 20) if a given automaton admits a split or not. In the remaining of this section we will give similar conditions sufficient for a deterministic automaton to be equivalent to a weak alternating automaton of index $(0,2)$, $(1,3)$ and $(1,4)$. This is the first step to the solution of the weak index problem for deterministic automata.

Proposition 5. *Each deterministic $(1,2)$-automaton is equivalent to a weak $(0,2)$-automaton.*

Proof. Fix a deterministic $(1,2)$-automaton A. We will construct a weak $(0,2)$-automaton B such that $L(A) = L(B)$. Basically, for each node v the automaton B should check whether on each path in the subtree rooted in v the automaton A will reach a state with rank 2. This can be done as follows. Take two copies of A. In the first copy, all the states are universal and have rank 0. The transitions are like in A plus, for each state $q^{(1)}$ there is an ε-transition to $q^{(2)}$, the counterpart of $q^{(1)}$ in the second copy. In the second copy all states are universal and have rank 1. For the states with rank 1 in A, the transitions are like in A. For the states with rank 2 in A, there is just one transition to an all-accepting state $\top$ (rank 2 in B). □

Before we proceed with the conditions, let us show a useful property of the replication.

[1] In the original paper this index is $(0,2)$ because of a slight difference in the definition of weak automata. The authors let the automata stop after reading a letter for which there is no transition from the current state, which is equivalent to having one extra rank.

Lemma 2 (Replication Lemma)**.** *A state occurs in infinitely many incomparable nodes of an accepting run iff it is productive and is replicated by an accepting loop.*

Proof. If a state p is replicated by an accepting loop, then by productivity one may easily construct an accepting run with infinitely many incomparable occurrences of p. Let us concentrate on the converse implication.

Let p occur in an infinite number of incomparable nodes $v_0, v_1, \ldots$ of an accepting run ρ. Let π_i be a path of ρ going through the node v_i. Since 2^ω is compact, we may assume, passing to a subsequence, that the sequence π_i converges to a path π. Since v_i are incomparable, v_i is not on π. Let the word α_i be the sequence of states labeling the path from the last common node of π and π_i to v_i. Cutting the loops off if needed, we may assume that $|\alpha_i| \leq |Q|$ for all $i \in \omega$. Consequently, there exist a word α repeating infinitely often in the sequence $\alpha_0, \alpha_1, \ldots$. Moreover, the path π is accepting, so the starting state of α must lay on an accepting productive loop. This loop replicates p. □

Proposition 6. *Each deterministic* $(0,1)$*-automaton which contains no weak* $(1,2)$*-flower replicated by an accepting loop is equivalent to a weak* $(1,3)$*-automaton.*

Proof. Let A be a deterministic $(0,1)$-automaton which contains no weak $(1,2)$-flower replicated by an accepting loop. Let us call a state of A *relevant* if it has the highest rank on some loop. We may change the ranks of productive irrelevant states to 0, and assume from now on that all odd states are relevant. We claim that the odd states occur only finitely many times on accepting runs of A. Suppose that an odd state p occurs infinitely many times in an accepting run ρ. Then it must occur in infinitely many incomparable nodes (otherwise we would get a rejecting path). By the Replication Lemma p is replicated by an accepting loop. As p is odd and relevant, it lies on some nontrivial rejecting loop. Since p is also productive, some accepting loop can be reached from p. Hence, A contains a weak $(1,2)$-flower replicated by an accepting loop - a contradiction

Now, we can easily construct a weak $(1,3)$-automaton recognizing $L(A)$. Intuitively, we will simulate A and check if A's odd states occur finitely many times. This can be done as follows. Take three copies of A. In the first copy all the states are universal and have rank 1. The transitions are just like in A, only they go to the second copy of A. In the second copy of A, all the

states are existential and have rank 1. From each state q'' there are two ε-transitions to $q^{(1)}$ in he first copy and to $q^{(3)}$ in the third copy. Finally, in the third copy of A all the states are universal and have rank 2. The transitions from the states ranked 0 in A are just like in A, and from the states ranked 1 in A they go to an all-rejecting state $\perp$ (rank 3 in B). It is easy to see that B recognizes $L(A)$. □

Proposition 7. *Each deterministic automaton containing no* $(0,1)$*-flower replicated by an accepting loop is equivalent to a weak alternating* $(1,4)$*-automaton.*

Proof. Let A be an automaton without $(0,1)$-flower replicated by an accepting loop. Consider the DAG of strongly connected components of A. For each SCC X containing at least one loop we will construct a weak automaton B_X recognizing the languages of trees t such that each path of A's run on t that enters X either leaves X or is accepting. Obviously, the conjunction of such automata recognizes exactly $L(A)$. Let us first consider components replicated by an accepting loop. By the hypothesis, such a component must not contain a $(0,1)$-flower. Therefore we may assume that X only uses ranks 1 and 2. To obtain B_X take a copy of A. The states outside X can be divided into three disjoint groups: those that can be reached from X, those from which X can be reached, and the rest. Give the states from the first group the rank 4, and the states from the second and third group the rank 2. Finally, following the method from Proposition 5, replace X with an equivalent weak alternating subautomaton using ranks 2,3, and 4.

The case of X not replicated by an accepting loop is more tricky. The key property follows from the Replication Lemma. Let ρ_X denote the restriction of the run ρ to the nodes labeled with a state from X or having a descendant labeled with a state from X. By the Replication Lemma, this tree has only finitely many branches (some of them may be infinite). What B_X should do is to guess a node v on each path such that in the subtree rooted in v, ρ_X is either empty or consists of one infinite accepting branch. In the latter case we may additionally demand that on this infinite path the highest rank that ever occurs, occurs infinitely many times.

B_X consists of the component C_{guess} realising the guessing, the component $C_{A\setminus X}$ checking that no path of the computation enters X, and components $C_{X,r}$ for all ranks r used in X, which check that in a given subtree of the run ρ there is exactly one branch of ρ_X and that on this branch r occurs infinitely often and no higher rank is used.

To construct C_{guess}, take a copy of A and declare all the states universal and set their ranks to 1. For each q add a fresh existential state q' of rank 1 with an ε-transition to q and either to $q^{A\setminus X} \in C_{A\setminus X}$ if $q \notin X$ (ρ_X is empty) or to $q^{X,r} \in C_{X,r}$ for all r if $q \in X$ (ρ_X is one infinite accepting path). Finally replace each transition $p \xrightarrow{\sigma} p_0, p_1$ with $\xrightarrow{\sigma} p'_0, p'_1$.

The component $C_{A\setminus X}$ is a copy of A with all ranks equal 2, and the SCC X replaced with one all-rejecting state $\perp$ with rank 3.

Finally, let us now describe the automaton $C_{X,r}$. The automaton, staying in rank 2, works its way down the input tree just like A would, with the following modifications:

- if A enters a state in X with rank greater than r, $C_{X,r}$ moves to an all rejecting state $\perp$ (rank 3),
- if A takes a transition exiting X on both branches or staying in X on both branches, $C_{X,r}$ moves to $\perp$,
- if A takes a transition whose left branch leaves X and the right branch stays inside, $C_{X,r}$ sends to the right a $(3,4)$-component looking for a state from X with the rank r, and moves on to the right subtree (and symmetrically).

In order two see that $C_{X,r}$ does the job, it is enough to observe that if the $(3,4)$ component always succeeds to find a state from X with the rank r, then on the unique path that stays forever in X the rank r repeats infinitely often. □

2.10 Solution

We have now collected all the ingredients for the solution of the weak index problem and the Borel rank problem for deterministic languages. So far we have sufficient conditions for index easiness and Borel hardness. We will now use Corollary 1 to glue them together and see that they are in fact also necessary conditions.

Theorem 8. *For deterministic tree languages the Borel hierarchy and the weak index hierarchy coincide (see Fig. 2.4) and are decidable in polynomial time.*

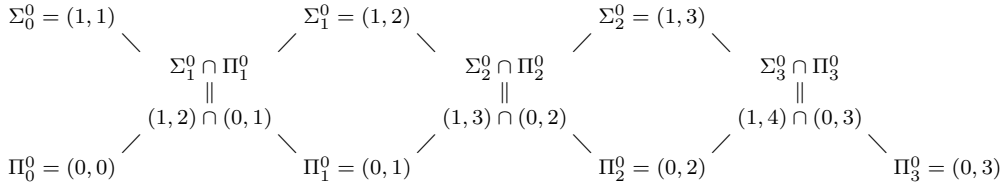


Figure 2.4: For deterministic tree languages the hierarchies coincide.

Proof. We will abuse the notation and write (ι, κ) to denote the class of languages recognized by weak (ι, κ)-automata. All the classes considered here are relativised to the deterministic languages.

By the two versions of the Gap Theorem we have the equality and decidability of the classes of the classes Π_3^0 and $(0,3)$.

Let us continue with the third level. Let us see that $\Sigma_3^0 = (1,4)$. We will show that both these classes are equal to the class of languages recognized by deterministic automata without a $(0,1)$-flower replicated by an accessible loop. If a deterministic automaton A does not contain this pattern, then it is equivalent to a weak $(1,4)$-automaton and by Corollary 1 recognizes a Σ_3^0 language. If A does contain this pattern, then by Proposition 4 it is not Σ_3^0 and so is not equivalent to a weak $(1,4)$-automaton. The decidability follows easily, since checking for the pattern above can be done effectively (in polynomial time).

For the equality $\Pi_2^0 = (0,2)$, prove that both classes are equal to the class of languages recognized by deterministic automata without a $(0,1)$-flower. Proceed just like before, only use Proposition 5 instead of Proposition 7. Analogously, using Proposition 6, show that both Σ_2^0 and $(1,3)$ are equal to the class of languages recognized by deterministic automata admitting neither a $(1,2)$-flower nor a weak $(1,0)$-flower replicated by an accepting loop.

For the first level use the characterisation given by Proposition 3. The level zero is trivial. □

Chapter 3

Wadge Ordering

Topological hierarchies stormed into the theory of formal languages with Klaus Wagner's fundamental works on regular word languages [40, 41]. The incredible coincidence of the Wagde ordering and the index hierarchy for these languages encouraged further investigation of the Wadge ordering of wider classes of word languages, corresponding to more powerful recognizing devices: push-down automata and Turing machines [6, 10, 33].

The beauty of these results inspires the search for a complete picture of the two hierarchies and the relations between them for recognizable tree languages. As a first step towards this end, we deal with the deterministic case. The index hierarchy for deterministic tree automata has already been explored thoroughly [27, 37]. Here, we investigate the Wadge ordering of deterministic tree languages.

3.1 Scenario

The notion of continuous reduction defined in the last chapter yields a preordering on sets. Let X and Y be topological spaces, and let $A \subseteq X$, $B \subseteq Y$. We write $A \leq_W B$ (to be read "A is *Wadge reducible* to B"), if there exists a continuous reduction of A to B, i. e., a continuous function $\varphi\colon X \to Y$ such that $A = \varphi^{-1}(B)$. We say that A is *Wadge equivalent* to B, in symbols $A \equiv_W B$, if $A \leq_W B$ and $A \leq_W B$, and $A <_W B$ if $A \leq_W B$ and $B \not\leq_W A$. The *Wadge ordering* is the ordering induced by $\leq_W$ on the $\equiv_w$-classes of subsets of Polish spaces. The Wadge ordering restricted to Borel sets is called the *Wadge hierarchy*.

In this study we only work with the spaces T_Σ and Σ^ω. Since we only consider finite Σ, these spaces are homeomorphic with the Cantor discontinuum $\{0,1\}^\omega$ as long as $|\Sigma| \geq 2$. In particular, all the languages we consider are Wadge equivalent to subsets of $\{0,1\}^\omega$. Note however that the homeomorphism need not preserve recognizability. In fact, no homeomorphism from T_Σ to $\{0,1\}^\omega$ does: the Borel hierarchy for regular tree languages is infinite, but for words it collapses on Δ^0_3. In other words, there are regular tree languages (even weak, or deterministic), which are not Wadge equivalent to regular word languages. Conversely, each regular word language L is Wadge equivalent to a deterministic tree language L' consisting of trees which have a word from L on the leftmost branch. As a consequence, the height of the Wadge ordering of regular word languages gives as a lower bound for the case of deterministic tree languages, and this is essentially everything we can conclude from the word case.

The remaining part of this chapter is in fact one long proof. The aim of this short section is to help the reader see it this way. The theorem we want to prove says that the Wadge ordering is decidable for deterministic tree languages. We could have tried to work on it directly, by comparing carefully the structure of two given automata, but we have chosen a different approach. We provide a collection of canonical deterministic automata, one for each $\equiv_W$-class that contains a deterministic tree language. Then we describe the ordering of the canonical automata induced by $\leq_W$. Finally, we provide an algorithm that for a given automaton constructs an equivalent canonical one, and so computes the $\equiv_W$ class of the language recognized by the automaton. This way we also get the height of the Wadge hierarchy restricted to deterministic tree languages.

The proof is divided into Sections 3.2 – 3.12. We start by reformulating the classical criterion of reducibility via Wadge games in terms of automata (Sect. 3.2). This will be the main tool of the whole argument. Then we define four ways of composing automata: sequential composition $\oplus$, parallel composition $\wedge$, alternative $\vee$, and (ι,κ)-replication $\xrightarrow{(\iota,\kappa)}$ (Sect. 3.3). The operations $\oplus$, $\vee$, and $\xrightarrow{(1,1)}$ are used to construct the canonical automata: we define automata C_α for each $0 < \alpha < \omega^{\omega\cdot 3}$, and D_α, E_α for $0 < \alpha < \omega$ or $\alpha = \omega^{\omega\cdot 2}\alpha_2 + \omega^\omega\alpha_1 + n$ with $\alpha_2 < \omega^\omega$, $0 < \alpha_1 < \omega^\omega$, and $n < \omega$ (Sect. 3.4). After some preparatory remarks (Sect. 3.5 and 3.6), we prove that the canonical automata form a strict hierarchy, roughly speaking, coinciding with the usual order on their ordinal indices (Sect. 3.7). This completes the

first part of the proof, the description of the hierarchy.

Next, we need to show the our hierarchy contains all deterministic languages (up to Wadge equivalence). Once again we turn to the methodology of patterns used widely in Chapter 2. We introduce a fundamental notion of admittance, which formalizes what it means to contain an automaton as a pattern (Sect. 3.8). Then we extend the family of the canonical automata with three more elements forming the top of the hierarchy, and rephrase the results on the Borel hierarchy and the Wagner hierarchy in terms of admittance of canonical automata (Sect. 3.9). Basing on these results, we show that the family of canonical automata is closed by the composition operations from Sect. 3.3 (Sect. 3.10), and prove the Completeness Theorem asserting that (up to Wadge equivalence) each deterministic automaton may be obtained by composing C_1 and D_1 using those operations (Sect. 3.11). As a consequence, each deterministic automaton is equivalent to a canonical one, which concludes the second part of the proof.

Finally, from the proof of the Completeness Theorem, we extract an algorithm calculating the equivalent canonical automata (Sect. 3.12). This solves the problem: given two deterministic automata, A and B, we decide if $L(A) \leq_W L(B)$ by comparing the equivalent canonical automata.

3.2 Games and Automata

A classical criterion for reducibility is based on the notion of *Wadge games*. Let us introduce a tree version of Wadge games (see [29] for word version). For any pair of tree languages $L \subseteq T_{\Sigma_1}, M \subseteq T_{\Sigma_2}$ the game $G_W(L, M)$ is played by Spoiler and Duplicator. Each player builds a tree, $t_S \in T_{\Sigma_1}$ and $t_D \in T_{\Sigma_2}$ respectively. In every round, first Spoiler adds at least one complete level to t_S and then Duplicator can either add some levels to t_D or skip a round (not forever). Duplicator wins the game if $t_S \in L \iff t_D \in M$. We say that Spoiler *is in charge of* L, and Duplicator *is in charge of* M.

Just like for the classical Wadge games, a winning strategy for Duplicator can be easily transformed into a continuous reduction, and vice versa.

Lemma 3. *Duplicator has a winning strategy in $G_W(L, M)$ if and only if $L \leq_W M$.*

Proof. A strategy for Duplicator defines a reduction in an obvious way. Conversely, suppose there exist a reduction $t \mapsto \varphi(t)$. It follows that there exist

a sequence n_k (without loss of generality, increasing) such that the level k of $\varphi(t)$ depends only on the levels $1, \ldots, n_k$ of t. Then the strategy for Duplicator is the following: if the number of the round is n_k, choose the k-th level of t_D according to φ; otherwise skip. □

We would like to point out that Wadge games are much less interactive than classical games. The move made by one player has no influence on the possible moves of the other. Of course, if one wants to win, one has to react to the opponent's actions, but the responses need not be immediate. As long as the player keeps putting some new letters, he may postpone the real reaction until he knows more about the opponent's plans. Because of that, we will often speak about strategies for some language without considering the opponent and even without saying if the player in charge of the language is Spoiler or Duplicator.

Since we only want to work with deterministically recognizable languages, let us redefine the games in terms of automata. Let A, B be deterministic tree automata as defined in Sect. 2.2. The *automata game* $G(A, B)$ starts with one token put in the initial state of each automaton. In every round players perform a finite number of the following actions:

fire a transition – for a token placed in a state q choose a transition $q \xrightarrow{\sigma} q_l, q_2$, take the old token away from q and put new tokens in q_1 and q_2,

remove – remove a token placed in a state different from $\bot$.

Spoiler plays on A and must perform one of these actions at least for all the tokens produced in the previous round. Duplicator plays on B and is allowed to postpone performing an action for a token, but not forever. Let us first consider plays in which the players never remove tokens. The paths visited by the tokens of each player define a run of the respective automaton. We say that Duplicator wins a play if both runs are accepting or both are rejecting. Now, removing a token from a state p is interpreted as plugging in an accepting subrun in the corresponding node of the constructed run. So, Duplicator wins if the runs obtained by plugging in an accepting subrun for every removed token are both accepting or both rejecting.

Observe that removing tokens in fact does not give any extra power to the players: instead of actually removing a token, a player may easily pick an accepting subrun, and in future keep realizing it level by level in the constructed run. The only reason for adding this feature in the game is that it simplifies the strategies. In a typical strategy, while some tokens have a

significant role to play, most are just moved along a trivially accepting path. It is convenient to remove them right off and keep concentrated on the real actors of the play.

We will write $A \leq B$ if Duplicator has a winning strategy in $G(A, B)$. Like for languages, define $A \equiv B$ iff $A \leq B$ and $A \geq B$. Finally, let $A < B$ iff $A \leq B$ and $A \not\geq B$.

Lemma 4. *For all deterministic tree automata A and B,*

$$A \leq B \iff L(A) \leq_W L(B)\,.$$

Proof. Suppose that Duplicator has a winning strategy in $G(A, B)$. We will show that Duplicator has a winning strategy in $G_W(L(A), L(B))$, and hence $L(A) \leq_W L(B)$. What Duplicator should do is to simulate a play of $G(A, B)$ in which an imaginary Spoiler keeps constructing the run of A on the tree t_S constructed by the real Spoiler in $G_W(L(A), L(B))$, and Duplicator replies according to his winning strategy that exists by hypothesis. In $G_W(L(A), L(B))$ Duplicator should simply construct a tree such that B's run on it is exactly Duplicator's tree from $G(A, B)$.

For the converse implication, Duplicator should simulate a play in the game $G_W(L(A), L(B))$ in which Spoiler keeps constructing a tree such that A's run on it is exactly the tree constructed by the real Spoiler in $G(A, B)$, and Duplicator replies according to his winning strategy. In $G(A, B)$ Duplicator should keep constructing the run of B on t_D constructed in the simulated play. □

As a corollary we have that all automata recognising a given language have the same "game power".

Corollary 2. *For deterministic tree automata A and B, if $L(A) = L(B)$, then $A \equiv B$.*

Classically, in automata theory we are interested in the language recognized by an automaton. One language may be recognized by many automata and we usually pick the automaton that fits best our purposes. Here, the approach is entirely different. We are not interested in the language itself, but in its Wadge equivalence class. This, as it turns out, is reflected in the general structure of the automaton. Hence, our main point of interest will be that structure.

We will frequently modify an automaton in a way that does change the recognized language, but only within one $\equiv_W$-class. One typical thing we

need to do with an automaton, is to treat it as an automaton over an extended alphabet in such a way, that the new recognized language is Wadge equivalent to the original one. This has to be done with some care, since the automaton is required to have transitions by each letter from every state. Suppose we want to extend the input alphabet by a fresh letter τ. Let us construct an automaton A_τ. First, if A has the all-rejecting state $\bot$, add a transition $\bot \xrightarrow{\tau} \bot, \bot$. Then add an all-accepting state $\top$ with transitions $\top \xrightarrow{\sigma} \top, \top$ for each $\sigma \in \Sigma \cup \{\tau\}$ (if A already has the state $\top$, just add a transition $\top \xrightarrow{\tau} \top, \top$). Then for each $p \notin \{\bot, \top\}$, add a transition $p \xrightarrow{\tau} \top, \top$.

Lemma 5. *For every deterministic tree automaton A over Σ and a letter $\tau \notin \Sigma$, $A \equiv A_\tau$.*

Proof. It is obvious that $A \leq A_\tau$: since A_τ contains all transitions of A, a trivial winning strategy for Duplicator in $G(A, A_\tau)$ is to copy Spoiler's actions. Let us see that new transitions do not give any real power. Consider $G(A_\tau, A)$. While Spoiler uses old transitions, Duplicator may again copy his actions. The only difficulty lies in responding to a move that uses a new transition. Suppose Spoiler does use a new transition. If Spoiler fires a transition $p \xrightarrow{\tau} \top, \top$ for a token x in a state $p \neq \bot$, Duplicator simply removes the corresponding token in p, and ignores the further behaviour of x and all his descendants. The only other possibility is that Spoiler fires $\bot \xrightarrow{\tau} \bot, \bot$. Then for the corresponding token Duplicator should fire $\bot \xrightarrow{\sigma} \bot, \bot$ for some $\sigma \in \Sigma$. The described strategy is clearly winning for Duplicator. □

An automaton for us is not as much a recognizing device, as a device to carry out strategies. Therefore even two automata with substantially different structure may be equivalent, as long as they enable us to use the same set of strategies. A typical thing we will be doing, is to replace a part of an automaton with a different part that gives the same strategical possibilities. Recall that by A_q we denote the automaton A with the initial state changed to q. For $q \in Q^A$ let $A_{q:=B}$ denote the automaton obtained from a copy of A and a copy of B by replacing each A's transition of the form $p \xrightarrow{\sigma,d} q$ with $p \xrightarrow{\sigma,d} q_0^B$. Note that $A_{q:=A_q}$ is equivalent to A.

Lemma 6 (Substitution Lemma)**.** *Let A, B, C be deterministic automata with pairwise disjoint sets of states, and let p be a state of C. If $A \leq B$, then $C_{p:=A} \leq C_{p:=B}$.*

Proof. Consider the game $G(C_{p:=A}, C_{p:=B})$ and the following strategy for Duplicator. In C Duplicator copies Spoiler's actions. If some Spoiler's token x enters the automaton A, Duplicator should put its counterpart y in the initial state of B, and then y and its descendants should use Duplicator's winning strategy from $G(A, B)$ against x and its descendants.

Let us see that this strategy is winning. Suppose first that Spoiler's run is rejecting. Then there is a rejecting path, say π. If on π the computation stays in C, in Duplicator's run π is also rejecting. Suppose π enters A. Let v be the first node of π in which the computation is in A. The subrun of Spoiler's run rooted in v is a rejecting run of A. Since Duplicator was applying a winning strategy form $G(A, B)$, the subrun of Duplicator's run rooted in v is also rejecting. In either case, Duplicator's run is rejecting.

Now assume that Spoiler's run is accepting, and let us see that so is Duplicator's. All paths staying in C are accepting, because they are identical to the paths in Spoiler's run. For every v in which the computation enters B, the subrun rooted in v is accepting thanks to the winning strategy form $G(A, B)$ used to construct it. □

3.3 Operations

It this section we introduce four operations that will be used to construct canonical automata representing all Wadge degrees of deterministic tree languages.

The first operation yields an automaton that lets a player choose between A and B. For two deterministic tree automata A and B over Σ, the *alternative* $A \vee B$ is an automaton with the input alphabet $\Sigma \cup \{a, b\}$ consisting of disjoint copies of A and B over the extended alphabet $\Sigma \cup \{a, b\}$, $A_{a,b}$ and $B_{a,b}$, and a fresh initial state q_0 with transitions

$$q_0 \xrightarrow{a} q_0^{A_{a,b}}, \top\,, \quad q_0 \xrightarrow{b} q_0^{B_{a,b}}, \top\,, \quad \text{and} \quad q_0 \xrightarrow{\sigma} \top, \top \quad \text{for} \quad \sigma \notin \{a, b\}$$

(only if $L(A) = L(B) = \emptyset$ put $q_0 \xrightarrow{\sigma} \bot, \bot$). By Lemma 6, $\equiv$ is a congruence with respect to $\vee$. Furthermore, $L(A \vee B)$ is Wadge equivalent to the disjoint union of $L(A)$ and $L(B)$, so $\vee$ is associative and commutative up to $\equiv$. Multiple alternatives are performed from left to right:

$$A_1 \vee A_2 \vee A_3 \vee A_4 = ((A_1 \vee A_2) \vee A_3) \vee A_4\,.$$

The *parallel composition* $A \wedge B$ is defined analogously, only now we extend the alphabet only by a and add transitions

$$q_0 \xrightarrow{a} q_0^A, q_0^B\,, \quad \text{and} \quad q_0 \xrightarrow{\sigma} \top, \top \quad \text{for} \quad \sigma \neq a$$

(only if $L(A) = \emptyset$ or $L(B) = \emptyset$, put $q_0 \xrightarrow{\sigma} \bot, \bot$). Note that, while in $A \vee B$ the computation must choose between A and B, here it continues in both. Again, $\equiv$ is a congruence with respect to $\wedge$. The language $L(A \wedge B)$ is Wadge equivalent to $L(A) \times L(B)$ and $\wedge$ is associative and commutative up to $\equiv$. Multiple parallel compositions are performed from left to right, and for $n > 0$ the symbol $(A)^n$ denotes $\underbrace{A \wedge \ldots \wedge A}_{n}$.

The *canonical* (ι, κ)*-flower* $F_{(\iota,\kappa)}$ is an automaton with the input alphabet $\{a_\iota, a_{\iota+1} \ldots, a_\kappa\}$, the states $q_\iota, q_{\iota+1}, \ldots, q_\kappa$ where the initial state is q_ι and $\mathrm{rank}(q_i) = i$, and transitions

$$q_\iota \xrightarrow{a_\iota} q_\iota, \top\,, \quad q_\iota \xrightarrow{a_j} q_j, \top\,, \quad q_j \xrightarrow{a_j} q_0, \top\,, \quad \text{and} \quad q_j \xrightarrow{a_k} \top, \top$$

for $j = \iota + 1, \iota + 2, \ldots, \kappa$ and $k \neq j$. A flower $F_{(\iota,\kappa)}$ is *nontrivial* if $\iota < \kappa$.

Let $A, A_\iota, \ldots, A_\kappa$ be deterministic tree automata over Σ. The (ι, κ)-*replication* $A \xrightarrow{(\iota,\kappa)} A_\iota, \ldots, A_\kappa$ is obtained as follows. Take a copy of $F_{(\iota,\kappa)}$ over the extended alphabet $\{a_\iota, a_{\iota+1} \ldots, a_\kappa\} \cup \Sigma \cup \{b\}$, where b is a fresh letter. Add single disjoint copies of $A_\iota, \ldots, A_\kappa$ and A over the extended alphabet $\Sigma \cup \{a_\iota, a_{\iota+1} \ldots, a_\kappa\} \cup \{b\}$. Finally, in $F_{(\iota,\kappa)}$ over the extended alphabet, replace the transition $q_\iota \xrightarrow{b,0} r$ (where $r \in \{\bot, \top\}$) with $q_\iota \xrightarrow{b,0} q_0^A$, and $q_\iota \xrightarrow{a_i,1} \top$ with $q_\iota \xrightarrow{a_i,1} q_0^{A_i}$ for $i = \iota, \ldots, \kappa$. Again, using Lemma 6 it is easy to see that the $\equiv$-class of the defined automaton depends only on (ι, κ) and the $\equiv$-classes of $A, A_\iota, \ldots, A_\kappa$. Hence, $\equiv$ is a congruence with respect to $\xrightarrow{(\iota,\kappa)}$ for every (ι, κ).

The last operation we define produces out of A and B an automaton that behaves as A, but in at most one point (on the leftmost path) may switch to B. Consider the DAG of strongly connected components of the automaton A. A component X is *leftmost* if no path connecting the initial state with a state in X uses a right transition. In other words, no run of A contains a state from X outside of the leftmost branch. In particular, all transitions within X are left. A leftmost component that is a leaf in the DAG of SCCs is called a *tail component*. We will also use the term *head component* to denote the root of the DAG of SCCs. Note that, while each automaton

has a unique head component, it may have many or none tail components. For deterministic tree automata A and B over Σ, the *sequential composition* $A \oplus B$ is an automaton with the input alphabet $\Sigma \cup \{b\}$, where b is a fresh letter. It is constructed by taking copies of A and B over the extended alphabet $\Sigma \cup \{b\}$ and replacing the transition $p \xrightarrow{b,0} r$ with $p \xrightarrow{b,0} q_0^{B_b}$ for all p in tail components of A and $r \in \{\bot, \top\}$. Like for $\wedge$ and $\vee$, we perform the multiple sequential compositions from left to right. For $n > 0$ we often use an abbreviation $nA = \underbrace{A \oplus \ldots \oplus A}_{n}$. Since the tail components of $A \oplus B$ are exactly those of B, the operation $\oplus$ is associative up to a permutation of the letters freshly added to the input alphabet. In particular, the $\equiv$-class of a multiple sequential composition does not depend on the way we put parentheses. An analog of $\oplus$ for word automata defines an operation on $\equiv$-classes, but for tree automata this is no longer true. We will also see later that $\oplus$ is not commutative even up to $\equiv$.

The priority of the operations is $\oplus, \wedge, \vee, \xrightarrow{(\iota,\kappa)}$. For instance $A_1 \to A_2 \oplus A_3 \wedge A_4 \vee A_5 = A_1 \to (((A_2 \oplus A_3) \wedge A_4) \vee A_5)$. Nevertheless, we usually use parentheses to make the expressions easier to read.

In the definitions above we often use an all-accepting state $\top$. This is in fact a way of saying that a transition is of no importance when it comes to possible strategies: a token moved to $\top$ has no use later in the play. Therefore, we may assume that players remove their tokens instead of putting them to $\top$. In particular, when a transition is of the form $p \xrightarrow{\sigma} q, \top$, it is convenient to treat it as a "left only" transition in which no new token is created, only the old token is moved from p to q. Consequently, when analyzing games on automata, we will ignore the transitions to $\top$.

3.4 Canonical Automata

In this section we define canonical automata, which – as we will later see – represent all $\equiv_W$-classes realised by deterministic tree languages, save for three which will be defined later. For each $\alpha < \omega^{\omega \cdot 3}$ we define the *canonical automaton* C_α. For some values of α we also define canonical automata D_α and E_α. All the defined automata have at least one tail component, so the operation $\oplus$ is always non-trivial.

Let $C_1 = F_{(0,0)}$, $D_1 = F_{(1,1)}$, and $E_1 = F_{(0,0)} \vee F_{(1,1)}$. For $1 < \alpha < \omega$ define

$$\begin{aligned} C_\alpha &= C_1 \oplus (\alpha - 1)E_1 \,, \\ D_\alpha &= D_1 \oplus (\alpha - 1)E_1 \,, \\ E_\alpha &= \alpha E_1 \,. \end{aligned}$$

For $\omega \leq \alpha < \omega^\omega$ we only define C_α. In what follows we use $\rightarrow$ to denote the operation $\xrightarrow{(1,1)}$. Let $C_\omega = C_1 \rightarrow C_3$ and $C_{\omega^{k+1}} = C_1 \rightarrow (C_1 \oplus C_{\omega^k})$ for $1 \leq k < \omega$. For every α from the considered range we have a unique presentation $\alpha = \omega^{l_k} n_k + \omega^{l_{k-1}} n_{k-1} \ldots + \omega^{l_0} n_0$, with $\omega > l_k > 0$, $l_k > l_{k-1} > \ldots l_0$ and $0 < n_i < \omega$. For $l_0 = 0$ define

$$\begin{aligned} C_\alpha &= C_{n_0} \oplus n_1 C_{\omega^{l_1}} \oplus \ldots \oplus n_k C_{\omega^{l_k}} \quad \text{for odd } n_0 \,, \\ C_\alpha &= D_{n_0} \oplus n_1 C_{\omega^{l_1}} \oplus \ldots \oplus n_k C_{\omega^{l_k}} \quad \text{for even } n_0 \,, \end{aligned}$$

and for $l_0 > 0$ set

$$C_\alpha = n_0 C_{\omega^{l_0}} \oplus n_1 C_{\omega^{l_1}} \oplus \ldots \oplus n_k C_{\omega^{l_k}} \,.$$

Now consider $\omega^\omega \leq \alpha < \omega^{\omega \cdot 2}$. For $k < \omega$ let $C_{\omega^{\omega+k}} = F_{(0,k+1)}$, $D_{\omega^{\omega+k}} = F_{(1,k+2)}$ and $E_{\omega^{\omega+k}} = F_{(0,k+1)} \vee F_{(1,k+2)}$. For every α from the considered range we have a unique presentation $\alpha = \omega^\omega \alpha_1 + \alpha_0$ with $\alpha_0, \alpha_1 < \omega^\omega$ and $\alpha_1 > 0$. Let $\alpha_1 = \omega^{l_k} n_k + \omega^{l_{k-1}} n_{k-1} \ldots + \omega^{l_0} n_0$, with $\omega > l_k > l_{k-1} > \ldots l_0$ and $0 < n_i < \omega$. For $\alpha_0 = 0$ and $l_0 = 1$ let

$$\begin{aligned} C_\alpha &= C_{\omega^{\omega+l_0}} \oplus n_1 E_{\omega^{\omega+l_1}} \oplus \ldots \oplus n_k E_{\omega^{\omega+l_k}} \,, \\ D_\alpha &= D_{\omega^{\omega+l_0}} \oplus n_1 E_{\omega^{\omega+l_1}} \oplus \ldots \oplus n_k E_{\omega^{\omega+l_k}} \,, \\ E_\alpha &= E_{\omega^{\omega+l_0}} \oplus n_1 E_{\omega^{\omega+l_1}} \oplus \ldots \oplus n_k E_{\omega^{\omega+l_k}} \,, \end{aligned}$$

for $\alpha_0 = 0$ and $l_0 > 1$ let

$$\begin{aligned} C_\alpha &= C_{\omega^{\omega+l_0}} \oplus (n_0 - 1) E_{\omega^{\omega+l_0}} \oplus n_1 E_{\omega^{\omega+l_1}} \oplus \ldots \oplus n_k E_{\omega^{\omega+l_k}} \,, \\ D_\alpha &= D_{\omega^{\omega+l_0}} \oplus (n_0 - 1) E_{\omega^{\omega+l_0}} \oplus n_1 E_{\omega^{\omega+l_1}} \oplus \ldots \oplus n_k E_{\omega^{\omega+l_k}} \,, \\ E_\alpha &= n_0 E_{\omega^{\omega+l_0}} \oplus n_1 E_{\omega^{\omega+l_1}} \oplus \ldots \oplus n_k E_{\omega^{\omega+l_k}} \,, \end{aligned}$$

for $\omega > \alpha_0 > 0$ let

$$\begin{aligned} C_\alpha &= C_{\alpha_0} \oplus E_{\omega^\omega \alpha_1} \,, \\ D_\alpha &= D_{\alpha_0} \oplus E_{\omega^\omega \alpha_1} \,, \\ E_\alpha &= E_{\alpha_0} \oplus E_{\omega^\omega \alpha_1} , \end{aligned}$$

and for $\alpha_0 > \omega$ let

$$C_\alpha \;=\; C_{\alpha_0} \oplus E_{\omega^\omega \alpha_1}\,.$$

Finally consider $\omega^{\omega\cdot 2} \le \alpha < \omega^{\omega\cdot 3}$. Let $C_{\omega^{\omega\cdot 2}} = C_1 \to F_{(0,2)}$, and for $k < \omega$ let $C_{\omega^{\omega\cdot 2+k+1}} = C_1 \to (C_1 \oplus C_{\omega^{\omega\cdot 2+k}})$. We have a unique presentation $\alpha = \omega^{\omega\cdot 2}\alpha_2 + \omega^\omega \alpha_1 + \alpha_0$ with $\alpha_0, \alpha_1, \alpha_2 < \omega^\omega$ and $\alpha_2 > 0$. Let $\alpha_2 = \omega^{l_k} n_k + \omega^{l_{k-1}} n_{k-1} \ldots + \omega^{l_0} n_0$, with $\omega > l_k > l_{k-1} > \ldots l_0$ and $0 < n_i < \omega$. For $\alpha_0 = \alpha_1 = 0$ let

$$C_\alpha \;=\; n_0 C_{\omega^{\omega\cdot 2+l_0}} \oplus n_1 C_{\omega^{\omega\cdot 2+l_1}} \oplus \ldots \oplus n_k C_{\omega^{\omega\cdot 2+l_k}}\,,$$

for $\alpha_0 = 0$ and $\alpha_1 > 0$ let

$$\begin{aligned} C_\alpha &= C_{\omega^\omega \alpha_1} \oplus C_{\omega^{\omega\cdot 2}\alpha_2}\,,\\ D_\alpha &= D_{\omega^\omega \alpha_1} \oplus C_{\omega^{\omega\cdot 2}\alpha_2}\,,\\ E_\alpha &= E_{\omega^\omega \alpha_1} \oplus C_{\omega^{\omega\cdot 2}\alpha_2}\,, \end{aligned}$$

for $\omega > \alpha_0 > 0$ and $\alpha_1 = 0$ let

$$\begin{aligned} C_\alpha &= C_{\alpha_0} \oplus C_{\omega^{\omega\cdot 2}\alpha_2} \quad \text{for odd } \alpha_0\,,\\ C_\alpha &= D_{\alpha_0} \oplus C_{\omega^{\omega\cdot 2}\alpha_2} \quad \text{for even } \alpha_0\,, \end{aligned}$$

and in the remaining case ($\alpha_0 > \omega$ or $\alpha_1 > 0$) let

$$C_\alpha \;=\; C_{\omega^\omega \alpha_1 + \alpha_0} \oplus C_{\omega^{\omega\cdot 2}\alpha_2}\,.$$

Let $\mathcal{C}$ denote the family of the canonical automata, i. e.,

$$\begin{aligned} \mathcal{C} \;=\;& \{C_\alpha : \alpha < \omega^{\omega\cdot 3}\} \cup \{D_n, E_n \colon n < \omega\} \cup\\ & \cup \{D_{\omega^{\omega\cdot 2}\alpha_2 + \omega^\omega \alpha_1 + n}, E_{\omega^{\omega\cdot 2}\alpha_2 + \omega^\omega \alpha_1 + n} \colon 0 < \alpha_1 < \omega^\omega\,,\ \alpha_2 < \omega^\omega\,,\ n < \omega\}\,. \end{aligned}$$

In the next three sections we will investigate the order induced on $\mathcal{C}$ by the Wadge ordering of the recognized languages.

Now, let us discuss briefly the taxonomy of the canonical automata. *Simple* automata are those canonical automata that cannot be decomposed with respect to $\oplus$, i. e., the automata on the levels ω^k, $\omega^{\omega+k}$, and $\omega^{\omega\cdot 2+k}$ for $k < \omega$. *Complex* automata are those obtained from simple ones by means of $\oplus$. If for some (simple) automata $A_1, A_2, \ldots, A_n$ we have $A = A_1 \oplus A_2 \oplus \ldots A_n$, we call A_i *(simple) components* of A. *Non-branching* canonical automata are those constructed from flowers without the use of $\to$, i. e., $C_{\omega^\omega \alpha+n}, D_{\omega^\omega \alpha+n}, E_{\omega^\omega \alpha+n}$ for $\alpha < \omega^\omega$ and $n < \omega$. The remaining automata are called *branching*.

3.5 Without Branching

In this section we briefly reformulate Wagner's results on regular word languages [41] in terms of canonical automata. For the sake of completeness, we reprove them in this framework.

The outline is just like for tree languages: define a collection of canonical automata, prove that they form a strict hierarchy with respect to the Wadge reducibility, check some closure properties, and provide an algorithm calculating the equivalent canonical automaton for a given deterministic automaton, thus proving that the hierarchy is complete for regular languages.

Since the non-branching canonical automata have only left transitions, they only check a regular word property on the leftmost path. It is easy to see that for each word language K, the language of trees whose leftmost branch is in K is Wadge equivalent to K. Based on this observation, we will treat the non-branching canonical automata as automata on words.

Let $L_{(\iota,\kappa)}$ denote the language of infinite words over $\{\iota, \iota+1, \ldots, \kappa\}$ that satisfy the parity condition, i. e., the highest number occurring infinitely often is even.

Lemma 7. *For every index (ι,κ) and every deterministic tree automaton A of index at most (ι,κ),*

1. $L(A) \leq L_{(\iota,\kappa)}$
2. $L(F_{(\iota,\kappa)}) \equiv_W L_{(\iota,\kappa)}$,
3. $L_{(\iota,\kappa)} \leq L_{(\iota,\kappa')}$ *iff* $(\iota,\kappa) \leq (\iota',\kappa')$.

Proof. The reduction showing (1) is given by $w \mapsto \mathrm{rank}(q_0)\mathrm{rank}(q_1)\mathrm{rank}(q_2)\ldots$, where $q_0q_1q_2\ldots$ is the run of A on the word w.

For (2) the remaining reduction is obtained by assigning to a sequence $n_1n_2n_3\ldots$ the tree with the word $a_{n_1}a_{n_1}a_{n_2}a_{n_2}a_{n_3}a_{n_3}\ldots$ on the leftmost branch, and a a_ι elsewhere.

Since $L_{(\iota,\kappa)}$ can be recognized by a (ι,κ) automaton, one implication in (3) follows from (1). To prove the remaining one, it is enough to show that $L_{(\iota,\kappa)} \not\leq L_{\overline{(\iota,\kappa)}}$. Let us fix ι and proceed by induction on κ. For $\iota = \kappa$ the claim holds trivially: $\emptyset \subseteq T_{\{1\}}$ and $T_{\{1\}}$ are not reducible to each other. Take $\iota < \kappa$ and let $(\iota',\kappa') = \overline{(\iota,\kappa)}$. Consider the game $G_\kappa = G_W(L_{(\iota,\kappa)}, L_{(\iota',\kappa')})$. As long as Duplicator does not play κ', Spoiler can follow the strategy from $G_{\kappa-1} = G_W(L_{(\iota,\kappa-1)}, L_{\overline{(\iota,\kappa-1)}})$. If Duplicator never plays κ', he loses. When

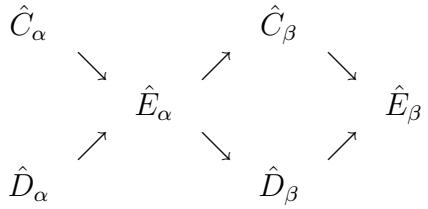
Duplicator plays κ', Spoiler should play κ, and then again follow the strategy from $G_{\kappa-1}$, and so on. Each time, Duplicator has to play κ' finally, otherwise he loses. But then he must play κ' infinitely many times, and he loses to, since κ' and κ have different parity. □

For the sake of convenience let us renumber the non-branching automata. For $\eta < \omega^\omega$ let

$$\hat{C}_{\omega\eta+n} = C_{\omega^\omega\eta+n}\,, \qquad \hat{D}_{\omega\eta+n} = D_{\omega^\omega\eta+n}\,, \qquad \hat{E}_{\omega\eta+n} = E_{\omega^\omega\eta+n}\,.$$

Let $\hat{\mathcal{C}} = \{\hat{C}_\alpha, \hat{D}_\alpha, \hat{E}_\alpha : 1 < \alpha < \omega^\omega\}$.

Proposition 8. *For $0 < \alpha < \beta < \omega^\omega$ we have*



where $\to$ means $<$. Furthermore, $\hat{C}_\alpha \not\leq \hat{D}_\alpha$ and $\hat{D}_\alpha \not\leq \hat{C}_\alpha$.

Proof. First, observe that $\hat{C}_\alpha \leq \hat{E}_\alpha$: a winning strategy for Duplicator in $G(\hat{C}_\alpha, \hat{E}_\alpha)$ is to move the initial token to $F_{(0,\kappa)}$, and then simply copy Spoiler's actions. Analogously, $\hat{D}_\alpha \leq \hat{E}_\alpha$.

Let us now suppose that $\beta = \omega^k$ for some $k < \omega$. Then $\alpha = \omega^{k-1}n_{k-1} + \ldots + n_0$. By definition, $\hat{E}_\alpha$, has index at most $(0, k)$. Hence, by Lemma 7, $\hat{E}_\alpha \leq F_{(0,k)} = \hat{C}_\beta$. If we increase the ranks in each $F_{(0,l)}$ in $\hat{E}_\alpha$ by 2, we obtain an automaton with index at most $(1, k+1)$ recognizing the same language. Hence, we also have $\hat{E}_\alpha \leq F_{(1,k+1)} = \hat{D}_\alpha$.

Now, consider the general case. We have a unique pair of presentations $\alpha = \omega^k m_k + \ldots + m_0$ and $\beta = \omega^k n_k + \ldots + n_0$ with $n_k > 0$. Let i be the largest number satisfying $m_i \neq n_i$. Since $\alpha < \beta$, $m_{i_0} < n_{i_0}$. Thus we have $\hat{E}_\alpha \equiv \hat{E}_{\alpha_0} \oplus \hat{E}_\gamma$, $\hat{C}_\beta \equiv \hat{E}_{\beta_0} \oplus \hat{C}_\gamma$, where $\gamma = \omega^k m_k + \ldots + \omega^i m_i$, $\alpha_0 = \omega^{i-1}m_{i-1} + \ldots + m_0$, $\beta_0 = \omega^i(n_i - m_i) + \omega^{i-1}m_{i-1} + \ldots + m_0$. Consider the game $G(\hat{E}_{\alpha_0} \oplus \hat{E}_\gamma, \hat{E}_{\beta_0} \oplus \hat{C}_\gamma)$. The strategy for Duplicator is as follows. First move the token to the last $F_{(0,i)}$ in $\hat{C}_{\beta_0}$. Then follow the strategy given by the inequality $\hat{E}_{\alpha_0} \leq F_{(0,i)}$, as long as Spoiler stays in $\hat{E}_{\alpha_0}$. If he stays there forever, Duplicator wins. If Spoiler moves to $\hat{E}_\gamma$, Duplicator should do

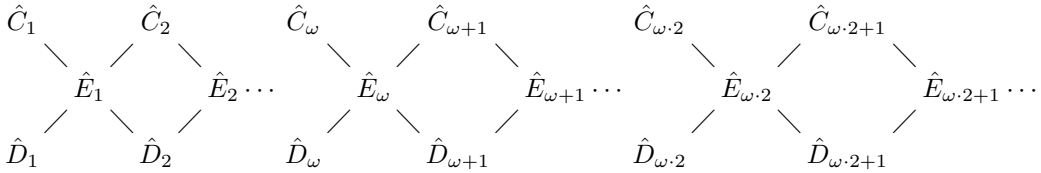


Figure 3.1: An initial segment of the Wagner hierarchy

the same and keep copying Spoiler's move from that moment on. This also guarantees winning. The proof for $\hat{D}_\beta$ is entirely analogous.

In order to prove that the inequalities are strict it is enough to show that $\hat{C}_\alpha \not\leq \hat{D}_\alpha$ and $\hat{D}_\alpha \not\leq \hat{C}_\alpha$. We only prove that $\hat{C}_\alpha \not\leq \hat{D}_\alpha$; the proof for $\hat{D}_\alpha \not\leq \hat{C}_\alpha$ is entirely analogous. Let us proceed by induction. The assertion holds for $\alpha = 1$: the whole space is not reducible to the empty set. Let us take $\alpha > 1$. By the definition, $\hat{C}_\alpha = F_{(0,k)} \oplus \hat{E}_\gamma$, $\hat{D}_\alpha = F_{(1,k+1)} \oplus \hat{E}_\gamma$, where $\alpha = \omega^k + \gamma$. Consider the game $G(F_{(0,k)} \oplus \hat{E}_\gamma, F_{(1,k+1)} \oplus \hat{E}_\gamma)$. We have to find a winning strategy for Spoiler. If Duplicator never leaves $F_{(1,k+1)}$ Spoiler can stay in $F_{(0,k)}$ and win using the strategy given by the Lemma 7 (3). Otherwise, after Duplicator enters $\hat{E}_\gamma$, he must make choice between $\hat{C}_\gamma$ and $\hat{D}_\gamma$. Spoiler should loop in any loop of $F_{(0,k)}$ waiting for Duplicator's choice. When Duplicator chooses one of $\hat{C}_\gamma$, $\hat{D}_\gamma$, Spoiler should choose the other one and use the strategy given by the induction hypothesis. □

The third step is proving closure by natural operations. For word automata only the operations $\oplus$ and $\vee$ make sense. The operation $\vee$ is defined just like for trees. To define $\oplus$, simply assume that the tail components of a word automaton are leaf SCCs. It is easy to see that $\equiv$ is a congruence with respect to $\oplus$ and $\wedge$. Both operations are associative up to $\equiv$.

Proposition 9. *For each $A_1, A_2 \in \hat{\mathcal{C}}$, one can find in polynomial time automata $A_\vee, A_\oplus \in \hat{\mathcal{C}}$ such that $A_1 \vee A_2 \equiv A_\vee$ and $A_1 \oplus A_2 \equiv A_\oplus$.*

Proof. Closure by $\vee$ is easy. If $A_1 \geq A_2$, $A_1 \vee A_2 \equiv A_1$. Without loss of generality we may assume that A_1 and A_2 are incomparable. But then $A_1 = \hat{C}_\alpha$, $A_1 = \hat{C}_\alpha$ for some $\alpha < \omega^\omega$. It is very easy to see that $C_\alpha \vee D_\alpha \equiv E_\alpha$.

Let us now consider $A_1 \oplus A_2$. Since $\oplus$ is associative up to $\equiv$ and only depends on the $\equiv$ classes of the input automata, it is enough to prove the claim for simple A_1; in order to obtain a canonical automaton for $(A_1^{(1)} \oplus \ldots \oplus A_1^{(n)}) \oplus A_2$, take $A_1^{(1)} \oplus (A_1^{(2)} \oplus \ldots (A_1^{(n)} \oplus A_2) \ldots)$. Let us first consider

$A_1 = C_{\omega^k}$. Observe that if $C_{\omega^k} \geq B$, $C_{\omega^k} \oplus B \equiv C_{\omega^k}$. It is enough to give a strategy for Duplicator in $G(C_{\omega^k} \oplus B, C_{\omega^k})$, since the other inequality is obvious. To win, Duplicator should first copy Spoiler's actions, as long as Spoiler stays in C_{ω^k}. When Spoiler moves to B, Duplicator should simply switch to the strategy from $G(B, C_{\omega^k})$.

Using the property above, we easily reduce the general situation to one of the following cases: $C_{\omega^k} \oplus C_{\eta\omega^{k+1}}$, $C_{\omega^k} \oplus D_{\eta\omega^k}$, or $C_{\omega^k} \oplus E_{\eta\omega^k}$. In the third case, the automaton is already canonical. Let us calculate the result in the first two cases.

In the first case we have $C_{\omega^k} \oplus C_{\eta\omega^{k+1}} \equiv C_{\eta\omega^{k+1}}$. Consider the game $G(C_{\omega^k} \oplus C_{\eta\omega^{k+1}}, C_{\eta\omega^{k+1}})$. Let C_{ω^l} be the head component of $C_{\eta\omega^{k+1}}$. It holds that $l > k$. In order to win the game, while Spoiler stays inside C_{ω^k}, Duplicator should stay in C_{ω^l} and use the strategy from $G(C_{\omega^k}, C_{\omega^l})$. When Spoiler enters $C_{\eta\omega^{k+1}}$, Duplicator may simply copy his actions. The converse inequality is trivial.

In the second case there are two possibilities. If the head component of $D_{\eta\omega^k}$ is D_{ω^l} with $l > k$, proceeding as before one proves $C_{\omega^k} \oplus D_{\eta\omega^{k+1}} \equiv D_{\eta\omega^{k+1}}$. But if $l = k$, we have $C_{\omega^k} \oplus D_{\eta\omega^k} \equiv C_{\eta\omega^k + \omega^k}$. Consider the game $G(C_{\eta\omega^k+\omega^k}, C_{\omega^k} \oplus D_{\eta\omega^k})$. While Spoiler stays in C_{ω^k}, Duplicator should copy his actions. When Spoiler leaves C_{ω^k}, he has to choose between D_{ω^k} and the next copy of C_{ω^k}. If he chooses D_{ω^k}, Duplicator also moves to his copy of D_{ω^k}, and mimics Spoiler actions. Suppose Spoiler chooses C_{ω^k}. Then Duplicator stays in his head component, and mimics Spoiler's actions, as long as he stays in C_{ω^k}. When Spoiler leaves C_{ω^k}, he enters the head state of $E_{\eta'\omega^k}$, where $\eta' + 1 = \eta$. Duplicator should exit C_{ω^k}, go past D_{ω^k}, and enter his copy of $E_{\eta'\omega^k}$. From now on, he can copy Spoiler's actions.

For $A_1 = D_{\omega^k}$, simply dualize the claims and the proofs. For $A_1 = E_{\omega^k}$, note that $E_{\omega^k} \oplus A_2 \equiv C_{\omega^k} \oplus A_2 \vee D_{\omega^k} \oplus A_2$, and the equivalent canonical automaton can be obtained by previous cases. □

Let us now see that the hierarchy is complete for word languages.

Theorem 9. *For each word automaton A one can find in polynomial time a canonical non-branching automaton B such that $L(A) \equiv_W L(B)$.*

Proof. We will proceed by induction on the height of the DAG of strongly connected components of A. Suppose that the automaton is just one strongly connected component. Let (ι, κ) be the highest index for which A contains a (ι, κ)-flower. It is well defined, because if A contains a $(0, k)$-flower and a

$(1, k+1)$-flower, it must also contain a $(0, k+1)$-flower. By Theorem 1, A is equivalent to a (ι, κ)-automaton and so, by Lemma 7, $A \leq F_{(\iota,\kappa)}$. On the other hand it is easy to see, that in $G(F_{(\iota,\kappa)}, A)$, Duplicator may easily use the (ι, κ)-flower in A to mimic Spoiler's actions in $F_{(\iota,\kappa)}$. Hence, $A \equiv F_{(\iota,\kappa)}$.

Now, suppose that the DAG of SCCs of A has at least two nodes. Let X be the root SCC. Like before, let (ι, κ) be the maximal index such that X contains a (ι, κ)-flower. Let $q_1, \ldots, q_m$ be all the states reached by the transitions exiting X (the "initial" states of the SCCs that are children of X). Recall that A_q is the automaton A with the initial state set to q. Let B_i be the canonical non-branching automaton equivalent to A_{q_i}. It is easy to see that $A \equiv F_{(\iota,\kappa)} \oplus (B_1 \vee B_2 \vee \ldots \vee B_m)$. □

3.6 The Use of Replication

According to the definition of the automata game, in a branching transition a token is split in two. However in branching canonical automata, the role to be played by two new tokens is very different. Therefore, we prefer to see the process of splitting a token as producing a new token that moves along the right branch of the transition, while the original one moves left. Thus the initial token moves along the leftmost path, bubbling out new tokens now and then.

Recall that we have defined the operation $\oplus$ in such a way, that the second automaton can only be reached via a leftmost path. This means that the only token that can actually move from one simple automaton to another is the initial token. Since passing between the simple automata forming a canonical automaton is usually the key strategic decision, we call the initial token *critical*, and the path it moves along, the *critical path*.

Branching automata are defined by iterating $\xrightarrow{(1,1)}$, shortly denoted by $\rightarrow$. The significance of $\rightarrow$ lies in the fact that closing the family of non-branching automata by this operation gives, up to Wadge equivalence, almost all deterministic tree languages (only $C_{\omega^{\omega\cdot 3}}$, $C_{\omega^{\omega\cdot 3}+1}$, and $C_{\omega^{\omega\cdot 3}+2}$ will be defined by means of a stronger replication). In particular, we will show that the operation $\wedge$ is not needed. In other words, $\rightarrow$ is everything that deterministic tree automata have, which word automata have not. Let us see then what the use of the operation $\rightarrow$ is.

There are two kinds of simple branching automata. The first one is obtained by iterating $\rightarrow$ on C_3, and generalizes C_n. Intuitively, C_n =

$C_1 \oplus (n-1)E_1$ lets a player in an automata game change his mind $n-1$ times in the following sense. First, the player moves his (only) token along the head loop. The head loop is accepting, so if he keeps looping there forever, the resulting run will be accepting. But after some time he may decide that producing an accepting run is not a good idea. In such a case he can move to the rejecting loop in the first copy of E_1. Later he may want to change his mind again, and again, until he reaches the last copy of E_1. Now, when the player is in charge of $C_\omega = C_1 \to C_3$ he can choose a number $n < \omega$, and looping in the head loop of C_ω produce n tokens in the head loop of his copy of C_3. We will see that with those tokens it is possible to simulate any strategy designed for C_{n+2}. In other words, C_ω offers the choice between C_n for arbitrarily high $n \geq 3$. The automaton $C_{\omega^2} = C_1 \to (C_1 \oplus (C_1 \to C_3))$ lets you choose the number of times you will be allowed to choose some C_n, and so on.

The second kind of simple branching automata, obtained by iterating $\to$ on $C_{\omega^{\omega+1}}$, does the same with $C_{\omega^{\omega+n}}$ instead of C_n. For instance, $C_{\omega^{\omega\cdot 2}} = C_1 \to C_{\omega^{\omega+1}}$ lets the player choose any $C_{\omega^{\omega+n}} = \hat{C}_{\omega^n}$ (see page 51), and in consequence $L(C_{\omega^{\omega\cdot 2}})$ is hard for the class of regular languages of words.

Let us now see the proofs. The first lemma justifies the name replication.

Lemma 8. *For all automata A, B and all $0 < k < \omega$,*

1. $A \to B \;\geq\; (A \to B) \wedge (B)^k$,

2. $C_1 \to B \;\geq\; (B)^k$.

Proof. To see that (1) holds, consider $G((A \to B) \wedge (B)^k, A \to B)$. Spoiler's initial moves produce a token x in the head loop of $A \to B$, and tokens $x_1, \ldots, x_k$, each in the head component of a different copy of B. Duplicator should loop his starting token y around the head loop of $A \to B$ exactly k times producing for each x_i a *doppelgänger* y_i and move them all to the initial state of B. From now on y mimics x, and y_i mimics x_i for $i = 1, \ldots, k$.

For the proof of (2) it is enough to check that $(C_1 \to B) \wedge (B)^k \geq (B)^k$. Clearly $C_1 \to B \geq C_1$. By Lemma 6, $(C_1 \to B) \wedge (B)^k \geq C_1 \wedge (B)^k$, and the claim follows. □

Next we need to calculate the value of $(C_3)^n$ and $(C_{\omega^{\omega+1}})^n$. Apart from canonical (ι, κ)-flowers $F_{(\iota,\kappa)}$, we consider the following automata containing

weak (ι,κ)-flowers (see page 25):

$$WF_{(0,n)} = \underbrace{C_1 \oplus D_1 \oplus C_1 \oplus D_1 \oplus \ldots}_{n+1}, \quad WF_{(1,n+1)} = \underbrace{D_1 \oplus C_1 \oplus D_1 \oplus C_1 \oplus \ldots}_{n+1}.$$

We will refer to these automata as weak (ι,κ)-flowers too. In fact, $WF_{(0,n)} \equiv C_{n+1}$, $WF_{(1,n+1)} \equiv D_{n+1}$, but we find the notation convenient.

A pair $(i_1,i_2) \in \omega \times \omega$ is called *even* if both i_1 and i_2 are even. Otherwise (i_1,i_2) is *odd*. Let $[\iota,\kappa]$ denote the set $\{\iota,\iota+1,\ldots,\kappa\} \subseteq \omega$ with the natural order. Consider the set $[\iota,\kappa]\times[\iota',\kappa']$ with the product order: $(x_1,y_1) \leq (x_2,y_2)$ if $x_1 \leq x_2$ and $y_1 \leq y_2$. An *alternating chain of type* (m,n), or (m,n)-chain, is a sequence $(x_m,y_m) < (x_{m+1},y_{m+1}) < \ldots < (x_n,y_n)$, such that (x_i,y_i) is even iff i is even. It is enough to consider $(0,n)$ and $(1,n)$ chains. Suppose we have a (m,n)-chain of maximal length in $[\iota,\kappa]\times[\iota',\kappa']$. The parity of n is equal to the parity of (κ,κ'), as defined above, for otherwise we could extend the alternating chain with (κ,κ') and get a $(m,n+1)$-chain. Consequently, the following operation is well-defined:

$$(\iota,\kappa)\wedge(\iota',\kappa') = \text{ the type of the longest alternating chain in } [\iota,\kappa]\times[\iota',\kappa'].$$

Lemma 9. *For all indices (ι_1,κ_1) and (ι_2,κ_2) it holds that*

$$\begin{aligned} F_{(\iota_1,\kappa_1)} \wedge F_{(\iota_2,\kappa_2)} &\equiv F_{(\iota_1,\kappa_1)\wedge(\iota_2,\kappa_2)}\,, \\ WF_{(\iota_1,\kappa_1)} \wedge WF_{(\iota_2,\kappa_2)} &\equiv WF_{(\iota_1,\kappa_1)\wedge(\iota_2,\kappa_2)}\,. \end{aligned}$$

In particular, $(F_{(0,2)})^k \equiv F_{(0,2k)}$ and $(WF_{(0,2)})^k \equiv WF_{(0,2k)}$. Equivalently, $(C_{\omega^{\omega+1}})^k \equiv C_{\omega^{\omega+1+2k}}$ and $(C_3)^k = C_{2k+1}$.

Proof. By Lemma 7, $L(F_{(i,j)}) \equiv_W L_{(i,j)}$, so $L(F_{(\iota_1,\kappa_1)} \wedge F_{(\iota_2,\kappa_2)}) \equiv_W L_{(\iota_1,\kappa_1)} \times L_{(\iota_2,\kappa_2)}$, where $L\times M = \{(x_1,y_1)(x_2,y_2)\ldots : x_1x_2\ldots \in L,\ y_1y_2\ldots \in M\}$. We will show that $L_{(\iota_1,\kappa_1)} \times L_{(\iota_2,\kappa_2)} \equiv_W L_{(\iota,\kappa)}$, where $(\iota,\kappa) = (\iota_1,\kappa_1)\wedge(\iota_2,\kappa_2)$.

Consider the following automaton A. The state space is the set

$$[\iota_1,\kappa_1]\times[\iota_2,\kappa_2] \to \{0,1,2\}$$

and the initial state is the function constantly equal 0. The transition relation δ is defined as $(f,\sigma,g) \in \delta$ iff for all i and j, $(f(i,j),\sigma,g(i,j)) \in \delta_{(i,j)}$, where $\delta_{(i,j)}$ is defined as

$$\begin{aligned} &0 \xrightarrow{(i,*)} 1\,, \qquad 0 \xrightarrow{(k,*)} 0 \text{ for all } k \neq i\,, \\ &1 \xrightarrow{(*,j)} 2\,, \qquad 1 \xrightarrow{(*,k)} 1 \text{ for all } k \neq j\,, \\ &2 \xrightarrow{(*,*)} 1\,, \end{aligned}$$

with $*$ denoting any letter.

Let us now define the rank function. For $i \in [\iota_1, \kappa_1]$ and $j \in [\iota_2, \kappa_2]$, let $(\iota', \kappa') = (\iota_1, i) \wedge (\iota_2, j)$ and $\mathrm{rank}(i,j) = \kappa'$. Observe that $\iota' = \iota$, so $\iota \leq \kappa' \leq \kappa$. Set the rank of the states that never take the value 2 to ι. For the remaining states set the rank to $\mathrm{rank}(\max_k i_k, \max_k j_k)$, where $(i_1, j_1), (i_2, j_2), \ldots, (i_r, j_r)$ are the arguments for which the value 2 is taken

Let us check that the automaton recognizes $L_{(\iota_1,\kappa_1)} \times L_{(\iota_2,\kappa_2)}$. Take a word $w = (x_1, y_1)(x_2, y_2)\ldots$. Let $x = \max_k x_k$ and $y = \max_k y_k$. In the run of A on w, the states f satisfying $f(x,y) = 2$ will occur infinitely often. Furthermore, from some moment on there only appear states f satisfying $\forall_{(x',y')}\ f(x',y') = 2 \implies (x',y') \leq (x,y)$. Since $(x,y) \leq (x',y') \implies \mathrm{rank}(x,y) \leq \mathrm{rank}(x',y')$, the highest rank used infinitely often in the run on w is $\mathrm{rank}(x,y)$. Finally, $\mathrm{rank}(x,y)$ is even iff x and y are even, so the run on w is accepting iff $w \in L_{(\iota_1,\kappa_1)} \times L_{(\iota_2,\kappa_2)}$.

Since A has the index (ι, κ), the automaton itself provides a reduction of $L_{(\iota_1,\kappa_1)} \times L_{(\iota_2,\kappa_2)}$ to $L_{(\iota,\kappa)}$.

By definition of (ι, κ), there exists a sequence of pairs

$$(x_\iota, y_\iota) < (x_{\iota+1}, y_{\iota+1}) < \ldots < (x_\kappa, y_\kappa)$$

such that for all i it holds that $\iota_1 \leq x_i \leq \kappa_1$, $\iota_2 \leq y_i \leq \kappa_2$, and x_i and y_i are even iff i is even. The reduction is given by the function

$$\varphi(i_1 i_2 i_3 \ldots) = (x_{i_1}, y_{i_1})(x_{i_2}, y_{i_2})(x_{i_3}, y_{i_3})\ldots .$$

The proof for weak flowers is entirely analogous. □

Lemma 10. *For all $0 < k, l < \omega$ and all $m < \omega$*

$$\begin{aligned} C_1 \oplus C_{\omega^m k} \wedge C_1 \oplus C_{\omega^m l} &\equiv C_1 \oplus C_{\omega^m (k+l)}\,, \\ C_1 \oplus C_{\omega^{\omega\cdot 2+m} k} \wedge C_1 \oplus C_{\omega^{\omega\cdot 2+m} l} &\equiv C_1 \oplus C_{\omega^{\omega\cdot 2+m} (k+l)}\,. \end{aligned}$$

In particular, $(C_1 \oplus C_{\omega^m})^k \equiv C_1 \oplus C_{\omega^m k}$ and $(C_1 \oplus C_{\omega^{\omega\cdot 2+m}})^k \equiv C_1 \oplus C_{\omega^{\omega\cdot 2+m} k}$.

Proof. Consider $G(C_1 \oplus C_{\omega^m k} \wedge C_1 \oplus C_{\omega^m l}, C_1 \oplus C_{\omega^m k} \oplus C_{\omega^m l})$. Duplicator's critical token moves along $WF_{(0,2k+2l)}$ formed by the alternating head and tail loops of consecutive copies of C_{ω^m}. Spoiler's initial token splits in the first move in two tokens which continue moving along $WF_{(0,2k)}$ and $WF_{(0,2l)}$. For the purpose of this proof, call them both critical.

The strategy for Duplicator is based on the fact that $WF_{(0,2k)} \wedge WF_{(0,2l)} \equiv WF_{(0,2k+2l)}$. Duplicator can loop his critical token inside an accepting loop as long as both Spoiler's critical tokens loop inside accepting loops. When Spoiler changes his mind and moves one of them to a rejecting loop, Duplicator should also move to a rejecting loop too, and keep looping there until both Spoiler's tokens are again in accepting loops. This can only repeat $k+l$ times, so Duplicator is able to realise this strategy.

This way, whenever Spoiler produces a new token x using one of the critical tokens, Duplicator can produce its *doppelgänger* y. The role of the *doppelgänger* is to mimic the original. The mimicking is in fact passed from generation to generation: if the original token bubbles a new token x', y should bubble a new *doppelgänger* y' which is to mimic x', and so on.

In order to see that the strategy is winning it is enough to observe two facts: Duplicator's critical token stays in a rejecting loop forever iff one of Spoiler's critical tokens does, and the sequence of ranks seen by any of Spoiler's non-critical tokens is equal to the one seen by its *doppelgänger*. Hence, $C_1 \oplus C_{\omega^m k} \wedge C_1 \oplus C_{\omega^m l} \leq C_1 \oplus C_{\omega^m (k+l)}$.

The converse inequality is proved in a similar way and for the second equivalence the same proof works. □

Corollary 3. *For all* $l, \iota, \kappa < \omega$ *and all* $0 < n < \omega$

1. $C_\omega > WF_{(\iota,\kappa)}$, $C_{\omega^{l+1}} \geq C_{\omega^l n}$,
2. $C_{\omega^{\omega\cdot 2}} > F_{(\iota,\kappa)}$, $C_{\omega^{\omega\cdot 2+l+1}} \geq C_{\omega^{\omega\cdot 2+l} n}$.

Proof. By Lemma 8 and Lemma 9, $C_\omega \geq (WF_{(0,2)})^m \equiv WF_{(0,2m)}$ and by the strictness of the hierarchy for word languages $C_\omega > WF_{(\iota,\kappa)}$. Similarly, using Lemma 8 and Lemma 10 we get $C_{\omega^{l+1}} \geq (C_1 \oplus C_{\omega^l})^n \equiv C_1 \oplus C_{\omega^l n} \geq C_{\omega^l n}$. The remaining two inequalities are analogous. □

3.7 Automata in Order

Let us start with a simple observation on canonical automata. Recall that, by convention, in transitions of the form $p \xrightarrow{\sigma} q, \top$ or $p \xrightarrow{\sigma} \top, q$ no new tokens are produced, only the old token moves from p to q, and in the remaining transitions the old token moves left bubbling the new token to the right. The following lemma relies on this convention.

Lemma 11. *Let A be a canonical automaton.*

1. *If a player in charge of A produces infinitely may tokens, the resulting run is rejecting.*
2. *If a run constructed by a player in charge of A is rejecting, one of the tokens realized a rejecting path.*

Proof. We will proceed by structural induction. The claim holds trivially for non-branching automata. Suppose now that $A = C_1 \to A'$. If the constructed run is to be accepting, the player can only produce a finite number of tokens by looping in the head loop of A. By the induction hypothesis for A', those tokens can only have finitely many descendants. Hence, in the whole play there can be only finitely many tokens.

Now, take $A = A' \oplus A''$. Suppose there were infinitely many tokens in some play on A. Observe that all the tokens in A'' are descendants of the critical token. Hence, if there were infinitely many tokens in A'', by the induction hypothesis for A'' the whole run is rejecting. Suppose there were infinitely many tokens in A'. Consider a play in which the critical token instead of moving to A'' stays in the last accepting loop of A' (it exists by the definition of canonical automata). In such a play a run of A' is build. Since there are infinitely many tokens used, the run is rejecting by the induction hypothesis for A'. Consequently, the run of A constructed in the original play must have been rejecting as well.

For the proof of (2) it is enough to observe that in a canonical automaton the only loop using right transitions is the loop around $\top$. In other words, each path of the constructed computation that does not reach $\top$ goes right only a bounded number of times (depending on A). Now, consider a rejecting run of A constructed during a play. It must contain a rejecting path π. The token created during the last right transition on π realises a suffix of π, which of course is a rejecting path. □

Using the above observation we can slowly start examining the order on canonical automata.

Lemma 12. *For all $0 < \alpha < \omega^\omega$*

$$C_\alpha \leq C_{\omega^\omega}\,, \qquad C_\alpha \leq D_{\omega^\omega}\,.$$

Proof. We give a proof for the first inequality; the second one is proved analogously. Consider the following strategy for Duplicator in $G(C_\alpha, C_{\omega^\omega})$.

In every move, if any of Spoiler's tokens is inside a rejecting loop, Duplicator should move his critical token around a 1-loop, otherwise he should loop around the 0-loop. Let us see that the strategy is winning.

By Lemma 11 (1) if Spoiler's run is to be accepting, he must produce only finitely many tokens. All of those tokens must finally get to some 0-loop, and stay there forever. This means that after some number of moves, all Spoiler's tokens are in 0-loops which they will never leave later. But from this moment on Duplicator's critical token will keep looping around the 0-loop, so Duplicator's run will also be accepting.

By Lemma 11 (2), if Spoiler's run is to be rejecting, there must be a token that from some moment on stays forever in a 1-loop. Then Duplicator's token will also get trapped in the rejecting loop in C_{ω^ω}, and Duplicator's run will be rejecting too. □

Let us now see that we can restrict the way the players use non-critical tokens. For a simple automaton A and a canonical automaton $B = B_1 \oplus \ldots \oplus B_n$ with B_i simple, we say that B *dominates* A if one of the following conditions holds

- A is non-branching
- $A = C_1 \to C_\alpha$, $B_1 = C_1 \to C_\beta$, and $\beta \geq \alpha$,
- $A = C_\omega^k$ and $B_1 = F_{(\iota,\kappa)}$ or $B = F_{(\iota,\kappa)} \vee F_{\overline{(\iota,\kappa)}}$ for $\iota < \kappa$.

Lemma 13. *Let $A_1, A_2, \ldots, A_n$ be simple and let B be a canonical automaton dominating all A_i. For every deterministic automaton C, if Spoiler has a winning strategy in $G(A_1 \oplus \ldots A_n \oplus B, C)$, then he also has a strategy in which he removes all non-critical tokens before entering B. Similarly for Duplicator in $G(C, A_1 \oplus \ldots A_n \oplus B)$.*

Proof. Let $B = B_1 \oplus \ldots \oplus B_n$ with B_i simple. Suppose that at some moment the strategy tells Spoiler to enter B (if this never happens, the claim is obvious). If there are no non-critical tokens left in $A_1, A_2, \ldots, A_n$, then we are done. However if there are, we have to take extra care of them. Suppose Spoiler has produced non-critical tokens $x_1, \ldots, x_r$, and x_i is in A_{m_i}. Since x_i is not on a critical path of A_{m_i}, by the definitions of canonical automata, it will stay within a copy of C_{α_i} over the alphabet extended to the alphabet of B.

Suppose $B_1 = C_1 \to C_\beta$. Since B dominates A_i, $\beta \geq \alpha_i$ for all i. Spoiler should replace the token x_i with x_i' and let x_i' take over the duties of x_i. To produce x_i', Spoiler should loop once in the head loop of B_1. If $B_1 = C_{\omega^k}$, or $A_i = C_{\omega^{\omega\cdot 2+k'}}$, Spoiler may simply move x_i' to a copy of C_{α_i} and let it perform exactly the actions x_i would take. If $\beta = C_{\omega^{\omega\cdot 2+k}}$, $\alpha_i = C_{\omega^{k'}}$, Spoiler should move x_i' to the copy of $F_{(0,2)}$ contained in C_β, and let it apply the strategy guaranteed by Lemma 12. To see that the strategy is applicable, it is enough to note that it does not require any waiting, and that $F_{(0,2)}$ contains a copy of $F_{(0,1)}$.

Suppose now that B_1 is non-branching. Then, $\alpha_i < \omega^\omega$ for all i. In this case Spoiler cannot produce a token to take over x_i's duties. Instead, he has to modify the actions of the critical token. He should move the critical token according to his original strategy moving from flower to flower, only when one of his non-critical tokens would be in a rejecting loop, he should choose a 1-loop in his current flower (instead of the loop suggested by the old strategy). Just like in the proof of Lemma 12, if in a play according to the original strategy one of the non-critical tokens stays forever in a rejecting loop, then in the game according to the new strategy the critical token finally also gets trapped in a 1-loop. Otherwise, there are only finitely many non-critical tokens, and all of them finally stabilize in an accepting loop. From that moment on, the critical token will see exactly the same ranks as it would see if Spoiler was playing with the original strategy. Hence, the modified strategy is also winning.

If the original strategy brings Spoiler to a branching automaton, he should produce counterparts of his non-critical tokens just like above. □

Corollary 4. *For every canonical automaton of the form $A \oplus B$ and every deterministic tree automaton C, if a Spoiler has a winning strategy in $G(A \oplus B, C)$, than he has also a winning strategy which removes all non-critical tokens before entering B. Similarly for Duplicator in $G(C, A \oplus B)$.*

Proof. Let $A = A_1 \oplus A_2 \oplus \ldots \oplus A_n$ with A_i simple. From the structure of canonical automata it follows that if $A \oplus B$ is canonical, B dominates A_i for $i = 1, 2, \ldots, n$. □

Now we are ready to get back to the order on $\mathcal{C}$.

Lemma 14. *If $0 < \alpha \leq \beta < \omega^{\omega\cdot 3}$ then $C_\alpha \leq C_\beta$ and whenever D_α and E_β are defined, $D_\alpha \leq E_\beta$, $C_\alpha \leq E_\beta$. If $\beta < \alpha$, then $E_\beta \leq D_\alpha$, $E_\beta \leq C_\alpha$.*

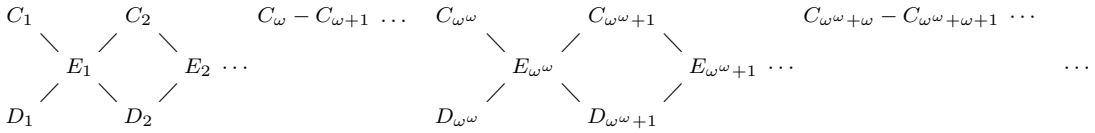


Figure 3.2: The Wadge ordering of the canonical automata.

Proof. As an auxiliary claim let us see that if $A \oplus B$ is canonical and $A' \geq A$, $A \oplus B \leq A' \oplus B$. Indeed, the following is a winning strategy for Duplicator in $G(A \oplus B, A' \oplus B)$. While Spoiler keeps inside A, apply the strategy from $G(A, A')$. If Spoiler enters B, by Corollary 4 we may assume he removes all non-critical tokens. Hence, Duplicator may remove non-critical tokens, move the critical token to B and copy Spoiler's actions. In an analogous way we prove $A \oplus B \leq A' \oplus B$.

Let us now see that $C_\alpha \leq C_\beta$ for $\alpha < \beta < \omega^{\omega \cdot 3}$; the other inequalities may be proved in an analogous way. We will proceed by induction on (α, β) with lexicographic order. If $\beta < \omega$, the result follows by the word languages case. Suppose that $\omega \leq \beta < \omega^\omega$. Let $\alpha = \omega^k m_k + \ldots + m_0$ and $\beta = \omega^k n_k + \ldots + n_0$, $n_k > 0$. First, assume that $m_k = 0$. Obviously $C_{\omega^k} \leq C_\beta$, simply because C_β contains a copy of C_{ω^k}. If $k = 1$ the claim follows directly from Corollary 3. For $k > 1$, using the induction hypothesis and Corollary 3, we get $C_\alpha \leq C_{\omega^{k-1}(m_{k-1}+1)} \leq C_{\omega^k}$. Now, assume that $m_k > 0$. Then $\alpha = \omega^k + \alpha'$, $\beta = \omega^k + \beta'$ for some ordinals $\alpha' < \beta'$. By definition $C_\alpha = C_{\alpha'} \oplus C_{\omega^k}$, $C_\beta = C_{\beta'} \oplus C_{\omega^k}$, and by induction hypothesis, $C_{\alpha'} \leq C_{\beta'}$. Hence, by the auxiliary claim above, $C_\alpha \leq C_\beta$.

Now, suppose that $\omega^\omega \leq \beta < \omega^{\omega \cdot 2}$. Let $\alpha = \omega^\omega \alpha_1 + \alpha_0$, $\beta = \omega^\omega \beta_1 + \beta_0$ for $\alpha_0, \alpha_1, \beta_0, \beta_1 < \omega^\omega$. If $\alpha_1 = \beta_1$, then by induction hypothesis $C_{\alpha_0} \leq C_{\beta_0}$, and $C_\alpha \leq C_\beta$ follows by the auxiliary claim above. Assume that $\alpha_1 < \beta_1$. By Lemma 12, $C_{\alpha_0} \leq C_{\omega^\omega}$. Replacing $G(C_{\alpha_0}, C_{\beta_0})$ with $G(C_{\alpha_0}, C_{\omega^\omega})$ in the above strategy, we get $C_\alpha = C_{\alpha_0} \oplus E_{\omega^\omega \alpha_1} \leq C_{\omega^\omega} \oplus E_{\omega^\omega \alpha_1} = C_{\omega^\omega(\alpha_1+1)}$. By Proposition 8 $C_{\omega^\omega(\alpha_1+1)} \leq C_{\omega^\omega \beta_1}$ and since $C_{\omega^\omega \beta_1}$ is contained in $C_{\omega^\omega \beta_1 + \beta_0}$, we get $C_{\omega^\omega \alpha_1 + \alpha_0} \leq C_{\omega^\omega \beta_1 + \beta_0}$. Observe that the argument works also for α_0 or β_0 equal to 0.

The case $\omega^{\omega \cdot 2} \leq \beta < \omega^{\omega \cdot 3}$ is analogous to $\omega \leq \beta < \omega^\omega$. □

For a complete description of the ordering on the canonical automata (see Fig. 3.2) we need the strictness of the inequalities from the previous lemma.

Theorem 10. *Let $0 < \alpha \leq \beta < \omega^{\omega \cdot 3}$. Whenever the respective automata are defined, it holds that $D_\alpha \not\leq C_\alpha$, $D_\alpha \not\geq C_\alpha$, $D_\alpha < E_\beta$, $C_\alpha < E_\beta$, and for*

$\alpha < \beta$, $C_\alpha < C_\beta$, $E_\alpha < D_\beta$, $E_\alpha < C_\beta$.

Proof. By Lemma 14 it is enough to prove $C_\alpha < C_{\alpha+1}$, $D_\alpha < E_\alpha$, $C_\alpha < E_\alpha$, $D_\alpha \not\leq C_\alpha$, $D_\alpha \not\geq C_\alpha$. We will only give a proof of the first inequality; the others can be argued similarly. We will proceed by induction on α. If $\alpha < \omega$, the claim follows by the word languages case.

Suppose $\alpha = \omega^k + \alpha'$, $k \geq 1$. Let $\alpha' \geq 1$ (the remaining case is similar). We shall describe a winning strategy for Spoiler in $G = G(C_{\omega^k+\alpha'+1}, C_{\omega^k+\alpha'})$. Spoiler should first follow the winning strategy for $G(C_{\alpha'+1}, C_{\alpha'})$, which exists by the induction hypothesis. Suppose that Duplicator enters the head loop of C_{ω^k}. We may assume that he removes all his non-critical tokens (Corollary 4). Spoiler should remove all his non-critical tokens, move his critical token to the (accepting) tail loop of $C_{\alpha'+1}$ and loop there until Duplicator leaves the head loop. If Duplicator stays forever in the head loop of C_{ω^k}, he loses. Suppose that Duplicator leaves the head loop of C_{ω^k} after producing r tokens. The rest of the game is equivalent to $G' = G(C_1 \oplus C_{\omega^k}, A)$ for $A = A_1 \wedge \ldots \wedge A_r$, where A_j is the part of C_{ω^k} accessible for the Duplicator's jth token. If $k = 1$, then $A_j \leq WF_{(0,2)}$ for each j. Hence $A \leq WF_{(0,2r)}$ and by Corollary 3 Spoiler has a winning strategy in G'. Let us suppose $k > 1$. Then $A_j \leq C_1 \oplus C_{\omega^{k-1}}$ for $j = 1, \ldots, r$ and so, by Lemma 6, $A \leq (C_1 \oplus C_{\omega^{k-1}})^r$. Hence, by Lemma 10, $A \leq C_{\omega^{k-1}r+1}$. Since $\omega^{k-1}r + 1 < \omega^{k-1}r + 2 \leq \alpha$, we may use the induction hypothesis to get a winning strategy for Spoiler in G'. In either case Spoiler has a winning strategy in G as well.

Now, assume $\omega^\omega \leq \alpha < \omega^{\omega \cdot 2}$. Let $\alpha = \omega^\omega \alpha_1 + \alpha_0$ with $\alpha_0 < \omega^\omega$, $1 \leq \alpha_1 < \omega^\omega$. Again, we describe a strategy for Spoiler in $G = G(C_{\omega^\omega \alpha_1 + \alpha_0 + 1}, C_{\omega^\omega \alpha_1 + \alpha_0})$ only for $\alpha_0 \geq 1$, leaving the remaining case to the reader. First follow the winning strategy from $G(C_{\alpha_0+1}, C_{\alpha_0})$. If Duplicator does not leave the C_{α_0} component, he will lose. After leaving C_{α_0}, Duplicator has to choose $D_{\omega^\omega \alpha_1}$ or $C_{\omega^\omega \alpha_1}$. Suppose he chooses $D_{\omega^\omega \alpha_1}$. Again, by Corollary 4 we may assume that he removes all non-critical tokens. Now, Spoiler has to remove all non-critical tokens and move the critical token to the initial state of $E_{\omega^\omega \alpha_1}$ and use the winning strategy from $G(E_{\omega^\omega \alpha_1}, D_{\omega^\omega \alpha_1})$.

For $\alpha = \omega^{\omega \cdot 2 + k} + \alpha'$ argue like for $\alpha = \omega^k + \alpha'$. □

3.8 Patterns in Automata

Compare the notion of a pattern called (ι, κ)-flower defined in Chapter 2 and the canonical flower $F_{(\iota,\kappa)}$. It is fairly clear that if A contains a (ι, κ)-flower,

Duplicator can win in $G(F_{(\iota,\kappa)}, A)$ by copying Spoiler's actions. It seems plausible to look at A as if it contained $F_{(\iota,\kappa)}$. In this section we provide a notion which captures this intuition.

Two paths $p \xrightarrow{\sigma'_1,d'_1} p'_1 \xrightarrow{\sigma'_2,d'_2} \dots \xrightarrow{\sigma'_m,d'_m} p'_m$ and $p \xrightarrow{\sigma''_1,d''_1} p''_1 \xrightarrow{\sigma''_2,d''_2} \dots \xrightarrow{\sigma''_n,d''_n} p''_n$ in a deterministic automaton A are *branching* iff there exists $i < \min(m,n)$ such that for all $j < i$ it holds that $(\sigma'_j, d'_j) = (\sigma''_j, d''_j)$, $\sigma'_i = \sigma''_i$, and $d'_i \neq d''_i$. Note that the condition implies that $p'_j = p''_j$ for $j \leq i$.

An automaton B can be *embedded* into an automaton A, if there exists a function $e_Q : Q^B \to Q^B$ and a function $e_\delta : Q^B \times \Sigma^B \times \{0,1\} \to \Pi^A$, where Π^A is the set of paths in A, satisfying the following conditions:

1. if $p \xrightarrow{\sigma,d} q$ and $e_\delta(p,\sigma,d) = r_0 \xrightarrow{\sigma_1,d_1} r_1 \xrightarrow{\sigma_2,d_2} \dots \xrightarrow{\sigma_n,d_n} r_n$ then $r_0 = e_Q(p)$, $r_n = e_Q(q)$,
2. for all p, σ the paths $e_\delta(p,\sigma,0)$ and $e_\delta(p,\sigma,1)$ are branching,
3. for every loop λ in B, the corresponding loop in A (obtained by concatenating the paths assigned to the edges of λ) is accepting iff λ is accepting.

For each tree automaton A, let A' be the automaton obtained from A by unravelling the DAG of strongly connected components into a tree (for the purpose of this definition, we allow multiple copies of $\bot$). An automaton A *admits* an automaton B, in symbols $B \sqsubseteq A$, if the automaton B' can be embedded into A. Note that if B can be embedded into A, then $A \sqsubseteq B$.

Lemma 15. *For all deterministic tree automata A and B*

$$A \sqsubseteq B \implies A \leq B\,.$$

Proof. Since $L(B') = L(B)$, without loss of generality we may assume that $B = B'$. We have to provide a winning strategy for Duplicator in $G(B,A)$. Without loss of generality, we may assume that Spoiler never removes his tokens. Let e_Q and e_δ be the embedding functions. We will show that Duplicator can keep a collection of *doppelgängers*, one for each Spoiler's token, such that if some Spoiler's token x is in the state p, its *doppelgänger* y is in the state $e_Q(p)$.

Let us first assume that $e_Q(q_0^B) = q_0^A$. Then the invariant above holds when the play starts. As long as Spoiler does not enter $\bot$, the invariant can be maintained by means of the function e_δ as follows. Suppose that Spoiler

fires a transition $q \xrightarrow{\sigma} q', q''$ for some token x obtaining new tokens x' and x''. Let

$$e_\delta(q, \sigma, 0) = p_0 \xrightarrow{\sigma_1, d_1} \dots \xrightarrow{\sigma_{l-1}, d_{l-1}} p_{l-1} \xrightarrow{\sigma_l, d'_l} p'_l \xrightarrow{\sigma'_{l+1}, d'_{l+1}} \dots \xrightarrow{\sigma'_m, d'_m} p'_m\,,$$

$$e_\delta(q, \sigma, 1) = p_0 \xrightarrow{\sigma_1, d_1} \dots \xrightarrow{\sigma_{l-1}, d_{l-1}} p_{l-1} \xrightarrow{\sigma_l, d''_l} p''_l \xrightarrow{\sigma''_{l+1}, d''_{l+1}} \dots \xrightarrow{\sigma''_n, d''_n} p''_n\,,$$

with $d'_l = 1 - d''_l$.

Let r_i, r'_j, r''_k be such that $p_{i-1} \xrightarrow{\sigma_i, \overline{d_i}} r_i$, $p'_{j-1} \xrightarrow{\sigma'_j, \overline{d'_j}} r'_j$, and $p''_{k-1} \xrightarrow{\sigma''_k, \overline{d''_k}} r''_k$ for $1 \le i < l$, $l+1 \le j \le m$, $l+1 \le k \le n$, where $\overline{d} = 1 - d$.

Recall that we assume that for every transition, either both target states are $\perp$ or none. Since $q' \neq \perp$ and $q'' \neq \perp$ then, by the condition (3) of the definition of admittance, $p'_m \neq \perp$ and $p''_n \neq \perp$ and consequently all the states p_i, r_i, p'_j, r'_j, p''_k, r''_k are not equal to $\perp$. Hence, Duplicator can proceed as follows. Starting with the token y (the *doppelgänger* of x), fire the transitions forming the common prefix of both paths, each time removing the token sent to r_i. Thus he reaches the state p_{l-1} with a descendant of the token y. Then he should fire the next transition producing two tokens y' and y'', and for each of them fire the remaining sequence of transitions (again removing the tokens in the states r'_j and r''_k). Thus he ends up with two tokens in the states $p'_m = e_Q(q')$ and $p''_n = e_Q(q'')$. Hence, the token in $e_Q(q')$ may be the *doppelgänger* of x', and the token in $e_Q(q'')$ may be the *doppelgänger* of x''.

Let us see that if Spoiler never enters $\perp$, Duplicator wins. Observe that the function e_δ induces a function e from the set of infinite paths in B to the set of infinite paths in A. Owing to the condition (3), $e(\pi)$ is accepting iff π is accepting. The strategy used by Duplicator guarantees that for each path π in Spoiler's run, Duplicator's run contains the path $e(\pi)$. The paths in Duplicator's run that are not images of paths from Spoiler's run were all declared accepting by removing the corresponding tokens. Hence, Duplicator's run is accepting iff Spoiler's run is accepting.

Now, if Spoiler enters $\perp$, Duplicator proceeds as before, only if some r_i, r'_j, or r''_k is equal to $\perp$, instead of removing the token from there (he is not allowed to do that), he lets the token and all its descendants loop there forever. In the end, again each path in Spoiler's run has a counterpart in Duplicator's run. The images of the rejecting paths (which exist in Spoiler's run), will be rejecting too. Hence, Duplicator also wins in this case.

Finally we have to consider the situation when $e_Q(q_0^B) \neq q_0^A$. In this case, Duplicator should first move his initial token to the state $e_Q(q_0^B)$, removing

the other tokens produced on the way whenever possible, and then proceed as before. □

Another property that makes admittance similar to containment is transitivity.

Lemma 16. *For all deterministic tree automata A, B, and C,*

$$A \sqsubseteq B \sqsubseteq C \implies A \sqsubseteq C\,.$$

Proof. Again, we may assume that $A' = A$. Furthermore, since the states from one SCC have to be mapped to states from one SCC, then A can be embedded directly into B'. Hence, we may also assume that $B = B'$. Let $e_Q^{X,Y}$, $e_\delta^{X,Y}$ be functions embedding the automaton X into Y. The embedding of A into C is simply a composition of two given embeddings: $e_Q^{A,C} = e_Q^{B,C} \circ e_Q^{A,C}$, $e_\delta^{A,C} = e_\Pi^{B,C} \circ e_\delta^{A,B}$, where $e_\Pi^{B,C} : \Pi^B \to \Pi^C$ is the function induced by $e_\delta^{B,C}$ in the natural way. It is easy to see that $e_Q^{A,C}$ and $e_\delta^{A,C}$ satisfy the conditions from the definition of admittance. □

Embedding for automata on words is defined analogously, only the function e_δ is defined on $Q \times \Sigma$ instead of $Q \times \Sigma \times \{0,1\}$, and the condition (2) is dropped. Admittance is defined identically. The two lemmas above carry over with analogous proofs.

3.9 Hard Automata

In previous sections we have described an extended hierarchy of canonical automata. As we have already mentioned there are still three canonical automata left to define. Let $C_{\omega^{\omega\cdot 3}} = C_1 \xrightarrow{(0,1)} C_1, C_{(0,2)}$ and $C_{\omega^{\omega\cdot 3}+1} = C_1 \xrightarrow{(0,0)} F_{(0,1)}$. The last automaton, $C_{\omega^{\omega\cdot 3}+2}$ consists of the states q_0, q_1, $\top$ with $\mathrm{rank}(q_i) = i$ and transitions

$$q_0 \xrightarrow{a} q_0, q_1\,, \quad q_0 \xrightarrow{b} \top, \top\,,$$

$$q_1 \xrightarrow{a} q_0, \top\,, \quad q_1 \xrightarrow{b} \top, \top\,.$$

Using canonical automata we can formulate results from Chapter 2 in a uniform way. In the proof we will need the following technical lemma.

Lemma 17. *Let A be a deterministic tree automaton. For every productive state p in A there exists a state q, a path π_p from p to q, and pair of branching paths π_q^0, π_q^1 from q to q forming accepting loops.*

Proof. We prove the following claim: every set $S \subseteq Q$ such that the graph of A restricted to S is strongly connected and for which there exists an accepting run that uses only states from S, contains a state q and a pair of branching paths from q to q forming accepting loops. We proceed by induction on the highest rank κ used in S.

If κ is even, the claim follows trivially. Take $\kappa = 2k + 1$. Consider an accepting run ρ using only states from S. In ρ there must be a subtree ρ' that does not contain the rank $2k + 1$ (otherwise, we would find a path with infinitely many occurrences of $2k + 1$). Consider the set $S' \subseteq S$ of the states used in ρ'. Clearly, the graph of A restricted to S' may not be strongly connected anymore. Let S'' be the set of vertices of some leaf SCC of this graph. Let $p \in S''$, and let ρ'' be the subtree of ρ' rooted in some node labeled with p. The run ρ'' is accepting and only uses states from S''. Since $S'' \subseteq S'$, the highest rank it uses is less then $2k + 1$. The claim follows by the induction hypothesis.

To obtain the lemma, take S equal to the states of some productive SCC reachable from p. □

Theorem 11. *Let A be a deterministic automaton.*

1. *$L(C_1 \oplus D_1)$ is Π_1^0-complete; $L(A) \in \Sigma_1^0$ iff A does not admit $C_1 \oplus D_1$.*
2. *$L(D_1 \oplus C_1)$ is Σ_1^0-complete; $L(A) \in \Pi_1^0$ iff A does not admit $D_1 \oplus C_1$.*
3. *$L(F_{(1,2)})$ and $L(C_1 \xrightarrow{(0,0)} (D_1 \oplus C_1))$ are Π_2^0-complete;*
   $L(A) \in \Sigma_2^0$ iff A does not admit $F_{(1,2)}$ nor $C_1 \xrightarrow{(0,0)} (D_1 \oplus C_1)$.
4. *$L(F_{(0,1)})$ is Σ_2^0-complete; $L(A) \in \Pi_2^0$ iff A does not admit $F_{(0,1)}$.*
5. *$L(C_{\omega^{\omega \cdot 3}+1})$ is Π_3^0-complete; $L(A) \in \Sigma_3^0$ iff A does not admit $C_{\omega^{\omega \cdot 3}+1}$.*
6. *$L(A) \in \Pi_3^0$ iff A does not admit $C_{\omega^{\omega \cdot 3}+2}$.*
7. *$L(C_{\omega^{\omega \cdot 3}+2})$ is Π_1^1-complete; $L(A)$ is Π_1^1-complete iff A admits $C_{\omega^{\omega \cdot 3}+2}$.*

Proof. Proving Theorem 8 (page 36) we in fact proved that the implications in Proposition 4 (page 31) were equivalences. It remains to be seen that for an automaton it is the same to contain some of the patterns from Proposition 4 and to admit the respective automaton. It is straightforward to check that it indeed is so. The only difficulty is embedding the transitions to all-accepting states, but this is solved by Lemma 17. Let us just see the case of $C_{\omega^{\omega\cdot 3}+2}$. If A admits $C_{\omega^{\omega\cdot 3}+2}$, then the image of the two loops in $C_{\omega^{\omega\cdot 3}+2}$ that contain the initial state is a split.

Suppose that A contains a split consisting of an i-loop $p \xrightarrow{\sigma,0} p_1 \xrightarrow{\sigma_1,d_1} \ldots \xrightarrow{\sigma_m,d_m} p_{m+1} = p$ and a j-loop $p \xrightarrow{\sigma,1} p'_1 \xrightarrow{\sigma'_1,d'_1} \ldots \xrightarrow{\sigma'_n,d'_n} p'_{n+1} = p$, such that i is even, j is odd, and $i < j$. Without loss of generality we may assume that $m, n \geq 1$. Let $p'_1 \xrightarrow{\sigma'_1,\overline{d'_1}} q'$, and let t_p, $t_{p'_1}$, $t_{q'}$ be the states guaranteed by Lemma 17 for p, p'_1, and q' respectively.

Let B be the automaton obtained from $C_{\omega^{\omega\cdot 3}+2}$ by unravelling the DAG of SCC's. The only way it differs from $C_{\omega^{\omega\cdot 3}+2}$ is that instead of one state $\top$ it contains 5 all-accepting states $\top_1, \ldots, \top_5$, one for each transition from the root SCC:

$$q_0 \xrightarrow{a} q_0, q_1\,, \quad q_0 \xrightarrow{b} \top_1, \top_2\,,$$
$$q_1 \xrightarrow{a} q_0, \top_3\,, \quad q_1 \xrightarrow{b} \top_4, \top_5\,.$$

Define $e_Q(q_0) = p$, $e_Q(q_1) = p'_1$, $e_Q(\top_1) = e_Q(\top_2) = t_p$, $e_Q(\top_3) = t_{q'}$, $e_Q(\top_4) = e_Q(\top_4) = t_{p'_1}$. The function e_δ is defined as follows:

$$\begin{aligned}
e_\delta(q_0, a, 0) &= p \xrightarrow{\sigma,0} p_1 \xrightarrow{\sigma_1,d_1} \ldots \xrightarrow{\sigma_m,d_m} p\,,\\
e_\delta(q_0, a, 1) &= p \xrightarrow{\sigma,1} p'_1\\
e_\delta(q_0, b, 0) &= \pi_p \pi^0_{t_p}\,,\\
e_\delta(q_0, b, 1) &= \pi_p \pi^1_{t_p}\,,\\
e_\delta(q_1, a, 0) &= p'_1 \xrightarrow{\sigma'_1,d'_1} \ldots \xrightarrow{\sigma'_n,d'_n} p\,,\\
e_\delta(q_1, a, 1) &= (p'_1 \xrightarrow{\sigma'_1,\overline{d'_1}} q')\pi_{q'}\,,\\
e_\delta(q_1, b, 0) &= \pi_{p'_1} \pi^0_{t_{p'_1}}\,,\\
e_\delta(q_1, b, 1) &= \pi_{p'_1} \pi^1_{t_{p'_1}}\,,\\
e_\delta(\top_i, *, 0) &= \pi^0_{e_Q(\top_i)}\,,\\
e_\delta(\top_i, *, 1) &= \pi^1_{e_Q(\top_i)}\,,
\end{aligned}$$

where $*$ denotes any letter and by $\pi_1\pi_2$ we mean the concatenation of two paths. Checking that this is an embedding is straightforward. □

The following theorem settles the position of the last canonical automaton, $C_{\omega^{\omega\cdot 3}}$.

Theorem 12. *$L(C_{\omega^{\omega\cdot 3}})$ is Wadge complete for deterministic Δ^0_3 tree languages. In particular, $A \leq C_{\omega^{\omega\cdot 3}}$ for each $A \in \mathcal{C}$.*

Proof. Since $C_{\omega^{\omega\cdot 3}}$ admits neither $C_{\omega^{\omega\cdot 3}+2}$ nor $C_{\omega^{\omega\cdot 3}+1}$, $L(C_{\omega^{\omega\cdot 3}})$ is a deterministic Δ^0_3 language (Theorem 11). Let us see that it is hard in that class.

Take a deterministic automaton A recognizing a Δ^0_3-language. By Theorem 11 (5), A does not admit $C_{\omega^{\omega\cdot 3}+1} = C_1 \xrightarrow{(0,0)} F_{(0,1)}$. Let us divide the states of A into two categories: a state is *blue* if it is replicated (see page 31) by a accepting loop, otherwise it is *red*.

Let A' be the automaton A with the ranks of red states set to 0, and let A'' be A with the ranks of the blue states set to 0. It is easy to prove that $A \leq A' \wedge A''$. Since A does not admit $C_1 \xrightarrow{(0,0)} F_{(0,1)}$, it follows that all $(0,1)$-flowers in A are red. Consequently, A' does not admit $F_{(0,1)}$, and so $L(A')$ is Π^0_2. Since $L(F_{(1,2)})$ is Π^0_2-hard (Theorem 11 (3)), $A' \leq F_{(1,2)}$.

Now consider A''. Note that every state reachable from a blue state is blue. Since all blue states have rank 0, we may actually replace all the blue states with one all-accepting state $\top$ without changing the recognized language. Recall that, by convention, instead of putting tokens into $\top$ we simply remove them. Hence, when for some token in p a transition of the form $p \xrightarrow{\sigma} \top, q$ or $p \xrightarrow{\sigma} q, \top$ is fired, we imagine that the token is moved to q without producing any new tokens. By the Replication Lemma (Lemma 2, page 34) the occurrences of red states in an accepting run may be covered by a finite number of infinite paths. Hence, by our convention, only finitely many tokens may be produced in a play if the constructed run is to be accepting.

Let us now show that Duplicator has a winning strategy in $G(A'', (C_1 \xrightarrow{(0,1)} C_1, F_{(\iota,\kappa)}))$, where (ι,κ) is the index of A. Whenever Spoiler produces a new token (including the starting token), Duplicator should loop once around the head 1-loop producing a *doppelgänger* in $F_{(\iota,\kappa)}$, and keep looping around the head 0-loop. The new token is to visit states with exactly the same ranks as the token produced by Spoiler. Let us see that this strategy works. Suppose Spoiler's run was accepting. Then, there were only finitely many red tokens produced, and hence the head 1-loop was visited only finitely often.

Furthermore, each Spoiler's token visited an accepting path. But then, so did its *doppelgänger*, and Duplicator's run was also accepting. Now suppose Spoiler's run was rejecting. If infinitely many red tokens were produced, the head 1-loop was visited infinitely often, and Duplicator's run was also rejecting. If there were finitely many tokens produced, then one of the tokens must have gone along a rejecting path, but so did its *doppelgänger* and Duplicator's run was also rejecting. Hence $A'' \leq (C_1 \xrightarrow{(0,1)} C_1, F_{(\iota,\kappa)})$.

By Lemma 6, $A' \wedge A'' \leq (C_1 \xrightarrow{(0,1)} C_1, F_{(\iota,\kappa)}) \wedge F_{(1,2)}$, so it is enough to check that $(C_1 \xrightarrow{(0,1)} C_1, F_{(\iota,\kappa)}) \wedge F_{(1,2)} \leq C_{\omega^{\omega\cdot 3}}$. Consider the following strategy for Duplicator in the game $G((C_1 \xrightarrow{(0,1)} C_1, F_{(\iota,\kappa)}) \wedge F_{(1,2)}, C_{\omega^{\omega\cdot 3}})$. First, loop once around the 1-loop and produce a new token in $F_{(0,2)}$ and use it to mimic Spoiler's actions in $F_{(1,2)}$. Then, for each new token x Spoiler produces in his 1-loop and sends to $F_{(\iota,\kappa)}$, Duplicator should produce tokens $y_1, \ldots, y_{\lfloor \frac{\kappa+1}{2} \rfloor}$ in $F_{(0,2)}$. By Lemma 9, $(F_{(0,2)})^{\lfloor \frac{\kappa+1}{2} \rfloor} \equiv F_{(0,2\lfloor \frac{\kappa+1}{2} \rfloor)} \leq F_{(\iota,\kappa)}$, so Duplicator has a winning strategy in $G(F_{(\iota,\kappa)}, (F_{(0,2)})^{\lfloor \frac{\kappa+1}{2} \rfloor})$. Adapting this strategy, Duplicator can simulate the actions of Spoiler's token x in $F_{(\iota,\kappa)}$ with the tokens $y_1, \ldots, y_{\lfloor \frac{\kappa+1}{2} \rfloor}$ in $F_{(0,2)}$. If Spoiler loops the 1-loop without producing a new token, or loops around the 0-loop, Duplicator should copy his actions. Clearly, this strategy is winning for Duplicator.

Finally, let us see that $A \leq C_{\omega^{\omega\cdot 3}}$ for each $A \in \mathcal{C}$. Take $n < \omega$. Observe that in $C_{\omega^{\omega\cdot 2+n}}$ no state is replicated by an accepting loop. Hence, $C_{\omega^{\omega\cdot 2+n}}$ may not admit $C_{\omega^{\omega\cdot 3}+1}$ nor $C_{\omega^{\omega\cdot 3}+2}$. By Theorem 11, $L(C_{\omega^{\omega\cdot 2+n}})$ is in Δ^0_3. By Theorem 14, for each $A \in \mathcal{C}$ there exists $m < \omega$ such that $A \leq C_{\omega^{\omega\cdot 2+n}}$. Hence, all for all $A \in \mathcal{C}$, $L(A) \in \Delta^0_3$, and $A \leq C_{\omega^{\omega\cdot 3}}$. □

From Theorems 11 and 12 we obtain the following picture of the top of the hierarchy:

$$\mathcal{C} < C_{\omega^{\omega\cdot 3}} < C_{\omega^{\omega\cdot 3}+1} < C_{\omega^{\omega\cdot 3}+2}\,.$$

Let $\mathcal{C}' = \mathcal{C} \cup \{C_{\omega^{\omega\cdot 3}}, C_{\omega^{\omega\cdot 3}+1}, C_{\omega^{\omega\cdot 3}+2}\}$. In the remaining of the chapter we will show that each deterministic automaton is Wadge equivalent to one of the canonical automata from $\mathcal{C}'$. Note that the hierarchy has the height $(\omega^\omega)^3+3$, which should be compared with ω^ω for regular word languages [41], $(\omega^\omega)^\omega$ for deterministic context-free word languages [6], $(\omega_1^{CK})^\omega$ for word languages recognized by deterministic Turing machines [33], or an unknown ordinal $\xi > \varepsilon_0$ for nondeterministic context-free word languages [10].

3.10 Closure Properties

Our aim is to show that each deterministic tree language is Wadge equivalent to the language recognized by one of the canonical automata. If this is to be true, the family of canonical automata should be closed (up to Wadge equivalence) by the operations introduced in Sect. 3.3. In this section we will see that it is so indeed. The closure properties carry substantial part of the technical difficulty of the main theorem, whose proof is thus made rather concise.

Proposition 10. *For $A, B \in \mathcal{C}$ one can find in polynomial time an automaton in $\mathcal{C}$ equivalent to $A \vee B$.*

Proof. Take $A, B \in \mathcal{C}$. Obviously, if $A \leq B$, then $A \vee B \equiv B$ and if $B \leq A$, then $A \vee B \equiv A$. If A and B are incomparable, by Lemma 14 we get that they must be equal to D_α and C_α. It follows immediately from the definitions of the automata that $D_\alpha \vee C_\alpha \equiv E_\alpha$. □

Proposition 11. *For $A, B \in \mathcal{C}$ one can find in polynomial time an automaton in $\mathcal{C}$ equivalent to $A \oplus B$.*

Proof. Recall that simple automata are those that cannot be written as $A_1 \oplus A_2$ for some canonical automata A_1, A_2. Let us first assume that A is a simple branching automaton. First let us prove that for $B < A$, $A \oplus B \equiv A$. Let us consider the game $G(A \oplus B, A)$. The following is a winning strategy for Duplicator. While Spoiler keeps inside the head loop of A, mimic his actions. When he exits the head loop, let all the non-critical tokens produced so far copy the actions of their counterparts belonging to Spoiler, and for the critical token (and all new tokens to be produced) proceed as follows. If $C_1 \oplus B$ is a canonical automaton, then, by the shape of the hierarchy, $C_1 \oplus B < A$ and Duplicator may use the winning strategy from $G(C_1 \oplus B, A)$. If $C_1 \oplus B$ is not canonical, then $B = F_{(\iota,\kappa)} \oplus B'$ for some $(\iota, \kappa) \neq (1, 1)$. It is very easy to see that $C_1 \oplus F_{(\iota,\kappa)} \oplus B' \equiv F_{(\iota,\kappa)} \oplus B'$, and again Duplicator can use the winning strategy from $G(C_1 \oplus B, A)$.

Let us assume now that $B = B_1 \oplus B_2 \oplus \ldots \oplus B_n$ where B_i are simple and $B_1 \geq A$. Suppose $B = C_{\omega^\omega \eta}$ for some $\eta < \omega^{\omega \cdot 3}$. Then $A \oplus B \equiv B$. Indeed, consider the game $G(A \oplus B, B)$. While Spoiler keeps inside A, Duplicator should keep in B_1 and apply the strategy from $G(A, B_1)$. Suppose Duplicator enters B. Since $B_1 \geq A$, it holds that B dominates A and we may assume

that Spoiler has removed his non-critical tokens before entering B. From now on Duplicator may simply mimic Spoiler's behaviour.

An analogous argument shows that for $B_1 = D_{\omega^\omega\eta}$, we get $A \oplus B \equiv B$. For $B_1 = E_{\omega^\omega\eta}$, $A \oplus B$ is a canonical automaton (up to a permutation of the input alphabet).

Now, consider $B = B_1 \oplus \ldots \oplus B_n \geq A$, B_i simple and $B_1 < A$. By the definition of canonical automata, $B_1 \leq B_2 \leq \ldots \leq B_n$, and since $B \geq A$, $B_n \geq A$. Let k be the least number for which $B_k \geq A$. Let $B' = B_1 \oplus \ldots \oplus B_{k-1}$ and $B'' = B_k \oplus \ldots \oplus B_n$. In order to reduce this case to the previous one it is enough to check that $A \oplus B \leq A \oplus B''$ (the converse inequality is obvious). Consider $G(A \oplus B, A \oplus B'')$. While Spoiler's critical token stays inside $A \oplus B'$, Duplicator follows the strategy from $G(A \oplus B', A)$. If Spoiler does not leave $A \oplus B'$, he loses. Suppose that Spoiler finally enters B''. Note that B'' dominates A and $B_1, \ldots, B_{k-1}$. Hence, by Lemma 13, we may assume that Spoiler removes all his non-critical tokens on entering B''. Duplicator should simply move his critical token to the head component of B'' and mimic Spoiler's actions.

Suppose now that $A = F_{(\iota,\kappa)}$ or $A = F_{(\iota,\kappa)} \vee F_{\overline{(\iota,\kappa)}}$. Let $B = B_1 \oplus \ldots \oplus B_n$ with B_i simple. For $\iota < \kappa$ proceeding like in Lemma 9 (page 52) one proves that

1. $B < A \implies A \oplus B \equiv A$,
2. $B_1 = F_{\overline{(\iota,\kappa)}} \implies A \oplus B \equiv F_{(\iota,\kappa)} \oplus (F_{(\iota,\kappa)} \vee F_{\overline{(\iota,\kappa)}}) \oplus B_2 \oplus \ldots \oplus B_n \in \mathcal{C}$,
3. $A \leq B_1 = (F_{(\iota',\kappa')} \vee F_{\overline{(\iota',\kappa')}}) \implies A \oplus B \in \mathcal{C}$,
4. $A \leq B_1 = F_{(\iota',\kappa')} \implies A \oplus B \equiv B$,
5. $A \leq B_1 = (C_1 \to B_1') \implies A \oplus B \equiv B$.

In the remaining case, $B_1 < A \leq B$, argue like for branching A. For $\iota = \kappa$, the implications (2), (3), and (4) also hold, and give a canonical form if B_1 is non-branching. If B_1 is branching, $A \oplus B \equiv B$ for $A = F_{(1,1)}$, and $A \oplus B \equiv F_{(0,0)} \oplus B \in \mathcal{C}$ for $A \in \{F_{(0,0)}, F_{(0,0)} \vee F_{(1,1)}\}$.

Finally let $A = A_1 \oplus A_2 \oplus \ldots \oplus A_r$, where A_i are simple. Using the fact that $\oplus$ is associative up to $\equiv$, and Lemma 6 (page 44), we get $(A_1 \oplus A_2 \oplus \ldots \oplus A_r) \oplus B \equiv (A_1 \oplus A_2 \oplus \ldots \oplus A_{r-1}) \oplus (A_r \oplus B) \equiv (A_1 \oplus A_2 \oplus \ldots \oplus A_{r-1}) \oplus B'$ where B' is a canonical automaton equivalent to $(A_r \oplus B)$. Repeating this $r-1$ times more we obtain a canonical automaton equivalent to $A \oplus B$. $\square$

In the following proofs we will need the following property. For simple branching automata $B = (C_1 \to C_\alpha)$, let $B^- = D_1 \to C_\alpha$.

Lemma 18. *For every $A \in \mathcal{C}$ and every simple branching B one can find in polynomial time a canonical automaton equivalent to $B^- \oplus A$.*

Proof. B is simple branching, so $B = C_\alpha$ where $\alpha = \omega^k$ or $\alpha = \omega^{\omega\cdot 2+k}$. Let $A = S \oplus A'$, where $A' \in \mathcal{C}$ and S is a simple automaton. Suppose first that S is a branching automaton. Then $S \equiv C^-_\beta \oplus C_1$ and $A \equiv C^-_\beta \oplus C_1 \oplus A'$ with $\beta = \omega^j$ or $\beta = \omega^{\omega\cdot 2+j}$. Let us check that $C^-_\alpha \oplus C^-_\beta \oplus C_1 \oplus A' \equiv C^-_{\max(\alpha,\beta)} \oplus C_1 \oplus A'$. Consider the following strategy for Duplicator in $G(C^-_\alpha \oplus C^-_\beta \oplus C_1 \oplus A', C^-_{\max(\alpha,\beta)} \oplus C_1 \oplus A')$. While Spoiler's critical token x can reach the head loop of C^-_α or C^-_β , Duplicator may keep his critical token y looping in the head loop of his automaton $C^-_{\max(\alpha,\beta)}$. For every new token produced by Spoiler in the head loop of C^-_α or C^-_β, Duplicator produces a *doppelgänger* in the head loop of $C^-_{\max(\alpha,\beta)}$. When Spoiler moves his critical token x to $C_1 \oplus A'$, Duplicator does the same with y and lets it copy x's actions. As the converse inequality is obvious, $C^-_{\max(\alpha,\beta)} \oplus C_1 \oplus A' \equiv C_{\max(\alpha,\beta)} \oplus A'$ gives the canonical form for $C^-_\alpha \oplus A$.

Now, let S be non-branching. Suppose first that S is one of the automata $D_{\omega^{\omega+k}}$, $C_{\omega^{\omega+k}}$, $E_{\omega^{\omega+k}}$ for $k \geq 0$. If $\alpha = \omega^k$, $C^-_\alpha \oplus S \oplus A' \leq C_\alpha \oplus S \oplus A' \equiv S \oplus A'$ by the proof of the closure by $\oplus$. The converse inequality is obvious. Similarly, if $\alpha = \omega^{\omega\cdot 2+k}$, $C^-_\alpha \oplus S \oplus A' \leq C_\alpha \oplus S \oplus A' \equiv C_\alpha \oplus A'$. The converse inequality is obvious again.

The remaining possible values for S are C_1, D_1 and E_1. If $S = C_1$, $C^-_\alpha \oplus C_1 \oplus A \equiv C_\alpha \oplus A$, and the canonical automaton is obtained via closure by $\oplus$. For $S = E_1$, observe that $C^-_\alpha \oplus E_1 \oplus A' \leq C^-_{\omega^k} \oplus C_2 \oplus A' = C_{\omega^k} \oplus D_1 \oplus A$. By the proof of the closure by $\oplus$ we get $C_{\omega^k} \oplus D_1 \oplus A \equiv C_{\omega^k} \oplus A$. Hence $C^-_{\omega^k} \oplus E_1 \oplus A \leq C_{\omega^k} \oplus A$. The converse inequality is obvious. Finally, if $S = D_1$, we get $C^-_\alpha \oplus D_1 \oplus A' \equiv C^-_\alpha \oplus A'$. By the structure of canonical automata, A' must start with E_1 or $C_{\omega^{\omega+k}}$. In both cases we can use one of the previous cases to get an equivalent canonical automaton.

If $A = S$ the whole argument is analogous, only in the last case, for $S = D_1$, we have $C^-_\alpha \oplus D_1 \equiv D_1$. □

Proposition 12. *For $A, B \in \mathcal{C}$ one can find in polynomial time an automaton in $\mathcal{C}$ equivalent to $A \wedge B$.*

Proof. We will proceed by induction on (A, B) with the product order induced by $\leq$. Let $A = A_1 \oplus A_2 \oplus \ldots \oplus A_m$, $B = B_1 \oplus B_2 \oplus \ldots \oplus B_n$ with A_i, B_j simple. Let $A' = A_2 \oplus \ldots \oplus A_m$ for $m > 1$ and $B' = B_2 \oplus \ldots \oplus B_n$ for $n > 1$.

First, assume that $B_1 = C_1 \to C_\beta$, and either $A_1 = F_{\iota,\kappa}$ for some (ι, κ), or $A_1 = C_1 \to C_\alpha$ for $\alpha \leq \beta$. Let $m, n > 1$. Let us see that $A \wedge B \equiv B_1^- \oplus (A' \wedge B \vee A \wedge C_1 \oplus B')$. In the first move Spoiler produces token x^A in A and x^B in B. While x^A stays in A_1 and x^B stays in the head loop of B_1, Duplicator should keep his critical token in the head loop of B_1^- and for each x, a child of x^B or x^A, produce a token y whose task is to play against x. The token x after being produced is put in the head loop of C_β or, if $A_1 = C_1 \to C_\alpha$, in the head loop of C_α. The token y is put in the head loop of C_β. Since $\alpha \leq \beta$, y can adapt the strategy from $G(C_\alpha, C_\beta)$ if x is in C_α, or simply copy x's actions if x is in C_β. Now, two things may happen. If x^A enters A' while x^B stays in the head loop of B_1, Duplicator should move his critical token to $A' \wedge B$ and split it into y^A sent to A' and y^B sent to B. Then y^A should mimic x^A, and y^B should mimic x^B. If x^B exits the head loop of B_1, Duplicator should move to $A \wedge C_1 \oplus B'$, produce two tokens, and mimic Spoiler's actions. The converse inequality is even simpler. In a similar way we prove $A \wedge B \equiv B_1^- \oplus (A \wedge C_1 \oplus B')$ for $n > m = 1$, $A \wedge B \equiv B_1^- \oplus (A' \wedge B \vee A \wedge C_1)$ for $m > n = 1$, and $A \wedge B \equiv B_1^- \oplus (A \wedge C_1)$ for $m = n = 1$. In all four cases using the induction hypothesis, the closure by $\vee$, $\oplus$, and the Substitution Lemma (Lemma 6, page 44) we obtain an automaton of the form $B_1^- \oplus C$, where C is canonical. Lemma 18 gives an equivalent canonical automaton.

Next, suppose that $A_1 = F_{(\iota,\kappa)}$, $B_1 = F_{(\iota',\kappa')}$. Assume $m, n > 1$. Using Lemma 9 one proves easily that $A \wedge B \equiv F_{(\iota,\kappa)\wedge(\iota',\kappa')} \oplus ((A \wedge B') \vee (A' \wedge B))$. Similarly, for $m > 1$, $n = 1$, we have $A \wedge B \equiv F_{(\iota,\kappa)\wedge(\iota',\kappa')} \oplus (A' \wedge B)$ and the canonical form follows from the induction hypothesis. For $m = 1$, $n > 1$ proceed symmetrically. For $m = n = 1$, $A \wedge B \equiv F_{(\iota,\kappa)\wedge(\iota',\kappa')}$. Again, using the induction hypothesis, the closure by $\vee$, $\oplus$, and the Substitution Lemma, we get an equivalent canonical automaton.

The general case may be reduced to one of the special cases above, because $E_\alpha \wedge A \equiv (C_\alpha \wedge A) \vee (D_\alpha \wedge A)$. □

Since (ι, κ)-replication requires a rather involved analysis, let us first consider the $(1, 1)$ case.

Proposition 13. *For $A, B \in \mathcal{C}$ one can find in polynomial time an automaton in $\mathcal{C}$ equivalent to $A \to B$.*

Proof. First, let us deal with two special cases for which the general method does not work. For $B \not\geq C_3$ simple calculations give the following equivalences: $A \to B \equiv (D_1 \oplus A) \wedge B$ for $B \in \{C_1, E_1, C_2, D_2, D_3\}$, $A \to D_1 \equiv A \vee D_1$, $A \to E_2 \equiv A \to D_3$. By the Substitution Lemma, the equivalent canonical forms follow from the closure by $\oplus$, $\vee$, and $\wedge$.

The second special case is when B contains non-trivial flowers but $B \not\geq F_{(0,2)}$. First, let us see that $A \to F_{(0,1)} \equiv (D_1 \oplus A) \wedge F_{(0,1)}$. The inequality $A \to F_{(0,1)} \geq (D_1 \oplus A) \wedge F_{(0,1)}$ follows easily from Lemma 8. For the converse it remains to observe that the following strategy is winning for Duplicator in $G(A \to F_{(0,1)}, (D_1 \oplus A) \wedge F_{(0,1)})$: in $D_1 \oplus A$ mimic Spoiler and in $F_{(0,1)}$ apply the strategy from $G(C_1 \to F_{(0,1)}, F_{(0,1)})$ given by Theorem 11 (3 and 4). An analogous argument shows that $A \to F_{(1,2)} \equiv (D_1 \oplus A) \wedge F_{(1,2)}$. For the remaining possible values of B we will show $A \to B \equiv (D_1 \oplus A) \wedge F_{(0,1)} \wedge F_{(1,2)}$. Again, $A \to B \geq (D_1 \oplus A) \wedge B \wedge B \geq (D_1 \oplus A) \wedge F_{(0,1)} \wedge F_{(1,2)}$ is easy. For the converse, observe that B only uses ranks $1, 2, 3$. Consider the following strategy for Duplicator in $G(A \to B, (D_1 \oplus A) \wedge F_{(0,1)} \wedge F_{(1,2)})$. In the component $D_1 \oplus A$ simply mimic the behaviour of Spoiler's critical token. In $F_{(0,1)}$ use the strategy from $G(C_1 \to B', F_{(0,1)})$, where B' denotes B with ranks 1 and 2 replaced by 0 and rank 3 replaced by 1. In $F_{(1,2)}$ use the strategy from $G(C_1 \to B'', F_{(1,2)})$, where B'' denotes B with all 3's replaced by 1's. The combination of these three strategies is winning for Duplicator.

For the remaining automata, we will show that what really matters is the maximal simple branching automaton contained in $C_1 \to B$. There are two main cases: either $C_{\omega^{k-1}} < B \leq C_{\omega^k}$ ($C_3 \leq B \leq C_\omega$ for $k = 1$), or $C_{\omega^{\omega \cdot 2 + (k-1)}} < B \leq C_{\omega^{\omega \cdot 2 + k}}$ ($F_{(0,2)} \leq B \leq C_{\omega^{\omega \cdot 2}}$ for $k = 1$). In the first case $A \to B \equiv C^-_{\omega^k} \oplus A$, in the second case $A \to B \equiv C^-_{\omega^{\omega \cdot 2 + k}} \oplus A$. Since the proofs are entirely analogous, we will only consider the first case. We only need to argue that $A \to B \leq C^-_{\omega^k} \oplus A$, since the converse inequality is obvious.

Let us start with $B = C_{\omega^k}$. Denote the head loop of C_{ω^k} by λ_0. It is enough to show a winning strategy in $G(A \to B, C^-_{\omega^k} \oplus A)$. Since no path from the head loop of $A \to B$ to λ_0 goes through an accepting loop, Duplicator may keep his critical token in the head loop of $C^-_{\omega^k}$ as long as at least one of Spoiler's tokens can reach λ_0. Hence, for every token produced by Spoiler in λ_0, Duplicator can produce a *doppelgänger*. When none of Spoiler's tokens can reach λ_0 any more, Duplicator moves his critical token to A and mimics

Spoiler.

Let us now suppose that $C_{\omega^{k-1}} < B < C_{\omega^k}$, $k \geq 2$ (for $k = 1$ the proof is very similar). The strategy for Duplicator in $G(A \to B, C^-_{\omega^k} \oplus A)$ is as follows. Let m be such that $B \leq C_{\omega^{k-1}m}$. For every token x_i produced by Spoiler using the head loop of $A \to B$, Duplicator produces m tokens $y_i^1, \ldots, y_i^m$ using the head loop of $C^-_{\omega^k}$. Then the tokens $y_i^1, \ldots, y_i^m$ play against x_i simulating Duplicator's winning strategy from $G(B, (C_1 \oplus C_{\omega^{k-1}})^m)$. When Spoiler moves his critical token to A, Duplicator does the same and keeps mimicking Spoiler in A.

Thus we managed to simplify $A \to B$ to $C^-_\alpha \oplus A$ where $\alpha = \omega^k$ or $\alpha = \omega^{\omega \cdot 2 + k}$. An equivalent canonical automaton is provided by Lemma 18. □

Now we are ready to deal with (ι, κ)-replication. Since $C_{\omega^{\omega \cdot 3}} = C_1 \xrightarrow{(0,1)} C_{\omega^{\omega+1}}$, the class $\mathcal{C}$ is not closed by $\xrightarrow{(\iota,\kappa)}$. However, adding the three top canonical automata is enough to get the closure property.

Proposition 14. *For $A, A_\iota, \ldots, A_\kappa \in \mathcal{C}$, $\iota, \kappa < \omega$, one can find in polynomial time an automaton in $\mathcal{C}'$ equivalent to $A \xrightarrow{(\iota,\kappa)} A_\iota, \ldots, A_\kappa$.*

Proof. Let $B = A \xrightarrow{(\iota,\kappa)} A_\iota, \ldots, A_\kappa$. If B admits any of the automata $C_{\omega^{\omega \cdot 3}}$, $C_{\omega^{\omega \cdot 3}+1}$, $C_{\omega^{\omega \cdot 3}+2}$, then it is equivalent to the maximal one it admits (see Theorems 11 and 12). Let us assume B admits none of the three automata above. Let us also assume that $\iota < \kappa$.

1. If some A_i contains a $(0,1)$-flower and some A_j contains a $(1,2)$-flower, then $B \equiv (F_{(\iota,\kappa)} \oplus A) \wedge F_{(1,2)} \wedge F_{(0,1)}$. It is easy to show that $(F_{(\iota,\kappa)} \oplus A) \wedge F_{(1,2)} \wedge F_{(0,1)} \leq B$ (c. f. Lemma 8). We shall concentrate on the converse inequality. From the hypothesis that B does not admit $C_{\omega^{\omega \cdot 3}+1}$, it follows easily that κ must be odd and $A_\iota, \ldots, A_{\kappa-1}$ must be $(1,2)$ automata. Furthermore, since B does not admit $C_{\omega^{\omega \cdot 3}}$, A_k uses only ranks $1, 2, 3$. The strategy for Duplicator in $G(B, (F_{(\iota,\kappa)} \oplus A) \wedge F_{(1,2)} \wedge F_{(0,1)})$ is analogous to the one used in the proof of the previous proposition. In the component $F_{(\iota,\kappa)} \oplus A$ simply mimic the behaviour of Spoiler's critical token. In $F_{(0,1)}$, loop around the 1-loop whenever Spoiler loops around the 1-loop of a $(0,1)$-flower in A_κ (again, if the run is to be accepting, this may happen only finitely many times), otherwise loop around 0-loop. For the strategy in $F_{(1,2)}$, treat all the ranks appearing in Spoiler's $F_{(\iota,\kappa)}$ or A as 2's, and the 3's in A_κ as 1's. Seen

this way, B is a $(1,2)$-automaton, and by Theorem 11 Spoiler's actions can be simulated in $F_{(1,2)}$.

2. If A_i contain only $(1,2)$-flowers, then $B \equiv (F_{(\iota,\kappa)} \oplus A) \wedge F_{(1,2)}$. This is proved just like the first case.

3. If A_i contain only $(0,1)$-flowers, then $B \equiv (A \xrightarrow{(\iota,\kappa)} A_\iota, \ldots, A_{\kappa-1}, C_1) \wedge F_{(0,1)}$ (use case 4 or 5 to get a canonical form). Like in the first case, κ must be odd, $A_\iota, \ldots, A_{\kappa-1}$ must be $(1,2)$-automata. Consequently, it must be A_κ that contains a $(0,1)$-flower. Since A_κ contain no $F_{(0,2)}$ (by the hypothesis no A_i does), $A_\kappa = F_{(0,1)}$. Again $B \geq ((A \xrightarrow{(\iota,\kappa)} A_\iota, \ldots, A_{\kappa-1}, C_1) \wedge F_{(0,1)})$ is easy. The strategy for Duplicator in $G(B, (A \xrightarrow{(\iota,\kappa)} A_\iota, \ldots, A_{\kappa-1}, C_1) \wedge F_{(0,1)})$ is to copy Spoiler's actions in $A \xrightarrow{(\iota,\kappa)} A_\iota, \ldots, A_{\kappa-1}, C_1$ and in $F_{(0,1)}$ keep record of all 1's appearing in A_κ (if the run is to be accepting, there may be only finitely many altogether).

4. If A_i contain no non-trivial flowers, $\iota = 0$, and A_ι contains a D_2, then $B \equiv (F_{(\iota,\kappa)} \oplus A) \wedge F_{(1,2)}$. The inequality $B \leq (F_{(\iota,\kappa)} \oplus A) \wedge F_{(1,2)}$ is proved just like in the first case. Let us see that the converse holds. Consider the game $G((F_{(\iota,\kappa)} \oplus A) \wedge F_{(1,2)}, B)$ and the following strategy for Duplicator. Copy Spoiler's actions in $F_{(\iota,\kappa)} \oplus A$, but whenever Spoiler enters the 1-loop in $(1,2)$, loop once around 0-loop, move the extra token to the head loop of D_2, and keep looping around until Spoiler leaves his 1-loop. Then remove your extra token, and so on. It is easy to see that the strategy is winning for Duplicator.

5. If A_i contain no non-trivial flowers and either $\iota \neq 0$ or A_ι contains no D_2, then $B \equiv F_{(\iota,\kappa)} \oplus A$. To prove it, we have to describe the strategy for Duplicator in $G(B, F_{(\iota,\kappa)} \oplus A)$. During the whole play keep numbering the new tokens produced by Spoiler according to their birth time. (As usual, the left token is considered a parent, the right token is born, transitions of the form $p \longrightarrow \top, q$ or $p \longrightarrow q, \top$ do not produce new tokens.) The strategy is as follows. While there are no new tokens in rejecting loops in $A_\iota, \ldots, A_\kappa$, keep copying Spoiler's moves in his $F_{(\iota,\kappa)}$. When the first new token, say x_{i_1}, enters a 1-loop, start looping around the 1-loop of your $F_{(\iota,\kappa)}$ (the loop exists since $\iota < \kappa$), and keep doing it until x_{i_1} leaves the 1-loop. If it does

not happen, Spoiler will lose. When it does happen, stop looping around 1-loop. Investigate all the ranks used by Spoiler in (ι, κ)-flower while you were simulating x_{i_1}, choose the highest one, say k, and loop once a k-loop. Afterwords, if there are no tokens in rejecting loops in A_i, copy Spoiler's moves. Otherwise, choose the token with the smallest number, say x_{i_2}, start looping around the loop with the highest rank 1 in your (ι, κ)-flower, and so on.

Let us see that if Spoiler does not enter A, he loses the game. If the run constructed by Spoiler is to be rejecting, either the highest rank used infinitely often in $F_{(\iota,\kappa)}$ is odd, or some token stays forever in a rejecting loop in one of $A_\iota, \ldots, A_\kappa$. In any case Duplicator's strategy guarantees a rejecting run for him as well. Let us suppose that Spoiler's run is accepting. If only finitely many new tokens entered rejecting loops in $A_\iota, \ldots, A_\kappa$, then there was a round such that from this round on Duplicator was simply mimicking Spoiler's actions in $F_{(\iota,\kappa)}$ and so Duplicator's run is also accepting. Suppose that infinitely many new tokens visited rejecting loops in $A_\iota, \ldots, A_\kappa$. We have assumed that either $\iota \neq 0$ or A_ι contains no D_2. In either case the ranks greater then 0 must have been used infinitely many times in $F_{(\iota,\kappa)}$. Consequently, the highest rank used in $F_{(\iota,\kappa)}$ is greater then 1, and Duplicator's run is accepting despite infinitely many 1's used in $F_{(\iota,\kappa)}$.

Suppose now that Spoiler leaves $F_{(\iota,\kappa)}$. Following the argument used in the proof of the closure by $\oplus$, we may suppose that the simple automaton containing the head loop of A is at least a $\overline{(\iota, \kappa)}$. When Spoiler enters A, he may produce no more tokens in $A_\iota, \ldots, A_\kappa$. From now on Duplicator should mimic Spoiler's behaviour in his copy of A, handling rejecting loops in $A_\iota, \ldots, A_\kappa$ in the usual way.

What is left is the case $\iota = \kappa$. If κ is odd, $B = A \to A_1$. If κ is even, A_0 must be a $(1, 2)$-automaton. In the cases 2 and 4 proceed just like before. In the case 5, the automaton A_0 cannot contain D_2. If $A_0 \in \{C_1, D_1, C_2\}$, then $B \equiv (C_1 \oplus A) \wedge A_0$. If $A_0 = E_1$, then $B \equiv C_1 \oplus (A \vee C_1)$. □

The following corollary sums up the closure results.

Corollary 5. *The class of canonical automata $\mathcal{C}'$ is closed by $\vee$, $\oplus$, $\wedge$, $\xrightarrow{(\iota,\kappa)}$, and the equivalent automaton can be found in polynomial time.*

Proof. The claim is an almost immediate consequence of the preceding propositions. Only the automata $C_{\omega^{\omega\cdot 3}}$, $C_{\omega^{\omega\cdot 3}+1}$, $C_{\omega^{\omega\cdot 3}+2}$ need special care: if the

result of the operation in question admits any of these automata, it is equivalent to the hardest one it admits (Theorems 11 and 12). □

3.11 Completeness

In this section we show that the canonical automata represent the $\equiv_W$-classes of all deterministically recognizable tree languages. We will implicitly use Corollary 5 and the Substitution Lemma (Lemma 6, page 44) on several occasions.

We will say that a transition is *positive* if one of its branches lies on an accepting loop, and *negative* if one of its branches lies on a rejecting loop. Note that a transition may be positive and negative at the same time. Recall the notion of replication (see page 31). We say that a state is j-replicated if it is replicated by a j-loop. An automaton is j-replicated if its initial state is j-replicated.

Finally, let us recall the *lifting operation* invented by Niwiński and Walukiewicz and used to prove the decidability of the deterministic index hierarchy (Theorem 1, page 21).

Lemma 19 (Niwiński and Walukiewicz [25])**.** *For each deterministic automaton A one can compute (in polynomial time if the productive states are given) an automaton $A\uparrow^0\uparrow^1\ldots\uparrow^n$ such that $L(A) = L(A\uparrow^0\uparrow^1\ldots\uparrow^n)$ and if a state q has the rank $j \leq n$ than q lies on a j-loop of a (j,n)-flower.*

Theorem 13. *For every deterministic tree automaton there exists an equivalent canonical automaton.*

Proof. Let A be a deterministic tree automaton. From Theorem 11 (7) it follows that if A admits $C_{\omega^{\omega\cdot 3}+2}$, $A \equiv C_{\omega^{\omega\cdot 3}+2}$. If A does not admit $C_{\omega^{\omega\cdot 3}+2}$, then by Theorem 11 (5 and 6) if A admits $C_{\omega^{\omega\cdot 3}+1}$, $A \equiv C_{\omega^{\omega\cdot 3}+1}$. Otherwise $L(A) \in \Delta_3$ and if A admits $C_{\omega^{\omega\cdot 3}}$, then $A \equiv C_{\omega^{\omega\cdot 3}}$ (Theorem 12). In the remaining of the proof we will assume that A admits none of these three automata. We will proceed by induction on the height of the DAG of SCCs of A. Let X denote the head SCC of A. We will say that X contains a transition $p \longrightarrow p', p''$, if X contains all three states, p, p', and p''. We consider four separate cases.

1. X contains a positive transition. Should A admit $F_{(0,1)}$, it would also admit $C_{\omega^{\omega\cdot 3}+1}$, which is excluded by our initial assumption. Consequently, A is a $(1,2)$-automaton and $L(A) \in \Pi^0_2$. Since $L(F_{(0,2)})$ is Π^0_2-complete, $A \leq F_{(1,2)}$. If A admits $F_{(1,2)}$ or $C_1 \xrightarrow{(0,0)} D_1 \oplus C_1$, then $L(A)$ is Π^0_2-complete and $A \equiv F_{(1,2)}$. Otherwise, X contains no rejecting loops and A does not admit $D_1 \oplus C_1$. By Theorem 11 (2), $L(A) \in \Pi^0_1$ and since $L(C_1 \oplus D_1)$ is Π^0_1-complete, $A \leq C_1 \oplus D_1$. If A admits D_1 it also admits $C_1 \oplus D_1$, and so $A \equiv C_2$. If A does not admit D_1 it means that it contains no rejecting loop. Hence, A accepts every tree and $A \equiv C_1$.

2. X contains an accepting loop and a negative transition, but no positive transitions. Let λ_+ be an accepting loop in X and λ_X be a loop visiting all X's nodes and containing a branch of the (negative) transition contained in X. Since X does not contain positive transitions, λ_X is rejecting. The loops λ_+ and λ_X form a $(0,1)$-flower. Hence, A admits $F_{(0,1)}$. Furthermore, should A contain a $(0,2)$-flower, it would obviously be replicated by λ_X and A would admit $C_{\omega^{\omega\cdot 3}}$, which contradicts our general hypothesis. Hence, A does not admit $F_{(0,2)}$, which means A is a $(1,3)$-automaton (Theorem 1, page 22). Without loss of generality we may assume that it uses only ranks $1, 2, 3$.

By Theorem 11 (3 and 4), if A admits neither $F_{(1,2)}$ nor $C_1 \xrightarrow{(0,0)} D_1 \oplus C_1$, then $A \equiv F_{(0,1)}$. Suppose that A admits one of these two automata. Consider the game $G(F_{(0,1)} \wedge F_{(1,2)}, A)$. Let x^1 and x^2 be Spoiler's tokens in $F_{(0,1)}$ and $F_{(1,2)}$, respectively. Since X contains a (negative) transition, Duplicator can split his critical token into y^1 and y^2 within X, and move y^1 to the $(0,1)$-flower in X, and y^2 to the $(1,2)$-flower, or to the accepting loop replicating a weak $(1,2)$-flower (if A admits $C_1 \xrightarrow{(0,0)} D_1 \oplus C_1$). Then y^1 should mimic x^1, and y^2 should mimic x^2 (either directly, or adapting the strategy from $G(F_{(1,2)}), C_1 \xrightarrow{(0,0)} D_1 \oplus C_1)$). Hence, Duplicator has a strategy to win the game.

It follows that $F_{(0,1)} \wedge F_{(1,2)} \leq A$. For the converse inequality, let us call the states with rank 3 contained in a $(0,1)$-flower *red*, and the remaining *blue*. Since A does not admit $C_1 \xrightarrow{(0,0)} F_{(0,1)}$, no red state is replicated by an accepting loop. Consider the game $G(A, F_{(0,1)} \wedge F_{(1,2)})$. For a strategy in $F_{(1,2)}$ Duplicator should treat all the red states as if they had rank 1; the automaton A modified this way does not admit $F_{(0,1)}$, so Duplicator may use

the strategy given by Theorem 11 (3 and 4). In $F_{(0,1)}$ Duplicator should loop a 1-loop whenever some Spoiler's token is in a red state. Otherwise, Duplicator should loop a 0-loop. Let us see that this strategy is winning. Suppose that Spoiler's run is accepting. After changing the ranks of red states from 3 to 1 it is still accepting, so Duplicator's token in $F_{(1,2)}$ visited an accepting path. By the Replication Lemma (Lemma 8, page 55), the occurrences of red states in Spoiler's run may be covered by a finite number of paths. Furthermore, each of these paths is accepting, so it may only contain a finite number of red states. Hence, there may be only finitely many red states in Spoiler's run and the path visited by Duplicator's token in $F_{(0,1)}$ is also accepting. Suppose now, that Spoiler's run is rejecting. If red states occurred only finitely often, Spoiler's run is still rejecting after changing their ranks to 1, so Duplicator's token in $F_{(1,2)}$ visited a rejecting path. If there were infinitely many red states in Spoiler's run, Duplicator's token in $F_{(0,1)}$ visited a rejecting path. By Lemma 9, $A \equiv F_{(1,3)}$.

3. X contains some transitions but no accepting loops. Let $q_i \xrightarrow{\sigma_i} q_i', q_i'', i = 1, \ldots, n$ be all the transitions such that $q_i \in X$ and $q_i', q_i'' \notin X$. Let $p_j \xrightarrow{\sigma_i, d} p_j'$ $j = 1, \ldots, m$ be all the remaining transitions such that $p_j \in X$ and $p_j' \notin X$. We will call the automata $(A)_{q_i'}$, $(A)_{q_i''}$ and $(A)_{p_j'}$ the *child automata of* X. By the induction hypothesis we may assume that they are in canonical forms. Let $B = ((A)_{q_1'} \wedge (A)_{q_1''}) \vee \ldots \vee ((A)_{q_n'} \wedge (A)_{q_n''}) \vee (A)_{p_1'} \vee \ldots \vee (A)_{p_m'}$. It is not difficult to see that A is equivalent to $C_1 \to B$.

4. X contains no transitions. Recall that this means exactly that at most one branch of every transition stays in X. First replace subtrees rooted in the target states of transitions whose all branches leave X with one canonical automaton B just like above. Let (ι, κ) denote the highest index of a flower contained in X. It is well defined, because a strongly connected component admitting $F_{(0,j)}$ and $F_{(1,j+1)}$ must also admit $F_{(0,j+1)}$. We may assume that X uses only ranks $\iota, \ldots, \kappa$, and that each j-loop is indeed a j-loop in a (j, κ)-flower (Lemma 19). For each $j = \iota, \ldots, \kappa$, let B_j be the alternative of all the child automata replicated by a j-loop in X. By induction hypothesis and the closure properties, we may assume that $B_\iota, \ldots, B_\kappa$ and B are canonical automata. Let $A' = B \xrightarrow{(\iota,\kappa)} B_\iota, \ldots, B_\kappa$. We will show that $A \equiv A'$.

If $\iota = \kappa$, the assertion is clear. Suppose that $\iota < \kappa$. Obviously, $A' \geq$

A. Let us see that $A' \leq A$. Let A'' denote the result of the following simplifications performed on A'.

- If some B_i contains a $(0,1)$-flower and some B_j contains a $(1,2)$-flower, replace B_κ with a $(1,3)$-flower.
- If some B_i contains a $(0,1)$-flower and no B_j contains a $(1,2)$-flower, replace B_κ with a $(0,1)$-flower.
- If some B_i contains a $(1,2)$-flower and no B_j contains a $(0,1)$-flower, replace B_κ with a $(1,2)$-flower.
- If $B_\iota, \ldots, B_\kappa$ admit no $F_{(\iota,\kappa)}$ with $\iota < \kappa$, remove B_κ.
- If $\iota = 0$ and B_ι admits D_2, replace B_ι with D_2. Otherwise, remove B_ι.
- Remove all $B_{\iota+1}, \ldots, B_{\kappa-1}$.

Examination of the five cases considered in the proof of Proposition 14 reveals that A' and A'' have identical canonical forms. Consequently, $A' \equiv A''$, and it is enough to show that $A'' \leq A$. Consider all (ι,κ)-flowers in X. Choose any one whose ι-loop replicates D_2, if there is one, or take any (ι,κ)-flower otherwise. Then, extend the κ-loop to a loop using all the transitions in X. Denote this flower, together with the subtrees replicated by ι-loop or κ-loop, by F. One can prove easily that $A'' \leq F \oplus B$, and obviously $F \oplus B \leq A$. □

3.12 Algorithm

From the proof of the completeness theorem, one easily extracts an algorithm to calculate the canonical form of a given deterministic automaton (Algorithm 1).

Corollary 6. *For a deterministic tree automaton, a Wadge equivalent canonical automaton can be calculated within the time of finding the productive states of the automaton.*

Proof. It is easy to see that the size of the canonical forms returned by the recursive calls of each depth is bounded by the size of A (up to a uniform constant factor). To prove the time complexity of the algorithm assume that the productive states of A are given. Checking if A admits any of the automata

Algorithm 1 The canonical form of deterministic tree automata

if A admits $C_{\omega^{\omega\cdot 3}+2}$ **then**
  return $C_{\omega^{\omega\cdot 3}+2}$
else if A admits $C_{\omega^{\omega\cdot 3}+1}$ **then**
  return $C_{\omega^{\omega\cdot 3}+1}$
else if A admits $C_{\omega^{\omega\cdot 3}}$ **then**
  return $C_{\omega^{\omega\cdot 3}}$
else
  $X :=$ the head component of A
  if X contains a positive transition **then**
    if A admits $F_{(1,2)}$ or A admits $\emptyset \xrightarrow{(0,0)} D_2$ **then**
      return $F_{(1,2)}$
    else if A admits D_1 **then**
      return C_2
    else
      return C_1
    end if
  else if X contains a negative transition **then**
    if X admits C_1 **then**
      if A admits $F_{(1,2)}$ or A admits $\emptyset \xrightarrow{(0,0)} D_2$ **then**
        return $F_{(1,3)}$
      else
        return $F_{(0,1)}$
      end if
    else
      $B :=$ the alternative of the canonical forms of X's children
      return $C_1 \to B$
    end if
  else {X contains no transitions}
    $B :=$ the alternative of the canonical forms of X's non-replicated children
    lift ranks in X
    $(\iota, \kappa) :=$ the index of the maximal flower
    for $j := \iota$ to κ **do**
      $B_j :=$ the alternative of the canonical forms of X's j-replicated children
    end for
    return $B \xrightarrow{(\iota,\kappa)} B_\iota, \ldots, B_\kappa$
  end if
end if

mentioned in the algorithm can be easily done in polynomial time. The operations on the automata returned by the recursive calls of the procedure (lines 25, 26, 29, 33, and 35) are polynomial in the size of those automata, and by the initial remark also in the size of the automaton. By Lemma 19 the lifting operation is also polynomial. Therefore, when implemented dynamically, this procedure takes polynomial time for each SCC. Processing the entire automaton increases this polynomial by a linear factor. □

Instead of a canonical automaton, the algorithm above can return its "name", i. e., a letter C, D, or E, and an ordinal $\alpha \leq \omega^{\omega \cdot 3} + 2$ presented as a polynomial in ω^ω, with the coefficients presented as polynomials in ω. Since for such presentation it is decidable in linear time if $\alpha \leq \beta$, as an immediate consequence of Corollary 6 and Lemma 14 we get an algorithm for Wadge reducibility.

Corollary 7. *For deterministic tree automata A, B it is decidable if $L(A) \leq_W L(B)$ (within the time of finding the productive states of the automata).*

Chapter 4

Wadge Degrees

A truly remarkable aspect of the Wagner hierarchy is how simply it embeds into the general Wadge hierarchy: a language from the level $\omega^k n_k + \ldots + \omega n_1 + n_0$ of the Wagner hierarchy is on the level $\omega_1^k n_k + \ldots + \omega_1 n_1 + n_0$ of the Wadge hierarchy. This fact was observed by many people independently in 1990s (according to J. Duparc, V. Selivanov might be the first, or P. Simonnet and J.-P. Ressayre) but never published. Indeed, the proof is almost straightforward if one uses a handy and powerful tool: simple set-theoretical operations corresponding to ordinal sum and multiplication of Wadge degrees discovered already by Wadge in his PhD thesis [38, 39].

Duparc completed the list of the operations with the exponentiation with the base ω_1 [5], and later used some of those to show that in the case of deterministic context-free word languages the embedding is equally elegant [6]. Applying this technique we will calculate the Wadge degrees of deterministic tree languages and provide a lower bound for the height of the Wadge ordering of weak tree languages. The latter is a joint work with Jacques Duparc.

4.1 More on Wadge Hierarchy

Martin's famous determinacy theorem [16] gives very precise information on the shape of the Wadge hierarchy.

Theorem 14 (Wadge Lemma)**.** *For Borel languages L, M it holds that*

$$L \leq_W M \quad \textit{or} \quad L^{\complement} \geq_W M\,.$$

Proof. By determinacy one of the players has a winning strategy in $G_W(L, M)$. If it is Duplicator, then $L \leq_W M$. Suppose it is Spoiler who has a winning strategy. One can easily transform this strategy into a winning strategy for Duplicator in $G_W(M, L^{\complement})$ (see [12] or [29] for details). □

In other words the theorem says that the width of the Wadge hierarchy is at most two, and if L and M are incomparable, then $L \equiv_W M^{\complement}$. It means that the Wadge ordering is almost linear. The second fundamental result states that it is also a well-ordering (see [12]).

Theorem 15 (Wadge, Martin, Monk)**.** *The Wadge hierarchy is well-founded.*

Altogether, the position of a language in the Wadge hierarchy is determined, up to complementation, by its height. It also follows that each $\equiv_W$-class has one or two direct successors: the minimal $\equiv_W$-classes above.

If $L \equiv_W L^{\complement}$ then L is called *self dual*. Otherwise L is not comparable with $L^{\complement}$ and is called *non self dual*. Both notions are naturally extended to $\equiv_W$-classes. Note that a level in the hierarchy contains only one $\equiv_W$-class if and only if the class is self dual. Otherwise it contains two non self dual $\equiv_W$-classes which are complementary two each other, i. e., each set in one class is Wadge equivalent to the complement of each set in the other class. Steel and Van Weesp proved that the self dual and non self dual levels alternate (see [12]). If the alphabet is finite, which is our case, on limit steps we have non self duals. Finally, the self dual classes are easily obtained from their non self dual predecessors by means of disjoint union (see the proposition below).

Before we formalize this, let us observe that the choice of the alphabet Σ is of no importance. Let Σ and Σ' be finite alphabets containing at least two letters. For any language L over Σ, one can find a Wadge equivalent language L' over Σ'. Furthermore, if L is recognized by an automaton, so can be L', and the construction of the new automaton is effective. Therefore, without loss of generality we may assume $\Sigma = \{a, b\}$.

Fix a language L over Σ. Let L^- consist of those trees t for which there exists n such that

- $t(0^n) = b$ and $t(0^m) = a$ for all $m < n$,
- $t.0^{n+1} \in L$,

and let $L^+ = L^- \cup \{t : \forall_n\, t(0^n) = a\}$. Define also $L^{\pm}$ as the set of trees t for which either $t(\varepsilon) = a$ and $t.0 \in L$ or $t(\varepsilon) = b$ and $t.0 \in L^{\complement}$.

Proposition 15. *Let L be a Borel tree language.*

1. *If L is self dual, L^+ and L^- are its two direct successors.*
2. *If L is non self dual, $L^\pm$ is its only direct successor.*
3. *If L is self dual, there exists $M <_W L$ such that $L \equiv_W M^\pm$.*

The proof of this fact can be found in [6]. It is worth noting that if L is self dual, $L^\pm \equiv_W L$.

The results summarized above make it reasonable to ignore self duals when counting the height. Hence, the following definition. The *Wadge degree* of a non self dual set is an ordinal given by the inductive formula

- $d_W(\emptyset) = d_W(\emptyset^{\complement}) = 1$,
- $d_W(L) = \sup\{d_W(M) + 1 : \ M \text{ is non self dual, } M <_W L\}$ for $L >_W \emptyset$.

For self duals, we set

- $d_W(L) = \sup\{d_W(M) : \ M \text{ is non self dual, } M <_W L\}$.

By our definition, each Wadge degree is shared by three $\equiv_W$-classes: L, $L^{\complement}$, and $L^\pm$ for a non self dual L. This definition is a slight modification of the one introduced by Duparc, the original definition by Wadge counted self duals too (see [5] for details).

As a final remark of this section we would like to give the reader a hint about the height of the Wadge hierarchy. Let $\Omega_1 = 1$, and $\Omega_{n+1} = \omega_1^{\Omega_n}$. The Wadge degree of a Π^0_n-complete set is Ω_n for $1 < n < \omega$.

4.2 Arithmetic

One of the most surprising discoveries about the Wadge hierarchy is that the arithmetical structure on the Wadge degrees is reflected in simple set-theoretical operations on languages that in addition have natural interpretations in terms of Wadge games. We will now present a number of such operations selected for the first main goal of this chapter: calculating the Wadge degrees of deterministic tree languages. The operations were introduced by Wadge for the languages of infinite words [38, 39], here we present a straightforward adaptation to trees.

For $L, M \subseteq T_\Sigma$ define $L + M$ as the set of trees $t \in T_\Sigma$ satisfying one of the following conditions:

- $t.0 \in M$ and $t(10^n) = a$ for all n,
- 10^n is the first node on the path 10^* labeled with b and either $t(10^n 0) = a$ and $t.10^n 00 \in L$ or $t(10^n 0) = b$ and $t.10^n 00 \in L^{\complement}$.

When playing a Wadge game, being in charge of $M + L$ is like being in charge of L with one extra move that erases everything the player has played so far and changes the set the player is in charge of to M or $M^{\complement}$. This move can be played only once during the play, and is executed by playing b on the path 10^* for the first time. By choosing the next letter on this path we make choice between M and $M^{\complement}$.

Multiple sum is performed from left to right, i. e., $L_1 + L_2 + L_3 + L_4 = ((L_1 + L_2) + L_3) + L_4$. Later it will become clear that the operation is associative up to Wadge equivalence.

The next operation is a generalization of $+$. It lets the player choose from a countable collection of languages. Let $L_n \subseteq T_\Sigma$ for $n < \omega$. Define $\sup^-_{n<\omega} L_n$ as the set of trees $t \in T_\Sigma$ satisfying the following conditions for some k:

- 1^k is the first node on 1^* labeled with b,
- $t.1^k 0 \in L_k$.

Define also

$$\sup\nolimits^+_n L_n = \left(\sup\nolimits^-_n L_n\right) \cup \{t : \forall_n\, t(1^n) = a\}\,.$$

Intuitively, in the first operation we reject the trees in which the Wadge game player did not choose any L_n, and in the second operation we accept them. The operations are dual:

$$\left(\sup\nolimits^+_n L_n\right)^{\complement} = \sup\nolimits^-_n \left(L_n^{\complement}\right)\,.$$

Later we will make use of a different kind of duality, given by the lemma below.

Lemma 20. *If $L_1, L_2, \ldots$ satisfy $\forall_i\, \exists_{j>i}\, L_i <_W L_j$ then*

$$\sup\nolimits^-_n L_n \equiv_W \left(\sup\nolimits^+_n L_n\right)^{\complement}\,.$$

Proof. By the last observation, for the inequality $\leq_W$ it is enough to consider $G_W(\sup^-_n L_n, \sup^-_n (L_n)^{\complement})$. The strategy for Duplicator is to wait until Spoiler chooses L_i and then choose $(L_j)^{\complement}$ such that $L_j >_W L_i$. Then $(L_j)^{\complement} >_W L_i$,

and Duplicator can use the strategy from $G_W(L_i, (L_j)^{\complement})$. If Spoiler chooses no L_n, both trees are outside the respective languages.

For the converse inequality consider $G_W(\sup_n^-(L_n)^{\complement}, \sup_n^- L_n)$ and proceed analogously. □

Using sup one could easily define multiplication by countable ordinals as iterated sums, but here we will need the multiplication by ω_1. Let $L \cdot \omega_1$ be the set of trees satisfying the following conditions for some n:

- $t(1^n) = b$ and $t(1^m) = a$ for all $m > n$,
- either $t(1^n 0) = a$ and $t(1^n 00) \in L$ or $t(1^n 0) = b$ and $t(1^n 00) \in L^{\complement}$.

By multiplying L by ω_1 we get a language that lets the player switch between L and $L^{\complement}$ entirely unboundedly.

The names of the operations and the notation used make the following theorem rather expected.

Theorem 16 (Wadge). *For $L_1, L_2, \ldots \subseteq T_\Sigma$ it holds that*

$$\begin{aligned} d_W(L_1 + L_2) &= d_W(L_1) + d_W(L_2)\,, \\ d_W(\sup_n^+ L_n) = d_W(\sup_n^- L_n) &= \sup_n d_W(L_n)\,, \\ d_W(L_1 \cdot \omega_1) &= d_W(L_1) \cdot \omega_1\,. \end{aligned}$$

An elegant proof of this theorem can be found in [6].

4.3 Calculating Degrees

The non-branching canonical automata discussed in Sect. 3.4 give a complete representation of the Wadge ordering of regular word languages. The beauty of the Wagner hierarchy lies in the correspondence already mentioned in the introduction. Abusing the notation we will write $d_W(A)$ instead of $d_W(L(A))$. For $\alpha < \omega^\omega$, $\alpha = \omega^k n_k + \ldots + \omega^1 n_1 + n_0$ and

$$j_{\mathrm{Reg}}(\omega^k n_k + \ldots + \omega^1 n_1 + n_0) = \omega_1^k n_k + \ldots + \omega_1^1 n_1 + n_0\,,$$

it holds that

$$d_W(\hat{C}_\alpha) = d_W(\hat{D}_\alpha) = d_W(\hat{E}_\alpha) = j_{\mathrm{Reg}}(\alpha)\,.$$

We call j_{Reg} the *Wagner function* for regular word languages. Later, we will give a proof of this result.

In [6], Wagner's results are extended to deterministic context free word languages. It is proved that the hierarchy has the height $(\omega^\omega)^\omega$ and that the analogue of the Wagner function is given by the formula

$$j_{\mathrm{DetCFree}}((\omega^\omega)^k\alpha_k + \ldots + (\omega^\omega)^1\alpha_1 + \alpha_0) = (\omega_1)^k\alpha_k + \ldots + (\omega_1)^1\alpha_1 + \alpha_0\,,$$

with $\alpha_i < \omega^\omega$.

In this section we describe the Wagner function for deterministic tree languages. Due to the lack of complementation for deterministic languages, the function will not be that elegant. To this end we need to relate the operations from the previous section with the composing operations from Sect. 3.3. Let us start with the replication $\to$. Recall that $(A)^n$ denotes $\underbrace{A \wedge \ldots \wedge A}_{n}$ (page 46).

Lemma 21. *For every automaton A*

$$L(C_1 \to A) \;\equiv_W\; \sup\nolimits_n^- L((A)^n)\,.$$

Proof. The strategy for Duplicator in $G(L(C_1 \to A), \sup_n^- L((A)^n))$ is as follows. While the leftmost branch of the run of $C_1 \to A$ on the tree constructed by Spoiler keeps in the head loop, Duplicator should play a's everywhere. Suppose that Spoiler finally plays a letter that makes the leftmost path of the run exit the head loop in 0^k. Then $t_S \in L(C_1 \to A)$ iff $t_S.0^i1 \in L(A)$ for $i = 0, 1, \ldots, k-1$. Now, if Duplicator plays b in 1^k, then $t_D \in \sup_n^- L((A)^n))$ iff $t_D.1^k0 \in L((A)^n))$. Hence, Duplicator wins if he copies Spoiler's actions.

For the converse inequality, consider $G(\sup_n^- L((A)^n), L(C_1 \to A))$. The winning strategy for Duplicator is as follows. As long as Spoiler does not choose any of $L((A)^n)$, Duplicator plays a tree that keeps the leftmost branch of the computation of $C_1 \to A$ in the head loop. If this continues forever, Duplicator wins. If finally Spoiler chooses $L((A)^k)$, then Duplicator simply applies the strategy given by $(A)^k \leq C_1 \to A$ (Lemma 8, page 55). □

The symbols chosen to denote canonical automata were to reflect the shape of the hierarchy: dual automata are called C_α and D_α, and self dual ones are denoted by E_α. Since for the canonical automata of the form $C_1 \to A$ the automaton $L(C_1 \to A)^{\complement}$ is not equivalent to a deterministic language, we will need the following lemma.

Lemma 22. *For every canonical automaton of the form $C_1 \to A$,*

$$L(C_1 \to A)^{\complement} \leq_W L(C_1 \oplus (C_1 \to A))\,.$$

Proof. By Lemma 21,

$$L(C_1 \to A)^{\complement} = (\sup_n^- L((A)^n))^{\complement} \equiv \sup_n^+ L((A)^n)^{\complement}\,.$$

The strategy for Duplicator in $G(\sup_n^+ L((A)^n)^{\complement}, L(C_1 \oplus (C_1 \to A)))$ is as follows. While Spoiler keeps playing a's on the path 1^*, Duplicator should play a tree which keeps the leftmost path of the run of $C_1 \oplus (C_1 \to A)$ in C_1. Suppose that Spoiler finally plays b in 1^k. Then $t_S \in \sup_n^+ L((A)^n)^{\complement}$ iff $t_S.1^k0 \in L((A)^k)$. Furthermore, $t_D \in L(C_1 \oplus (C_1 \to A))$ iff $t_D.0^k \in L(C_1 \oplus (C_1 \to A))$. Hence, Duplicator could apply a winning strategy from $G(L((A)^k)^{\complement}, L(C_1 \oplus (C_1 \to A)))$, if it existed.

Let us check that $L((A)^k)^{\complement} \leq_W L(C_1 \oplus (C_1 \to A))$. By the Wadge Lemma (Lemma 14), it is enough to prove that $L((A)^k) <_W L(C_1 \oplus (C_1 \to A))$. By Lemma 8 (page 55), $(A)^k \leq C_1 \to A$. Since $C_1 \to A$ is canonical, $C_1 \to A = C_\alpha$ for $\alpha = \omega^l$, $0 < l < \omega$ or $\alpha = \omega^{\omega \cdot 2 + l}$, $0 \leq l < \omega$. By Theorem 10, in either case $C_\alpha < C_{\alpha+1} = C_1 \oplus C_\alpha$. □

Now, let us move to $\oplus$. For automata on words, one can prove that $L(A \oplus B) \equiv L(A) + L(B)$ for arbitrary A and B. For tree automata this is only true if B dominates (see page 60) all the simple automata that constitute A. That suffices, since we only need this property for canonical $A \oplus B$.

Lemma 23. *For every canonical automaton $A = A_1 \oplus \ldots \oplus A_n$, where A_i are simple automata,*

$$L(A) \equiv_W L(A_1) + \ldots + L(A_n)\,.$$

Proof. We proceed by induction on n. For $n = 1$ the claim is trivial. Take $n > 1$. Let $A' = A_1 \oplus \ldots \oplus A_{n-1}$. By induction hypothesis, $L(A') \equiv_W L(A_1) + \ldots + L(A_{n-1})$, so it is enough to prove that $L(A' \oplus A_n) \equiv_W L(A') + L(A_n)$. Abusing slightly the definition of canonical automata we will assume that A' and A_n are over the same input alphabet Σ.

Let us first consider $G_W(L(A' \oplus A_n), L(A') + L(A_n))$. While the tree constructed by Spoiler does not force the corresponding run of $A' \oplus A_n$ into A_n, Spoiler is in fact in charge of $L(A'_\tau)$, where A'_τ is a copy of A' over an alphabet extended with one fresh letter τ. By Lemma 5 (page 44), $L(A') \equiv_W L(A'_\tau)$. Hence, Duplicator wins if he applies the strategy from $G_W(L(A'), L(A'_\tau))$ in $t_D.0$, and plays a's elsewhere. Suppose that Spoiler does force the run of $A' \oplus A_n$ into the initial state of A_n. By definition of $\oplus$, this must happen in $v = 0^k$ for some $k < \omega$. It is easy to see, adapting the proof of Lemma

13 (page 60) and Corollary 4 (page 61), that if Spoiler has a strategy in this game, he also has a strategy such that, if the computation of $A' \oplus A_n$ on the constructed tree enters A_n, than all the paths staying within A' are accepting. Then, by definition of $\oplus$, $t_S \in L(A' \oplus A_n)$ iff $t_S.v0 \in L((A_n)_\tau)$. Hence, Duplicator should play ba on the path 10^*, say in the nodes 10^l and 10^l0, in $t_D.10^l00$ use the strategy from $G_W(L((A_n)_\tau)), L(A_n))$, and elsewhere play a's.

For the converse inequality, consider $G_W(L(A') + L(A_n), L(A' \oplus A_n))$. If Spoiler does not play b on the path 10^*, then $t_S \in L(A') + L(A_n)$ iff $t_S.0 \in L(A')$. Hence, while Spoiler plays a's on 10^*, Duplicator may apply the strategy from $G_W(L(A'), L(A'_\tau))$. Suppose Spoiler plays b in 10^k, and a in 10^k0. Then $t_S \in L(A') + L(A_n)$ iff $t_S.10^k00 \in L(A_n)$. Duplicator should now play on the leftmost branch a sequence that will force the computation to A_n in a node v' and than apply the strategy from $G_W(L(A_n), L((A_n)_\tau)$ in $t_D.v'0$. Elsewhere Duplicator should play τ's.

Suppose now that Spoiler plays b in 10^k and in 10^k0. Then $t_S \in L(A') + L(A_n)$ iff $t_S.10^k00 \in L(A_n)^{\complement}$. If A_n is a non-branching automaton, then by the definition of canonical automata it is an alternative of two dual flowers $A_n \equiv A_n^{\complement}$ and $A_n \vee A_n^{\complement} \equiv A_n$. Hence, Duplicator may proceed like before, only instead of the strategy from $G_W(L(A_n), L((A_n)_\tau))$, he should use the winning strategy from $G_W(L(A_n)^{\complement}, L((A_n)_\tau))$.

Finally, let A_n be a branching automaton. By Lemma 22, $L(A_n)^{\complement} \leq_W L(C_1 \oplus A_n)$. Recall that $WF_{(0,n)} \equiv C_{n+1}$, $WF_{(1,n+1)} \equiv D_{n+1}$ (see page 56). Hence, replacing C_n and D_n with $WF_{(\iota,\kappa)}$ in the definition of C_α for $\alpha = \alpha\omega^{\omega\cdot 2} + \alpha_1\omega^\omega + \alpha_0\omega + n$, with $\alpha_i < \omega^\omega$ and either $\alpha_0 > 0$ or $\alpha_2 > \alpha_1$ (page 47), we may assume that A_{n-1} is not equal to D_1 nor E_1. Hence, on the leftmost branch Duplicator can always play a sequence that will move the computation to an accepting loop in A_{n-1} in a node 0^l for some $l < \omega$. In the nodes that are not descendants of 0^l, Duplicator should play τ and in the subtree rooted in 0^l Duplicator should simulate the strategy from the game $G_W(L(A_n)^{\complement}, L(C_1 \oplus A_n))$ given by Lemma 22. □

Let us now compute the Wadge degrees of canonical flowers.

Lemma 24. *For every index* (ι, κ)

$$d_W(F_{(\iota,\kappa)}) = \omega_1^{\kappa-\iota}.$$

Proof. We will proceed by induction on $\kappa - \iota$. For $\iota = \kappa$, we have $L(F_{(0,0)}) = \emptyset^{\complement}$ and $L(F_{(1,1)}) = \emptyset$, so $d_W((F_{(0,0)}) = d_W((F_{(1,1)}) = 1$ by definition. Let

us take $\iota < \kappa$ with κ odd. By Theorem 16 and the induction hypothesis, it is enough to prove that $L(F_{(\iota,\kappa)}) \equiv_W L(F_{(\iota,\kappa-1)}) \cdot \omega_1$. By Lemma 9 (page 56), $L(F_{(\iota,\kappa)}) \equiv_W L_{(\iota,\kappa)} \equiv_W L'_{(\iota,\kappa)}$, where $L'_{(\iota,\kappa)}$ is a language of trees over the alphabet $\{\iota, \iota+1, \ldots, \kappa\}$, such that the highest number occurring infinitely often on the leftmost path is even. We will prove that $L'_{(\iota,\kappa)} \equiv_W L'_{(\iota,\kappa-1)} \cdot \omega_1$.

Let us consider $G_W(L'_{(\iota,\kappa)}, L'_{(\iota,\kappa-1)} \cdot \omega_1)$. While Spoiler does not play κ on the leftmost branch, Duplicator should copy his actions in $t_D.0$, and play a's elsewhere. If Spoiler never uses κ on the leftmost branch, Duplicator wins. Suppose that Spoiler does play κ. Then Duplicator should play the erasing letter in the first free node on the rightmost path, say 1^k, and again copy Spoiler's actions in $t_D.1^k0$. If after a few repetitions of this scenario, Spoiler never uses κ on the leftmost branch again, Duplicator wins. If Spoiler uses κ infinitely often on the leftmost branch, Duplicator plays the "erasing" letter infinitely often. Hence, both trees will be outside of the respective languages.

In $G_W(L'_{(\iota,\kappa-1)} \cdot \omega_1, L'_{(\iota,\kappa)})$ the strategy for Duplicator is similar. As long as Spoiler does not play the "erasing" letter, Duplicator should simply copy the leftmost branch, and play ι elsewhere. When Spoiler plays the "erasing" letter in 1^k, Duplicator should play κ on the leftmost branch and then copy the letters played by Spoiler on the path $1^k0, 1^k00, \ldots$, until Spoiler plays the "erasing" letter again. Like before, it is easy to see that this strategy is winning.

For $\iota < \kappa$ and κ even, observe that $L(F_{(\iota,\kappa)}) \equiv_W L(F_{\overline{(\iota,\kappa)}})^{\complement}$. Hence, $d_W(F_{(\iota,\kappa)}) = d_W(F_{\overline{(\iota,\kappa)}}) = \omega_1^{\kappa-\iota}$ by the previous case. □

Using the above results we can calculate the Wadge degrees of the languages recognized by simple automata.

Lemma 25.

1. $d_W(C_1) = d_W(D_1) = d_W(E_1) = 1$.
2. $d_W(C_{\omega^{\omega+k}}) = d_W(D_{\omega^{\omega+k}}) = d_W(E_{\omega^{\omega+k}}) = \omega_1^{k+1}$ *for* $0 \leq k < \omega$.
3. $d_W(C_{\omega^k}) = \omega^k$, *for all* $k \geq 1$.
4. $d_W(C_{\omega^{\omega\cdot 2+k}}) = \omega_1^{\omega}\omega^k$, *for all* $k \geq 0$.
5. $d_W(C_{\omega^{\omega\cdot 3}}) = \omega_1^{\omega+1}$.

Proof. First, observe that for each (ι,κ), $L(F_{\overline{(\iota,\kappa)}}) \equiv_W L(F_{(\iota,\kappa)})^{\complement}$. In consequence, $L(F_{(\iota,\kappa)} \vee F_{\overline{(\iota,\kappa)}}) \equiv_W L(F_{(\iota,\kappa)}))^{\pm}$, so $d_W(F_{(\iota,\kappa)}) = d_W(F_{(\iota,\kappa)} \vee F_{\overline{(\iota,\kappa)}})$. By Lemma 24, we immediately get (1) and (2).

The proof of (3) is by induction. For $k = 1$, $C_\omega = C_1 \to C_3$. By Lemma 21 and Lemma 9 (page 56)

$$L(C_\omega) = \sup^-_{n\geq 1} L\left((C_3)^n\right) = \sup^-_{n\geq 1} L\left(C_{1+2n}\right) .$$

By (1) and Lemma 23, $d_W(C_k) = k$. Hence,

$$d_W(C_\omega) = \sup_{n\geq 1} d_W\left(C_{1+2n}\right) = \sup_{n\geq 1}\left(1+2n\right) = \omega .$$

The inductive step is similar: $C_{\omega^{k+1}} = C_1 \to C_1 \oplus C_{\omega^k}$, so by Lemma 21, Lemma 10 (page 57), and the induction hypothesis

$$\begin{aligned} d_W(C_{\omega^{k+1}}) &= \sup_{n\geq 1} d_W\left((C_1 \oplus C_{\omega^k})^n\right) = \sup_{n\geq 1} d_W\left(C_1 \oplus nC_{\omega^k}\right) = \\ &= \sup_{n\geq 1}\left(1+\omega^k n\right) = \omega^{k+1} . \end{aligned}$$

Analogously we get (4):

$$\begin{aligned} d_W(C_{\omega^{\omega\cdot 2}}) &= \sup_{n\geq 1} d_W\left((F_{(0,2)})^n\right) = \sup_{n\geq 1} d_W\left(F_{(0,2n)}\right) = \\ &= \sup_{n\geq 1} \omega_1^{2n} = \omega_1^{\omega} , \end{aligned}$$

$$\begin{aligned} d_W(C_{\omega^{\omega\cdot 2+k+1}}) &= \sup_{n\geq 1} d_W\left((C_1 \oplus C_{\omega^{\omega\cdot 2+k}})^n\right) = \sup_{n\geq 1} d_W\left(C_1 \oplus nC_{\omega^{\omega\cdot 2+k}}\right) = \\ &= \sup_{n\geq 1}(1+\omega_1^{\omega}\omega^k n) = \omega_1^{\omega}\omega^{k+1} . \end{aligned}$$

For (5) we need to prove that $L(C_1 \xrightarrow{(0,1)} F_{(0,2)}) \equiv_W L(C_1 \to F_{(0,2)}) \cdot \omega_1$. Let us first consider $G_W(L(C_1 \xrightarrow{(0,1)} F_{(0,2)}), L(C_1 \to F_{(0,2)}) \cdot \omega_1)$. Suppose that for some time Spoiler plays in such a way that on the leftmost branch the computation stays in the 1-loop of the head component. Then on paths of the form $0^{2i+1}10^*$ the computation stays in $F_{(0,2)}$. While Spoiler plays like this, Duplicator should keep the computation in the head loop of $C_1 \to F_{(0,2)}$, and on the path 0^i10^* mimic the path $0^{2i+1}10^*$. Suppose that finally, in the node 0^{2k}, the computation on t_S moves to the 0-loop in the head component of $C_1 \xrightarrow{(0,1)} F_{(0,2)}$. Then Duplicator should play a letter that makes the leftmost branch of the computation move to the accepting tail loop of $C_1 \to F_{(0,2)}$, an on keep mimicking Spoiler on the paths 0^i10^* for $i = 1, 2, \ldots, k$. If the

computation on t_S never returns to the head 1-loop, Duplicator wins. If it does, Duplicator should play the letter τ, that "erases" everything played so far (see page 89), choose $L(C_1 \to F_{(0,2)})$ again (not the complement), make the computation loop k times in the head loop sending k paths of the form 0^i10^* to $F_{(0,2)}$, and then proceed as before, producing for each new path of the computation on t_S in $F_{(0,2)}$, a path of the computation on t_D in $F_{(0,2)}$, and so on. If after a few iterations of this scenario the leftmost branch of the computation on t_S stabilizes in the 0-loop, Duplicator wins. Otherwise, the leftmost branch of the computation on t_S will be rejecting, but then Duplicator will play the "erasing" letter infinitely often, and t_D will not belong to $L(C_1 \to F_{(0,2)})\cdot\omega_1$. Hence, $L(C_1 \xrightarrow{(0,1)} F_{(0,2)}) \leq_W L(C_1 \to F_{(0,2)})\cdot\omega_1$,

For the converse inequality consider $G_W(L(C_1 \to F_{(0,2)}) \cdot \omega_1, L(C_1 \xrightarrow{(0,1)} F_{(0,2)}))$. As long as Spoiler does not use the "erasing" letter, Duplicator should follow the strategy given by $L(C_1 \to F_{(0,2)}) \leq L(C_1 \xrightarrow{(0,1)} F_{(0,2)})$. Suppose that Spoiler plays the "erasing" letter, and chooses $L(C_1 \to F_{(0,2)})$ again. Let 0^l be the last node played by Duplicator on the leftmost branch. In $t_D.0^l0$, Duplicator should again apply the strategy given by $L(C_1 \to F_{(0,2)}) \leq_W L(C_1 \xrightarrow{(0,1)} F_{(0,2)})$, and in the remaining paths play letters that will make the computation accepting. If Spoiler chooses $L(C_1 \to F_{(0,2)})^{\complement}$, Duplicator proceeds like above, only uses the strategy given by $L(C_1 \to F_{(0,2)})^{\complement} \leq_W L(C_1 \oplus (C_1 \to F_{(0,2)})) \leq_W L(C_1 \xrightarrow{(0,1)} F_{(0,2)})$ (Lemma 22 and Theorem 12, page 69). If Spoiler plays the "erasing" letter infinitely many times, on the leftmost branch of t_D the computation loops infinitely often in 1-loop, and so t_D is rejected. □

We have now everything we need to get the Wagner function for deterministic tree languages. For every $\alpha < \omega^{\omega\cdot 3}$ there is a unique presentation

$$\alpha = \omega^{\omega\cdot 2}(\omega^p k_p + \ldots + \omega^0 k_0) + \omega^\omega(\omega^q l_q + \ldots + \omega^0 l_0) + \omega^r m_r + \ldots + \omega^0 m_0\,,$$

with $k_p, l_q, m_r > 0$. Define $j_{\mathrm{Det}}(\alpha)$ for $\alpha < \omega^{\omega\cdot 3}$ with the formula

$$j_{\mathrm{Det}}(\alpha) = \omega_1^\omega(\omega^p k_p + \ldots + \omega^0 k_0) + \omega_1(\omega_1^q l_q + \ldots + \omega_1^0 l_0) + \omega^r m_r + \ldots + \omega^0 m_0$$

and let $j_{\mathrm{Det}}(\omega^{\omega\cdot 3}) = \omega_1^{\omega+1}$, $j_{\mathrm{Det}}(\omega^{\omega\cdot 3}+1) = \omega_1^{\omega_1}$.

Theorem 17. *For all $\alpha \leq \omega^{\omega\cdot 3}+1$ (whenever the automata are defined)*

$$d_W(C_\alpha) = d_W(D_\alpha) = d_W(E_\alpha) = j_{\mathrm{Det}}(\alpha)\,.$$

Proof. Recall that the Wadge degree of a Π^0_3-complete language is $\omega_1^{\omega_1}$. Hence, the case of $C_{\omega^{\omega\cdot 3}+1}$ is obvious. The remaining cases follow directly from Lemmas 23 and 25. □

The result on the word languages mentioned in the beginning of the section follows immediately from this theorem.

Corollary 8. *For all* $\alpha < \omega^\omega$

$$d_W(\hat{C}_\alpha) = d_W(\hat{D}_\alpha) = d_W(\hat{E}_\alpha) = j_{\mathrm{Reg}}(\alpha)\,.$$

4.4 Conciliatory World

In the remaining part of the present chapter we work mainly with binary trees that may not be full, i.e., partial functions $t : \{0,1\} \to \Sigma$ with a prefix closed domain. Let $\tilde{T}_\Sigma$ denote the set of such trees. A *conciliatory tree language* over Σ is a subset of $\tilde{T}_\Sigma$.

For conciliatory languages L, M we define a *conciliatory version of the Wadge game* $G_C(L, M)$ (see [5]). The classical Wadge game was defined for languages of full infinite trees. The players had to add both child nodes under each node they had put in the previous round. Only Duplicator was allowed to skip, but not forever. He had to make infinitely many real moves, so that the resulting tree was full. Here, this requirement is not needed: in each round both players are allowed to put arbitrary (finite) number of new nodes, including no nodes at all. Obviously, the resulting trees may contain finite branches, or even be finite.

For conciliatory languages L, M we use the notation $L \leq_C M$ iff Duplicator has a winning strategy in the game $G_C(L, M)$. If $L \leq_C M$ and $M \leq_C L$, we will write $L \equiv_C M$. The *conciliatory hierarchy* is the order induced by $\leq_C$ on the $\equiv_C$ classes of conciliatory languages.

Let us examine the relations between the conciliatory hierarchy and the Wadge hierarchy. Consider $T_{\Sigma\cup\{s\}}$, where s stands for "skip". For a tree $t \in T_{\Sigma\cup\{s\}}$ we will define $u(t) \in \tilde{T}_\Sigma$, called the *undressing* of t. Informally, we want to omit the skips in a top-down manner. Suppose we are in a node v such that $t(v) = s$. We would like to ignore this node and replace it with the next one. However, in case of trees we have two nodes to choose from: $v0$ and $v1$. Let us always choose $v0$. Another problem is that we may encounter an infinite sequence of s's. This would keep us replacing the current node with its left child, and never get to a symbol different from s. In that case,

$u(t)$ simply does not contain v. Now, let us see a formal definition. For each $v \in \{0,1\}^*$ consider two possibly infinite sequences:

- $w_0 = \varepsilon$, $v_0 = v$,
- for $v_i = bv'$, $w_{i+1} = w_i b$, $v_{i+1} = 0v'$ if $t(w_{i+1}) = s$, and $v_{i+1} = v'$ otherwise.

If $v_n = \varepsilon$ for some n, then $v \in \mathrm{dom} u(t)$ and $u(t)(v) = t(w_n)$. Otherwise, $v \notin \mathrm{dom} u(t)$.

For a conciliatory language L, define L_s as the set of trees that belong to L after undressing, i. e., $L_s = \{t \in T_{\Sigma \cup \{s\}} : \ u(t) \in L\}$. The mapping $L \mapsto L_s$ gives a natural embedding of the conciliatory hierarchy into the Wadge hierarchy.

Lemma 26. *For all conciliatory languages L and M,*

$$L \leq_C M \iff L_s \leq_W M_s \,.$$

Proof. A strategy in one game can be translated easily to a strategy in the other: arbitrary skipping in $G_C(L, M)$ gives the same power as the s labels in $G_W(L_s, M_s)$. In particular, in $G_W(L_s, M_s)$ Duplicator does not need skipping at all. □

Recall that a language is called *self dual* if it is equivalent to its complement. The conciliatory hierarchy does not contain self dual languages: a strategy for Spoiler in $G_C(L, L^{\complement})$ is to skip in the first round, and then copy Duplicator's moves. By the lemma above, L_s is non self dual in terms of ordinary Wadge reducibility. Altogether, this shows that the conciliatory languages correspond to certain non self dual languages. To which ones?

To answer this question we make a detour via word languages. A conciliatory word language is simply $L \subseteq \Sigma^* \cup \Sigma^\omega$, i. e., a set of finite or infinite words. As for trees, we define L_s as the set of words over $\Sigma \cup \{s\}$, such that when we ignore all the s we obtain a word (finite or infinite) from L. Obviously, Lemma 26 holds also for words, but we get much more than that.

Theorem 18. *(Duparc [5]) For every $L \subseteq \Sigma^\omega$ of finite Borel rank, L is non self dual iff there exists $F \subseteq \Sigma^*$ such that $L \equiv_W (F \cup L)_s$.*

In particular, a word language of finite Borel rank is non self dual iff it is Wadge equivalent to C_s for some conciliatory language C. Using Theorem 18, we will show that this also holds for tree languages.

Corollary 9. *For every $L \subseteq T_\Sigma$ of finite Borel rank, L is non self dual iff $L \equiv_W C_s$ for some conciliatory tree language C.*

Proof. First, observe that L is Wadge equivalent to L_w, which is the set of sequences obtained by writing down the trees from L level by level from left to right. The "writing down" and its inverse are suitable continuous reductions. By Theorem 18, L_w is equivalent to $(L_w \cup F)_s$ for some set of finite words F. Now, we need a conciliatory tree language C, such that $C_s \equiv_W (L_w \cup F)_s$.

For $t \in \tilde{T}_\Sigma$ let fixed(t) denote the sequence obtained by writing down the tree level by level from left to right until the first missing node is reached. Note that fixed(t) is infinite iff t is a full tree. Let $C = \{t \in \tilde{T}_\Sigma : \text{ fixed}(t) \in L_w \cup F\}$. The identity function reduces L to C_s, so $C_s \geq_W L \equiv_W (L_w \cup F)_s$.

To prove the converse inequality consider a hybrid game $G(C_s, (L_w \cup F)_s)$. (Formally, instead of $(L_w \cup F)_s$ one can take a Wadge equivalent language $T(L_w \cup F)$, consisting of trees which have the leftmost path in $(L_w \cup F)_s$.) In this game Spoiler constructs a tree t_S, and Duplicator constructs a word w_D. A winning strategy for Duplicator is to undress on-line the tree constructed by Spoiler and write it down level by level, from left to right. When Duplicator reaches a place where the node is missing, he plays s until Spoiler plays the missing node. If this never happens, Duplicator plays s forever. At the end of the play, fixed$(u(t_S)) = u(w_D)$. Hence, $t_S \in C_s \iff w_D \in (L_w \cup F)_s$. □

Since the conciliatory hierarchy can be embedded into the Wadge hierarchy (Lemma 26), we can define inductively the *conciliatory degree* of a language:

- $d_C(\emptyset) = d_C(\emptyset^{\complement}) = 1$,
- $d_C(L) = \sup\{d_C(M) + 1 : M <_C L\}$ for $L >_C \emptyset$.

By Corollary 9 and Lemma 26, for conciliatory L such that L_s has finite Borel rank, $d_C(L) = d_W(L_s)$. This observation lets us work with the conciliatory hierarchy instead of the Wadge hierarchy, as long as we restrict ourselves to non self dual sets of finite Borel ranks.

The operations $+$, $\sup^+$ and $\sup^-$ can be generalised easily to conciliatory sets. Now, we will also need the multiplication by countable ordinals. It is defined inductively by means of $+$ and $\sup^+$:

- $L \cdot 1 = L$,

- $L \cdot (\alpha + 1) = L + L \cdot \alpha$,
- $L \cdot \lambda = \sup^{+}_{\gamma<\lambda} L \cdot \gamma$ for limit ordinals λ.

Yet another arithmetical operation that has an almost exact counterpart is exponentiation. Let $L \subseteq \tilde{T}_\Sigma$. For $t \in \tilde{T}_\Sigma$ let

$$i(t)(a_1 a_2 \dots a_n) = \begin{cases} s & \text{if } \exists_k\, t(a_1 0 a_2 \dots 0 a_n 1^k) = b \\ t(a_1 0 a_2 \dots 0 a_n 0) & \text{otherwise} \end{cases} .$$

Intuitively, the rightmost path starting in $a_1 0 a_2 \dots 0 a_n$ tells us whether to skip the node $a_1 0 a_2 \dots 0 a_n 0$ or not. Let

$$\exp L = \{t \in \tilde{T}_\Sigma :\ u(i(t)) \in L\} .$$

A player in charge of $\exp L$ is like a player in charge of L with an extra possibility to decide that a chosen node he played in the past (and the subtree rooted in its right child) is to be ignored.

Like before, the operations correspond to order-arithmetical operations on conciliatory degrees, only with exponentiation we get a slight discrepancy. The *pseudoexponentiation* is defined as $\exp' \alpha = \omega_1^{\alpha+\varepsilon}$, where

$$\varepsilon = \begin{cases} -1 & \text{if } d_C(L) < \omega \\ 0 & \text{if } d_C(L) = \beta + n \text{ and } \mathrm{cof}\beta = \omega_1 \\ +1 & \text{if } d_C(L) = \beta + n \text{ and } \mathrm{cof}\beta = \omega \end{cases} .$$

Theorem 19 (Duparc [5]). *For $L, M \subseteq \tilde{T}_\Sigma$, L_s, M_s Borel of finite rank, and a countable ordinal α it holds that*

$$\begin{aligned} d_C(L + M) &= d_C(L) + d_C(M) , \\ d_C(L \cdot \alpha) &= d_C(L) \cdot \alpha , \\ d_C(\exp L) &= \exp' d_C(L) . \end{aligned}$$

4.5 A Lower Bound

In this section we give a lower bound for the height of the Wadge hierarchy restricted to weak languages. We obtain the bound by showing that weak automata are closed by some of the operations from the previous section. For that purpose we need to adapt the definition of weak automata to the

conciliatory case. Apparently, we could simply feed the automaton with a tree that may not be full, get a "partial" computation tree, and use the standard definition of acceptance via weak parity games. However, for technical reasons (and for the sake of elegance) we prefer to use a slightly modified version of weak parity games. We say that a play (finite or infinite) is winning for Eve if the highest rank used at least once is even. Thus we put an end to the special treatment of the situations, when a player cannot move.

In this section, the *language recognized by the automaton*, $L(A)$, is the set of accepted trees, full or not. For the set of full trees accepted by the automaton we will use the symbol $L^\omega(A)$.

The extension of weak recognizability we propose is preserved by the embedding operation used in Sect. 4.4.

Corollary 10. *The mapping $L \mapsto L_s$ embeds the conciliatory hierarchy restricted to weakly recognizable languages into the Wadge hierarchy restricted to weakly recognizable languages.*

Proof. By Lemma 26 it is enough to prove that each weak automaton A can be transformed into a weak automaton A' such that $L^\omega(A') = (L(A))_s$. The automaton A' simply moves deterministically to the left without changing the state whenever it sees a node labeled with s. In other words, it is enough to add $\{q \xrightarrow{s,0} q : q \in Q\}$ to the transition relation of A. □

Let A, B be weak alternating tree automata with input alphabet $\{a, b\}$. We will construct weak automata $A+B$, $A\cdot\omega$, $\exp A$, such that $L(A+B) \equiv_C L(B) + L(A)$, $L(A \cdot \omega) \equiv_C L(A) \cdot \omega$, and $L(\exp A) \equiv_C \exp L(A)$.

Sum. Consider the automaton $B + A$ defined on Fig. 4.1. The symbol $*$ denotes any letter. The diamond states are existential and the box states are universal. The circle states can be treated as existential, but in fact they give no choice to either player. The transitions leading to A, B and $B^{\complement}$ should be understood as transitions to the initial states of the according automata. The priority functions of B and $B^{\complement}$ should be increased by 2, so that they do not use the value 0. It is easy to check that $L(B + A) = L(B) + L(A)$.

Multiplication by ω. The automaton $A \cdot \omega$ is shown on Fig. 4.1. The language recognized by $A \cdot \omega$ consists of trees having no b's on the path 1^* or satisfying the following conditions for some $0 < i \leq k$ and n:

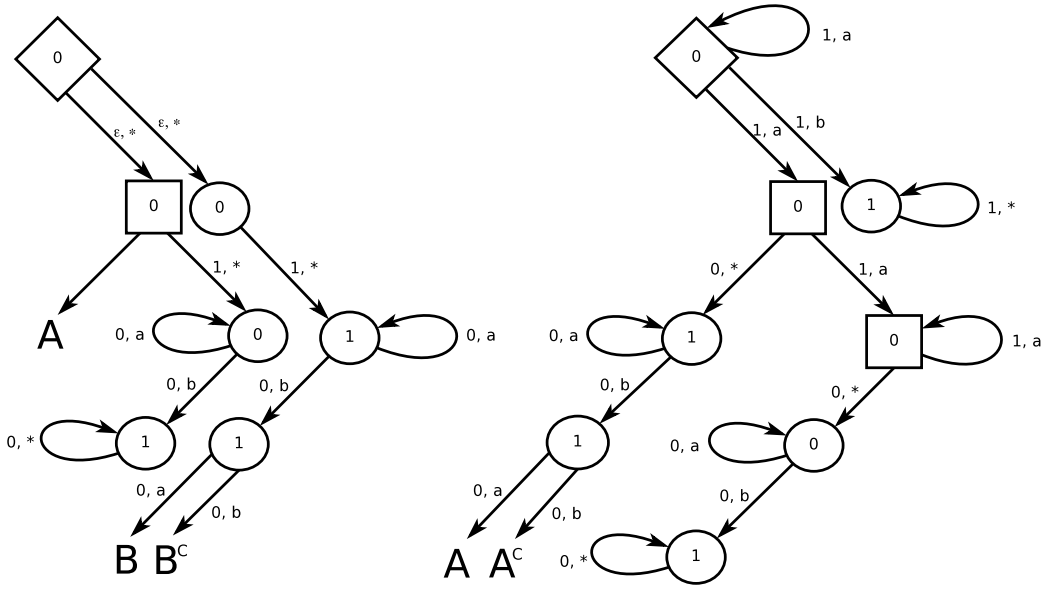


Figure 4.1: The automata $B + A$ and $A \cdot \omega$.

- 1^k is the first node labeled with b on the path 1^*,
- i is minimal such that for all $i < j \leq k$ the path $1^j 0^+$ contains no b's,
- $1^i 0^n$ is the first node labeled with b lying on the path $1^i 0^+$,
- either $t(1^i 0^n 0) = a$ and $t.1^i 0^n 00 \in L(A)$ or $t(1^i 0^n 0) = b$ and $t.1^i 0^n 00 \in L(A)^{\complement}$.

Denote by L_k the set of trees satisfying the four conditions above for a fixed k. Let us see that $L_k \equiv_W \emptyset + L(A) \cdot k$.

Consider $G_W(L_k, \emptyset + L(A) \cdot k)$. If Spoiler does not play the word from $a^{k-1}b\{a, b\}^\omega$ on the rightmost branch, Duplicator should fill the whole tree with a's . This way $t_D \in \emptyset + L(A) \cdot k$ iff $t_D.0 \in \emptyset$. Hence, $t_D \notin \emptyset + L(A) \cdot k$ and Duplicator wins. Suppose that Spoiler played $a^{k-1}b$ on the rightmost branch. Now, while Spoiler plays a's on the leftmost paths starting in $1, 11, \ldots, 1^k$, Duplicator should play a's everywhere. Again, this guarantees a win for Duplicator. Suppose that finally, Spoiler plays b in $1^{i_1}0^{j_1}$ for some $0 < i_1 \leq k$ and $j_1 < \omega$. Duplicator should now play the "erasing" letter, say in the node v_1. From now on, $t_D \in \emptyset + L(A) \cdot k$ iff $t_D.v_1 0 \in L(A) \cdot k$. As long as Spoiler

does not play b in a node $1^{i_2}0^{j_2}$ for some $i_1 < i_2 \leq k$ and some $j_2 < \omega$, Duplicator should observe Spoiler's actions in $t_S.1^{i_1}0^{j_1}$, copy them to $t_D.v_100$ and play a's elsewhere. If this continues, Duplicator wins. When Spoiler does play b in $1^{i_2}0^{j_2}$ for some $i_1 < i_2 \leq k$ and some $j_2 < \omega$, Duplicator should play the erasing letter, say in v_2. Afterwords, $t_D \in \emptyset + L(A) \cdot k \iff t_D.v_20 \in L(A) \cdot (k-1)$ and Duplicator should proceed as before.

If this process stops before the numbers $i_1, i_2, \ldots$, reach k, Duplicator wins. Suppose that for some l, $i_l = k$. Then $t_S \in L_K$ iff either $t_S(1^k0^k0) = a$ and $t_S.1^k0^{j_l}00 \in L(A)$ or $t_S(1^{i_l}0^{j_l}0) = b$ and $t_S.1^{i_l}0^{j_l}00 \in L(A)^{\complement}$. An in the case of Duplicator, $t_D \in \emptyset + L(A) \cdot k$ iff Duplicator does not play the erasing letter (if $l = k$ he actually cannot) and either $t_D(v_l0) = a$ and $t_D.v_l00 \in L(A)$ or $t_D(v_l0) = b$ and $t_D.v_l00 \in L(A)^{\complement}$. Hence, Duplicator wins too.

Thus we have shown $L_k \leq_W \emptyset + L(A) \cdot k$. Analogously we prove the converse inequality and get the equivalence.

Now, consider $G_C(L(A) \cdot \omega, L(A \cdot \omega))$. By definition, $L(A) \cdot \omega$ consists of trees having no b on the rightmost path, or such that 1^k is the first node on this path labeled with b, and $t.1^k0 \in L(A) \cdot k$. Consider the following strategy for Duplicator. First, only observe the rightmost path of Spoiler's tree t_S. While Spoiler plays a's, keep playing a's in t_D (Duplicator's tree). If Spoiler never plays a b, Duplicator wins. Suppose Spoiler plays his first b in the node 1^k. Duplicator should also play b in the node 1^k. Now, the result of the play depends only on whether $t_S.1^k0 \in L(A) \cdot k \iff t_D \in L_k$, and Duplicator should simply use the strategy from $G_C(L(A) \cdot k, L_k)$.

In the game $G_C(L(A \cdot \omega), L(A) \cdot \omega))$ the only difference is that Duplicator should play one more a: if Spoiler plays the first b on the rightmost path in the node 1^k, then Duplicator should put his first b in 1^{k+2}, so that he can later use the winning strategy from $G_C(L_k, L(A) \cdot (k+2))$. Let us see that such a strategy exists. It is enough to show a strategy in $G_W(\emptyset + L(A) \cdot k, L(A) \cdot (k+2))$. Duplicator should immediately play the "erasing" letter, and choose $L(A)^{\complement}$ if $L(A)^{\complement} \neq \emptyset$, and $L(A)$ otherwise. Then, while Spoiler does not play the "erasing" letter, Duplicator should play in $t_D.0$ a tree that is not in the chosen language, and a's elsewhere. When Spoiler does play the "erasing" letter, say in v, Duplicator should also play the "erasing" letter, say in v'. Then Duplicator should copy $t_S.v0$ into $t_D.v'0$.

Pseudoexponentiation. Both previous constructions were performed by combining two or three automata with a particularly chosen gadget. The

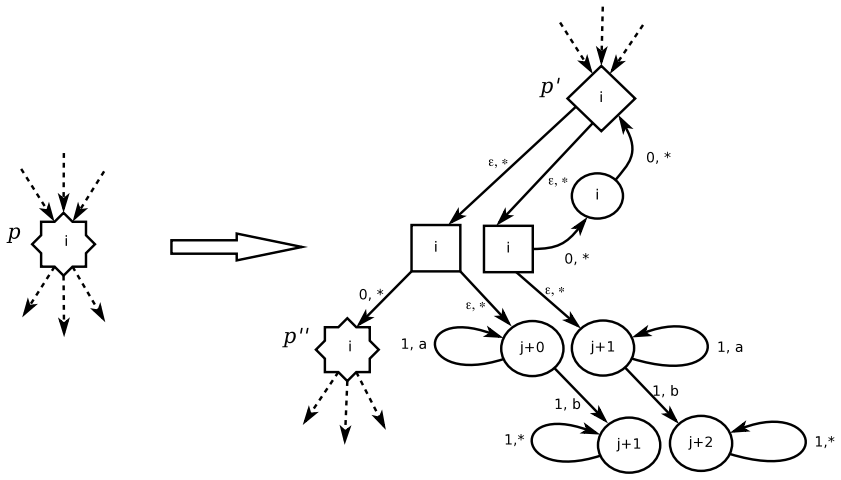


Figure 4.2: The gadget to replace p in the construction of $\exp A$. The state p'' is existential iff p is existential, $i = \operatorname{rank} p$, and j is the least even number greater or equal to i.

automaton $\exp A$ is a bit more tricky. This time, we have to change the whole structure of the automaton. Instead of adding one gadget, we replace each state of A by a different gadget.

The gadget for a state p is shown on Fig. 4.2. By replacing p with the gadget we mean that all the transitions ending in p should now end in p' and all the transitions starting in p should start in p''. Note that the state p'' is the place where the original transition is chosen, so p'' should be existential iff p is existential. It is not difficult to see that $\exp A$ recognizes exactly $\exp L(A)$.

Now we are ready to prove the promised lower bound.

Theorem 20. *The Wadge hierarchy restricted to weakly recognizable tree languages has the height of at least* ε_0.

Proof. The class of weakly recognizable conciliatory languages is closed by sum, multiplication by ω, and pseudoexponentiation. By iterating finitely many times sum and multiplication by ω we obtain the closure by multiplication by ordinals of the form $\omega^n k_n + \ldots + \omega k_1 + k_0$, i.e., all ordinals less then ω^ω.

In other words, we can find a weakly recognizable language of any conciliatory degree from the closure of $\{1\}$ by ordinal sum, multiplication by ordinals $< \omega^\omega$ and $\exp'$. It is easy to see that the order type of this set is not changed if we replace $\exp'$ with $\alpha \mapsto \omega_1^\alpha$. This in turn is isomorphic with the closure of $\{1\}$ by ordinal sum, multiplication by ordinals $< \omega^\omega$, and exponentiation with the base ω^ω. This last set is obviously ε_0, the least fixed point of the exponentiation with the base ω. The result follows by Corollary 10. □

Our intuition tells us the bound is tight, but we have no evidence for that. Hence, we end the chapter – and the whole thesis – with a conjecture.

Let $\alpha < \varepsilon_0$. It has a unique presentation in the base ω^ω. Let $\alpha = (\omega^\omega)^{\beta_n}\alpha_n + \ldots + (\omega^\omega)^{\beta_1}\alpha_1 + (\omega^\omega)^{\beta_0}\alpha_0$, with $\beta_0 < \beta_1 < \ldots < \beta_n < \alpha$, and $0 < \alpha_i < \omega^\omega$ for $i = 0, 1, \ldots, n$. Define $j_{\mathrm{Weak}}(0) = 0$ and for $\alpha > 0$ inductively

$$j_{\mathrm{Weak}}(\alpha) = \omega_1^{j_{\mathrm{Weak}}(\beta_n)}\alpha_n + \ldots + \omega_1^{j_{\mathrm{Weak}}(\beta_1)}\alpha_1 + \omega_1^{j_{\mathrm{Weak}}(\beta_0)}\alpha_0\,.$$

Conjecture 2. *The height of the Wadge hierarchy of the weakly recognizable tree languages is ε_0 and the Wagner function is given by j_{Weak}.*

Bibliography

[1] A. Arnold. The μ-calculus alternation-depth hierarchy is strict on binary trees. RAIRO-Theoretical Informatics and Applications **33** (1999) 329–339.

[2] A. Arnold, D. Niwiński. Continuous separation of game languages. Manuscript, submitted, 2006.

[3] J. C. Bradfield. The modal mu-calculus alternation hierarchy is strict. Theoret. Comput. Sci. **195** (1998) 133–153.

[4] A. Browne, E. M. Clarke, S. Jha, D. E. Long, W. Marrero. An improved algorithm for the evaluation of fixpoint expressions. Theoret. Comput. Sci. **178** (1997) 237–255.

[5] J. Duparc. Wadge hierarchy and Veblen hierarchy. Part I: Borel sets of finite rank. The Journal of Symbolic Logic **66** (2001).

[6] J. Duparc. A hierarchy of deterministic context-free ω-languages. Theoret. Comput. Sci. **290** (2003) 1253–1300.

[7] J. Duparc, F. Murlak. On the topological complexity of weakly recognizable tree languages. Proc. FCT 2007, LNCS 4639 (2007) 261-273.

[8] E. A. Emerson, C. S. Jutla The complexity of tree automata and logics of programs. *Proc. FoCS '88*, IEEE Computer Society Press 1988, 328–337.

[9] E. A. Emerson, C. S. Jutla. Tree automata, mu-calculus and determinacy. *Proc. FoCS '91*, IEEE Computer Society Press 1991, 368–377.

[10] O. Finkel. Wadge Hierarchy of Omega Context Free Languages. Theoret. Comput. Sci. **269** (2001) 283–315.

[11] M. Jurdziński, J. Vöge. A discrete strategy improvement algorithm for solving parity games. *Proc. CAV '00*, LNCS 1855 (2000) 202–215.

[12] A. S. Kechris. *Classical Descriptive Set Theory.* Graduate Texts in Mathematics, Vol. 156, Springer-Verlag 1995.

[13] O. Kupferman, S. Safra, M. Vardi. Relating Word and Tree Automata. *Proc. LICS '96*, 322–332.

[14] L. H. Landweber. Decision problems for ω-automata. Math. Systems Theory **3** (1969) 376–384.

[15] G. Lenzi. A hierarchy theorem for the mu-calculus. *Proc. ICALP '96*, LNCS 1099 (1996) 87–109.

[16] D. A. Martin. Borel determinacy. Ann. Math. **102** (1975) 363–371.

[17] A. W. Mostowski. Hierarchies of weak automata and weak monadic formulas. Theoret. Comput. Sci. **83** (1991) 323-335.

[18] A. W. Mostowski. Games with forbidden positions. Technical Report 78, Instytut Matematyki, University of Gdansk, 1991.

[19] D. E. Muller, A. Saoudi, P. E. Schupp. Alternating automata. The weak monadic theory of the tree, and its complexity. *Proc. ICALP '86*, LNCS 226 (1986) 275–283.

[20] D. E. Muller, P. E. Schupp. Alternating automata on infinite objects, determinacy and Rabin's theorem. *Proc. Automata on Infinite Words 1984*, LNCS 192 (1985) 100–107.

[21] F. Murlak. On deciding topological classes of deterministic tree languages. *Proc. CSL '05*, LNCS 3634 (2005) 428–441.

[22] F. Murlak. The Wadge hierarchy of deterministic tree languages. *Proc. ICALP '06*, Part II, LNCS 4052 (2006) 408-419.

[23] D. Niwiński. An example of non-Borel set of infinite trees recognizable by a Rabin automaton. Manuscript, Warsaw Univeristy, 1985 (in polish).

[24] D. Niwiński. On fixed point clones. *Proc. ICALP '86*, LNCS 226 (1986) 464–473.

[25] D. Niwiński, I. Walukiewicz. Relating hierarchies of word and tree automata. *Proc. STACS '98*, LNCS 1373 (1998) 320–331.

[26] D. Niwiński, I. Walukiewicz. A gap property of deterministic tree languages. Theoret. Comput. Sci. **303** (2003) 215–231.

[27] D. Niwiński, I. Walukiewicz. Deciding nondeterministic hierarchy of deterministic tree automata. *Proc. WoLLiC '04*, Electronic Notes in Theoret. Comp. Sci. 2005, 195–208.

[28] M. Otto. Eliminating recursion in μ-calculus. *Proc. STACS'99*, LNCS 1563 (1999) 531–540.

[29] D. Perrin, J.-E. Pin. *Infinite Words. Automata, Semigroups, Logic and Games.* Pure and Applied Mathematics Vol. 141, Elsevier 2004.

[30] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. Trans. Amer. Soc. **141** (1969) 1–35.

[31] M. O. Rabin. Weakly definable relations and special automata. *Mathematical Logic and Foundations of Set Theory*, North-Holland 1970, 1–70.

[32] H. Seidl. Fast and simple nested fixpoints. Information Processing Letters **59** (1996) 303–308.

[33] V. Selivanov. Wadge Degrees of ω-languages of deterministic Turing machines. Theoret. Informatics Appl. **37** (2003) 67-83.

[34] J. Skurczyński. The Borel hierarchy is infinite in the class of regular sets of trees. Theoret. Comput. Sci. **112** (1993) 413–418.

[35] W. Thomas. A hierarchy of sets of infinite trees. *Proc. Theoretical Computer Science*, LNCS 145 (1982) 335–342.

[36] W. Thomas. Languages, automata, and logic. Handbook of Formal Languages, Vol. 3, Springer-Verlag 1997, 389–455.

[37] T. F. Urbański. On deciding if deterministic Rabin language is in Büchi class. *Proc. ICALP '00*, LNCS 1853 (2000) 663–674.

[38] W. W. Wadge. Degrees of complexity of subsets of the Baire space. Notice Amer. Math. Soc. (1972) A-714.

[39] W. W. Wadge. Reducibility and determinateness on the Baire space. Ph. D. Thesis, Berkeley, 1984.

[40] K. Wagner. Eine topologische Charakterisierung einiger Klassen regulärer Folgenmengen. J. Inf. Process. Cybern. EIK **13** (1977) 473–487.

[41] K. Wagner. On ω-regular sets. Inform. and Control **43** (1979) 123–177.

Index

B^-, 73
C_α, 47
D_α, 47
E_α, 47
$L \cdot \omega^1$, 89
$L + M$, 87
$L \cdot \alpha$, 98
$L^\omega(A)$, 100
T_Σ, 18
X^*, 17
X^ω, 17
$\bot$, 19
$\exp L$, 99
$\wedge$, 46
$\vee$, 45
$\sup^+_{n<\omega} L_n$, 88
$\sup^-_{n<\omega} L_n$, 88
ω, 17
$\oplus$, 47
$\xrightarrow{(\iota,\kappa)}$, 46
j-loop, 21
$t.v$, 18
$\mathcal{C}$-complete, 26
$\mathcal{C}$-hard, 26

$\lim t_n$, 18

algorithm, 82
alternating chain, 21, 56
alternative, 45
analytical set, 26
automata game, 42
automaton on infinite trees
  alternating, 23
    computation tree, 23
  branching, 49
  canonical, 47
  complex, 49
  deterministic
    partial run, 30
  non-branching, 49
  nondeterministic, 19
    run, 19
  simple, 49
  weak, 24
automaton on infinite words
  deterministic, 19
  nondeterministic, 19
    run, 19

Borel hierarchy, 26
  of deterministic languages, 30, 37
Borel rank, 30
Borel set, 26

Cantor discontinuum, 26
closure properties, 71
component
  head, 46
  leftmost, 46
  tail, 46
concatenation of

tree languages, 18
trees, 18
word languages, 18
words, 17
conciliatory degree, 98
conciliatory hierarchy, 96
conciliatory Wadge game, 96
continuous reduction, 26, 39
critical path, 54

domination, 60
doppelgänger, 58
Duplicator, 41, 42

embedding, 64

flower, 21
canonical, 46
nontrivial, 46

game languages, 28
Gap Theorem, 30, 32

index, 20
dual, 20
index hierarchy, 20
alternating, 23
of deterministic languages, 23
deterministic, 21
nondeterministic, 21
of deterministic languages, 22
weak, 25
of deterministic languages, 33, 37
index problem
deterministic, 21
nondeterministic, 21, 22
weak, 25

leaf, 18

lifting operation, 79
loop, 21
j-loop, 21
accepting, 21
rejecting, 21

Mostowski–Rabin index, *see* index
multiplication, 89, 98

negative transition, 79
node, 18
non self dual set, 86

parallel composition, 46
parity game, 22
positional determinacy, 23
weak, 24
pattern, 31
Polish space, 26
positive transition, 79
projective hierarchy, 26
pseudoexponentiation, 99

replication, 31, 46
Replication Lemma, 34

segment, 30
self dual set, 86
sequential composition, 47
Spoiler, 41, 42
state
all-accepting, 19
all-rejecting, 19
productive, 19
replicated, 31
substitution, 18
Substitution Lemma, 44
sum, 87

token, 42

critical, 54
tree, 18
Σ-labeled, 18
n-ary, 18

undressing, 96

Wadge degree, 87
Wadge game, 41
Wadge hierarchy, 39
Wadge ordering, 39
Wadge reducibility, 39
Wadge reduction, *see* continuous reduction
Wagner hierarchy, 52