Logical Methods in Computer Science
Vol. 4 (4:15) 2008, pp. 1–44
www.lmcs-online.org

Submitted Nov. 2, 2006
Published Dec. 23, 2008

THE WADGE HIERARCHY OF DETERMINISTIC TREE LANGUAGES *

FILIP MURLAK

Institute of Informatics, University of Warsaw, ul. Banacha 2, 02–097 Warszawa, Poland
e-mail address: fmurlak@mimuw.edu.pl

Abstract. We provide a complete description of the Wadge hierarchy for deterministically recognisable sets of infinite trees. In particular we give an elementary procedure to decide if one deterministic tree language is continuously reducible to another. This extends Wagner's results on the hierarchy of ω-regular languages of words to the case of trees.

1. Introduction

Two measures of complexity of recognisable languages of infinite words or trees have been considered in literature: the index hierarchy, which reflects the combinatorial complexity of the recognising automaton and is closely related to μ-calculus, and the Wadge hierarchy, which is the refinement of the Borel/projective hierarchy that gives the deepest insight into the topological complexity of languages. Klaus Wagner was the first to discover remarkable relations between the two hierarchies for finite-state recognisable (ω-regular) sets of infinite words [14]. Subsequently, decision procedures determining an ω-regular language's position in both hierarchies were given [4, 7, 15].

For tree automata the index problem is only solved when the input is a deterministic automaton [9, 13]. As for topological complexity of recognisable tree languages, it goes much higher than that of ω-regular languages, which are all Δ^0_3. Indeed, co-Büchi automata over trees may recognise Π^1_1-complete languages [8], and Skurczyński [12] proved that there are even weakly recognisable tree languages in every finite level of the Borel hierarchy. This may suggest that in the tree case the topological and combinatorial complexities diverge. On the other hand, the investigations of the Borel/projective hierarchy of deterministic languages [5, 8] reveal some interesting connections with the index hierarchy.

Wagner's results [14, 15], giving rise to what is now called the Wagner hierarchy (see [10]), inspire the search for a complete picture of the two hierarchies and the relations between them for recognisable tree languages. In this paper we concentrate on the Wadge hierarchy of deterministic tree languages: we give a full description of the Wadge-equivalence classes forming the hierarchy, together with a procedure calculating the equivalence class

1998 ACM Subject Classification: F.4.3, F.4.1, F.1.1, F.1.3.

Key words and phrases: Wadge hierarchy, deterministic automata, infinite trees, decidability.

* An extended abstract of this paper was presented at ICALP'06 in Venice, Italy.
Supported by KBN Grant 4 T11C 042 25.

DOI:10.2168/LMCS-4 (4:15) 2008

of a given deterministic language. In particular, we show that the hierarchy has the height $\omega^{\omega\cdot 3}+3$, which should be compared with ω^ω for regular ω-languages [15], ω^{ω^2} for deterministic context-free ω-languages [1], $(\omega_1^{CK})^\omega$ for ω-languages recognised by deterministic Turing machines [11], or an unknown ordinal $\xi > (\omega_1^{CK})^\omega$ for ω-languages recognised by nondeterministic Turing machines, and the same ordinal ξ for nondeterministic context-free languages [2].

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 collection of canonical automata representing the Wadge degrees of all deterministic tree languages. Then we provide a procedure calculating the canonical form of a given deterministic automaton, which 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).

2. Automata

We use the symbol ω to denote the set of natural numbers $\{0,1,2,\dots\}$. For an alphabet Σ, Σ^* is the set of finite words over Σ and Σ^ω is the set of infinite words over Σ. The *concatenation* of words $u\in\Sigma^*$ and $v\in\Sigma^*\cup\Sigma^\omega$ will be denoted by uv, and the empty word by ε. The concatenation is naturally generalised for infinite sequences of finite words $v_1v_2v_3\dots$. The concatenation of sets $A\subseteq\Sigma^*$, $B\subseteq\Sigma^*\cup\Sigma^\omega$ 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-*labelled tree* (or a tree over Σ) is a function $t : \operatorname{dom} t \to \Sigma$ such that $\operatorname{dom} t$ is a tree. For $v\in\operatorname{dom} t$ we define $t.v$ as a subtree of t rooted in v, i. e., $\operatorname{dom}(t.v)=\{u : vu\in\operatorname{dom} t\}$, $t.v(u)=t(vu)$.

A *full* n*-ary* Σ*-labeled* tree is a function $t:\{0,1,\dots,n-1\}^*\to\Sigma$. The symbol T_Σ will denote the set of full binary trees over Σ. From now on, if not stated otherwise, a "tree" will mean a full binary tree over some alphabet.

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 $A=\langle\Sigma,Q,\delta,q_0,\operatorname{rank}\rangle$, where Σ is a finite input alphabet, Q is a finite set of states, $\delta\subseteq Q\times\Sigma\times Q$ is the transition relation, and $q_0\in Q$ is the initial state. The meaning of the function $\operatorname{rank}: Q\to\omega$ 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 recognised 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 recognised by a deterministic automaton.

A *nondeterministic automaton on trees* is a tuple $A=\langle\Sigma,Q,\delta,q_0,\operatorname{rank}\rangle$, 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*

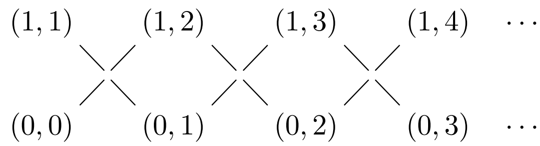

Figure 1: The Mostowski–Rabin index hierarchy.

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 $q_0 \xrightarrow{\sigma_0,d_0} q_1 \xrightarrow{\sigma_1,d_1} \dots \xrightarrow{\sigma_n,d_n} q$ 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 $i = 0, 1, \dots, n$.

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 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 $\cap$ co-NP and 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 it is particularly important.

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 recognised by (ι, κ)-automata (see Fig. 1).

The fundamental question about the hierarchy is the strictness, i. e., the existence of languages recognised 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 [15]: 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 [6]. In fact, the languages showing

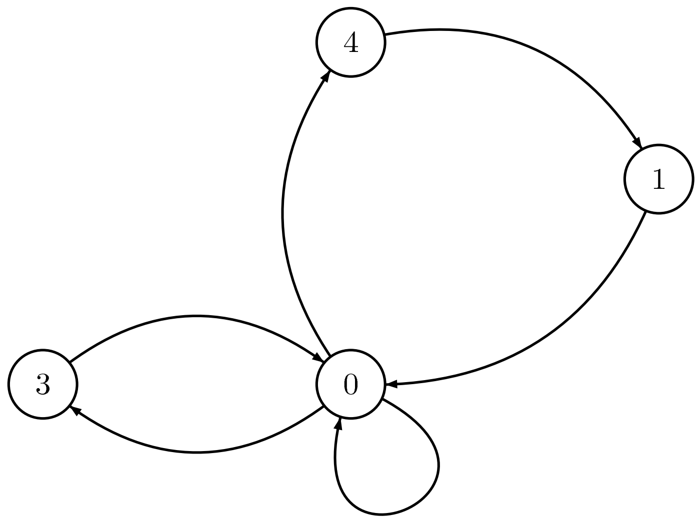

Figure 2: A $(0, 2)$-flower.

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 [7].

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

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

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 (see Fig. 2). [1]

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

Theorem 2.1 (Niwiński, Walukiewicz [7])**.** *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 generalised 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 word automaton recognising $\mathrm{Paths}(L(A))$. 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

[1]This is a slight modification of the original definition from [7].

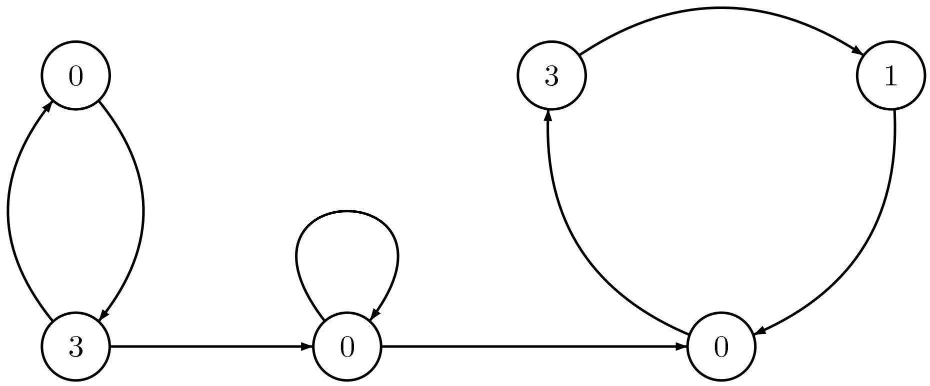

Figure 3: A weak $(1, 3)$-flower.

$(p, (\sigma, d), q) \in \delta' \iff (p, \sigma, d, q) \in \delta$. Conversely, given a deterministic word automaton recognising $\mathrm{Paths}(L(A))$, one may interpret it as a tree automaton, obtaining thus a deterministic automaton recognising $L(A)$. Hence, applying Theorem 2.1 one gets the following result.

Proposition 2.2. *For a deterministic tree automaton A the language $L(A)$ is recognised by a deterministic (ι, κ)-automaton iff A does not contain a $\overline{(\iota, \kappa)}$-flower.* □

In [5] it is shown how to compute the *weak deterministic index* of a given deterministic language. An automaton is called *weak* if the ranks may only decrease during the run, i. e., if $p \longrightarrow q$, then $\mathrm{rank}(p) \geq \mathrm{rank}(q)$. The weak deterministic index problem is to compute a weak deterministic automaton with minimal index recognising a given language. The procedure in [5] is again based on the method of difficult patterns used in Theorem 2.1 and Proposition 2.2. 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 $\lambda_\iota, \lambda_{\iota+1} \dots, \lambda_\kappa$ such that λ_{j+1} is reachable from λ_j, and λ_j is accepting iff j is even (see Fig. 3).

Proposition 2.3 ([5])**.** *A deterministic automaton A is equivalent to a weak deterministic (ι, κ)-automaton iff it does not contain a weak $\overline{(\iota, \kappa)}$-flower.* □

For a deterministic language one may also 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 consisting of trees whose leftmost paths are in a regular word language M. It can be recognised 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 [9]. Decidability of the general index problem for nondeterministic automata is one of the most important open questions in the field.

3. Topology

We start with a short recollection of elementary notions of descriptive set theory. For further information see [3].

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 unions 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 not go beyond 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 omit 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.*

In 2002 Niwiński and Walukiewicz discovered a surprising dichotomy in the topological complexity of deterministic tree languages: a deterministic tree language has either a very low Borel rank or it is not Borel at all (see Fig. 4). 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 3.1 (Niwiński, Walukiewicz [8])**.** *For a deterministic automaton A, $L(A)$ is on the level Π^0_3 of the Borel hierarchy iff A does not admit split; otherwise $L(A)$ is Π^1_1-complete (hence non-Borel).* □

$$
\begin{tikzcd}[column sep=small]
\Sigma^0_0 \arrow[dr, no head] & & \Sigma^0_1 \arrow[dr, no head] & & \Sigma^0_2 \arrow[dr, no head] & & & \\
& \Delta^0_1 \arrow[ur, no head] \arrow[dr, no head] & & \Delta^0_2 \arrow[ur, no head] \arrow[dr, no head] & & \Delta^0_3 = \Sigma^0_3 \arrow[dr, no head] & & \\
\Pi^0_0 \arrow[ur, no head] & & \Pi^0_1 \arrow[ur, no head] & & \Pi^0_2 \arrow[ur, no head] & & \Pi^0_3 & \big| \ \Pi^1_1 - \text{complete}
\end{tikzcd}
$$

Figure 4: The Borel hierarchy for deterministic tree languages.

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 a 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. 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).

In [5] a similar characterisation was obtained for the remaining classes from the above hierarchy. Before we formulate these result, let us introduce one of the most important technical notions of this study. A state p is *replicated* by a loop $q_1 \xrightarrow{\sigma,d_0} q_2 \longrightarrow \ldots \longrightarrow q_1$ if there exist a path $q_1 \xrightarrow{\sigma,d_1} q_2' \longrightarrow \ldots \longrightarrow p$ 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 occur in the proposition below.

Theorem 3.2 (Murlak [5])**.** *Let A be a deterministic automaton.*
(1) $L(A) \in \Sigma^0_1$ *iff A does not contain a weak* $(0,1)$*-flower.*
(2) $L(A) \in \Pi^0_1$ *iff A does not contain a weak* $(1,2)$*-flower.*
(3) $L(A) \in \Sigma^0_2$ *iff A does not contain a* $(1,2)$*-flower nor a weak* $(1,2)$*-flower replicated by an accepting loop.*
(4) $L(A) \in \Pi^0_2$ *iff A does not contain a* $(0,1)$*-flower.*
(5) $L(A) \in \Sigma^0_3$ *iff A does not contain a* $(0,1)$*-flower replicated by an accepting loop.* □

4. The Main Result

The notion of continuous reduction defined in Sect. 3 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 $A \leq_W B$. Similarly we write $A <_W B$ 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 recognisability. 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 us a lower bound for the case of deterministic tree languages, and this is essentially everything we can conclude from the word case.

The starting point of this study is the Wadge reducibility problem.

PROBLEM:	Wadge reducibility
INPUT:	Deterministic tree automata A and B
QUESTION:	$L(A) \leq_W L(B)$?

An analogous problem for word automata can be solved fairly easy by constructing a tree automaton recognising Duplicator's winning strategies (to be defined in the next section). This method however does not carry over to trees. One might still try to solve the Wadge reducibility problem directly by comparing carefully the structure of two given automata, but we have chosen a different approach. We will provide a family of *canonical* deterministic tree automata $\mathcal{A} = \{A_i \colon i \in I\}$ such that

(1) given $i, j \in I$, it is decidable if $L(A_i) \leq_W L(A_j)$,
(2) for each deterministic tree automaton there exists exactly one $i \in I$ such that $L(A) \equiv_W L(A_i)$, and this i can be computed effectively for a given A.

The decidability of the Wadge reducibility problem follows easily from the existence of such a family: given two deterministic automata A and B, we compute i and j such that $L(A) \equiv_W L(A_i)$ and $L(B) \equiv_W L(A_j)$, and check if $L(A_i) \leq_W L(A_j)$.

More precisely, we prove the following theorem.

Theorem 4.1. *There exists a family of deterministic tree automata*

$$\mathcal{C}' = \{C_\alpha \colon \alpha \in I\} \cup \{D_\alpha, E_\alpha \colon \alpha \in J\}$$

with $I = \{\alpha \colon 0 < \alpha \leq \omega^{\omega\cdot 3} + 2\}$, $J = \{n \colon 0 < n < \omega\} \cup \{\omega^{\omega\cdot 2}\alpha_2 + \omega^\omega \alpha_1 + n \colon \alpha_2 < \omega^\omega,\ 0 < \alpha_1 < \omega^\omega,\ n < \omega\}$ *such that*

(1) *for* $0 < \alpha < \beta \leq \omega^{\omega\cdot 3} + 2$, *whenever the respective automata are defined, we have*

$$\begin{tikzcd}
L(C_\alpha) \arrow[dr] & & L(C_\beta) \arrow[dr] & \\
& L(E_\alpha) \arrow[ur] \arrow[dr] & & L(E_\beta) \\
L(D_\alpha) \arrow[ur] & & L(D_\beta) \arrow[ur] &
\end{tikzcd}$$

where $\rightarrow$ *means* $<_W$, *and* $L(C_\alpha)$ *and* $L(D_\alpha)$ *are incomparable,*
(2) *for each deterministic tree automaton* A *there exists exactly one automaton* $A' \in \mathcal{C}'$ *such that* $L(A') \equiv_W L(A)$ *and it is computable, i.e., there exists an algorithm computing for a given* A *a pair* $(\Xi, \alpha) \in \{C\} \times I \cup \{D, E\} \times J$ *such that* $L(A) \equiv_W L(\Xi_\alpha)$.

The family $\mathcal{C}'$ satisfies the conditions postulated for the family of canonical automata $\mathcal{A}$: for ordinals presented as arithmetical expressions over ω in Cantor normal form the ordinal order is decidable, so we can take $\{C\} \times I \cup \{D, E\} \times J$ as the indexing set of $\mathcal{A}$.

Observe that the pair (Ξ, α) computed for a given A can be seen as a name of the $\equiv_W$-class of $L(A)$. Hence, the set $\{C\} \times I \cup \{D, E\} \times J$ together with the order defined in the statement of theorem provides a complete effective description of the Wadge hierarchy restricted to deterministic tree languages. One thing that follows is that the height of the hierarchy is $\omega^{\omega\cdot 3}+3$.

The remaining part of the paper is in fact a single long proof. We start by reformulating the classical criterion of reducibility via Wadge games in terms of automata (Sect. 5). This will be the main tool of the whole argument. Then we define four ways of composing automata: sequential composition $\oplus$, replication $\to$, parallel composition $\wedge$, and alternative $\vee$ (Sect. 6). Using the first three operations we construct the canonical automata, all but top three ones (Sect. 7). Next, to rehearse our proof method, we reformulate and prove Wagner's results in terms of canonical automata (Sect. 8). Finally, after some preparatory remarks (Sect. 9), we prove the first part of Theorem 4.1, modulo three missing canonical automata.

Next, we need to show that our family our family contains all deterministic tree automata up to Wadge equivalence of the recognised languages. Once again we turn to the methodology of patterns used in Sect. 2 and Sect. 3. We introduce a fundamental notion of admittance, which formalises what it means to contain an automaton as a pattern (Sect. 11). Then we generalise $\to$ to (ι,κ)-replication $\xrightarrow{(\iota,\kappa)}$ in order to define the remaining three canonical automata, and rephrase the results on the Borel hierarchy and the Wagner hierarchy in terms of admittance of canonical automata (Sect. 12). Basing on these results, we show that the family of canonical automata is closed by the composition operations (Sect. 13), and prove the Completeness Theorem asserting that (up to Wadge equivalence) each deterministic automaton may be obtained as an iterated composition of C_1 and D_1 (Sect. 14). As a consequence, each deterministic automaton is equivalent to a canonical one. From the proof of the Completeness Theorem we extract an algorithm calculating the equivalent canonical automata, which concludes the proof of Theorem 4.1.

5. 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 [10] for word version). By *the nth level* of a tree we understand the set of nodes $\{0,1\}^{n-1}$. The 1st level consists of the root, the 2nd level consists of all the children of the root, etc. 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 some levels to t_S and then Duplicator can either add some levels to t_D or skip a round (not forever). The result of the play is a pair of full binary trees. Duplicator wins the play 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 5.1. *Duplicator has a winning strategy in $G_W(L,M)$ iff $L\leq_W M$.*

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, strictly increasing) such that the level k of $\varphi(t)$ depends only on the levels $1,2,\ldots,n_k$ of t. Then the strategy for Duplicator is the following: if the number of the round is n_k, play the kth level of t_D according to φ; otherwise skip. $\square$

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 recognisable languages, let us redefine the games in terms of automata. Let A, B be deterministic tree automata. 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 actions described below.

Fire a transition: for a token placed in a state q choose a transition $q \xrightarrow{\sigma} q_1, q_2$, 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 $\perp$.

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 realising 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 5.2. *For all deterministic tree automata A and B,*

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

Proof. First consider a modified Wadge game $G'_W(L, M)$, where players are allowed to build their trees in an arbitrary way provided that the nodes played always form one connected tree, and in every round Spoiler must provide both children for all the nodes that were leaves in the previous round. It is very easy to see that Duplicator has a winning strategy in $G'_W(L, M)$ iff he has a winning strategy in $G_W(L, M)$.

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 $G(A, B)$.

Let us move to the converse implication. Now, 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 5.3. *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 recognised by an automaton. One language may be recognised 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 recognised language, but not its $\equiv_W$-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 recognised 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 5.4. *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 recognising 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

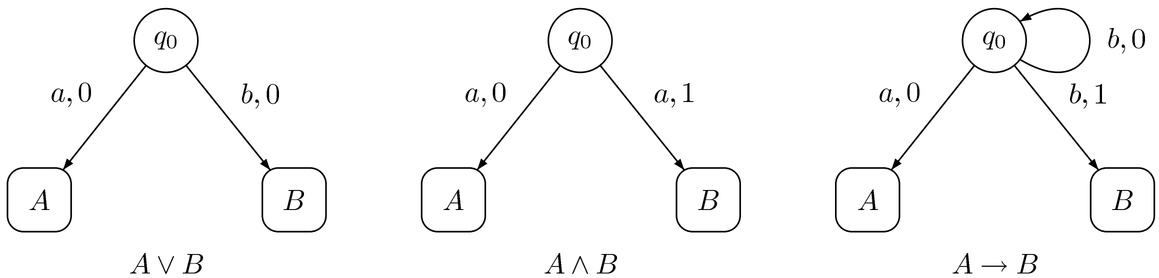

Figure 5: The alternative $A \vee B$, the parallel composition $A \wedge B$, and the replication $A \to B$ (transitions to $\bot$ and $\top$ are omitted).

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 5.5 (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. $\square$

6. Operations

It this section we introduce four operations that will be used to construct canonical automata representing 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* $A \vee B$ (see Fig. 5) 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

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

(only if $L(A) = L(B) = \emptyset$ put $q_0 \xrightarrow{\sigma} \bot, \bot$). By Lemma 5.5, $\equiv$ is a congruence with respect to $\vee$. Furthermore, $\vee$ is associative and commutative up to $\equiv$. Multiple alternatives are

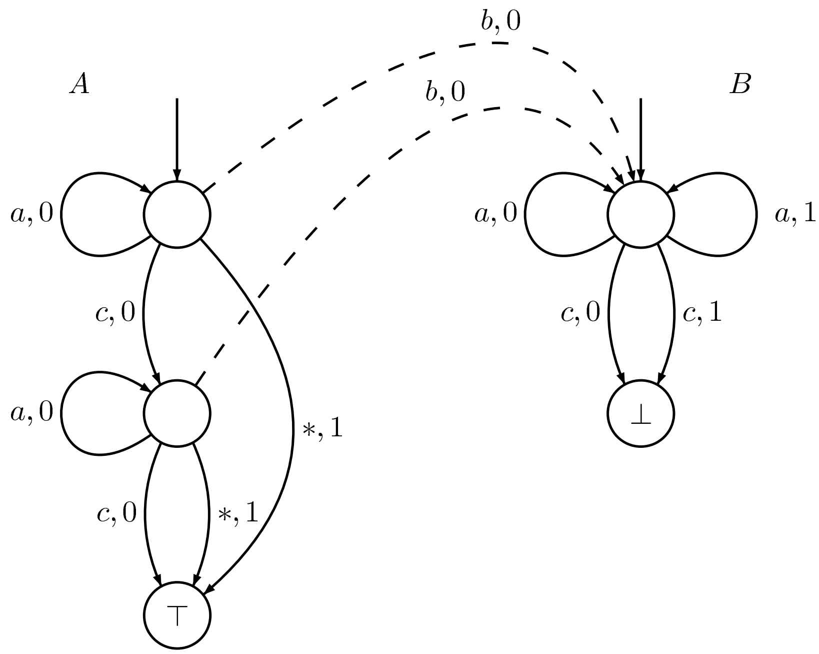

Figure 6: The sequential composition $A \oplus B$.

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

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

(only if $L(A) = \emptyset$ or $L(B) = \emptyset$, put $q_0 \xrightarrow{\sigma} \perp, \perp$). 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 $\underbrace{A \wedge \ldots \wedge A}_{n}$.

To obtain the *replication* $A \to B$, extend the alphabet again by $\{a, b\}$, set $\operatorname{rank}(q_0) = 1$, and add and transitions

$$q_0 \xrightarrow{a} q_0^A, \top\,, \quad q_0 \xrightarrow{b} q_0, q_0^B\,, \quad \text{and} \;\; q_0 \xrightarrow{\sigma} \perp, \perp \;\text{ for }\; \sigma \notin \{a, b\}\,.$$

Like for two previous operations, $\equiv$ is a congruence with respect to $\to$.

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. A state p is *leftmost* if no path connecting the initial state with p uses a right transition. In other words, leftmost states are those which can only occur in the leftmost path of a run. Note that an automaton may have no leftmost states. Furthermore, a leftmost state cannot be reachable from a non-leftmost state. In particular, if an automaton has any leftmost states at all, the initial state has to be leftmost. For deterministic tree automata A and B over Σ, the *sequential composition* $A \oplus B$ (see Fig. 6) is an automaton with the input alphabet

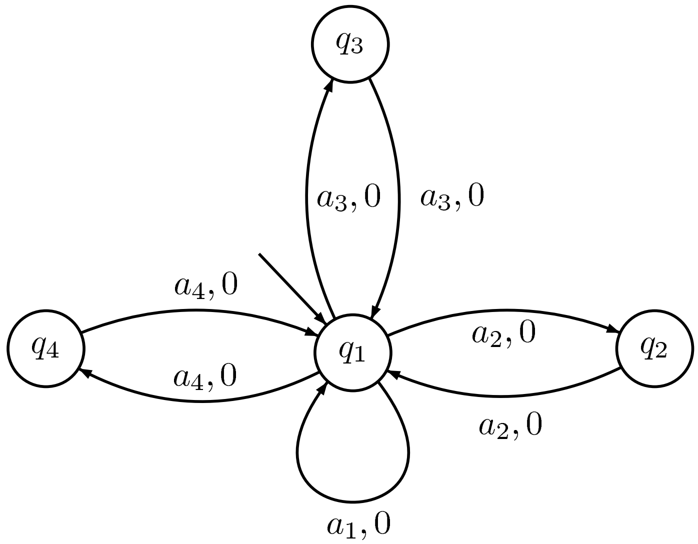

Figure 7: The canonical $(1,4)$-flower $F_{(1,4)}$.

$\Sigma \cup \{b\}$, 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 each leftmost state p 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 $nA = \underbrace{A \oplus \ldots \oplus A}_{n}$.

Observe that if A has a leftmost state, then a state in $A \oplus B$ is leftmost iff it is a leftmost state of A or a leftmost state of B. It follows that 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 $\oplus, \wedge, \vee, \rightarrow$. For instance $A_1 \rightarrow A_2 \oplus A_3 \wedge A_4 \vee A_5 = A_1 \rightarrow (((A_2 \oplus A_3) \wedge A_4) \vee A_5)$. Nevertheless, we usually use parentheses to make the expressions easier to read.

Finally, let us define the basic building blocks, to which we will apply the operations defined above. The *canonical* (ι, κ)*-flower* $F_{(\iota,\kappa)}$ (see Fig. 7) 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 $\operatorname{rank}(q_i) = i$, 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 $\iota < \kappa$.

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 analysing games on automata, we will ignore the transitions to $\top$.

7. Canonical Automata

For convenience, in this section we put together the definitions of all canonical automata (save for three which will be defined much later) together with some very simple properties. More explanations and intuitions come along with the proofs in the next three sections.

For each $\alpha < \omega^{\omega\cdot 3}$ we define the *canonical automaton* C_α. The automata D_α and E_α will only be defined for $0 < \alpha < \omega$ and $\alpha = \omega^{\omega\cdot 2}\alpha_2 + \omega^\omega\alpha_1 + n$ with $0 < \alpha_1 < \omega^\omega$, $\alpha_2 < \omega^\omega$, $n < \omega$. All the defined automata have at least one leftmost state, 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. Let $C_\omega = C_1 \to C_3$ and $C_{\omega^{k+1}} = C_1 \to (C_1 \oplus C_{\omega^k})$ 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 $\omega^{\omega\cdot 2} \leq \alpha < \omega^{\omega\cdot 3}$. 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.,

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

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

Now, let us discuss briefly the anatomy and 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 automata $A_1, A_2, \ldots, A_n$ we have $A = A_1 \oplus A_2 \oplus \ldots \oplus A_n$, we call A_i *components* of A. If A_i are simple, they are called *simple components* of A, A_1 is the *head component*, and A_n is the *tail component*. *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*. The term *head loop* refers to any minimal-length loop around the initial state. If the head component of a canonical automaton is branching, then the automaton has only one head loop. Similarly, if the head component is C_1 or D_1.

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 each token moves along the leftmost path from the node it was born in, bubbling out new tokens to the right. Let us prove the following simple yet useful property of those paths.

Proposition 7.1. *If a run constructed by a player in charge of a canonical automaton is rejecting, one of the tokens has visited a rejecting path.*

Proof. 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 the automaton). Now, consider a rejecting run constructed during a play. It must contain a rejecting path π. The token created during the last right transition on π visits a suffix of π, which of course is a rejecting path. □

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*.

Since we can remove the tokens from $\top$ with no impact on the outcome of the game, we can assume that 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. The following fact relies on this convention.

Proposition 7.2. *If a player in charge of a canonical automaton produces infinitely may tokens, the resulting run is rejecting.*

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 loop a finite number of times in the head loop of A, thus producing only a finite number of new tokens. 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 A'''s 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. □

8. Without Branching

In this section we briefly reformulate Wagner's results on regular word languages [15] in terms of canonical automata. For the sake of completeness, we reprove them in our present framework.

The scenario 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 $\{\iota, \iota+1, \dots, \kappa\}$ that satisfy the parity condition, i. e., the highest number occurring infinitely often is even.

Lemma 8.1. *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. A 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 recognised 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. $\square$

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

$$\begin{tikzcd} \hat{C}_\alpha \arrow[dr] & & \hat{C}_\beta \arrow[dr] & \\ & \hat{E}_\alpha \arrow[ur] \arrow[dr] & & \hat{E}_\beta \\ \hat{D}_\alpha \arrow[ur] & & \hat{D}_\beta \arrow[ur] & \end{tikzcd}$$

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 8.1, $\hat{E}_\alpha \leq F_{(0,k)} = \hat{C}_\beta$. If we increase the ranks in each $F_{(0,l)}$ in $\hat{E}_\alpha$ by 2, we obtain an automaton with index at most $(1,k+1)$ recognising 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 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 8.1 (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 all states are leftmost. It is easy to see that $\equiv$ is a congruence with respect to $\oplus$ and $\wedge$. Both operations are associative up to $\equiv$.

Proposition 8.3. *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. For $A_1 \geq A_2$ it holds that $A_1 \vee A_2 \equiv A_1$. Indeed, $A_1 \vee A_2 \geq A_1$, as $A_1 \vee A_2$ contains a copy of A_1. For the converse inequality consider $G(A_1 \vee A_2, A_1)$. In the first move, Spoiler moves his initial token either to A_1 or to A_2. If Spoiler chooses A_1, Duplicator may simply mimic Spoiler's actions in his copy of A_1. If Spoiler chooses A_2, Duplicator wins by applying the strategy from $G(A_2, A_1)$, guaranteed by the inequality $A_1 \geq A_2$.

In the remaining case A_1 and A_2 are incomparable. But then $A_1 = \hat{C}_\alpha$, $A_2 = \hat{D}_\alpha$ for some $\alpha < \omega^\omega$ (or symmetrically). It is very easy to see that $\hat{C}_\alpha \vee \hat{D}_\alpha \equiv \hat{E}_\alpha$.

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

Using the property above, we easily reduce the general situation to one of the following cases: $\hat{C}_{\omega^k} \oplus \hat{C}_{\eta\omega^{k+1}}$, $\hat{C}_{\omega^k} \oplus \hat{D}_{\eta\omega^k}$, or $\hat{C}_{\omega^k} \oplus \hat{E}_{\eta\omega^k}$. 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 $\hat{C}_{\omega^k} \oplus \hat{C}_{\eta\omega^{k+1}} \equiv \hat{C}_{\eta\omega^{k+1}}$. Consider the game $G(\hat{C}_{\omega^k} \oplus \hat{C}_{\eta\omega^{k+1}}, \hat{C}_{\eta\omega^{k+1}})$. Let $\hat{C}_{\omega^l}$ be the head component of $\hat{C}_{\eta\omega^{k+1}}$. It holds that $l > k$. In order to win the game, while Spoiler stays inside $\hat{C}_{\omega^k}$, Duplicator should stay in $\hat{C}_{\omega^l}$ and use the strategy from $G(\hat{C}_{\omega^k}, \hat{C}_{\omega^l})$. When Spoiler enters $\hat{C}_{\eta\omega^{k+1}}$, Duplicator may simply copy his actions. The converse inequality is trivial.

$$\begin{array}{cccccccccccc}
\hat{C}_1 & & \hat{C}_2 & & \hat{C}_\omega & & \hat{C}_{\omega+1} & & \hat{C}_{\omega\cdot 2} & & \hat{C}_{\omega\cdot 2+1} & \\
 & \diagdown\quad\diagup & & \diagdown & & \diagdown\quad\diagup & & \diagdown & & \diagdown\quad\diagup & & \diagdown \\
 & \hat{E}_1 & & \hat{E}_2 \cdots & & \hat{E}_\omega & & \hat{E}_{\omega+1} \cdots & & \hat{E}_{\omega\cdot 2} & & \hat{E}_{\omega\cdot 2+1} \cdots \\
 & \diagup\quad\diagdown & & \diagup & & \diagup\quad\diagdown & & \diagup & & \diagup\quad\diagdown & & \diagup \\
\hat{D}_1 & & \hat{D}_2 & & \hat{D}_\omega & & \hat{D}_{\omega+1} & & \hat{D}_{\omega\cdot 2} & & \hat{D}_{\omega\cdot 2+1} &
\end{array}$$

Figure 8: An initial segment of the Wagner hierarchy

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

For $A_1 = \hat{D}_{\omega^k}$, simply dualise the claims and the proofs. For $A_1 = \hat{E}_{\omega^k}$, note that $\hat{E}_{\omega^k} \oplus A_2 \equiv \hat{C}_{\omega^k} \oplus A_2 \vee \hat{D}_{\omega^k} \oplus A_2$, and the equivalent canonical automaton can be obtained by previous cases. □

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

Theorem 8.4. *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. Without loss of generality we may assume that all states of A are reachable from the initial state. In such case, the DAG of SCCs is connected and has exactly one *root* component, the one containing the initial state of the automaton.

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 2.2, A is equivalent to a (ι, κ)-automaton and so, by Lemma 8.1, $A \leq F_{(\iota,\kappa)}$. 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 $A \equiv F_{(\iota,\kappa)} \oplus (B_1 \vee B_2 \vee \ldots \vee B_m)$. □

9. The Use of Replication

Branching automata are defined by iterating $\rightarrow$. The significance of $\rightarrow$ lies in the fact that closing the family of non-branching automata by this operation gives, up to Wadge

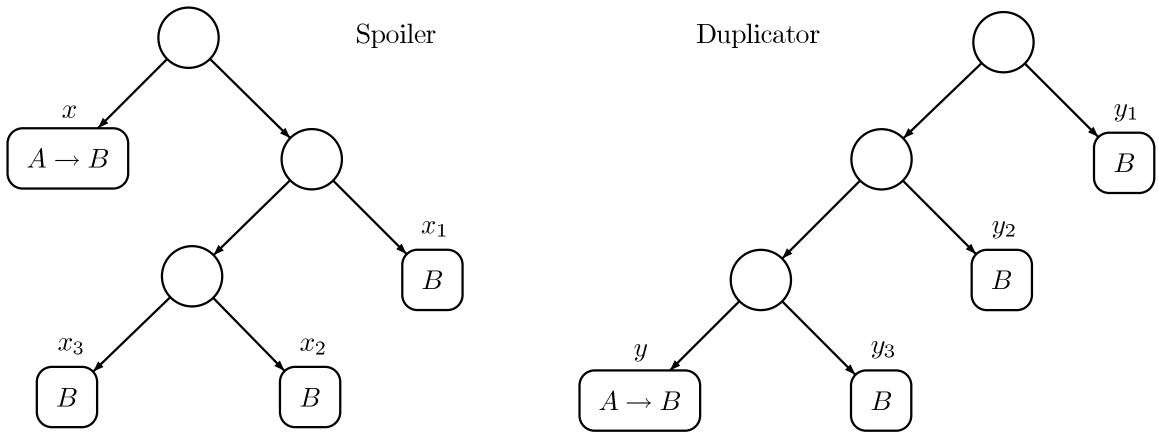

Figure 9: An initial part of the play in $G((A \to B) \wedge (B)^3, A \to B)$.

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, $\to$ is everything that deterministic tree automata have, which word automata have not. Let us see then what the use of the operation $\to$ is.

There are two kinds of simple branching automata. The first one is obtained by iterating $\to$ on C_3, and generalises 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 18), 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 9.1. *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 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 (see Fig. 9). 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 5.5, $(C_1 \to B) \wedge (B)^k \geq C_1 \wedge (B)^k$, and the claim follows. $\square$

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 5):

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

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 $\{\iota,\iota+1,\dots,\kappa\} \subseteq \omega$ 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 $y_1 \leq y_2$. For $m = 0,1$ and $n \geq m$ define an *alternating chain of type* (m,n), or (m,n)-chain, as a sequence $(x_m,y_m) < (x_{m+1},y_{m+1}) < \dots < (x_n,y_n)$, such that (x_i,y_i) is even iff i is even. 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 9.2. *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 8.1, $L(F_{(i,j)}) \equiv_W L_{(i,j)}$, 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 $L \times M = \{(x_1,y_1)(x_2,y_2)\dots : x_1x_2\dots \in L,\ y_1y_2\dots \in M\}$. 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 $\operatorname{rank}(i,j) = \kappa'$. 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 $\operatorname{rank}(\max_k i_k, \max_k j_k)$, where $(i_1,j_1),(i_2,j_2),\dots,(i_r,j_r)$ are the arguments for which the value 2 is taken.

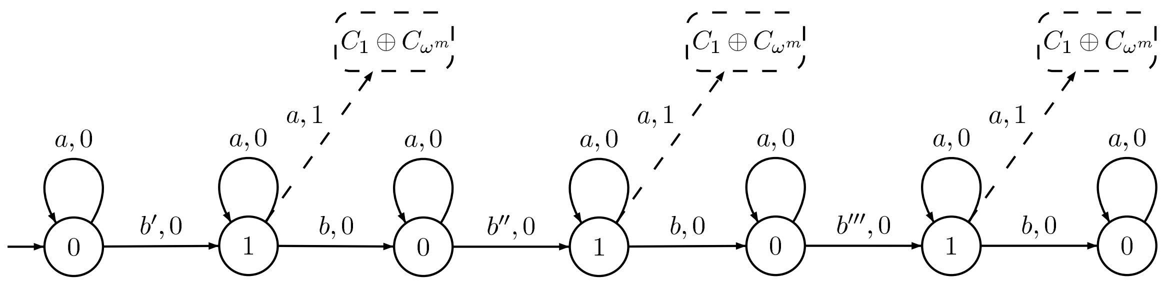

Figure 10: The weak flower $WF_{(0,6)}$ formed by the leftmost states of $C_1 \oplus C_{\omega^{m+1}\cdot 3}$.

Let us check that the automaton recognises $L_{(\iota_1,\kappa_1)} \times L_{(\iota_2,\kappa_2)}$. Take a word $w = (x_1, y_1)(x_2, y_2)\dots$. 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 $\forall_{(x',y')}\ f(x', y') = 2 \implies (x', y') \le (x, y)$. Since $(x, y) \le (x', y') \implies \mathrm{rank}(x, y) \le \mathrm{rank}(x', y')$, 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

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

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

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

The proof for weak flowers is entirely analogous. □

Lemma 9.3. *For all* $0 < k, l < \omega$ *and all* $m < \omega$

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

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 $G(C_1 \oplus C_{\omega^m k} \wedge C_1 \oplus C_{\omega^m l}, C_1 \oplus C_{\omega^m k} \oplus C_{\omega^m l})$. Observe that Duplicator's critical token will move along a copy of $WF_{(0,2k+2l)}$ formed by the leftmost states of consecutive copies of C_{ω^m} (see Fig. 10). 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 $WF_{(0,2k)} \wedge WF_{(0,2l)} \equiv WF_{(0,2k+2l)}$ (Lemma 9.2). 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 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. $\square$

Corollary 9.4. *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. Since $C_\omega = C_1 \to C_3 \equiv C_1 \to WF_{(0,2)}$, by Lemma 9.1 and Lemma 9.2 we get $C_\omega \geq (WF_{(0,2)})^m \equiv WF_{(0,2m)}$ and by the strictness of the hierarchy for word languages $C_\omega > WF_{(\iota,\kappa)}$. Similarly, using Lemma 9.1 and Lemma 9.3 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. $\square$

10. Automata in Order

Let us start examining the order on canonical automata with the following simple observation.

Lemma 10.1. *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 Proposition 7.2 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 Proposition 7.1, 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. $\square$

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 $\beta \geq \alpha$,
- $A = C_{\omega^m}$ and $B_1 = F_{(\iota,\kappa)}$ or $B_1 = F_{(\iota,\kappa)} \vee F_{\overline{(\iota,\kappa)}}$ for $\iota < \kappa$.

Lemma 10.2. *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 $G(A_1 \oplus \ldots \oplus A_n \oplus B, C)$, then he also has a strategy in which he removes all non-critical tokens before entering B. Similarly for Duplicator in $G(C, A_1 \oplus \ldots \oplus A_n \oplus B)$.*

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 10.1. 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 10.1, 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 stabilise 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 10.3. *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 10.4. *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 $E_\beta \leq D_\alpha$, $E_\beta \leq C_\alpha$.*

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 10.3 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.

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 9.4. For $k > 1$, using the induction hypothesis and Corollary 9.4, 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 10.1, $C_{\alpha_0} \leq C_{\omega^\omega}$. 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 8.2, $C_{\omega^\omega(\alpha_1+1)} \leq C_{\omega^\omega \beta_1}$ 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. 11) we need the strictness of the inequalities from the previous lemma.

Theorem 10.5. *Let $0 < \alpha \leq \beta < \omega^{\omega\cdot 3}$. Whenever the respective automata are defined, it holds that $D_\alpha \nleq C_\alpha$, $D_\alpha \ngeq C_\alpha$, $D_\alpha < E_\beta$, $C_\alpha < E_\beta$, and for $\alpha < \beta$, $C_\alpha < C_\beta$, $E_\alpha < D_\beta$, $E_\alpha < C_\beta$.*

Proof. By Lemma 10.4 it is enough to prove $C_\alpha < C_{\alpha+1}$, $D_\alpha < E_\alpha$, $C_\alpha < E_\alpha$, $D_\alpha \nleq C_\alpha$, $D_\alpha \ngeq C_\alpha$. 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 $\omega \leq \alpha < \omega^\omega$. Then $\alpha = \omega^k + \alpha'$ with $k \geq 1$, $\alpha' < \omega^{k+1}$. 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 10.3). Spoiler should remove all his non-critical tokens, move his critical token to any accepting loop in $C_{\alpha'+1}$. Let us check that such a loop is always reachable for the critical token.

Let $C_{\alpha'+1} = A \oplus B$ with B simple. If $B = C_1 \to B'$, Spoiler can move his critical token to C_1. If B is not of this form, then by definition of canonical automata, $C_{\alpha'+1} = C_{2n+1}$ or $C_{\alpha'+1} = D_{2n}$. Recall that $C_{2n+1} \equiv WF_{(0,2n)} \equiv WF_{(0,2n-1)} \oplus C_1$ and $D_{2n} \equiv WF_{(1,2n)} \equiv WF_{(1,2n-1)} \oplus C_1$ (see page 22). It follows that in any play on C_{2n+1} or D_{2n}, if one has a winning strategy, one also has a winning strategy never entering the rejecting loop of the tail component. Hence, the accepting loop in the tail component is always reachable (or has been reached already).

Thus, Spoiler can move his critical token to an accepting loop in the tail component 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

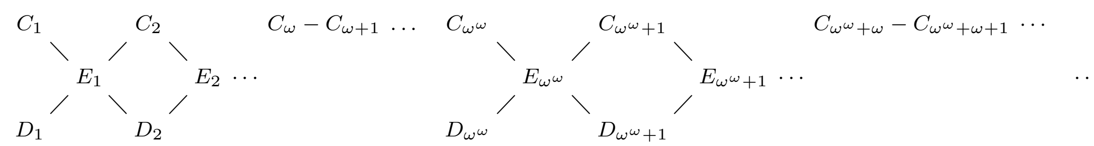

Figure 11: The Wadge ordering of the canonical automata.

$A = A_1 \wedge \ldots \wedge A_r$, 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 9.4 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 5.5, $A \leq (C_1 \oplus C_{\omega^{k-1}})^r$. Hence, by Lemma 9.3, $A \leq C_{\omega^{k-1}r+1}$. Since $\omega^{k-1}r + 1 < \omega^{k-1}r + 2 \leq \omega^k$, 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 10.3 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 $\alpha = \omega^k + \alpha'$. □

11. Patterns in Automata

Compare the notion of (ι, κ)-flower defined in Sect. 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. In that case it seems plausible to look at A as if it "contained" a copy of $F_{(\iota,\kappa)}$. In this section we provide a notion which captures this intuition.

Two paths $p \xrightarrow{\sigma'_1, d'_1} p'_1 \xrightarrow{\sigma'_2, d'_2} \ldots \xrightarrow{\sigma'_m, d'_m} p'_m$ and $p \xrightarrow{\sigma''_1, d''_1} p''_1 \xrightarrow{\sigma''_2, d''_2} \ldots \xrightarrow{\sigma''_n, d''_n} p''_n$ 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 $e_Q : Q^B \to Q^A$ 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 $e_\delta(p, \sigma, d) = r_0 \xrightarrow{\sigma_1, d_1} r_1 \xrightarrow{\sigma_2, d_2} \ldots \xrightarrow{\sigma_n, d_n} r_n$ 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 $\perp$). 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 11.1. *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 $1 \leq i < l$, $l+1 \leq j \leq m$, $l+1 \leq k \leq n$, where $\overline{d} = 1 - d$.

Recall that we assume that for every transition, either both target states are $\bot$ or none. Since $q' \neq \bot$ and $q'' \neq \bot$ then, by the condition (3) of the definition of admittance, $p_m' \neq \bot$ and $p_n'' \neq \bot$ and consequently all the states p_i, r_i, p_j', r_j', p_k'', r_k'' are not equal to $\bot$. 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 $\bot$, 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 $\bot$, Duplicator proceeds as before, only if some r_i, r_j', or r_k'' is equal to $\bot$, 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. $\square$

Another property that makes admittance similar to containment is transitivity.

Lemma 11.2. *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. $\square$

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.

12. 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. In their definition we will make the first use of a stronger variant of the operation $\to$.

In the operation $\xrightarrow{(\iota,\kappa)}$, instead of one (rejecting) loop replicating an automaton, we have a whole flower whose each loop replicates a different automaton. Recall that $F_{(\iota,\kappa)}$ is an automaton whose input alphabet is $\{a_\iota, a_{\iota+1} \ldots, a_\kappa\}$, the states are $q_\iota, q_{\iota+1}, \ldots, q_\kappa$, $\mathrm{rank}(q_i) = i$, the initial state is q_ι, 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 $k \neq j$. Let $A, A_\iota, \ldots, A_\kappa$ be deterministic tree automata over Σ. The (ι,κ)*-replication* $A \xrightarrow{(\iota,\kappa)} A_\iota, \ldots, A_\kappa$ (see Fig. 12) 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 $i = \iota, \ldots, \kappa$.

Using Lemma 5.5 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).

Note also that $A \to B$ and $A \xrightarrow{(1,1)} B$ are equal up to the names of letters and states. In particular $L(A \to B) \equiv_W L(A \xrightarrow{(1,1)} B)$.

Let us now define the three missing automata. 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

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

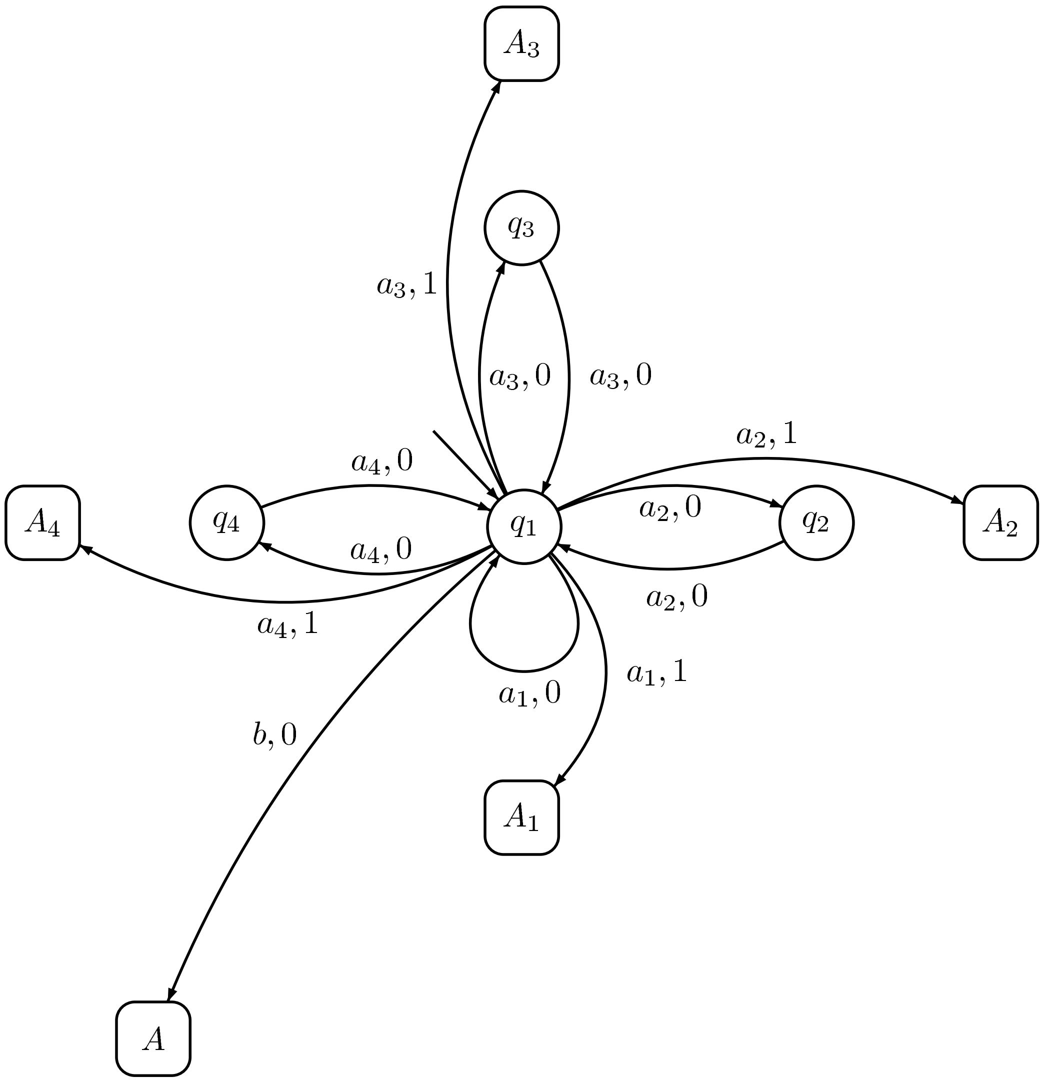

Figure 12: The $(1,4)$-replication $A \xrightarrow{(1,4)} A_1, A_2, A_3, A_4$.

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

Using canonical automata we can formulate results from Sect. 3 in a uniform way. In the proof we will need the following technical lemma.

Lemma 12.1. *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. Take an accepting run starting in p. For each node v of the run let S_v be the set of states of the automaton that appear below v. Note, that if v' is a descendant of v, $S_{v'} \subseteq S_v$. Since all S_v's are non-empty, there exists a node u such that for all descendants u' of u, $S_{u'} = S_u$. Pick a state $q \in S_u$. There exists a node w under u, labeled with q. Both $w0$ and $w1$ are labeled with S_u, so there exist a nodes w_0 under $w0$ and w_1 under $w1$ that are also labeled with q. To conclude, let π_p be the path in A induced by the path from ε to w, and let π_q^i be the path induced by the path between w and w_i for $i = 0, 1$. [1] ☐

Theorem 12.2. *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. It is enough to check that for an automaton it is the same to contain the patterns from Theorem 3.2 (page 7) and to admit the respective automata. 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 12.1. 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 $p \xrightarrow{\sigma,0} p_1 \xrightarrow{\sigma_1,d_1} \dots \xrightarrow{\sigma_m,d_m} p_{m+1} = p$ and a j-loop $p \xrightarrow{\sigma,1} p'_1 \xrightarrow{\sigma'_1,d'_1} \dots \xrightarrow{\sigma'_n,d'_n} p'_{n+1} = p$, 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 12.1 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 SCCs. 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 $\top_1, \dots, \top_5$, 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:

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

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. ☐

[1]This elegant proof was suggested by one of the referees in place of a clumsier inductive argument.

For the proof of the next theorem, settling the position of the last canonical automaton, $C_{\omega^{\omega\cdot 3}}$, we will need the following property of replication.

Lemma 12.3 (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 labelling 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. □

Theorem 12.4. *$L(C_{\omega^{\omega\cdot 3}})$ 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 12.2). Let us see that it is hard in that class.

Take a deterministic automaton A recognising a Δ^0_3-language. By Theorem 12.2 (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 7) by an accepting loop, otherwise it is *red*. Note that every state reachable from a blue state is blue.

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. Let us see that $A \leq A' \wedge A''$. The strategy for Duplicator in $G(A, A' \wedge A'')$ is to copy Spoiler's actions in A, both in A' and A''. To show that this strategy is winning it is enough to show that for each t a run of A on t is accepting iff the runs of A' and A'' on t are accepting. Take a path π of the run of A. Let π' and π'' be the corresponding paths of the computations of A' and A''. If π only visits red states, then the ranks on π and π'' are identical, and π' contains only 0's. Otherwise, π enters a blue state at some point, and then stays in blue states forever. In such case, the blue suffixes of π and π' have the same ranks, and the blue suffix of π'' contains only 0's. Thus, π is accepting iff π' and π'' are accepting and the claim follows.

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 12.2 (3)), $A' \leq F_{(1,2)}$.

Now consider A''. Once you enter a blue state, you can never move to a red state. Consequently, since in A'' all blue states have rank 0, we may actually replace them all with one all-accepting state $\top$ without changing the recognised 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 12.3) 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 5.5, $A' \wedge A'' \leq (C_1 \xrightarrow{(0,1)} C_1, F_{(\iota,\kappa)}) \wedge F_{(1,2)}$, 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 $y_1, \dots, y_{\lfloor \frac{\kappa+1}{2} \rfloor}$ in $F_{(0,2)}$. By Lemma 9.2, $(F_{(0,2)})^{\lfloor \frac{\kappa+1}{2} \rfloor} \equiv F_{(0,2\lfloor \frac{\kappa+1}{2} \rfloor)} \leq F_{(\iota,\kappa)}$, 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 $y_1, \dots, y_{\lfloor \frac{\kappa+1}{2} \rfloor}$ 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 12.2, $L(C_{\omega^{\omega}\cdot 2+n})$ is in Δ^0_3. By Lemma 10.4, for each $A \in \mathcal{C}$ there exists $m < \omega$ such that $A \leq C_{\omega^{\omega}\cdot 2+m}$. Hence, for all $A \in \mathcal{C}$, $L(A) \in \Delta^0_3$, and $A \leq C_{\omega^{\omega}\cdot 3}$. $\square$

From Theorems 12.2 and 12.4 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 $\mathcal{C}' = \mathcal{C} \cup \{C_{\omega^{\omega}\cdot 3}, C_{\omega^{\omega}\cdot 3+1}, C_{\omega^{\omega}\cdot 3+2}\}$. Note that it already follows that the height of the Wadge hierarchy of deterministic tree languages is at least $(\omega^{\omega})^3 + 3$. In the remaining of the paper we will show that each deterministic automaton is Wadge equivalent to one of the canonical automata from $\mathcal{C}'$, thus providing a matching upper bound.

13. Closure Properties

Our aim is to show that each deterministic tree language is Wadge equivalent to the language recognised 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. 6. 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 13.1. *For $A, B \in \mathcal{C}$ one can find in polynomial time an automaton in $\mathcal{C}$ equivalent to $A \vee B$.*

Proof. Proceed just like for nonbranching automata (Proposition 8.3, page 19). Take $A, B \in \mathcal{C}$. 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 10.4 we get that they must be equal to D_α and C_α. It follows easily from the definitions of the canonical automata that $D_\alpha \vee C_\alpha \equiv E_\alpha$. □

Proposition 13.2. *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 10.2, we may assume that Spoiler removes all his non-critical tokens on entering B''. Duplicator should simply move his critical token to the initial state 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 Proposition 8.3 (page 19) 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 5.5 (page 12), 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 $A \oplus B$. □

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 13.3. *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 13.4. *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 5.5, page 12) we obtain an automaton of the form $B_1^- \oplus C$, where C is canonical. Lemma 13.3 gives an equivalent canonical automaton.

Next, suppose that $A_1 = F_{(\iota,\kappa)}$, $B_1 = F_{(\iota',\kappa')}$. Assume $m, n > 1$. Using Lemma 9.2 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 $\to$.

Proposition 13.5. *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 9.1. 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 12.2 (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 13.3. $\square$

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 13.6. *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 12.2 and 12.4). 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 9.1). 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 12.2 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 $A_\iota, \dots, A_\kappa$. From now on Duplicator should mimic Spoiler's behaviour in his copy of A, handling rejecting loops in $A_\iota, \dots, A_\kappa$ 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 13.7. *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 12.2 and 12.4). □

14. Completeness

In this section we show that the canonical automata represent the $\equiv_W$-classes of all deterministically recognisable tree languages. We will implicitly use Corollary 13.7 and the Substitution Lemma (Lemma 5.5, page 12) 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 7). 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 2.1, page 4).

Lemma 14.1 (Niwiński and Walukiewicz [7]). *For each deterministic automaton A one can compute (in polynomial time if the productive states are given) an automaton $A\uparrow^0\uparrow^1 \dots \uparrow^n$ such that $L(A) = L(A\uparrow^0\uparrow^1 \dots \uparrow^n)$ and if a state q has the rank $j \le n$ than q lies on a j-loop of a (j,n)-flower.* □

Theorem 14.2. *For every deterministic tree automaton there exists an equivalent canonical automaton.*

Proof. Let A be a deterministic tree automaton. From Theorem 12.2 (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 12.2 (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 12.4). 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 root 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. Observe that each state of A is replicated by an accepting loop. Therefore, if A admits $F_{(0,1)}$, it must also admit $C_1 \xrightarrow{(0,0)} F_{(0,1)} = C_{\omega^{\omega\cdot 3}+1}$, which is excluded by our initial assumption. Consequently, A is a $(1,2)$-automaton (Theorem 2.2). Without loss of generality we may assume that A uses only ranks 1 and 2.

By Theorem 12.2 (3 and 4), $L(F_{(1,2)})$ is Π^0_2-complete and $L(A) \in \Pi^0_2$, which implies that $A \leq F_{(1,2)}$. If A admits $D_1 \oplus C_1$, then it also admits $C_1 \xrightarrow{(0,0)} D_1 \oplus C_1$, and so $A \geq C_1 \xrightarrow{(0,0)} D_1 \oplus C_1$. From Theorem 12.2 (3) it follows that $C_1 \xrightarrow{(0,0)} D_1 \oplus C_1 \equiv F_{(1,2)}$. Consequently, $A \equiv F_{(1,2)}$.

Suppose that A does not admit $D_1 \oplus C_1$, but X contains a rejecting loop λ_1. Let p_1 be a state on that loop. Since X contains a positive transition, it must contain an accepting loop and, in particular, a state with rank 2, say p_2. Since X is strongly connected, we may find a loop λ_2 going from p_1 to p_1 via p_2. Since X only uses ranks 1 and 2, and $\mathrm{rank}(p_2) = 2$, λ_2 is accepting. Hence, λ_1 and λ_2 form a $(1,2)$-flower. In consequence, $A \geq F_{(1,2)}$. Hence, $A \equiv F_{(1,2)}$.

Finally, suppose that X contains no rejecting loops and A does not admit $D_1 \oplus C_1$. By Theorem 12.2 (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 $C_1 \to F_{(0,2)} = C_{\omega^{\omega\cdot 3}}$, which contradicts our general hypothesis. Hence, A does not admit $F_{(0,2)}$, which means A is a $(1,3)$-automaton (Theorem 2.2, page 5). Without loss of generality we may assume that it uses only ranks $1, 2, 3$.

By Theorem 12.2 (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 12.2 (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 9.1, page 21), 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.

Hence, $A \equiv F_{(0,1)} \wedge F_{(1,2)}$ and by Lemma 9.2, $A \equiv F_{(1,3)}$.

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 14.1). 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, 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 13.6 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 $F \oplus B \leq A$. $\square$

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 root SCC 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 \kappa do
      B_j := the alternative of the canonical forms of X's j-replicated children
    end for
    return B \xrightarrow{(\iota,\kappa)} B_\iota, ..., B_\kappa
  end if
end if
```

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 14.3. *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 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 14.1 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. $\square$

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 14.3 and Theorem 10.5 we get an algorithm for Wadge reducibility.

Corollary 14.4. *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).* $\square$

Acknowledgements

The author thanks Damian Niwiński for drawing his attention to the Wadge hierarchy problems, and for reading carefully a preliminary version of this paper. The author is also grateful to the anonymous referees for their sharp yet extraordinarily useful comments that had a great impact on the present version of the paper.

References

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

[2] O. Finkel. Borel ranks and Wadge degrees of omega context free languages. Mathematical Structures in Computer Science **16** (2006) 813–840.

[3] A. S. Kechris. *Classical Descriptive Set Theory.* Graduate Texts in Mathematics Vol. 156, 1995.

[4] O. Kupferman, S. Safra, M. Vardi. Relating Word and Tree Automata. *11th IEEE Symp. on Logic in Comput. Sci.* (1996) 322–332.

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

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

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

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

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

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

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

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

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

[14] K. Wagner. Eine topologische Charakterisierung einiger Klassen regulärer Folgenmengen. J. Inf. Process. Cybern. EIK **13** (1977) 473–487.
[15] K. Wagner. On ω-regular sets. Inform. and Control **43** (1979), 123–177.