The Wadge Hierarchy of Deterministic Tree Languages

Filip Murlak*

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

Abstract. We provide a complete description of the Wadge hierarchy for deterministically recognizable 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 to the case of trees.

1 Introduction

Two measures of complexity of recognizable languages of infinite words or trees have been considered in literature: the index hierarchy, which reflects the combinatorial complexity of the recognizing 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 recognizable (ω-regular) sets of infinite words [14]. Subsequently, elementary 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 in the deterministic case [9, 13]. As for topological complexity of recognizable 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 recognize Π^1_1-complete languages [8], and Skurczyński [12] proved that there are even weakly recognizable 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 recognizable tree languages. In this paper we solve the Wadge hierarchy problem in the deterministic case. The obtained hierarchy has the height $(\omega^\omega)^3 + 3$, which should be compared with ω^ω for regular ω-languages [15], $(\omega^\omega)^\omega$ for deterministic context-free ω-languages [1], $(\omega_1^{CK})^\omega$ for ω-languages recognized by deterministic Turing machines [11], or an unknown ordinal $\xi > \varepsilon_0$ for nondeterministic context-free ω-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

* Supported by KBN Grant 4 T11C 042 25.

M. Bugliesi et al. (Eds.): ICALP 2006, Part II, LNCS 4052, pp. 408–419, 2006.

languages. Finally, we give 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).

In presence of space limitations we omit some of the proofs; they can be found in the full version of the paper (see [6] for a draft).

2 Automata

A *binary tree over* Σ is a partial function $t : \{l, r\}^* \rightharpoonup \Sigma$ with a prefix closed domain. A tree t is *full* if $\operatorname{dom} t = \{l, r\}^*$. Let T_Σ denote the set of full binary trees over Σ, and let $\tilde{T}_\Sigma$ be the set of all binary trees over Σ.

A *partial deterministic automaton* is a tuple $A = \langle \Sigma, Q, q_I, \delta, \mathrm{rank} \rangle$ where Σ is the input alphabet, Q is the set of states with a specified initial state q_I, the function rank maps states to naturals, and the transition relation δ is a partial function $\delta : Q \times \Sigma \times \{l, r\} \rightharpoonup Q$. A *run* of a partial automaton over $t \in T_\Sigma$ is a binary tree $\rho_t \in \tilde{T}_Q$ such that $\rho_t(\varepsilon) = q_I$ and for $v \in \operatorname{dom} \rho_t$, $\rho_t(v) = p$, $d = l, r$ it holds that $\delta(p, t(v), d) = q$ if and only if $vd \in \operatorname{dom} \rho_t$ and $\rho_t(vd) = q$. An infinite path $q_1, q_2, \ldots$ in a run is *accepting* if $\limsup_i \mathrm{rank}(q_i)$ is even. A run ρ_t is *infinitely accepting* if all its maximal paths are infinite and accepting. A state q is *productive* if there exists an infinitely accepting run starting in that state. A run is *accepting* if all its infinite paths are accepting and all finite maximal paths end in productive states. The *language recognised by* A, in symbols $L(A)$, is the set of trees that admit an accepting run of A. Partial deterministic automata over ω-words are defined analogously.

In classical deterministic automata the transition relation is a complete function. Consequently, all the paths in a run are infinite, and a run is accepting if and only if all the paths are accepting. Our definition is only a slight extension of the classical one. Adding transitions to an all-accepting state $\top$ or an all-rejecting state $\bot$ accordingly, we may transform a partial deterministic automaton A into a classical deterministic automaton $\tilde{A}$ recognizing the same language.

A *branching* transition is a pair $p \xrightarrow{\sigma,l} p_l$, $p \xrightarrow{\sigma,r} p_r$. For branching transitions we will sometimes write $p \xrightarrow{\sigma} p_l, p_r$. Transitions with one branch undefined will be called *non-branching*. We will also say that a partial deterministic automaton is non-branching if it contains only non-branching transitions.

The *head component* of an automaton A is the root of the directed acyclic graph (DAG) of strongly connected components (SCCs) of A. In the definitions of automata we will often specify the *tail components*, always a subset of the leaf SCCs.

3 Reductions and Games

T_Σ and the space of ω-words over Σ are equipped with the standard Cantor-like topology. For trees it is induced by the metric

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

L is *Wadge reducible* to M, in symbols $L \leq_W M$, if there exists a continuous function φ such that $L = \varphi^{-1}(M)$. M is *$\mathcal{C}$-hard* if for all $L \in \mathcal{C}$, $L \leq_W M$. If M is $\mathcal{C}$-hard and $M \in \mathcal{C}$, then M is *$\mathcal{C}$-complete*.

Let us introduce a *tree version of Wadge games* (see [10]). For any pair of tree languages L, M the game $G_W(L, M)$ is played by Spoiler and Duplicator. Each player builds a tree, t_S and t_D respectively. In every round, first Spoiler adds at least one level to t_S and then Duplicator can either add some levels to t_D or skip a round (not forever). Duplicator wins the game if $t_S \in L \iff t_D \in M$. Just like for the classical Wadge games, a winning strategy for Duplicator can be easily transformed into a continuous reduction, and vice versa.

Lemma 1. *Duplicator has a winning strategy in $G_W(L, M)$ if and only if $L \leq_W M$.*

For regular languages we find it useful to interpret the game in terms of automata. Let A, B be partial 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 following actions:

fire a transition – for a token placed in a state q choose a transition $q \xrightarrow{\sigma} q_l, q_2$, remove the old token from q and put new tokens in q_1 and q_2 (similarly for non-branching transitions),

remove – remove a token placed in a productive state.

Spoiler plays on A and must perform one of these actions at least for 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. During such a play the paths visited by the tokens of each player define a run of the respective automaton. Removing a token is interpreted as a declaration that this part of the run will be accepting: simply complete the run in construction by any accepting run starting in this state. Duplicator wins the game if both runs are accepting or both are rejecting.

Lemma 2. *Duplicator has a winning strategy in $G(A, B)$ iff $L(A) \leq_W L(B)$.*

Proof. Let $\mathrm{Acc}(C)$ denote the language of accepting runs of an automaton C. For deterministic automata, $\mathrm{Acc}(C) \equiv_W L(C)$. It is enough to observe that $G(A, B)$ is in fact $G_W(\mathrm{Acc}(A), \mathrm{Acc}(B))$. □

We will write $A \leq B$ for $L(A) \leq_W L(B)$, $A \equiv B$ for $L(A) \equiv_W L(B)$, and $A < B$ for $L(A) <_W L(B)$. For $q \in Q^A$ let $A_{q:=B}$ denote the automaton obtained by replacing each A's transition of the form $p \xrightarrow{\sigma,d} q$ with $p \xrightarrow{\sigma,d} q^B_I$. Recall that by A_q we denote the automaton A with the initial state changed to q. Note that $A_{q:=A_q}$ is equivalent to A. The following fact, which will be used implicitly throughout the paper, follows easily from Lemma 2.

Corollary 1. *Let A, B, C be deterministic partial automata and $p \in Q^C$. If $A \leq B$, then $C_{p:=A} \leq C_{p:=B}$.*

4 Gadgets

A partial automaton A' is a *transformation* of A if it was obtained from A by a finite number of the following operations:

relabeling – replacing $p \xrightarrow{\sigma} q_1, q_2$ with $p \xrightarrow{\sigma'} q_1, q_2$ for a fresh letter σ' (analogously for non-branching transitions),

swapping directions – replacing $p \xrightarrow{\sigma} q_1, q_2$ with $p \xrightarrow{\sigma} q_2, q_1$, or replacing a non-branching transition $p \xrightarrow{\sigma,d} q$ with $p \xrightarrow{\sigma,d'} q$.

edge subdivision – replacing $p \xrightarrow{\sigma,d} q$ with $p \xrightarrow{\sigma,d} p' \xrightarrow{\sigma,d} q$, where p' is a fresh state and $\mathrm{rank}(p') = \mathrm{rank}(p)$,

moving entering points in SCC – replacing $p \xrightarrow{\sigma,d} q$ with $p \xrightarrow{\sigma,d} q'$ for $q, q' \in X$, $p \notin X$, where X is a SCC of A (or replacing q_I with a different state from the head component),

moving exit points in SCC – replacing $p \xrightarrow{\sigma} q_1, q_2$ with $p' \xrightarrow{\sigma'} q_1, q_2$ for $p, p' \in X$, $q_1, q_2 \notin X$, and a fresh letter σ' (analogously for non-branching transitions).

Partial automata A and B over Σ are called *isomorphic* if there exists a bijection $\eta : Q^A \to Q^B$, preserving the initial state, such that $p \xrightarrow{\sigma,d} q$ if and only if $\eta(p) \xrightarrow{\sigma,d} \eta(q)$, and for each loop $p_1 \to \ldots \to p_n$, $\max_i \mathrm{rank}^A(p_i)$ and $\max_i \mathrm{rank}^B(\eta(p_i))$ have the same parity. In particular, $L(A) = L(B)$.

We say that A and B are *similar*, in symbols $A \sim B$, if there exist transformations A' and B' that are isomorphic. It can be checked easily that $\sim$ is an equivalence relation. The equivalence class of A will be denoted by $[A]$ and called a *gadget represented by* A.

An easy inductive argument shows that $A \sim B$ implies $L(A) \equiv_W L(B)$ (the converse need not hold). For a gadget Γ, we will denote by $L(\Gamma)$ the Wadge degree of the language recognized by an automaton representing Γ. The meaning of the symbols $<, \leq, \equiv$ introduced for partial automata extends to gadgets in the natural way. By abusing notation we will write $L(\Gamma) \leq_W L(A)$, $\Gamma \leq A$, etc.

Note that the transformations preserve the structure of the DAG of SCCs up to trivial components (singletons with exactly one parent and one child). Therefore, the head component of a gadget is well defined. Analogously, the tail components of a gadget are the tail components of any of the representing automata. Note also that for a single SCC the notion of similarity coincides with the classical notion of graph homeomorphism (up to relabeling and swapping directions).

It follows from the above that non-branching gadgets are well-defined. A non-branching gadget Γ can be represented by a (partial deterministic) ω-automaton over Σ: take any representing tree automaton and ignore the second coordinate of the arrow labels. Obviously, the ω-language recognized by the obtained automaton is Wadge equivalent to $L(\Gamma)$.

B is a *subautomaton of* A if $Q^B \subseteq Q^A$, $\delta^B \subseteq \delta^A$, $\mathrm{rank}^B = (\mathrm{rank}^A)\restriction_{Q^B}$ (the initial states need not be equal). A partial automaton A *contains* a gadget Γ if A contains a (productive) subautomaton B such that $[B] = \Gamma$. A *admits* Γ if it contains a productive subautomaton B which can be obtained from an automaton representing Γ by identifying some states. It follows easily that in both cases $[A] \geq \Gamma$.

5 Operations

The *alternative* $[A_1] \vee [A_2]$ is a gadget represented by an automaton consisting of disjoint copies of A_1 and A_2, and a fresh initial state q_I with the transitions $q_I \xrightarrow{\sigma_1,d_1} q_I^{A_1}$ and $q_I \xrightarrow{\sigma_2,d_2} q_I^{A_2}$, with $\sigma_1 \neq \sigma_2$. The tail components are inherited from $[A_1]$ and $[A_2]$. Note that $L([A_1] \vee [A_2])$ is Wadge equivalent to the disjoint sum of $L([A_1])$ and $L([A_2])$. Consequently, $\vee$ is associative and commutative up to Wadge equivalence.

The *parallel composition* $[A_1] \wedge [A_2]$ is defined analogously, only now $\sigma_1 = \sigma_2$ and $d_1 \neq d_2$. Note that, while in $[A_1] \vee [A_2]$ the computation must choose one of the branches, here it continues in both. The language $L([A_1] \wedge [A_2])$ is Wadge equivalent to $\{t : \ t.l \in L([A_1]), t.r \in L([A_2])\}$ and $\wedge$ is associative and commutative up to Wadge equivalence. Multiple parallel compositions are performed from left to right: $[A_1] \wedge [A_2] \wedge [A_3] \wedge [A_4] = (([A_1] \wedge [A_2]) \wedge [A_3]) \wedge [A_4]$. We will often write $([A])^n$ to denote $\underbrace{[A] \wedge \ldots \wedge [A]}_{n}$.

A (ι, κ)-*flower* $F_{(\iota,\kappa)}$ is a gadget represented by an automaton $A_{(\iota,\kappa)}$ with states $p, q_\iota, q_{\iota+1}, \ldots, q_\kappa$, $\mathrm{rank}(p) = \iota$, $\mathrm{rank}(q_i) = i$, and transitions $p \xrightarrow{\sigma_i,d_i} q_i \xrightarrow{\sigma_i',d_i'} p$ for $i = \iota, \iota+1, \ldots, \kappa$ such that $\sigma_i \neq \sigma_j$ for $i \neq j$. The only SCC of the (ι, κ)-flower is both the head component and the tail component.

The (ι, κ)-*composition* $[A] \xrightarrow{(\iota,\kappa)} [A_\iota], \ldots, [A_\kappa]$ is a gadget represented by an automaton obtained from the $A_{(\iota,\kappa)}$ above by adding (a single copy of) $A, A_\iota, \ldots, A_\kappa$ and transitions $p \xrightarrow{\sigma,d} q_I^A$, $p \xrightarrow{\sigma_i,\bar{d}_i} q_I^{A_i}$ such that $\sigma \neq \sigma_i$ and $\bar{d}_i \neq d_i$ for $i = \iota, \ldots, \kappa$, where d_i, σ_i and p are the ones from the definition of $A_{(\iota,\kappa)}$. The head component is simply the (ι, κ)-flower but for the tail components we choose only the tail components of A. Again, using Corollary 1 it is easy to see that the Wadge degree of the defined automaton depends only on (ι, κ) and the Wadge degrees of $L(A), L(A_\iota), \ldots, L(A_\kappa)$, and so the (ι, κ)-composition also defines an operation on Wadge degrees.

Let $C_1, \ldots, C_k$ be the tail components of $[A_1]$. The *sequential composition* $[A_1] \oplus [A_2]$ is represented by an automaton consisting of (a single copy of) A_1 and A_2 with transitions $p_i \xrightarrow{\sigma_i,d_i} q_I^{A_2}$, $i = 1, \ldots, k$, such that $p_i \in C_i$ and $\sigma_1, \ldots, \sigma_k$ are fresh letters. $[A_1] \oplus [A_2]$ inherits its head component from A_1 and the tail components from $[A_2]$. The operation $\oplus$ is associative, but it need not be commutative even up to Wadge equivalence. For any gadget Γ and $n < \omega$ let $n\Gamma = \underbrace{\Gamma \oplus \ldots \oplus \Gamma}_{n}$.

6 Canonical Gadgets

Let $C_1 = F_{(0,0)}$, $D_1 = F_{(1,1)}$. Note that in this case we simply get an accepting and a rejecting loop. $L(C_1)$ is Wadge equivalent to the whole space and $L(D_1) \equiv_W \emptyset$. Let $E_1 = C_1 \vee D_1$. Let $C_{\omega^{\omega+k}} = F_{(1,k+2)}$, $D_{\omega^{\omega+k}} = F_{(0,k+1)}$, and $E_{\omega^{\omega+k}} = C_{\omega^{\omega+k}} \vee D_{\omega^{\omega+k}}$. The above gadgets are called *simple non-branching gadgets*.

Let $\Gamma \to \Gamma'$ denote $\Gamma \xrightarrow{(1,1)} \Gamma'$. Let $E_\omega = C_1 \to C_3$ and $E_{\omega^{k+1}} = C_1 \to (C_1 \oplus E_{\omega^k})$ for $k \geq 1$. Let $E_{\omega^{\omega\cdot 2}} = C_1 \to F_{(0,2)}$ and $E_{\omega^{\omega\cdot 2+k+1}} = C_1 \to (C_1 \oplus E_{\omega^{\omega\cdot 2+k}})$ for $k \geq 1$. These in turn are called *simple branching gadgets*.

For every non-zero ordinal $\alpha < \omega^{\omega\cdot 3}$ we have a unique presentation

$$\alpha = \omega^{\omega\cdot 2+k} l_k + \ldots + \omega^{\omega\cdot 2+0} l_0 + \omega^{\omega\cdot +k} m_k + \ldots + \omega^{\omega\cdot +0} m_0 + \omega^k n_k + \ldots + \omega^0 n_0\,,$$

where at least one of l_k, m_k, n_k is non-zero ($\omega^0 = 1$, by convention). Let us define

$$E_\alpha = n_0 E_{\omega^0} \oplus \ldots \oplus n_k E_{\omega^k} \oplus m_0 E_{\omega^{\omega\cdot+0}} \oplus \ldots \oplus m_k E_{\omega^{\omega\cdot+k}} \oplus l_0 E_{\omega^{\omega\cdot 2+0}} \oplus \ldots \oplus l_k E_{\omega^{\omega\cdot 2+k}}\,.$$

For $\alpha = \omega^{\omega\cdot 2}\alpha_2 + \omega^\omega \alpha_1 + n + 1$, with $\alpha_1, \alpha_2 < \omega^\omega$, $n < \omega$ (at least one non-zero), let

$$C_\alpha = C_1 \oplus E_{\omega^{\omega\cdot 2}\alpha_2 + \omega^\omega \alpha_1 + n}\,, \quad D_\alpha = D_1 \oplus E_{\omega^{\omega\cdot 2}\alpha_2 + \omega^\omega \alpha_1 + n}$$

and for $\alpha = \omega^{\omega\cdot 2}\alpha_2 + \omega^{\omega+k}(\alpha_1 + 1)$, with $\alpha_1, \alpha_2 < \omega^\omega$ (at least one non-zero) and $k < \omega$, let

$$C_\alpha = C_{\omega^{\omega+k}} \oplus E_{\omega^{\omega\cdot 2}\alpha_2 + \omega^{\omega+k}\alpha_1}\,, \quad D_\alpha = D_{\omega^{\omega+k}} \oplus E_{\omega^{\omega\cdot 2}\alpha_2 + \omega^{\omega+k}\alpha_1}\,.$$

Let $\mathcal{G}$ denote the family of all the gadgets defined above.

By $\emptyset \xrightarrow{(\iota,\kappa)} \Gamma_\iota, \ldots, \Gamma_\kappa$ we understand a gadget obtained from $C_1 \xrightarrow{(\iota,\kappa)} \Gamma_\iota, \ldots, \Gamma_\kappa$ by removing the tail loop and the path joining it with the head loop. Let $E_{\omega^{\omega\cdot 3}}$ denote a gadget consisting of an accepting loop λ_0 and $\emptyset \xrightarrow{(1,1)} F_{(0,2)}$, such that λ_0 and the head loop of $\emptyset \xrightarrow{(1,1)} F_{(0,2)}$ form a $(0,1)$-flower. Let $C_{\omega^{\omega\cdot 3}+1} = \emptyset \xrightarrow{(0,0)} F_{(0,1)}$.

The gadgets defined in this section are called *canonical*. Observe that in a play involving any of the canonical gadgets there is always at most one token which can reach a tail component. Let us call such a token *critical*. Whenever a critical token splits in two, exactly one of its children can reach a tail component. We shall identify it with its parent and the other one will be treated as new.

7 Effective Hierarchies

The *index* of an automaton A is a pair $(\min \operatorname{rank}, \max \operatorname{rank})$. Scaling down the ranks by an even integer, we may assume that $\min \operatorname{rank}$ is 0 or 1. A is a (ι,κ)-*automaton* if $L(A)$ can be recognized by a deterministic automaton with index (ι,κ). The (ι,κ)-automata form the *deterministic index hierarchy*. The following theorem shows that the deterministic index hierarchy is effective for tree automata and ω-automata. Let $\overline{(0,m)} = (1,m+1)$, $\overline{(1,m)} = (0,m-1)$.

Theorem 1 (Niwiński & Walukiewicz [7]). *A deterministic automaton over words or trees is a (ι,κ)-automaton iff it does not admit a $\overline{(\iota,\kappa)}$-flower.*

Let Σ^0_n, Π^0_n denote the finite levels of the Borel hierarchy. As usually, $\Delta^0_n = \Sigma^0_n \cap \Pi^0_n$. The class of co-analytical sets (which need not be Borel) is denoted by Π^1_1. We showed in [5] that the Borel hierarchy is effective for deterministic tree languages. The proof relies on two facts, which will also be useful here. By a *split* we will understand a gadget Ω represented by $p \xrightarrow{\sigma,l} p_0 \longrightarrow p$, $p \xrightarrow{\sigma,r} p_1 \longrightarrow p$, with $\operatorname{rank} p = \operatorname{rank} p_0 = 0$, $\operatorname{rank} p_1 = 1$.

Theorem 2 (Niwiński & Walukiewiecz [8]). *Let A be a deterministic automaton. If A admits a split, $L(A)$ is Π^1_1-complete (and hence, non-Borel). If A does not admit a split, $L(A) \in \Pi^0_3$.*

Theorem 3 (Murlak [5]). *For a deterministic automaton A, $L(A) \in \Pi^0_2$ iff A does not admit $F_{(0,1)}$, $L(A) \in \Sigma^0_2$ iff A admits neither $F_{(1,2)}$ nor $\emptyset \xrightarrow{(0,0)} D_2$, and $L(A) \in \Sigma^0_3$ iff A does not admit $C_{\omega^{\omega\cdot 3}+1}$.*

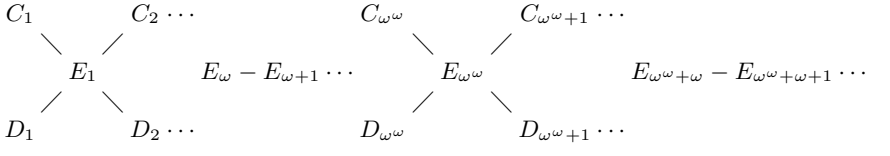
In fact, $L(F_{(1,2)})$ and $L(\emptyset \xrightarrow{(0,0)} D_2)$ are Π^0_2-complete, $L(F_{(1,2)})$ is Σ^0_2-complete, and $L(C_{\omega^{\omega\cdot 3}+1})$ is Π^0_3-complete.

Let $\mathcal{G}'$ denote the set of all non-branching canonical gadgets $\{C_\alpha, D_\alpha, E_\alpha : \alpha = \omega^\omega\gamma + m, \gamma < \omega^\omega, m < \omega\}$. In [15] Wagner showed that the Wadge hierarchy for regular ω-languages is constituted by the Wadge degrees of gadgets from $\mathcal{G}'$ and that the hierarchy is effective.

Theorem 4 (Wagner [15]). *For an ω-automaton A and $C_\alpha, D_\alpha, E_\alpha \in \mathcal{G}'$ it holds that $L(A) \leq_W L(E_\alpha)$ iff A admits neither $C_{\alpha+1}$ nor $D_{\alpha+1}$, and $L(A) \leq_W L(C_\alpha)$ iff A does not admit D_α (and dually).*

Let us state our main result. The remaining of the paper is devoted to the proof of it.

Theorem 5. *The Wadge hierarchy of deterministic tree languages is as follows*



8 Closure Properties

The main result of this section is that the family $\mathcal{G}$ is closed by the basic operations.

Proposition 1. *The family $\mathcal{G}$ is closed by the operations $\vee$, $\wedge$, $\oplus$, $\to$ up to Wadge equivalence, and the equivalent gadget may be found in polynomial time.*

Instead of giving the whole proof, which is quite technical (see [6]), we present a handful of special cases in which the result of the operation can be given explicitly and which will turn out useful later.

Lemma 3. *$E_\gamma \oplus C_{\omega^\omega\alpha} \equiv C_{\omega^\omega\alpha}$ and $E_\gamma \oplus D_{\omega^\omega\alpha} \equiv D_{\omega^\omega\alpha}$ for all $0 < \gamma, \alpha < \omega^\omega$.*

Proof. We shall only consider the case of C_α; the D_α case is dual. A strategy for Duplicator in $G(E_\gamma \oplus C_\alpha, C_\alpha)$ is as follows. While Spoiler keeps inside E_γ, stay in the first flower of C_α. If one of Spoiler's tokens is inside a rejecting loop, loop a rejecting loop in your flower, otherwise loop an accepting one. When he enters C_α, simply copy his actions moving from one flower to another. Only when one of his tokens in E_γ is

in a rejecting loop, choose a rejecting loop in your current flower (instead of copying Spoiler's move in C_α). Since a path in E_γ can only be rejecting if one of the tokens stays forever in a rejecting loop, this strategy is winning. □

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

The following auxiliary gadgets will be called *weak flowers*:

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

In fact, $WF_{(0,n)} \equiv C_{n+1}$, $WF_{(1,n+1)} \equiv D_{n+1}$, but we find the notation convenient.

Lemma 4. *For all $i, j, m, n < \omega$ it holds that $F_{(i,m)} \wedge F_{(j,n)} \equiv F_{(i,m)\wedge(j,n)}$ and $WF_{(i,m)} \wedge WF_{(j,n)} \equiv WF_{(i,m)\wedge(j,n)}$.*

Proof. For ω-regular languages L, M, let $A_{L\times M}$ be the canonical product automaton recognizing $L \times M = \{(v_1, w_1)(v_2, w_2) \ldots : v_1 v_2 \ldots \in L,\ w_1 w_2 \ldots \in M\}$. Let $L = L(F_{(i,m)})$, $M = L_\omega(F_{(j,n)})$. Since flowers are non-branching gadgets, we may assume that L, M are ω-regular. It holds that $A_{L\times M} \equiv F_{(i,m)} \wedge F_{(j,n)}$. It is easy to see that alternating chains of loops in $A_{L\times M}$ correspond directly to alternating chains in $[i, m] \times [j, n]$. Hence, $A_{L\times M}$ admits a $(i, m) \wedge (j, n)$-flower, and does not admit a $(i, m) \wedge (j, n)$-flower. From Thm. 4 it follows that $A_{L\times M} \equiv F_{(i,m)\wedge(j,n)}$. The proof for weak flowers is analogous. □

Lemma 5. *For all $0 < k, l < \omega$ and all $m < \omega$ it holds that $C_1 \oplus E_{\omega^m k} \wedge C_1 \oplus E_{\omega^m l} = C_1 \oplus E_{\omega^m (k+l)}$ and $C_1 \oplus E_{\omega^{\omega\cdot 2+m} k} \wedge C_1 \oplus E_{\omega^{\omega\cdot 2+m} l} = C_1 \oplus E_{\omega^{\omega\cdot 2+m} (k+l)}$.*

Proof. We will only give a proof of the first equivalence. Let us consider $G(C_1 \oplus E_{\omega^m k} \wedge C_1 \oplus E_{\omega^m l}, C_1 \oplus E_{\omega^m k} \oplus E_{\omega^n l})$. Duplicator has only one critical token which can move along $WF_{(0,2(k+l))}$ formed by the alternating head and tail loops of consecutive copies of E_{ω^m}. Spoiler's starting token splits in the first move into two critical tokens which continue moving along $WF_{(0,2k)}$ and $WF_{(0,2l)}$. The strategy for Duplicator is to loop his critical token inside an accepting loop as long as both Spoiler's critical tokens loop inside accepting loops; if at least one of them moves to a rejecting loop, Spoiler should also move to a rejecting loop, and so on (c. f. Lemma 4). This way, whenever Spoiler produces a new token x using one of the critical tokens, Duplicator can produce its *doppelgänger* y, and let it mimic x. Hence, $C_1 \oplus E_{\omega^m k} \wedge C_1 \oplus E_{\omega^m l} \leq C_1 \oplus E_{\omega^m (k+l)}$. The converse inequality is obvious. □

9 Wadge Ordering

In this section we will investigate the Wadge ordering of the canonical gadgets. First, let us see what the use of the operation $\to$ is.

Lemma 6. *For all gadgets Γ, Δ and all $0<k<\omega$, $\Gamma \to \Delta \geq (\Gamma \to \Delta) \wedge (\Delta)^k$.*

Proof. Consider $G((\Gamma \to \Delta) \wedge (\Delta)^k, \Gamma \to \Delta)$. Spoiler's initial moves produce a token x in the head loop of $\Gamma \to \Delta$, and tokens $x_1, \ldots, x_k$, each in the head component of a different copy of Δ. Duplicator should loop his starting token y around the head loop of $\Gamma \to \Delta$ exactly k times producing tokens $y_1, \ldots, y_k$ and move them to the head component of Δ. From now on y mimics x, and y_i mimics x_i for $i = 1, \ldots, k$. □

Corollary 2. *For all $k, \iota, \kappa < \omega$ and all $0 < n < \omega$, $E_\omega > WF_{(\iota,\kappa)}$, $E_{\omega^{k+1}} \geq E_{\omega^k n}$, $E_{\omega^{\omega\cdot 2}} > F_{(\iota,\kappa)}$, $E_{\omega^{\omega\cdot 2+k+1}} \geq E_{\omega^{\omega\cdot 2+k} n}$.*

Proof. It is easy to see that $(0, 2m) \wedge (0, 2n) = (0, 2m+2n)$. Consequently, by Lemma 6 and Lemma 4, $E_{\omega^\omega} \geq (WF_{(0,2)})^m \equiv WF_{(0,2m)}$ and by the strictness of the hierarchy for ω-languages $\Gamma_{\omega^\omega} > WF_{(\iota,\kappa)}$. Similarly, using Lemma 6 and Lemma 5 we get $E_{\omega^{k+1}} \geq (C_1 \oplus E_{\omega^k})^n \equiv C_1 \oplus E_{\omega^k n} \geq E_{\omega^k n}$. The remaining inequalities are analogous. □

From the facts above we obtain the following lemma (see [6] for details).

Lemma 7. *If $0 < \alpha \leq \beta \leq \omega^{\omega\cdot 3}$ then $E_\alpha \leq E_\beta$ and whenever C_α and D_α are defined, $C_\alpha \leq E_\beta$, $D_\alpha \leq E_\beta$. If $\alpha < \beta$ then $E_\alpha \leq C_\beta$, $E_\alpha \leq D_\beta$.*

Now we can show that the hierarchy induced on the family of the canonical gadgets by the Wadge ordering actually coincides with the one described in Sect. 7.

Theorem 6. *Let $0 < \alpha \leq \beta \leq \omega^{\omega\cdot 3}$. Whenever the respective gadgets are defined, it holds that $C_\alpha \not\leq D_\alpha$, $C_\alpha \not\geq D_\alpha$, $C_\alpha < E_\beta$, $D_\alpha < E_\beta$, and for $\alpha < \beta$, $E_\alpha < E_\beta$, $E_\alpha < C_\beta$, $E_\alpha < D_\beta$.*

Proof. By Lemma 7 it is enough to prove $E_\alpha < E_{\alpha+1}$, $C_\alpha < E_\alpha$, $D_\alpha < E_\alpha$, $C_\alpha \not\leq D_\alpha$, $C_\alpha \not\geq D_\alpha$. 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 ω-languages case.

Suppose $\alpha = \omega^k + \alpha'$, $k \geq 1$. Let $\alpha' \geq 1$ (the remaining case is similar). We shall describe a winning strategy for Spoiler in $G = G(E_{\omega^k+\alpha'+1}, E_{\omega^k+\alpha'})$. Spoiler should first follow the winning strategy for $G(E_{\alpha'+1}, E_{\alpha'})$, which exists by the induction hypothesis. When Duplicator enters the head loop of E_{ω^k}, Spoiler removes all his non-critical tokens, moves his critical token to the (accepting) tail loop of $E_{\alpha'+1}$ and loops there until Duplicator leaves the head loop. If Duplicator stays forever in the head loop of E_{ω^k}, he looses. After Duplicator has left the head loop, the play is equivalent to $G' = G(C_1 \oplus E_{\omega^k}, \Gamma)$ for $\Gamma = \Gamma_1 \wedge \ldots \wedge \Gamma_r$, where Γ_j is the part of E_α accessible for the Duplicator's jth token. If $k = 1$, then $\Gamma_j \leq WF_{(0,2)}$ for each j. Hence

$\Gamma \leq WF_{(0,2r)}$ and G' is winning for Spoiler by Corollary 2. Let us suppose $k > 1$. Then $\Gamma_j \leq C_1 \oplus E_{\omega^{k-1}}$ for $j = 1, \ldots, r$. Again, by Corollary 2, $\Gamma \leq E_{\omega^{\omega \cdot 2+k-1}r+1}$. Since $\omega^{k-1}r + 1 < \omega^{k-1}r + 2 \leq \alpha$, we may use the induction hypothesis to get a winning strategy for Spoiler in G'. In either case Spoiler has a winning strategy in G as well.

Now, assume $\omega^\omega \leq \alpha < \omega^{\omega \cdot 2}$. Let $\alpha = \omega^\omega \alpha_1 + \alpha_0$ with $\alpha_0 < \omega^\omega$, $1 \leq \alpha_1 < \omega^\omega$. Again, we describe a strategy for Spoiler in $G = G(E_{\omega^\omega \alpha_1 + \alpha_0 + 1}, E_{\omega^\omega \alpha_1 + \alpha_0})$ only for $\alpha_0 \geq 1$, leaving the remaining case to the reader. First follow the winning strategy from $G(E_{\alpha_0+1}, E_{\alpha_0})$. If Duplicator does not leave the E_{α_0} component, he will lose. After leaving E_{α_0}, Duplicator has to choose $C_{\omega^\omega \alpha_1}$ or $D_{\omega^\omega \alpha_1}$. Suppose he chooses $C_{\omega^\omega \alpha_1}$. By Lemma 3, $E_{\alpha_0} \oplus C_{\omega^\omega \alpha_1} \equiv C_{\omega^\omega \alpha_1}$, and $E_{\omega^\omega \alpha_1} > C_{\omega^\omega \alpha_1}$. Therefore, Spoiler has a winning strategy in $G_C = G(E_{\omega^\omega \alpha_1}, E_{\alpha_0} \oplus C_{\omega^\omega \alpha_1})$. Imagine a play in G_C in which Duplicator ignores Spoiler's move in the first round and copies the token pattern from the stopped play we were considering above. Of course Spoiler's strategy must work in this case too. The strategy for Spoiler in G is to move the critical token to the first node of the $E_{\omega^\omega \alpha_1}$ component, take all the other tokens away, and then follow the strategy from G_C merging the first two moves into one.

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

10 Completeness

In this final section we show that the canonical gadgets represent Wadge degrees of all deterministically recognizable tree languages. To this end we will need the following technical lemma which follows from the closure properties [6].

Lemma 8. *Let A be a deterministic automaton A whose SCCs contain no complete transitions. If A admits neither $E_{\omega^{\omega \cdot 3}}$ nor $C_{\omega^{\omega \cdot 3}+1}$, one can find effectively a gadget $\Gamma \in \mathcal{G}$ such that $\Gamma \equiv A$.*

Theorem 7. *For a deterministic tree automaton A admitting neither $E_{\omega^{\omega \cdot 3}}$ nor $C_{\omega^{\omega \cdot 3}+1}$, one can find effectively an equivalent gadget $\Gamma \in \mathcal{G}$.*

Proof. We may assume that A has a *tree form*, by which we mean that the DAG of SCCs of A is a tree and that the only components containing unproductive states are leaves containing only one all-rejecting state. We will proceed by induction on the structure of this tree. Let X denote the head component of A. Suppose first that X contains a branching transition with one of its branches lying on an accepting loop. Should A admit $F_{(0,1)}$, it would also admit $C_{\omega^{\omega \cdot 3}+1}$, which is excluded by the hypothesis. Consequently, A is a $(1, 2)$-automaton. If A admits $F_{(1,2)}$ or $\emptyset \xrightarrow{(0,0)} D_2$, then $A \equiv F_{(1,2)}$. Otherwise, X contains no rejecting loops and canonical forms of the subtrees rooted in the child components of X are at most C_2. If A contains D_1, then $A \equiv C_2$, otherwise $A \equiv C_1$.

Suppose that the above does not happen, but there is a branching transition with one of its branches lying on a rejecting loop and X contains an accepting loop. It follows immediately that X admits $F_{(0,1)}$ and A does not admit $F_{(0,2)}$, which means it is a $(1, 3)$ automaton. If A admits neither $F_{(1,2)}$ nor $\emptyset \xrightarrow{(0,0)} D_2$, then $A \equiv F_{(0,1)}$. If A admits one

of this two gadgets, it follows easily that $F_{(1,3)} \leq A$. Without loss of generality we may assume that only states lying on $(0,1)$-flowers may have rank 3. Let $\mathrm{rank}'(q) = (\mathrm{rank}(q))'$, where $1' = 1$, $2' = 2$, $3' = 1$, and $\mathrm{rank}''(q) = (\mathrm{rank}(q))''$, where $1'' = 0$, $2'' = 0$, $3'' = 1$. Let $A' = \langle \Sigma, Q, q_I, \delta, rank' \rangle$, $A'' = \langle \Sigma, Q, q_I, \delta, rank'' \rangle$. It is obvious that $A \leq A' \wedge A''$: a winning strategy for Duplicator in $G(A, A' \wedge A'')$ is to copy Spoilers behaviour both in A' and A''. A' is a $(1,2)$-automaton, so by Theorem 3, $A' \leq F_{(1,2)}$. Since A does not admit $\emptyset \xrightarrow{(0,0)} F_{(0,1)}$, A'' will not admit $\emptyset \xrightarrow{(0,0)} D_2$, and hence $A'' \leq F_{(0,1)}$. It follows that $A \equiv F_{(1,3)}$.

If X contains a branching transition but contains no accepting loops proceed as follows. 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 $p_j' \notin X$. By the induction hypothesis we may assume that $(A)_{q_i'}$, $(A)_{q_i''}$ and $(A)_{p_j'}$ are in canonical forms. Let $\Delta = ((A)_{q_1'} \wedge (A)_{q_1''}) \vee \ldots \vee ((A)_{q_n'} \wedge (A)_{q_n''}) \vee (A)_{p_1'} \vee \ldots \vee (A)_{p_m'}$. It is not difficult to see that A is equivalent to $C_1 \to \Delta$. By Proposition 1 we get a canonical gadget equivalent to $C_1 \to \Delta$.

Finally, let X contain no branching transitions. By induction hypothesis we may assume that the subtrees rooted in the children of X are in the canonical form. Consequently, no SCC of A contains a branching transition and we may use the lemma. □

Theorem 8. *$L(E_{\omega^{\omega \cdot 3}})$ is Wadge complete for deterministic Δ_3^0 tree languages.*

Proof. Take a deterministic automaton A recognizing a Δ_3^0-language. By Thm. 3, A does not admit $\emptyset \xrightarrow{(0,0)} F_{(0,1)}$. When a token splits in a branching transition, we will imagine that it goes left, and bubbles a new token to the right. Thus, in every transition only one token is produced. Let us divide the states of A into two categories: a state q is *blue* if there exists a (productive) accepting loop $p \xrightarrow{\sigma, d} p' \to \ldots \to p$ and a (productive) path $p \xrightarrow{\sigma, \bar{d}} p'' \to \ldots \to q$, $d \neq \bar{d}$. The remaining states are *red*. The tokens get the color of their birth state. The essential observation is that during an accepting run the occurrences of red states may be covered by a finite number of infinite paths (see [5], the proof of Thm. 4). Consequently, only finitely many red tokens may be produced during an accepting run.

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. Like before, $A \leq A' \wedge A''$. Since A does not admit $\emptyset \xrightarrow{(0,0)} F_{(0,1)}$, it follows that all $(0,1)$-flowers in A are red. Consequently, A' does not admit $F_{(0,1)}$, and $A' \leq F_{(1,2)}$.

Let Λ denote a gadget produced out of $E_{\omega^{\omega \cdot 3}}$ by replacing $F_{(0,2)}$ with $F_{(\iota,\kappa)}$, where (ι, κ) is the index of A. Consider the game $G(A'', \Lambda)$. In A'' the blue tokens are always in the states with rank 0, so they do not influence the result of the computation. Whenever Spoiler produces a new red token (including the starting token), Duplicator should loop once around the head 1-loop producing a new token 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. Using the assertion on red tokens, one checks easily that the strategy is winning. Hence $A'' \leq \Lambda$. By Lemma 6 and Lemma 4 it follows that $\Lambda \wedge F_{(1,2)} \leq E_{\omega^{\omega \cdot 3}}$. □

The following corollary sums up the results of this section and the whole paper.

Corollary 3. *For a deterministic tree automaton A the exact position of $L(A)$ in the Wadge hierarchy of deterministic tree languages (see Thm. 5) can be calculated within the time of finding the productive states of the automaton.*

Proof. From Thm. 2 it follows that if A admits Ω, $A \equiv \Omega$. If A does not admit Ω, then by Thm. 3 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 $E_{\omega^{\omega\cdot 3}}$, then $A \equiv E_{\omega^{\omega\cdot 3}}$ (Thm. 8). The remaining case is settled by Thm. 7.

If the productive states are given, checking if an automaton admits Ω, $C_{\omega^{\omega\cdot 3}+1}$, or $E_{\omega^{\omega\cdot 3}}$ can be done in polynomial time. The algorithm sketched in the proof of Thm. 7 can be implemented polynomially as well, by realizing the described procedure bottom-up on the original DAG of SCCs without constructing the tree form of A explicitly. □

Acknowledgements

The author thanks Damian Niwiński for drawing his attention to the Wadge hierarchy problems and for inspiring discussions, and the anonymous referees for useful comments.

References

1. J. Duparc. A hierarchy of deterministic context-free ω-languages. Theoret. Comput. Sci. **290** (2003) 1253–1300.
2. O. Finkel. Wadge Hierarchy of Omega Context Free Languages. Theoret. Comput. Sci. **269** (2001) 283–315.
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. F. Murlak. The Wadge hierarchy of deterministic tree languages. Draft version, `http://www.mimuw.edu.pl/~fmurlak/papers/conred.pdf`.
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 2004*, 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 2000*, 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.