The Non-deterministic Mostowski Hierarchy and Distance-Parity Automata

Thomas Colcombet[1,*] and Christof Löding[2]

[1] LIAFA/CNRS, France
[2] RWTH Aachen, Germany

Abstract. Given a Rabin tree-language and natural numbers i, j, the language is said to be i, j-feasible if it is accepted by a parity automaton using priorities $\{i, i+1, ..., j\}$. The i, j-feasibility induces a hierarchy over the Rabin-tree languages called the Mostowski hierarchy.

In this paper we prove that the problem of deciding if a language is i, j-feasible is reducible to the uniform universality problem for distance-parity automata. Distance-parity automata form a new model of automata extending both the nested distance desert automata introduced by Kirsten in his proof of decidability of the star-height problem, and parity automata over infinite trees. Distance-parity automata, instead of accepting a language, attach to each tree a cost in $\omega + 1$. The uniform universality problem consists in determining if this cost function is bounded by a finite value.

1 Introduction

Finite automata running on infinite trees, originally introduced by Rabin in his seminal work [15] are now widely considered as one of the key paradigms for understanding many logics relevant to verification. Those automata are known to be effectively equivalent to monadic second-order logic, μ-calculus, and to subsume all the standard temporal logics.

An important parameter in the definition of the automaton model is the acceptance condition. This acceptance condition determines, given a run of the automaton, whether it is accepting or not. Different (often equivalent) choices of acceptance conditions are known from the literature such as Büchi, Rabin, Muller, or Streett conditions (cf. [16]). Though all possess their own interest, the *parity condition* has emerged for many reasons as the central condition in the theories of automata, logic and games.

When using a parity condition, each state of the automaton is labelled by a natural number – called a priority – belonging to a fixed finite interval $[i, j]$. A run is accepting if on every branch the highest priority seen infinitely often is even. A language is said to be i, j-feasible if there exists a finite automaton using the interval of priorities $[i, j]$ accepting this language. Of course, the language does not change if we shift all priorities by steps of 2 or -2. This is why we

---

$$(1, k+2) \qquad (0, k+1)$$
$$(1, k+1) \qquad (0, k)$$

$$(1, 3) \qquad (0, 2)$$
$$(1, 2) \qquad (0, 1)$$
$$(1, 1) \qquad (0, 0)$$
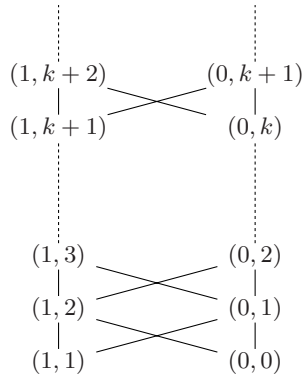
Fig. 1. Hierarchy of Mostowski indices

can restrict ourselves to $i = 0$ or $i = 1$. It is also clear that the bigger the interval $[i, j]$ is, the more tree languages are i, j-feasible. Mostowski first studied this parameter [10], and the corresponding ladder-shaped hierarchy – depicted Figure 1 – is named after him.

This Mostowski hierarchy exists in different variants according to the nature of the transition relation used by the automaton. Over trees the hierarchy is strict for all standard models of automata: deterministic [18] (even over words), non-deterministic [11], and alternating [1] (in combination with [4,9]), see also [2]. The hierarchy collapses over words for non-deterministic automata to the Büchi level $(1, 2)$, and in the alternating case to the intersection of levels $(0, 1)$ and $(1, 2)$.

The next step in the study of this hierarchy is the question of decidability. The problem is the following: given a regular language of infinite trees L (i.e., accepted by a non-deterministic or alternating automaton), and natural numbers $i \leq j$, is the language i, j-feasible? This question is parameterised both by the nature of the language L, that we call the *input* language, and the nature of the class of automata for which we test the i, j-feasibility, the *output* class.

In the case of any non-deterministic automaton as input, the 1, 1-feasibility and the 0, 0-feasibility in the non-deterministic Mostowski hierarchy is decidable for simple reasons: a language is at level $(1, 1)$ iff it consists solely of finite trees, and a language is at level $(0, 0)$ iff it is closed for the standard topology over infinite trees. Those two properties are easily shown to be decidable. Note also that the non-deterministic and the alternating hierarchy coincide over those two levels.

More interestingly, the problem is known to be decidable in the Mostowski hierarchy of languages accepted by deterministic automata. The problem is decidable if both the input language and the output class are deterministic [13] (see also [14] for more details). The problem is also known to be decidable if the input language is deterministic, and the output class is non-deterministic [14] (which is a refinement of the case of a path language as input [13]). The special case of deciding if a deterministic language is accepted by a non-deterministic Büchi automaton (i.e., 1, 2-feasible in the non-deterministic Mostowski hierarchy) was

formerly settled in [17]. Let us finally remark that every deterministic language is alternating co-Büchi, i.e., that every deterministic language falls in the level $(0, 1)$ of the alternating Mostowski hierarchy. And hence the case of a deterministic language as input and an alternating one as output is also settled.

This paper is the first part in an attempt to show the decidability of the following problem.

Problem 1. Given a regular tree language L and natural numbers $i \leq j$, answer whether L is i, j-feasible in the non-deterministic Mostowski hierarchy or not.

The scheme of the proof is inspired from the proof of decidability of the (restricted) star-height problem due to Kirsten [7] (the problem was originally solved by Hashiguchi [6]). The star-height problem is the following: given a regular language of finite words and a natural number k, is it possible to describe the language by a regular expression using at most k nesting of Kleene stars?

For showing the decidability of the star-height problem, Kirsten introduces a new class of automata: nested distance desert automata. Those are non-deterministic finite automata running on finite words and equipped with some counting features. The semantics of such automata is either to reject a word, or to compute a natural number for it, that we can see as the price to pay for accepting it. Hence a nested distance desert automaton defines a partial mapping from words to natural numbers. The proof then goes in two steps.

- Reduce the star-height problem to the limitedness problem for nested distance desert automata (the limitedness is the problem of determining if the partial mapping defined by the automaton is bounded).
- Solve the limitedness problem for nested distance desert automata.

We want to follow exactly the same scheme to solve Problem 1. We introduce the family of distance-parity automata running over infinite trees. Those automata combine the features of nested distance desert automata and of parity automata. If we restrict a distance-parity automaton to run on finite words, we fall back to the class of automata defined by Kirsten. If we restrict those automata to infinite words, we get a family of automata equivalent to the hierarchical ωB-automata in [3]. The limitedness problem still makes sense for the distance-parity automata, but we prefer to work with the uniform universality problem: a distance-parity automaton is uniformly universal if the function it defines is both total and bounded. This problem is decidable for finite words from [7], and can be shown to be decidable over infinite words using [3].

Our proof scheme then goes as follows in two steps:

- Reduce the i, j-feasibility problem to the uniform universality problem for distance-parity automata (Theorem 2).
- Show the decidability of the uniform universality problem for distance-parity automata (open).

This paper is concerned with the first item. This reduction part is very different from the one in the proof of Kirsten. Indeed, there is an intrinsic difficulty in

the study of regular languages of infinite trees: there is no known notion of a canonical object describing a language. When dealing with finite words, one can use the minimal deterministic automaton or the syntactic monoid. When dealing with infinite words, one can use the syntactic ω-semigroup. When dealing with finite trees, there exist a minimal bottom-up deterministic automaton, and a corresponding algebraic presentation. Most methods for characterising classes of languages begin by taking such a canonical presentation. But for infinite trees no canonical type of acceptor is known. This problem is very deep as it can be witnessed by the following fact: some languages are inherently ambiguous, i.e., it is impossible to provide an automaton that would possess a single accepting run over every accepted input (the proof of this result is given in [12] but has not been published; see also [5]). One contribution of this paper is the notion of a _guidable automaton_, i.e., an automaton that is able to 'mimic' the behaviour of every automaton accepting the same language. This guidable automaton plays the role of the canonical presentation in our reduction. We show that every regular tree language is accepted by a guidable automaton (Theorem 1).

The second part of our reduction is an 'on the fly' optimisation of the guidable automaton. This optimisation process makes use of the distance features of distance-parity automata. It is shown to be both correct, and optimal.

The remainder of the paper is organised as follows. Section 2 is devoted to definitions, in particular automata and the acceptance conditions we use. Section 3 presents the notion of guidable automaton and we prove that such an automaton exists for each regular language of trees. In Section 4 we establish Theorem 2 reducing the i, j-feasibility problem to the uniform universality problem.

2   Definitions

Words are finite sequences of letters. The set of words over an alphabet A is denoted by A^*. The empty word is ε, and uv is the concatenation of u and v. We denote by $\sqsubseteq$ the prefix relation over words and by $\sqsubset$ its strict variant. The length of a word u is $|u|$, and $|u|_a$ for $a \in A$ is the number of occurrences of a in u. An ω-word is an infinite sequence of letters and by A^ω we denote the set of infinite words over A. The ordered set of natural numbers is written as ω, and $\omega + 1$ is ω augmented with the maximal value ω.

For the sake of simplicity, we assume that trees are binary and complete (i.e., with no leaves)[1]. A _tree_ labelled by a finite _alphabet_ A (we also say an A-tree) is a mapping from $\{0, 1\}^*$ to A. The elements of $\{0, 1\}^*$ are called _nodes_. A _branch_ is a maximal totally ordered set of nodes. A branch naturally induces an ω-word in A^ω. It is sometimes convenient to identify a branch with this ω-word.

An _automaton_ is a tuple $\mathcal{A} = (Q, A, I, \Delta, col)$ in which:

---

[1] We can code leaves in an infinite binary tree by marking all nodes below by a special dummy symbol. It is easy to show that if the interval $[i, j]$ contains an even priority, then the original language is i, j-feasible iff the one after coding is i, j-feasible. This means that we cannot treat the case of $1, 1$-feasibility. However we have seen that $1, 1$-feasibility is easy by other arguments.

- Q is a finite set of *states*, $I \subseteq Q$ is the set of *initial states*,
- A is the alphabet,
- $\Delta \subseteq Q \times A \times Q \times Q$ is the *transition relation*,
- $col : Q \to Cols$ is a *colour mapping* to some finite set $Cols$ of *colours*.

A *run* ρ of an automaton over an A-tree t is a Q-tree such that $\rho(\varepsilon) \in I$ and for all $v \in \{0,1\}^*$, $(\rho(v), t(v), \rho(v0), \rho(v1)) \in \Delta$. The *Cols*-tree $col(\rho)$ denotes $(col \circ \rho)$. Each automaton – depending on its nature – comes with a mapping *val* from $Cols^\omega$ to $\omega + 1$. The *value* $val(\rho)$ of a run ρ is the supremum of $val(\beta)$ over all branches β of $col(\rho)$. The *value* $\mathcal{A}(t)$ of an A-tree t is the minimum of $val(\rho)$ for ρ ranging over all runs over t (by default ω if there is no such run).

We are now ready to introduce the different value mappings used throughout the paper. The *parity condition* corresponds to an interval $[i,j]$ of natural numbers – called *priorities* – as set of colours. Given an infinite sequence of priorities $u \in [i,j]^\omega$, $val(u)$ is 0 if the maximal priority appearing infinitely often is even, else it is ω. A *parity automaton* is an automaton using a parity condition. A *tree* t is accepted by such an automaton iff $\mathcal{A}(t) = 0$. The *parity index* (or Mostowski index) of a parity automaton is the pair (i,j) of the minimal and maximal priorities used in the automaton. We designate by $L(\mathcal{A})$ the set of trees that are accepted by the parity automaton $\mathcal{A}$. A language is *regular* if it is equal to $L(\mathcal{A})$ for some parity automaton $\mathcal{A}$, and is i,j-*feasible* if furthermore $\mathcal{A}$ has parity index (i,j).

A *distance condition* (corresponding to nested distance desert automata in [7], and hierarchical B-automata in [3]) is defined for a totally ordered set of colours $D = \{d_1, r_1, \ldots, d_k, r_k\}$ with the order $d_1 < r_1 < \cdots < d_k < r_k$. The colours $d_1, \ldots, d_k$ are called *distance colours*, while the colours $r_1, \ldots, r_k$ are *reset colours*. Given an infinite sequence $u \in D^\omega$, its value $val(u)$ is the supremum of $|v|_{d_i}$ where v ranges over all finite factors of u such that the maximal colour of v is d_i. One can see this as having k counters numbered from 1 to k. When seeing d_i the corresponding counter is incremented and all the counters below are reset. When seeing r_i, all the counters up to i are reset. The value of a sequence is the supremum of all values of counters seen during this process (starting from 0).

We can derive from the two previous mappings a last one, the *distance-parity mapping*, which can be seen as a conjunction of a distance and a parity condition. It is described for a set of colours of the form $Cols = D \times [i,j]$ where D is the ordered set of colours of a distance condition and $[i,j]$ is a finite interval of natural numbers. For a sequence $u \in Cols^\omega$ one derives the two corresponding sequences $u_1 \in D^\omega$ and $u_2 \in [i,j]^\omega$ obtained by projection to the first and second components of the elements in u, respectively. The value $val(u)$ is the maximum of $val(u_1)$ (as a distance condition) and $val(u_2)$ (as a parity condition).

Given a distance-parity automaton $\mathcal{A}$, we define for each $N < \omega$ the language $L^{(N)}(\mathcal{A}) := \{t : \mathcal{A}(t) \leq N\}$. In this way $\mathcal{A}$ defines a non-decreasing ω-sequence of languages, i.e., an ω-chain of languages. It is easy to observe that for each N, $L^{(N)}(\mathcal{A})$ is a regular language: For a fixed N the counters for the distance colours can be coded in the states of the automaton. This construction gives a parity automaton of index (i,j) for $L^{(N)}(\mathcal{A})$, hence we define the *parity index*

therefore, for the value to be finite, the parity condition has to be satisfied, and the distance condition has to be finite.

of a distance-parity automaton to be the parity index of the underlying parity condition. From the above explanation we easily conclude the following.

Fact 1. *Given a distance-parity automaton $\mathcal{A}$ of parity index (i, j) and a natural number N, the language $L^{(N)}(\mathcal{A})$ is i, j-feasible.*

The *limitedness problem* is the following: given a distance-parity automaton $\mathcal{A}$, determine if $L^{(N)}(\mathcal{A})$ is ultimately constant (in this case, the automaton is said to be *limited*). In this paper we prefer the *uniform universality problem*: given an automaton $\mathcal{A}$, determine if $L^{(N)}(\mathcal{A})$ is equal to the set of all trees for some natural number N (in this case the automaton is said to be *uniformly universal*).

3   Guidable Automata

As mentioned in the introduction, the first step of the proof is to construct a so-called guidable automaton accepting the language L. The underlying idea is that we want to be able to relate accepting runs of an automaton for L that has a minimal number of priorities to accepting runs of our guidable automaton.
It simulates every automaton recognising a sublanguage

Definition 1. *A parity automaton $\mathcal{A} = (Q_{\mathcal{A}}, A, \{q_0\}, \Delta_{\mathcal{A}}, col_{\mathcal{A}})$ is guidable if for every parity automaton $\mathcal{B} = (Q_{\mathcal{B}}, A, I_{\mathcal{B}}, \Delta_{\mathcal{B}}, col_{\mathcal{B}})$ such that $L(\mathcal{B}) \subseteq L(\mathcal{A})$ there exist a mapping $g : Q_{\mathcal{A}} \times \Delta_B \to \Delta_{\mathcal{A}}$ with the following properties:*

- *$g(p, (q, a, q', q'')) = (p, a, p', p'')$ for some $p', p'' \in Q_{\mathcal{A}}$.*
- *For every accepting run ρ of $\mathcal{B}$ over a tree t, $g(\rho)$ is an accepting run of $\mathcal{A}$ over t, where $g(\rho) = \rho'$ is the unique run such that $\rho'(\varepsilon) = q_0$, and for all $u \in \{0, 1\}^*$:*

$$(\rho'(u), t(u), \rho'(u0), \rho'(u1)) = g(\rho'(u), (\rho(u), t(u), \rho(u0), \rho(u1))) .$$

In this case we say that $(\mathcal{B}, g)$ guides $\mathcal{A}$.

One way to see this definition is that g is a deterministic transducer with state set $Q_{\mathcal{A}}$ that takes as input a run ρ of $\mathcal{B}$ and outputs a run ρ' of $\mathcal{A}$ such that if ρ is accepting (for $\mathcal{B}$), then ρ' is accepting (for $\mathcal{A}$).
   An example of an automaton over finite words that is not guidable (the definition of guidable automaton can easily be translated by the reader to the case of finite words) is the automaton that accepts all $\{a, b\}$-words by guessing in the first step of the run if the last letter is a or b, and then proceeds to two subautomata, one accepting words ending with a, the other accepting words ending with b. It is quite clear that it is not possible to guide such an automaton. This example carries the important intuition that an automaton is guidable if it is never forced to make an unnecessary guess concerning the remaining input. Also consistent with this intuition, note that every deterministic automaton is obviously guidable.
   In our context of infinite trees, the only way we use the property of being guidable is by the following *simultaneous pumping* argument.

Lemma 2. *Suppose* $(\mathcal{B}, g)$ *guides* $\mathcal{A}$ *and consider accepting runs* ρ, ρ' *as in Definition 1. Let* $u \sqsubset v$ *be nodes such that* $\rho(u) = \rho(v)$ *and* $\rho'(u) = \rho'(v)$. *If the maximal priority in* ρ *between* u *and* v *is even, then the maximal priority in* ρ' *between* u *and* v *is also even.*

Proof. Consider the run τ obtained from ρ by repeating infinitely often the part between u and v (i.e., positions x such that $u \sqsubseteq x$ and $v \not\sqsubseteq x$), and τ' be obtained from ρ' in the same way. If the maximal priority between u and v in ρ is even, then the run τ is accepting. But it is not difficult to see that $g(\tau) = \tau'$ and hence by definition of guidable automata, τ' is also accepting. Thus, the maximal priority n appearing infinitely often on the infinite branch obtained by pumping is even. Since this branch is obtained by pumping ρ' between u and v, the priority n appears as the maximal one between u and v in ρ'. $\qquad\square$

The main result of this section is that for each regular tree language we can construct a guidable automaton. (this procedure is doubly exponential)

Theorem 1. *For each regular tree language* L *there exists effectively a guidable parity automaton* $\mathcal{A}$ *accepting* L.

Proof. We start from a parity automaton $\mathcal{C} = (Q_{\mathcal{C}}, A, I_{\mathcal{C}}, \Delta_{\mathcal{C}}, col_{\mathcal{C}})$ accepting the complement of L. What we show is that applying a standard complementation procedure to $\mathcal{C}$ – as it can be found, e.g., in [16] – yields a guidable automaton $\mathcal{A}$ (which obviously accepts L). The rough idea why the automaton is guidable is that a positional winning strategy in a game witnessing that an automaton $\mathcal{B}$ has empty intersection with $\mathcal{C}$ can be rewritten into a mapping g which guides $\mathcal{A}$. This is a reasonable approach because both the mapping g and the winning strategy in the above game witness that $L(\mathcal{B}) \subseteq L(\mathcal{A})$.

We now formally describe the automaton $\mathcal{A}$ and then show that it is guidable. By G we denote the set of all mappings $f : \Delta_{\mathcal{C}} \to \{0, 1\}$. Each path in an $A \times G$ labelled tree corresponds to an infinite sequence over $(A \times G \times \{0, 1\})$, where the $\{0, 1\}$-component indicates the direction the path takes in each step. We say that such a sequence

$$(a_0, f_0, i_0)(a_1, f_1, i_1) \cdots \in (A \times G \times \{0, 1\})^{\omega}$$

is $\mathcal{C}$-accepting if there is a transition sequence $\tau_0 \tau_1 \cdots \in \Delta_{\mathcal{C}}^{\omega}$ such that

- for each j the transition τ_j is of the form $(q_j, a_j, q_j^{(0)}, q_j^{(1)})$ with $f_j(\tau_j) = i_j$ and $q_j^{(i_j)} = q_{j+1}$,
- q_0 is an initial state of $\mathcal{C}$, and
- the acceptance condition of $\mathcal{C}$ is satisfied by $q_0 q_1 \cdots$.

It is easy to see that the set of all $\mathcal{C}$-accepting sequences over $(A \times G \times \{0, 1\})$ is a regular language of infinite words (a non-deterministic automaton can guess the transition sequence of $\mathcal{C}$ and verify the local properties). Hence, the set of all sequences that are not $\mathcal{C}$-accepting is also a regular language, and from a deterministic parity word automaton for this language one constructs a deterministic parity tree automaton (with a single initial state) $\mathcal{A}' = (Q_{\mathcal{A}}, A \times$

description of the standard complementation construction for parity tree automata

i.e. trees and rejecting Spoiler's strategies

$G, \{q_I^A\}, \Delta_{A'}, col_A)$ accepting all $(A \times G)$-trees in which all paths correspond to non-C-accepting sequences. By projecting away the G-component one obtains the automaton $A = (Q_A, A, \{q_I^A\}, \Delta_A, col_A)$ for the language L. Note that A' and A only differ on the inputs in the transit Spoiler's strategy is guessed stepwise

We show that A is indeed a guidable automaton. Let $B = (Q_B, A, I_B, \Delta_B, col_B)$ be a tree automaton with $L(B) \subseteq L(A)$. The mapping g is constructed from a strategy in the emptiness game for the language $L(B) \cap L(C)$. In this game, Adam wants to verify that $L(B) \cap L(C) \neq \emptyset$ and Eve wants to show the contrary. In other words, Adam plays for constructing both a run of B and a run of C corresponding to the same tree, while Eve wants to show the failure of this construction by witnessing an invalid branch (rejectin for B or for C). The rules of the game are as follows:

1. Adam starts by choosing a starting position $(p_I, q_I) \in I_B \times I_C$.
2. From a position $(p, q) \in Q_B \times Q_C$, Adam picks two transitions $(p, a, p^{(0)}, p^{(1)}) \in \Delta_B$ and $(q, a, q^{(0)}, q^{(1)}) \in \Delta_C$ from these states (both using the same input letter a). The game is now in position $((p, a, p^{(0)}, p^{(1)}), (q, a, q^{(0)}, q^{(1)}))$.
3. Eve chooses a direction $i \in \{0, 1\}$ and the game moves to $(p^{(i)}, q^{(i)})$.

The result of the game (the part that is interesting for the winning condition) is an infinite sequence $(p_0, q_0)(p_1, q_1) \cdots \in (Q_B \times Q_C)^\omega$. Eve wins if either $p_0 p_1 \cdots$ does not satisfy the parity condition of B or $q_0 q_1 \cdots$ does not satisfy the parity condition of C.

This is the standard game for verifying emptiness of tree automata (see [16]) and Eve has a winning strategy iff $L(B) \cap L(C) = \emptyset$. Because $L(B) \subseteq L$, we know that Eve has indeed a winning strategy (C accepts the complement of L).

The winning condition for Eve is the disjunction of two parity conditions and hence can be written as a Rabin condition. Therefore Eve has a positional winning strategy (see [8,19]). Such a positional winning strategy is a mapping $\sigma_E : \Delta_B \times \Delta_C \to \{0, 1\}$ assigning to each pair of transitions a direction (for the transition pairs that do not correspond to valid game positions, an arbitrary value can be chosen). It can be equivalently written as a mapping $\sigma_E : \Delta_B \to (\Delta_C \to \{0, 1\})$ assigning to each B-transition a mapping from the set of C-transitions to $\{0, 1\}$. From this we first define $g' : Q_A \times \Delta_B \to \Delta_{A'}$ by $g'(p, (q, a, q', q'')) = (p, (a, f), p', p'')$ for the unique $p', p'' \in Q_A$ such that $(p, (a, f), p', p'') \in \Delta_{A'}$ with $f = \sigma_E(q, a, q', q'')$. The mapping g is then obtained from g' by projecting away the G-component.

We need now to show that g translates accepting B-runs into accepting A-runs. Let ρ be an accepting run of B on some tree t. Assume that $g(\rho)$ is rejecting. Then also the run $g'(\rho)$ is rejecting, where $g'(\rho)$ is a run over the input tree t' that is obtained from t by adding the G-components of the transitions used in $g'(\rho)$ to the node labels of t. This means that there is an infinite branch $i_0 i_1 \cdots$ with labels $(a_0, f_0)(a_1, f_1) \cdots$ in t' such that the sequence $(a_0, f_0, i_0)(a_1, f_1, i_1) \cdots$ is C-accepting. Let $\tau_0 \tau_1 \cdots \in \Delta_C^\omega$ be the transition sequence from the definition of C-acceptance.

Let $\tau'_0 \tau'_1 \cdots \in \Delta_B^\omega$ be the transition sequence of B along the path $i_0 i_1 \cdots$ in the run ρ. Now assume that in the emptiness game described above Eve

plays according to σ_E. One can verify that Adam can play the transition pairs $(\tau_0', \tau_0), (\tau_1', \tau_1), \dots$ against σ_E (because $\sigma_E(\tau_j', \tau_j) = f_j(\tau_j) = i_j$) and thus wins against σ_E. This contradicts the choice of σ_E as a winning strategy for Eve. $\square$

4   Reduction from Parity Rank to Uniform Universality

In this section we describe how to reduce the problem of deciding whether a regular tree language is i, j-feasible to the problem of deciding the uniform universality of a distance-parity automaton. We fix a regular language L and an interval $[i, j]$ of natural numbers. We also fix a guidable parity automaton $\mathcal{A} = (Q, A, q_0, \delta, col)$ with $L(\mathcal{A}) = L$ using priorities in some interval P. In the following, we often identify runs ρ of $\mathcal{A}$ with their colouring $col(\rho)$, i.e., with P-trees.

The idea is to construct a distance-parity automaton $\mathcal{T}_{i,j}$ of parity index (i, j) reading P-trees such that:

(Correctness). For all P-trees t, if $\mathcal{T}_{i,j}(t) < \omega$ then t is accepting (Lemma 4).
(Completeness). For every i, j-feasible language $K \subseteq L$, there exists $M \in \omega$ such that for all trees $t \in K$, there exists an accepting run ρ of $\mathcal{A}$ over t with $\mathcal{T}_{i,j}(col(\rho)) \le M$ (Lemma 5).

Then we can cascade the automata $\mathcal{T}_{i,j}$ and $\mathcal{A}$ into a single one denoted $\mathcal{A}_{i,j}$ using the same distance-parity condition as $\mathcal{T}_{i,j}$: this automaton guesses a run of $\mathcal{A}$ and applies the automaton $\mathcal{T}_{i,j}$ on the colouring of this run. This resulting distance-parity automaton is such that $L^{(N)}(\mathcal{A}_{i,j}) \subseteq L$ for all N (correctness), and such that for all i, j-feasible $K \subseteq L$, $K \subseteq L^{(M)}(\mathcal{A}_{i,j})$ for some M (completeness). Let us now construct a distance-parity automaton $\mathcal{U}_{i,j}$ which at the beginning non-deterministically decides either to execute the automaton $\mathcal{A}_{i,j}$ or an automaton $\mathcal{C}$ accepting the complement of L.

Lemma 3. *The automaton $\mathcal{U}_{i,j}$ is uniformly universal iff L is i, j-feasible.*

Proof. Assume that $L^{(N)}(\mathcal{U}_{i,j})$ contains all trees for some N. This means that $L^{(N)}(\mathcal{A}_{i,j}) \cup L(\mathcal{C})$ contains all trees, and thus $L \subseteq L^{(N)}(\mathcal{A}_{i,j})$. Since furthermore $L^{(N)}(\mathcal{A}_{i,j}) \subseteq L$ (correctness), we have $L = L^{(N)}(\mathcal{A}_{i,j})$. And by Fact 1, L is i, j-feasible. Conversely, assume that L is i, j-feasible. Then by completeness, there exists M such that $L \subseteq L^{(M)}(\mathcal{A}_{i,j})$. Hence $L \cup L(\mathcal{C}) \subseteq L^{(M)}(\mathcal{U}_{i,j})$ contains all trees. The automaton $\mathcal{U}_{i,j}$ is uniformly universal. $\square$

From this we directly get our main theorem:

Theorem 2. *The problem of deciding the i, j-feasibility of a regular tree language is reducible to the uniform universality of distance-parity automata.*

We now describe the automaton $\mathcal{T}_{i,j}$ in detail. Intuitively, $\mathcal{T}_{i,j}$ maps each priority in P to a priority in $[i, j]$. When reading a priority $k \in P$ it "outputs" the priority associated to k. To implement this idea, the main objects used by $\mathcal{T}_{i,j}$ are partial mappings from P to $[i, j]$ (for technical reasons we allow some priorities to be undefined). To ensure correctness of $\mathcal{T}_{i,j}$ (in the sense mentioned above), these

mappings have to respect some conditions. First of all, <mark>odd priorities of P should be mapped to odd priorities of $[i, j]$</mark>. Second, the ordering of the priorities should be respected in the following sense: <mark>the image of every odd priority is required to dominate the image of all even priorities below it</mark>.

Formally, let S be the set of partial mappings $s : P \to [i, j]$ such that for all k for which $s(k)$ is defined:

1. If k is odd, then $s(k)$ is odd.
2. If k is even, then for all odd $l > k$ such that $s(l)$ is defined, $s(l) \geq s(k)$.

The transitions of $\mathcal{T}_{i,j}$ allow to change the mapping s in a safe way. It is not very difficult to observe that on reading priority $k \in P$, $\mathcal{T}_{i,j}$ can safely change the values for P-priorities strictly below k. Additionally, we also allow a bounded number of arbitrary changes of the values. This bound is not fixed a priori and is controlled by the distance condition.

We now proceed to the formal definition of the distance-parity automaton $\mathcal{T}_{i,j} = (S \times P \times S, P, S_I, \delta, col)$.

- The set of initial states is $S_I = \{\langle s_\perp, k, s \rangle \mid s \in S, \ k \in P\}$, where $s_\perp$ is the mapping that is undefined everywhere.
- For all $s', s, s_0, s_1 \in S$, $k, k_0, k_1 \in P$ we allow the transition

$$\big(\langle s', k, s \rangle, \ k, \ (0, \langle s, k_0, s_0 \rangle), \ (1, \langle s, k_1, s_1 \rangle) \big) \ .$$

  In fact, every transition is allowed, provided the first component is equal to the last one used at the parent node, and the second component remembers the label of the P-tree at the current node. As a consequence, a run of $\mathcal{T}_{i,j}$ on a given input tree is completely determined by the tree and the last components of its states.
- Distance colours are $D = \{d_k, r_{k-1} \ : \ k \in P\}$, and the priorities lie in $[i, j]$.
- For all $q \in S \times P \times S$ we set $col(q) = (dst(q), pri(q))$ with:

$$dst(\langle s', k, s \rangle) = \begin{cases} r_{k-1} & \text{if } s(k) \text{ is defined and } s'(l) = s(l) \text{ for all } l \geq k \\ d_k & \text{if } s(k) \text{ is undefined and } s'(l) = s(l) \text{ for all } l \geq k \\ d_l & \text{for the maximal } l \text{ with } s'(l) \neq s(l) \text{ otherwise.} \end{cases}$$

$$pri(\langle s', k, s \rangle) = \begin{cases} s(k) & \text{if } s(k) \text{ is defined,} \\ i & \text{if } s(k) \text{ is undefined.} \end{cases}$$

We now establish that $\mathcal{T}_{i,j}$ satisfies the correctness condition mentioned at the beginning of this section.

Lemma 4 (correctness). *For all P-trees t, if $\mathcal{T}_{i,j}(t) < \omega$ then t satisfies the parity condition of P on each branch.*

Proof. Let t be a P-tree and let ρ be a run of $\mathcal{T}_{i,j}$ on t. Consider a branch B and let k be the maximal P-priority that appears infinitely often on B in t. We show that if k is odd, then B is rejecting in ρ.

In the following, the s-value of a priority in P refers to the value it is mapped to by the sates of $T_{i,j}$. First note that r_{k-1} is the maximal release that can occur infinitely often on B in ρ. Hence, if the s-value of some $l \geq k$ changes infinitely often along B, then the distance condition is not satisfied and B is rejecting. Otherwise, the s-value of k becomes stable eventually on B. This value is odd because k is odd (property 1 of S), and furthermore it is bigger than the s-values of the smaller even priorities (property 2 of S). Hence, the maximal priority assumed infinitely often in ρ along B is odd and thus B is rejecting in ρ.   □

Lemma 5 (completeness). *For every i, j-feasible language $K \subseteq L$, there exists $M \in \omega$ such that for all trees $t \in K$, there exists an accepting run ρ of $\mathcal{A}$ over t with $T_{i,j}(col(\rho)) \leq M$.*

This is the difficult part of the proof. The principle is the following. Consider an automaton $\mathcal{B}$ of parity index (i, j) that accepts K and a mapping g such that $(\mathcal{B}, g)$ guides $\mathcal{A}$ (this is possible since $\mathcal{A}$ is guidable). Let us consider an accepting run τ of $\mathcal{B}$ over a tree $t \in K$. We set ρ to be $g(\tau)$, and we aim at constructing a run of $T_{i,j}$ witnessing that $T_{i,j}(col(\rho)) \leq M$ for a bound M which does not depend on t (but depends on $\mathcal{B}$).

The very informal idea for constructing the run of $T_{i,j}$ is to try to mimic the priorities used by the run τ. For defining the states of $T_{i,j}$ used at each position of the run, we heavily rely on Lemma 2 which relates the use of priorities in τ to the use of priorities in ρ. This lemma is only usable in presence of loops of $\mathcal{B}$ in τ. Therefore, in parts of τ where $\mathcal{B}$ has not yet entered a loop or enters a new strongly connected component of its transition graph, the states of $T_{i,j}$ map some priorities to an undefined value. The distance part of the condition is exactly used as a counter of such kind of "errors". But as $\mathcal{B}$ can only finitely often change the strongly connected component, one can imagine that the number of those errors is bounded by some M depending solely on the size of $\mathcal{B}$.

5   Conclusion

We have shown in this paper that the problem of deciding the levels of the non-deterministic Mostowski hierarchy can be reduced to the problem of uniform universality for distance-parity automata. The next step is of course to show the decidability of this latter problem. We have already obtained partial results showing that uniform universality is decidable for special classes of distance-parity automata (over trees). We expect the decidability of the general problem.

A key tool in our reduction is the notion of guidable automaton. We have shown that each regular language of infinite trees can be accepted by such an automaton. This model is interesting in its own right because it somehow shows that there is a canonical way of using non-determinism for accepting a language of infinite trees. We plan to investigate this model further and to see if it can be applied in other contexts.

References

1. Arnold, A.: The μ-calculus alternation-depth hierarchy is strict on binary trees. Informatique Théorique et Applications 33(4/5), 329–340 (1999)
2. Arnold, A., Duparc, J., Murlak, F., Niwiński, D.: On the topological complexity of tree languages. In: Flum, J., Grädel, E., Wilke, T. (eds.) Logic and automata: History and Perspectives, pp. 9–28. Amsterdam University Press (2007)
3. Bojańczyk, M., Colcombet, T.: Bounds in ω-regularity. In: Proceedings of LICS 2006, pp. 285–296. IEEE Computer Society Press, Los Alamitos (2006)
4. Bradfield, J.C.: The modal μ-calculus alternation hierarchy is strict. Theor. Comput. Sci. 195(2), 133–153 (1998)
5. Carayol, A., Löding, C.: MSO on the infinite binary tree: Choice and order. In: Duparc, J., Henzinger, T.A. (eds.) CSL 2007. LNCS, vol. 4646, pp. 161–176. Springer, Heidelberg (2007)
6. Hashiguchi, K.: Algorithms for determining relative star height and star height. Inf. Comput. 78(2), 124–169 (1988)
7. Kirsten, D.: Distance desert automata and the star height problem. RAIRO – Theoretical Informatics and Applications 3(39), 455–509 (2005)
8. Klarlund, N.: Progress measures, immediate determinacy, and a subset construction for tree automata. Annals of Pure and Applied Logic 69(2–3), 243–268 (1994)
9. Lenzi, G.: A hierarchy theorem for the μ-calculus. In: Meyer auf der Heide, F., Monien, B. (eds.) ICALP 1996. LNCS, vol. 1099, pp. 87–97. Springer, Heidelberg (1996)
10. Mostowski, A.W.: Regular expressions for infinite trees and a standard form of automata. In: Skowron, A. (ed.) SCT 1984. LNCS, vol. 208, pp. 157–168. Springer, Heidelberg (1985)
11. Niwiński, D.: On fixed-point clones. In: Kott, L. (ed.) ICALP 1986. LNCS, vol. 226, pp. 464–473. Springer, Heidelberg (1986)
12. Niwiński, D., Walukiewicz, I.: Ambiguity problem for automata on infinite trees (unpublished note)
13. Niwiński, D., Walukiewicz, I.: Relating hierarchies of word and tree automata. In: Morvan, M., Meinel, C., Krob, D. (eds.) STACS 1998. LNCS, vol. 1373, pp. 320–331. Springer, Heidelberg (1998)
14. Niwiński, D., Walukiewicz, I.: Deciding nondeterministic hierarchy of deterministic tree automata. Electr. Notes Theor. Comput. Sci. 123, 195–208 (2005)
15. Rabin, M.O.: Decidability of second-order theories and automata on infinite trees. Transactions of the American Mathematical Society 141, 1–35 (1969)
16. Thomas, W.: Languages, automata, and logic. In: Handbook of Formal Language Theory, vol. III, pp. 389–455. Springer, Heidelberg (1997)
17. Urbański, T.F.: On deciding if deterministic Rabin language is in Büchi class. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 663–674. Springer, Heidelberg (2000)
18. Wagner, K.W.: Eine topologische Charakterisierung einiger Klassen regulärer Folgenmengen. J. Inf. Process. Cybern. EIK 13(9), 473–487 (1977)
19. Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theoretical Computer Science 200(1–2), 135–183 (1998)