The non-deterministic Mostowski hierarchy and distance-parity automata

Thomas Colcombet and Christof Löding

LIAFA/CNRS, France and RWTH Aachen, Germany
Submitted to Track B of ICALP'08

Abstract. Given a Rabin tree-language and natural numbers i, j, the language is said 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 [14] 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 accepting condition. This accepting condition allows to determine, given a run of the automaton, whether it should be considered as accepting or not. Different (often equivalent) choices of accepting conditions are known from the literature such as Büchi, Rabin, Muller, or Streett conditions (cf. [15]). Though all possess their own interest, the notion of *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 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 can restrict ourselves to $i = 0$ or $i = 1$. It is also clear that the bigger is the interval $[i, j]$, the more tree languages are i, j-feasible. Mostowski first studied this parameter [9],

and the corresponding ladder-shaped hierarchy – depicted Figure 1 – is named after him.

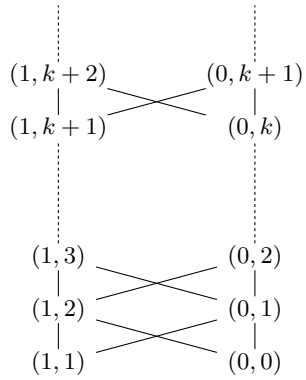

Fig. 1. Hierarchy of Mostowski indices.

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 [17] (even over words), non-deterministic [11], and alternating [1] (in combination with [3, 8]). 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. Remark 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 [12] (see also [13] for more details). The problem is also known to be decidable if the input language is deterministic, and the output class is non-deterministic [13] (which is a refinement of the case of a path languages as input [12]). 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 [16]. 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 [6] (the problem was originally solved by Hashiguchi [5]). 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 semantic of such automata is either to reject a word, or to associate to it a natural number, 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 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 [2]. The limitedness problem still makes sense for the distance-parity automata, but we prefer to it the uniform universality problem[1]: 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 [6], and can be derived decidable over infinite words using [2].

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).

[1] In fact Kirsten's proof can be slightly simplified by using the uniform universality problem instead of the limitedness one. It avoids the value ∞ used in matrices.

– 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 [10] but has not been published; see also [4]). 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 both correct, and optimal.

The remainder of the paper is organised as follows. Section 2 is devoted to definitions, in particular automata and the accepting conditions we use. Section 4 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 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 letter a in u. An ω-word is an infinite sequence of letters and by A^ω we denote the set of infinite words over alphabet 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)[2]. A *tree* labelled by a finite *alphabet* A (we also say an A-tree) is

[2] 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 for other arguments.

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 a ω-word in A^ω. It is sometime convenient to identify branches with this ω-word.

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

- 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 mapping from $\{0,1\}^*$ to Q such that $\rho(\varepsilon) \in I$ and for all $v \in \{0,1\}^*$, $(\rho(v), t(v), \rho(v0), \rho(v1)) \in \delta$. Given a run ρ, $col(\rho)$ is the $Cols$-tree $(col \circ \rho)$.

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 over all infinite branches 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 correspond to a 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 ω. A *parity automaton* is an automaton using a parity condition. A *tree* t is accepted by such an automaton iff $val(t) = 0$. The *parity index* (or Mostowski index) of a parity automaton is the pair (i, j) of maximal and minimal 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* (usually called nested distance desert) is defined for a totally ordered set of colours D of the following structure:

$$d_1 < r_1 < \cdots < d_k < r_k \ .$$

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 $|v|_{d_k}$ for v ranging over finite factors of u of maximal colour d_k. One can see this as having k counters numbered from 1 to k. When seeing d_k the corresponding counter is incremented and all the counters below are reset. When seeing r_k, all the counters up to k are reset. The value of a sequence is the supremum of all values of counters seen during this process (starting with all counters set to 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* 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 *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 *uniformly universal*).

3 Guidable automata

As mentioned in our introduction, the first step of the proof is to construct an automaton accepting the language L that has a special property, called guidable. 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.

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 $g : Q_\mathcal{A} \times \delta_B \to \delta_A$ 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 $\mathcal{A}$).

Let us provide some intuition on guidable automata in the context of finite words, even if those explanations are not necessary in the proofs of the paper.

The definition of guidable automaton can easily be translated by the reader to the case of finite words.

An example of an automaton that is not guidable 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, though this automaton is unambiguous[3] if we choose the two subautomata to be deterministic. In fact, this example carries the important intuition that an automaton is guidable if it is never forced to make an unnecessary guess concerning the remaining of the input.

It is easy to see that every deterministic automaton is guidable (even over infinite trees). The example above shows that unambiguous automata may not be guidable. It is also easy to construct an ambiguous automaton that is guidable (for instance an automaton of two states accepting all inputs and containing all possible transitions). This shows that the concept of guidable automata is of a different nature than the one of unambiguous automata.

In our context of infinite trees, the only way we use the property of guidable automata is by the following *simultaneous pumping* argument.

Lemma 2. *Consider automata $\mathcal{A}$, $\mathcal{B}$ and 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 piece of run 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 also 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'. □

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

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

Proof. What show that a standard complementation procedure, as it can be found in [15], yields a guidable automaton $\mathcal{A}$. We start by describing the automaton resulting from this complementation procedure and then show why it is guidable. Roughly, the idea is that a strategy in a game witnessing that an automaton $\mathcal{B}$ has empty intersection with the complement of $\mathcal{A}$ can be used to construct the mapping g to guide $\mathcal{A}$.

[3] Unambiguous automata have a unique accepting run for each word in the language.

Formally, let $\mathcal{C} = (Q_\mathcal{C}, A, I_\mathcal{C}, \delta_\mathcal{C}, col_\mathcal{C})$ be a parity automaton accepting the complement of L. 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 $q_0q_1 \cdots$.

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) $\mathcal{A}' = (Q_\mathcal{A}, A \times G, \{q_I^\mathcal{A}\}, \delta_{\mathcal{A}'}, col_\mathcal{A})$ accepting all $(A \times G)$-trees in which all paths correspond to $\mathcal{C}$-accepting sequences. By projecting away the G-component one obtains the automaton $\mathcal{A} = (Q_\mathcal{A}, A, \{q_I^\mathcal{A}\}, \delta_\mathcal{A}, col_\mathcal{A})$ for the language L. Note that $\mathcal{A}'$ and $\mathcal{A}$ only differ on the inputs in the transitions.

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

1. Adam starts by choosing a starting position $(p_I, q_I) \in I_\mathcal{B} \times I_\mathcal{C}$.
2. From a position $(p, q) \in Q_\mathcal{B} \times Q_\mathcal{C}$, Adam picks two transitions $(p, a, p^{(0)}, p^{(1)}) \in \delta_\mathcal{B}$ and $(q, a, q^{(0)}, q^{(1)}) \in \delta_\mathcal{C}$ from these states, both using the same input letter. 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 $(p_0, q_0)(p_1, q_1) \cdots \in (Q_\mathcal{B} \times Q_\mathcal{C})^\omega$. Eve wins if either $p_0p_1 \cdots$ does not satisfy the parity condition of $\mathcal{B}$ or $q_0q_1 \cdots$ does not satisfy the parity condition of $\mathcal{C}$.

This is the standard game for verifying emptiness of tree automata (see [15]) and Eve has a winning strategy iff $L(\mathcal{B}) \cap L(\mathcal{C}) = \emptyset$. Because $L(\mathcal{B}) \subseteq L$, we know that Eve indeed has a winning strategy ($\mathcal{C}$ 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 [7, 18]). Such a positional winning strategy is a mapping $\sigma_E : \delta_{\mathcal{B}} \times \delta_{\mathcal{C}} \to \{0, 1\}$ 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 $\sigma_E : \delta_{\mathcal{B}} \to (\delta_{\mathcal{C}} \to \{0, 1\})$ assigning to each $\mathcal{B}$-transition a mapping from the set of $\mathcal{C}$-transitions to $\{0, 1\}$. From this we first define $g' : Q_{\mathcal{A}} \times \delta_{\mathcal{B}} \to \delta_{\mathcal{A}'}$ by $g(p, (q, a, q', q'')) = (p, (a, f), p', p'')$ for the unique $p', p'' \in Q_{\mathcal{A}}$ such that $(p, (a, f), p', p'') \in \delta_{\mathcal{A}'}$ with $f = \sigma_E(q, a, q', q'')$. The mapping g is then obtained by projecting away the G-component.

We need now to show that g translates accepting $\mathcal{B}$-runs into accepting $\mathcal{A}$-runs. Let ρ be an accepting run of $\mathcal{B}$ 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 $\mathcal{C}$-accepting. Let $\tau_0 \tau_1 \cdots \in \delta_{\mathcal{C}}^{\omega}$ be the transition sequence from the definition of $\mathcal{C}$-acceptance.

Let $\tau'_0 \tau'_1 \cdots \in \delta_{\mathcal{B}}^{\omega}$ be the transition sequence of $\mathcal{B}$ 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 $(\tau'_0, \tau_0), (\tau'_1, \tau_1), \ldots$ 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. □

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 $T_{i,j}$ of parity index (i, j) reading P-trees such that:

(Correctness) For all P-trees t, if $T_{i,j}(t) < \omega$ 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 $T_{i,j}(col(\rho)) \leq M$ (Lemma 5).

Then we can cascade the automata $T_{i,j}$ and $\mathcal{A}$ into a single one noted $\mathcal{A}_{i,j}$ using the same distance-parity condition as $T_{i,j}$: this automaton non-deterministically chooses a run of $\mathcal{A}$ and applies the $T_{i,j}$ automaton 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 the distance-parity automaton $\mathcal{U}$ 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 $\mathcal{U}$ is uniformly universal iff L is i,j-feasible.*

Proof. Assume $L^{(N)}(\mathcal{U})$ accepts 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 L is i,j-feasible. Then by completeness, there exists M such that $L \subseteq L^{(M)}(\mathcal{A}_{i,j})$. Hence $L \cup L(\mathcal{C}) \subseteq L^{(M)}(\mathcal{U})$ contains all trees. The automaton $\mathcal{U}$ is uniformly universal. □

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 $T_{i,j}$ in detail. Intuitively, $T_{i,j}$ associates to each priority in P 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 $T_{i,j}$ are partial mappings from P to $[i,j]$ (for technical reasons we allow some priorities to be undefined). To ensure correctness of $T_{i,j}$ (in the sense mentioned above), these mappings have to respect some conditions. First of all, odd priorities of P should be mapped to odd priorities of $[i,j]$. Furthermore, the ordering of the priorities should be respected. However, we relax this second condition by only requiring that the image of every odd priority dominates the image of all even priorities below it.

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 $T_{i,j}$ allow to change the mapping s in a safe way. It is not very difficult to observe that on reading priority $k \in P$, $T_{i,j}$ can safely change the values for P-priorities strictly below P. 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 additional distance condition.

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

- The set of initial states is $S_I = \{(s_\perp, k, s) \mid s \in S,\ k \in P\}$, 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

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

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 $T_{i,j}$ 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:

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

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

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

Lemma 4 (correctness). *For all P-trees t, if $T_{i,j}(t) < \omega$ 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 $T_{i,j}$ 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.

First note that r_{k-1} is the maximal release that can occur infinitely often on B in ρ. Hence, if the value of some $l \geq k$ changes infinitely often along B, then the distance condition is not satisfied and B is rejecting. Otherwise, the 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 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 (given in the appendix). The principle is the following. Consider an automaton $\mathcal{B}$ of parity index (i, j) that accepts K. There exists 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 $t \in K$. The run ρ we consider is $g(\tau)$. The difficulty is to construct 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 6 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 "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 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. André Arnold. The μ-calculus alternation-depth hierarchy is strict on binary trees. *Informatique Théorique et Applications*, 33(4/5):329–340, 1999.
2. Mikolaj Bojanczyk and Thomas Colcombet. Bounds in ω-regularity. In *Proceedings of LICS 2006*, pages 285–296. IEEE Computer Society Press, 2006.
3. Julian C. Bradfield. The modal μ-calculus alternation hierarchy is strict. *Theor. Comput. Sci.*, 195(2):133–153, 1998.
4. A. Carayol and C. Löding. MSO on the infinite binary tree: Choice and order. In *CSL'07*, volume 4646 of *LNCS*, pages 161–176. Springer, 2007.
5. Kosaburo Hashiguchi. Algorithms for determining relative star height and star height. *Inf. Comput.*, 78(2):124–169, 1988.
6. D. Kirsten. Distance desert automata and the star height problem. *RAIRO*, 3(39):455–509, 2005.
7. Nils Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Annals of Pure and Applied Logic*, 69(2–3):243–268, 1994.
8. Giacomo Lenzi. A hierarchy theorem for the μ-calculus. In *ICALP*, pages 87–97, 1996.
9. Andrzej Wlodzimierz Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Computation Theory*, volume 208 of *LNCS*, pages 157–168. Springer, 1985.
10. D. Niwiński and I. Walukiewicz. Ambiguity problem for automata on infinite trees. Unpublished note.
11. Damian Niwiński. On fixed-point clones (extended abstract). In *ICALP '86*, volume 226 of *LNCS*, pages 464–473. Springer, 1986.

12. Damian Niwiński and Igor Walukiewicz. Relating hierarchies of word and tree automata. In Michel Morvan, Christoph Meinel, and Daniel Krob, editors, *Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science, STACS '98*, volume 1373 of *Lecture Notes in Computer Science*, pages 320–331. Springer, February 1998.
13. Damian Niwinski and Igor Walukiewicz. Deciding nondeterministic hierarchy of deterministic tree automata. *Electr. Notes Theor. Comput. Sci.*, 123:195–208, 2005.
14. Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, July 1969.
15. Wolfgang Thomas. *Handbook of Formal Language Theory*, volume III, chapter Languages, Automata, and Logic, pages 389–455. Springer, 1997.
16. Tomasz Fryderyk Urbanski. On deciding if deterministic Rabin language is in Büchi class. In *ICALP 2000*, pages 663–674, 2000.
17. Klaus W. Wagner. Eine topologische Charakterisierung einiger Klassen regulärer Folgenmengen. *Elektronische Informationsverarbeitung und Kybernetik*, 13(9):473–487, 1977.
18. Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1–2):135–183, 1998.

Appendix: Proof of the Completeness Lemma

We come back to the situation of the presentation in Section 4. In particular, we aim at proving Lemma 5. From now on i, j are fixed, as well as a guidable parity automaton $\mathcal{A} = (Q_\mathcal{A}, A \times G, q_I^\mathcal{A}, \delta_\mathcal{A}, col_\mathcal{A})$ such that $L = L(\mathcal{A})$ and the priorities used by $\mathcal{A}$ are from some interval P. Let also $\mathcal{B} = (Q_\mathcal{B}, A, q_I^\mathcal{B}, \delta_\mathcal{B}, col_\mathcal{B})$ be some parity automaton such that $L(\mathcal{B}) \subseteq L$ and the priorities used by $\mathcal{B}$ are from the interval $[i, j]$ (cf. Lemma 5). Because $\mathcal{A}$ is guidable there is a mapping g such that $(\mathcal{B}, g)$ guides $\mathcal{A}$.

We aim at showing that there is an N such that for every A-tree $t \in L(\mathcal{B})$ there is an accepting run of $\mathcal{A}$ that is accepted by $T_{i,j}$ with distance parameter N. Let $t \in L(\mathcal{B})$ and $\rho_\mathcal{B}$ be an accepting run of $\mathcal{B}$ on t. Set $\rho_\mathcal{A} = g(\rho_\mathcal{B})$. We show below that $\rho_\mathcal{A}$ is accepted by $T_{i,j}$ by constructing an accepting run $\rho_{T_{i,j}}$ based on the run $\rho_\mathcal{B}$. At the end of this construction, we establish Lemma 14 from which Lemma 5 is directly obtained.

We need first some more definitions. First of all, to simplify notation, for a position $x \in \{0,1\}^*$ we write $col_\mathcal{A}(x)$ and $col_\mathcal{B}(x)$ to denote the respective priorities of the states in the runs $\rho_\mathcal{A}$ and $\rho_\mathcal{B}$ at position x, i.e., $col_\mathcal{A}(\rho_\mathcal{A}(x))$ and $col_\mathcal{B}(\rho_\mathcal{B}(x))$.

Let $y, z \in \{0,1\}^*$ be two nodes with $y \sqsubset z$. For $\ell \in [i, j]$ we say that (y, z) is an *ℓ-loop* if

- $\rho_\mathcal{A}(y) = \rho_\mathcal{A}(z)$,
- $\rho_\mathcal{B}(y) = \rho_\mathcal{B}(z) =: q \in Q_\mathcal{B}$ with $col_\mathcal{B}(q) = \ell$, and
- $col_\mathcal{B}(x) \leq \ell$ for all x with $y \sqsubset x \sqsubset z$.

So an ℓ-loop is a loop for $\mathcal{A}$, and it has the same $\mathcal{B}$-state with priority ℓ at the first and the last position and no state with higher priority in-between (for $\mathcal{B}$).

We say that an ℓ-loop (y, z) is *dominated* by a priority $k \in P$ (a priority of $\mathcal{A}$) if k is the highest $\mathcal{A}$ priority appearing in the loop (z excluded), i.e., if

- $col_\mathcal{A}(x) \leq k$ for all $y \sqsubseteq x \sqsubset z$, and
- $col_\mathcal{A}(x) = k$ for some $y \sqsubseteq x \sqsubset z$.

The following lemma is easy to prove but it is the central reason why we use guidable tree automata. It states that an accepting loop of $\mathcal{B}$ has to be dominated by an accepting priority of $\mathcal{A}$. This lemma the same as Lemma 2, but presented with terminology used along the proof.

Lemma 6. *Let (y, z) be an ℓ-loop dominated by k. Then ℓ even implies k even.*

Proof. Assume that ℓ is even. The proof is a simple pumping argument. Let $\rho_\mathcal{B}^{[y,z)^*}$ be the run obtained from $\rho_\mathcal{B}$ by iterating the part between y and z. Formally, let $u \in \{0,1\}^*$ be such that $z = yu$. Then $\rho_\mathcal{B}^{[y,z)^*}$ is defined by

$$\rho_\mathcal{B}^{[y,z)^*}(x) = \begin{cases} \rho_\mathcal{B}(x) & \text{if } y \not\sqsubseteq x, \\ \rho_\mathcal{B}(yv) & \text{if } x = yu^n v \text{ for some } n \geq 0 \text{ and } v \text{ with } u \not\sqsubseteq v. \end{cases}$$

Because $\rho_{\mathcal{B}}(y) = \rho_{\mathcal{B}}(z)$ this indeed defines a run of $\mathcal{B}$, namely on the tree $t^{[y,z)^*}$ (which is defined in the same way). Furthermore, $\rho_{\mathcal{B}}^{[y,z)^*}$ is accepting because the maximal priority seen on the branch yu^ω is ℓ. All the other infinite branches in $\rho_{\mathcal{B}}^{[y,z)^*}$ correspond from some point onwards to a branch in $\rho_{\mathcal{B}}$ and are therefore accepting.

Now consider $g(\rho_{\mathcal{B}}^{[y,z)^*})$. Because g only locally translates the transitions of $\mathcal{B}$ into transitions of $\mathcal{A}$, and because $\rho_{\mathcal{A}}(y) = \rho_{\mathcal{A}}(z)$, one can observe that $g(\rho_{\mathcal{B}}^{[y,z)^*}) = (g(\rho_{\mathcal{B}}))^{[y,z)^*} = \rho_{\mathcal{A}}^{[y,z)^*}$, i.e., applying g to $\rho_{\mathcal{B}}^{[y,z)^*}$ yields the same as applying g to $\rho_{\mathcal{B}}$ and then iterating the loop (y, z).

In $\rho_{\mathcal{A}}^{[y,z)^*}$ the highest priority seen infinitely often on the branch yu^ω is k. As $\rho_{\mathcal{B}}^{[y,z)^*}$ is accepting, $\rho_{\mathcal{A}}^{[y,z)^*}$ must also be accepting and therefore k must be even. □

Having established this connection between priorities of $\mathcal{A}$ and $\mathcal{B}$ on loops, we now start the construction of an accepting run of $T_{i,j}$ on $\rho_{\mathcal{A}}$. Recall that the runs of $T_{i,j}$ are in one-to-one correspondence with mappings that assign to each node an element of S. We are now going to specify such a mapping that associates to each node $x \in \{0,1\}^*$ an element $s_x \in S$ and show that it defines an accepting run of $T_{i,j}$ on $\rho_{\mathcal{A}}$.

Construction of the run of $T_{i,j}$. The essential idea is the following: In an ℓ-loop we map (more precisely the elements of S used by the run in the loop map) the highest priority for $\mathcal{A}$ on this loop (i.e., the priority dominating this loop) to the highest priority of $\mathcal{B}$ on this loop (i.e., to ℓ). Then Lemma 6 ensures that we indeed map odd priorities of $\mathcal{A}$ to odd priorities of $\mathcal{B}$. To make this simple idea work we have to tune the definitions a bit.

We say that a position x is *covered* by k in an ℓ-loop (y, z) if this loop is dominated by k, and there is an x' with $y \sqsubseteq x' \sqsubseteq x \sqsubset z$ such that $col_{\mathcal{A}}(x') = k$. That is, a position with priority k has to appear before x in the loop. We just say that x is covered by k if it is covered by k in some ℓ-loop.

We let

$$cov_x(k) = \max\{\ell \in [i, j] \mid x \text{ is covered by } k \text{ in an } \ell\text{-loop}\}$$

where the maximum is undefined in case x is not covered by k.

Lemma 7. *The function cov_x is nondecreasing (over its domain). Furthermore, for all k such that $cov_x(k)$ is defined, k odd implies $cov_x(k)$ odd.*

Proof. Let $k \leq k'$ be such that $cov_x(k) = \ell$ and $cov_x(k') = \ell'$ are defined. By definition, x is k-covered in some ℓ-loop (y, z) and k'-covered in some ℓ'-loop (y', z'). As (y, z) is k-dominated and $k \leq k'$ we conclude that no position inside (y, z) can have priority k'. This implies that $y' \sqsubset y \sqsubseteq x \sqsubset z'$. Since (y', z') is an ℓ'-loop, ℓ' is the highest $\mathcal{B}$-priority on (y', z') and in particular $\ell = col_{\mathcal{B}}(y) \leq \ell'$.

The second statement is a direct consequence of Lemma 6. □

Using cov_x we define $s_x : P \to [i, j]$ as follows:

$$s_x(k) = \begin{cases} \bot & \text{if } cov_x(k) \text{ is undefined,} \\ cov_x(k) & \text{if } k \text{ is odd or } cov_x(k) \text{ is even or } cov_x(k) = i, \\ cov_x(k) - 1 & \text{if } k \text{ is even and } cov_x(k) \neq i \text{ is odd.} \end{cases}$$

Remark in this definition the special treatment of the priority i that is here just for preventing the use of the disallowed value $i - 1$. Note also (see the remark following the definition of $cov_x(k)$), that the second line in the definition of s_x is used iff k and $cov_x(k)$ have the same parity (even or odd). While in the third case the parity of k and $cov_x(k)$ are different: the -1 is a correction to the value of $cov_x(k)$ to make it of the same parity as k. Hence, the two last cases can be understood as taking the maximal priority below or equal to $cov_x(k)$ that has the same parity as k.

A very important thing to remark is that, though cov_x is monotonic over its domain, this is not the case for s_x: Consider, e.g., the case $cov_x(2) = cov_x(1) = 1$. This mapping is monotonic, but we get $s_x(2) = 0$ and $s_x(1) = 1$, which is not monotonic (and this case can happen for real). That is why the item 2 in the definition of S is weaker than monotonicity.

Lemma 8. *For each x, the mapping s_x belongs to S.*

Proof. 1. If k is odd, then $cov_x(k)$ is odd (Lemma 7). The second line in the definition of s_x has to be taken, and hence $s_x(k) = cov_x(k)$ is odd.
2. Let $k' > k$, k' odd such that $s_x(k)$ and $s_x(k')$ are defined. From the definition of s_x, $s_x(k) \leq cov_x(k)$. Still from the definition, and by Lemma 7, $s_x(k') = cov_x(k')$ is odd. Furthermore by Lemma 7, $cov_x(k') \geq cov_x(k)$. Hence $s_x(k') = cov_x(k') \geq cov_x(k) \geq s_x(k)$. □

Thus, we have assigned to each node x some $s_x \in S$, defining a unique run $\rho_{T_{i,j}}$ of $T_{i,j}$ on ρ_A. It remains to show that this run of $T_{i,j}$ is accepting. Since $T_{i,j}$ uses a distance-parity condition, we need to show that the parity condition of $T_{i,j}$ evaluates to 0, and that the distance condition evaluates to at most N where N depends on $\mathcal{B}$ but not on the specific run under consideration (nor the underlying tree). Our first step is to show that $\rho_{T_{i,j}}$ is accepting for the parity condition.

Lemma 9. *The run $\rho_{T_{i,j}}$ is accepting for the parity condition.*

Proof. Recall that the priority output by the run of $T_{i,j}$ when in state (s', k, s) is:

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

This means that the priority assumed by $\rho_{T_{i,j}}$ is $s_x(col_{\mathcal{A}}(x))$ if defined, and i otherwise.

Take an infinite branch B and let k and ℓ be the maximal priorities assumed by $\rho_{\mathcal{A}}$ and $\rho_{\mathcal{B}}$ respectively that appear infinitely often on this branch. Both k

and ℓ are even since $\rho_{\mathcal{A}}$ and $\rho_{\mathcal{B}}$ are accepting. Then there is a suffix B' of B on which k and ℓ are the maximal priorities of $\mathcal{A}$ and $\mathcal{B}$, and furthermore each position on B' is k-covered in some ℓ-loop. By definition of $cov_x(k)$ as a maximum, $cov_x(k) \geq \ell$. This implies $s_x(k) \geq \ell$ is even (since both ℓ and $cov_x(k)$ are even). This priority $s_x(k)$ is seen infinitely often on B'.

For the sake of contradiction, consider the maximal priority n appearing infinitely often in the run $\rho_{T_{i,j}}$ over B', and assume it is odd. Two cases can happen, either $n = i$ is odd, and in this case $n < \ell$, a contradiction. Else (definition of s_x), this priority has to be seen on some node x of B' such that $col_{\mathcal{A}}(x) = k'$ is odd, meaning $s_x(k') = n$. Because k is the maximal priority on B' and is even, we get $k' < k$. Since $s_x(k') = n$ is defined, k' dominates some n-loop (y', z'). By the choice of B' we know that x is also k-covered in an ℓ-loop (y, z). Because (y', z') is dominated by k' and not by k we conclude that $y \sqsubset y' \sqsubseteq x$. Thus, $n \leq \ell$ (definition of an ℓ-loop). And for parity reasons, we get $n < \ell$. Again a contradiction. □

We now turn to the distance condition. Recall that the distance values are of the form d_k and r_{k-1} for each $k \in P$. The distance condition is satisfied if there is an N such that for each k the number of times that d_k occurs without any $d_{k'}$ or $r_{k'-1}$ for $k' > k$ in-between is bounded by N. We start by presenting Lemmas 10 and 11 that are concerned with the evolution of the value of $s_x(k)$ when x changes.

Lemma 10. *Let $x \sqsubset x'$ be such that $s_x(k) \neq s_{x'}(k)$ and $col_{\mathcal{A}}(u) \neq k$ for all $x \sqsubseteq u \sqsubseteq x'$. Then $s_x(k) > s_{x'}(k)$.*

Proof. If $s_x(k)$ is undefined, then x' is k-covered in an ℓ'-loop (y', z') in which x is not k-covered (or that does not even contain x). Then the position k-covering x' must be between x and x', contradicting the assumption that $col_{\mathcal{A}}(u) \neq k$ for all $x \sqsubseteq u \sqsubseteq x'$.

Let $\ell = s_x(k)$ and $\ell' = s_{x'}(k)$. Then x is covered by k in an ℓ-loop (y, z) and x' is covered by k in an ℓ'-loop (y', z'). Assume that $\ell' > \ell$. Then y' cannot be between y and z since this would contradict the fact that (x, y) is an ℓ-loop. If y' is above y, then x would also be k covered in the ℓ'-loop (y', z') and hence $s_x(k) \geq \ell'$. If y' is below y, then it has to be below x because otherwise it would be between y and z. But then the position covering x' in the loop (y', z') must be between x and x', contradicting the assumption that $col_{\mathcal{A}}(u) \neq k$ for all $x \sqsubseteq u \sqsubseteq x'$. □

Lemma 11. *Let (y, z) be a k-dominated ℓ-loop and $x \sqsubset x'$ positions that are k-covered in (y, z). Then $s_x(k) \geq s_{x'}(k)$.*

Proof. Assume that $s_x(k) < s_{x'}(k)$. Then x' must be k-covered in an ℓ'-loop (y', z') with $\ell' > \ell$ such that x is not k-covered in this ℓ'-loop. From $\ell' > \ell$, we deduce that y' cannot be between y and z and hence $y' \sqsubset y$. But then x is also k-covered in (y', z') because it is in (y, z). □

Let β be a finite piece of branch such that $\max\{dst(x) \mid x \in \beta\} = d_k$ (we simplify notation using $dst(x) = dst(s, k, s_x)$). By piece of branch we mean that β is linearly ordered, and $x, x' \in \beta$ with $x \sqsubset x'$ implies that $u \in \beta$ for all $x \sqsubset u \sqsubset x'$. Let $D_k = \{x \in \beta \mid dst(x) = d_k\}$ be the positions on β with distance value d_k. We want to find a uniform bound on the size of D_k. We order the positions in D_k as $x_1 \sqsubset \cdots \sqsubset x_M$. Our goal is to prove that $|D_k|$ is bounded by a value that depends solely on $\mathcal{B}$. We first prove it inside a loop in the following lemma before getting the more general Lemma 13.

Lemma 12. *Let (y, z) be an ℓ-loop with $y, z \in \beta$. There cannot be more than $2(j - i + 2)$ many elements from D_k between y and z .*

Proof. Let us remark the following property (call it $\star$) that directly results from the definition of *dst*: let x be the parent of x_m for some $m > 1$, then $s_x(k) \neq s_{x_m}(k)$ if $s_{x_m}(k)$ is defined.

Let $y \sqsubseteq x \sqsubset z$ be such that $col_A(u) \neq k$ for all $y \sqsubseteq u \sqsubset x$. Then for all x_m between y and x the value $s_{x_m}(k)$ is defined because $col_{\mathcal{A}}(x) \neq k$ but $dst(x_m) = d_k$. Hence, by Lemma 10 and $(\star)$ we get that the values assigned by $T_{i,j}$ to the D_k positions in this initial part of the loop must strictly decrease (whenever there is a change it must be a decrease by Lemma 10, and between two D_k positions there must be a change by $(\star)$). There can be at most $j - i + 1$ such decreases. Thus, if (y, z) contains more than $j - i + 2$ positions from D_k, we can choose x such that $col_{\mathcal{A}}(x) = k$ and the part between y and x contains at most $j - i + 2$ positions from D_k. All the positions from D_k that are between x and z are thus k-covered (no priority higher than k can occur in β because $col_{\mathcal{A}}(x) > k$ implies that $dst(x) > d_k$). Hence, we can use Lemma 11 and $(\star)$ to infer that the values assigned to these D_k positions by $T_{i,j}$ must also strictly decrease. Thus, on the second part of the loop (y, z) there can also be at most $j - i + 2$ many positions from D_k. □

Finally we can give a bound in all cases.

Lemma 13. $|D_k| \leq (2(j - i + 2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}| + 1)^{j-i+1}$.

Proof. Let $\ell = \max\{col_{\mathcal{B}}(x) \mid x \in \beta\}$ be the maximal $\mathcal{B}$-priority on β. For each $m \in \{1, \ldots, M - 1\}$ let $\ell_m = \max\{col_{\mathcal{B}}(x) \mid x_m \sqsubset x \sqsubseteq x_{m+1}\}$. Assume that ℓ appears more than $2(j-i+2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}|$ often among the ℓ_m. Then there are two positions y', z' in β that form an ℓ-loop such that this ℓ-loop contains more than $2(j - i + 2)$ many positions from D_k. This contradicts Lemma 12. Thus, there are at most $2(j - i + 2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}|$ many of the ℓ_m that are equal to ℓ.

If $\ell = i$, then this implies that $|D_k|$ is bounded by $2(j - i + 2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}| + 1$. For $\ell = i + h$ with $h > 0$, we can split β in at most $2(j-i+2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}|+1$ many pieces on which the maximal $\mathcal{B}$-priority is less than ℓ. By induction hypothesis, each of these pieces contains at most $(2(j - i + 2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}| + 1)^h$ many elements from D_k. In total we get that $|D_k| \leq (2(j - i + 2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}| + 1)^{j-i+1}$. □

Putting together Lemmas 9 and 13, we get to the conclusion of the proof.

Lemma 14. $val(\rho_{T_{i,j}}) \leq (2(j - i + 2)|Q_{\mathcal{A}}||Q_{\mathcal{B}}| + 1)^{j-i+1}$.