Uniformization Problems for Tree-automatic Relations and Top-down Tree Transducers*

Christof Löding and Sarah Winter

Lehrstuhl für Informatik 7, RWTH Aachen University, Aachen, Germany

Abstract

For a given binary relation of finite trees, we consider the synthesis problem of deciding whether there is a deterministic top-down tree transducer that uniformizes the relation, and constructing such a transducer if it exists. A uniformization of a relation is a function that is contained in the relation and has the same domain as the relation. It is known that this problem is decidable if the relation is a deterministic top-down tree-automatic relation. We show that it becomes undecidable for general tree-automatic relations (specified by non-deterministic top-down tree automata). We also exhibit two cases for which the problem remains decidable. If we restrict the transducers to be path-preserving, which is a subclass of linear transducers, then the synthesis problem is decidable for general tree-automatic relations. If we consider relations that are finite unions of deterministic top-down tree-automatic relations, then the problem is decidable for synchronous transducers, which produce exactly one output symbol in each step (but can be non-linear).

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases tree transducers, tree automatic relation, uniformization

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.66

1 Introduction

A uniformization of a (binary) relation is a function that selects for each element in the domain of the relation a unique image that is in relation with this element. Originally, uniformization has been studied in set theory, where the complexity of a class of definable relations is related with the complexity of uniformizations for these relations (see [20] for results of this kind). The basic uniformization question for a class $\mathcal{C}$ of relations is whether each relation from $\mathcal{C}$ has a uniformization in $\mathcal{C}$.

Automata provide a natural framework for defining relations (over words or trees), and uniformization problems in this setting have been studied since the early days of automata theory. Word relations defined by asynchronous finite automata [9], also called rational relations, were first shown to have rational uniformizations in [15, Theorem 3] (with many alternative and simplified proofs following later). For relations of infinite words that are accepted by synchronous finite automata, or equivalently definable in monadic second-order logic (MSO) over the structure consisting of natural numbers equipped with the successor relation, the uniformization property is shown in [23]. Over infinite trees, the uniformization property fails for MSO definable relations (corresponding to synchronous tree automata) [12, 2], while it has been shown recently that uniformization is possible for synchronous relations over finite trees [16, 4]. These relations defined by synchronous automata are usually

* This work was supported by the DFG grant "Transducer Synthesis from Automaton Definable Specifications" (LO 1174/3-1)

referred to as automatic, ω-automatic, tree-automatic and ω-tree-automatic relations (for finite words, infinite words, finite trees, and infinite trees, respectively).

In a more algorithmic setting, uniformization is often referred to as synthesis: The relation is viewed as a specification between inputs and outputs, and the function is supposed to be realized by a device that produces the output while processing the input. This means, that the class for the uniformizations is usually different from the class of the specifications, and the problem of interest is now the decision problem whether a given relation admits a uniformization in the desired class.

The classical setting, originating from Church's synthesis problem [3], is the one of infinite words. The specification is given by an ω-automatic relation (orginally in MSO), and the question is whether it can be uniformized by a synchronous sequential transducer that produces, letter by letter, an infinite output word while reading an infinite input word. The seminal paper of Büchi and Landweber [1] shows the decidability of this problem, and has been extended later to uniformizations by asynchronous sequential transducers [14, 13]. A detailed study of the synthesis of sequential transducers for asynchronous automata on finite words is provided in [6].

Our aim is to study these uniformization questions for relations over finite trees. Tree automata are used in many fields, for example as a tool for analyzing and manipulating rewrite systems or XML Schema languages (see [7]). Tree transformations that are realized by finite tree transducers thus become interesting in the setting of translations from one document scheme into another [19]. As class for the uniformizations we consider deterministic top-down tree transducers (D↓TTs), which are a natural extension of sequential transducers on words. A first result in this setting was obtained in [18], where we show that it is decidable whether a tree-automatic relation that is defined by a deterministic top-down tree automaton (D↓TA) can be uniformized by a D↓TT. A representation of the specification by a deterministic automaton model is essential in many synthesis algorithms for automata. A standard approach is to build a game in which the two players produce input and output. The aim of the output player is to ensure that the pair of input and output produced along a play satisfies the specification. This property is ensured in the game by simulating a deterministic automaton for the specification on the moves of the players. A winning strategy for the output player then corresponds to a uniformizer.

In this paper, we show that the synthesis problem for D↓TT from nondeterministic tree-automatic relations is indeed undecidable, showing that the nondeterminism does not only destroy the game theoretic approach (as sketched above) but makes the problem intractable in general. On the positive side, we prove two decidability results for restricted classes of uniformizers and specifications: **1.** For nondeterministic tree-automatic relations uniformization by path-preserving D↓TTs is decidable. Intuitively, we call a D↓TT path-preserving if every node of the output tree is produced from a node of the input tree that is above or below the output node (this implies that each path-preserving D↓TT is in particular linear). For this class of uniformizers we can adapt the game theoretic approach by using guidable automata [5, 17] instead of deterministic automata for the specification. **2.** If we restrict the specifications to unions of D↓TAs with disjoint domain, we obtain decidability for uniformizations by synchronous D↓TTs. We call a D↓TT synchronous if it produces one output symbol in each transition (but the transitions can be non-linear). While this is a rather specific result, it is the first decidability result for synthesis of transducers in which in the synthesized transducer may need to be non-linear.

The paper is structured as follows. In Section 2 we fix some basic definitions and terminology. In Section 3 we show undecidability for synthesis of D↓TTs from tree-automatic

specifications, and the decidability results are presented in Section 4.

2 Preliminaries

Words and trees. The set of natural numbers containing zero is denoted by $\mathbb{N}$. For a set S, the powerset of S is denoted by 2^S. An *alphabet* Σ is a finite non-empty set of letters. A finite *word* is a finite sequence of letters. The set of all finite words over Σ is denoted by Σ^*. The length of a word $w \in \Sigma^*$ is denoted by $|w|$, the empty word is denoted by ε. For $w = a_1 \dots a_n \in \Sigma^*$ for some $n \in \mathbb{N}$ and $a_1, \dots, a_n \in \Sigma$, let $w[i]$ denote the ith letter of w, i.e., $w[i] = a_i$. Furthermore, let $w[i,j]$ denote the infix from the ith to the jth letter of w, i.e., $w[i,j] = a_i \dots a_j$. We write $u \sqsubseteq w$ if there is some v such that $w = uv$ for $u, v \in \Sigma^*$. A subset $L \subseteq \Sigma^*$ is called *language* over Σ.

A *ranked alphabet* Σ is an alphabet where each letter $f \in \Sigma$ has a rank $rk(f) \in \mathbb{N}$. The set of letters of rank i is denoted by Σ_i. A tree domain *dom* is a non-empty finite subset of $(\mathbb{N} \setminus \{0\})^*$ such that *dom* is prefix-closed and for each $u \in (\mathbb{N} \setminus \{0\})^*$ and $i \in \mathbb{N} \setminus \{0\}$ if $ui \in dom$, then $uj \in dom$ for all $1 \leq j < i$. We speak of ui as successor of u for each $u \in dom$ and $i \in \mathbb{N} \setminus \{0\}$.

A (finite Σ-labeled) *tree* is a pair $t = (dom_t, val_t)$ with a mapping $val_t : dom_t \to \Sigma$ such that for each node $u \in dom_t$ the number of successors of u is a rank of $val_t(u)$. The height h of a tree t is the length of its longest path, i.e., $h(t) = max\{|u| \mid u \in dom_t\}$. The set of all Σ-labeled trees is denoted by T_Σ. A subset $T \subseteq T_\Sigma$ is called *tree language* over Σ.

A *subtree* $t|_u$ of a tree t at node u is defined by $dom_{t|_u} = \{v \in \mathbb{N}^* \mid uv \in dom_t\}$ and $val_{t|_u}(v) = val_t(uv)$ for all $v \in dom_{t|_u}$. In order to formalize concatenation of trees, we introduce the notion of special trees. A *special tree* over Σ is a tree over $\Sigma \cup \{\circ\}$ such that $\circ$ occurs exactly once at a leaf. Given $t \in T_\Sigma$ and $u \in dom_t$, we write $t[\circ/u]$ for the special tree that is obtained by deleting the subtree at u and replacing it by $\circ$. Let S_Σ be the set of special trees over Σ. For $t \in S_\Sigma$ and $s \in T_\Sigma$ or $s \in S_\Sigma$ let the *concatenation* $t \cdot s$ be the tree that is obtained from t by replacing $\circ$ with s.

Let X_n be a set of n variables $\{x_1, \dots, x_n\}$ and Σ be a ranked alphabet. We denote by $T_\Sigma(X_n)$ the set of all trees over Σ which additionally can have variables from X_n at their leaves. We define X_0 to be the empty set, the set $T_\Sigma(\emptyset)$ is equal to T_Σ. Let $X = \bigcup_{n>0} X_n$. A tree from $T_\Sigma(X)$ is called *linear* if each variable occurs at most once. For $t \in T_\Sigma(X_n)$ let $t[x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n]$ be the tree that is obtained by substituting each occurrence of $x_i \in X_n$ by $t_i \in T_\Sigma(X)$ for every $1 \leq i \leq n$.

A tree from $T_\Sigma(X_n)$ such that all variables from X_n occur exactly once and in the order $x_1, \dots, x_n$ when reading the leaf nodes from left to right, is called *n-context* over Σ. Given an n-context, the node labeled by x_i is referred to as ith hole for every $1 \leq i \leq n$. A special tree can be seen as a 1-context, a tree without variables can be seen a 0-context. If C is an n-context and $t_1, \dots, t_n \in T_\Sigma(X)$ we write $C[t_1, \dots, t_n]$ instead of $C[x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n]$.

Tree automata. We fix our notations. For a detailed introduction to tree automata see e.g. [10] or [7]. Let $\Sigma = \bigcup_{i=0}^m \Sigma_i$ be a ranked alphabet. A *non-deterministic top-down tree automaton* (an N↓TA) over Σ is of the form $\mathcal{A} = (Q, \Sigma, Q_0, \Delta)$ consisting of a finite set of states Q, a set $Q_0 \subseteq Q$ of initial states, and $\Delta \subseteq \bigcup_{i=0}^m (Q \times \Sigma_i \times Q^i)$ is the transition relation. For $i = 0$, we identify $Q \times \Sigma_i \times Q^i$ with $Q \times \Sigma_0$. Let t be a tree and $\mathcal{A}$ be an N↓TA, a *run* of $\mathcal{A}$ on t is a mapping $\rho : dom_t \to Q$ compatible with Δ, i.e., $\rho(\varepsilon) \in Q_0$ and for each node $u \in dom_t$ with $i \geq 0$ successors $(\rho(u), val_t(u), \rho(u1), \dots, \rho(ui)) \in \Delta$. A tree $t \in T_\Sigma$ is *accepted* if, and only if, there is a run of $\mathcal{A}$ on t. The tree language *recognized* by $\mathcal{A}$ is

$T(\mathcal{A}) = \{t \in T_\Sigma \mid \mathcal{A} \text{ accepts } t\}$. A tree language $T \subseteq T_\Sigma$ is called *regular* if T is recognizable by a non-deterministic top-down tree automaton.

A top-down tree automaton $\mathcal{A} = (Q, \Sigma, Q_0, \Delta)$ is *deterministic* (a D↓TA) if the set Q_0 is a singleton set and for each $f \in \Sigma_i$ and each $q \in Q$ there is at most one transition $(q, f, q_1, \ldots, q_i) \in \Delta$. However, non-deterministic and deterministic top-down automata are not equally expressive. It is effectively decidable whether a regular tree language is top-down deterministic [10].

In Section 4.1 we use *guidable tree automata* [17]. The concept of guidable tree automata is that another tree automaton can act as a guide, meaning that a tree automaton $\mathcal{B}$ can guide a tree automaton $\mathcal{A}$ if an accepting run of $\mathcal{B}$ on a tree t can be translated deterministically into an accepting run of $\mathcal{A}$ on t.

Formally, an N↓TA $\mathcal{A}$ can be *guided by* an N↓TA $\mathcal{B}$ if there is a mapping $g : Q_\mathcal{A} \times \Delta_\mathcal{B} \to \Delta_\mathcal{A}$ such that $g(q, (p, a, p_1, \ldots, p_i)) = (q, a, q_1, \ldots, q_i)$ for some $q_1, \ldots, q_i \in Q_\mathcal{A}$, and 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 $\rho'(\varepsilon) = q_0^\mathcal{A}$, and for all $u \in dom_t : (\rho'(u), val_t(u), \rho'(u1), \ldots, \rho'(ui)) = g\,(\rho'(u), (\rho(u), val_t(u), \rho(u1), \ldots, \rho(ui)))$. An N↓TA $\mathcal{A}$ is called *guidable* if it can be guided by every N↓TA $\mathcal{B}$ such that $T(\mathcal{B}) \subseteq T(\mathcal{A})$.

Tree-automatic relations are defined by using tree automata over a product alphabet. For nodes that belong only to one of the trees one uses a padding symbol. Formally, let Σ, Γ be ranked alphabets and let $\Sigma_\perp = \Sigma \cup \{\perp\}$, $\Gamma_\perp = \Gamma \cup \{\perp\}$, where $\perp$ is a new symbol with rank 0. For an i-ary symbol $f \in \Sigma_\perp$ and a j-ary symbol $g \in \Gamma_\perp$, let $rk((f, g)) = max\{i, j\}$. The *convolution* of (t_1, t_2) with $t_1 \in T_\Sigma$, $t_2 \in T_\Gamma$ is the $\Sigma_\perp \times \Gamma_\perp$-labeled tree $t = t_1 \otimes t_2$ defined by $dom_t = dom_{t_1} \cup dom_{t_2}$, and $val_t(u) = (val^\perp_{t_1}(u), val^\perp_{t_2}(u))$ for all $u \in dom_t$, where $val^\perp_{t_i}(u) = val_{t_i}(u)$ if $u \in dom_{t_i}$ and $val^\perp_{t_i}(u) = \perp$ otherwise for $i \in \{1, 2\}$. As a special case, given $t \in T_\Sigma$, we define $t \otimes \perp$ to be the tree with $dom_{t\otimes\perp} = dom_t$ and $val_{t\otimes\perp}(u) = (val_t(u), \perp)$ for all $u \in dom_t$. Analogously, we define $\perp \otimes t$. We define the *convolution of a tree relation* $R \subseteq T_\Sigma \times T_\Gamma$ to be the tree language $T_R := \{t_1 \otimes t_2 \mid (t_1, t_2) \in R\}$.

We call a (binary) relation R *tree-automatic* if there exists a regular tree language T such that $T = T_R$. For ease of presentation, we say a tree automaton $\mathcal{A}$ recognizes R if it recognizes the convolution T_R and denote by $R(\mathcal{A})$ the induced relation R.

A *uniformization* of a relation $R \subseteq X \times Y$ is a function $f_R : X \to Y$ such that $(x, f_R(x)) \in R$ for all $x \in dom(R)$. We are interested in uniformizations of tree-automatic relations by deterministic top-down tree transducers.

Tree transducers. We consider top-down tree transducers, which read the tree from the root to the leaves and produce finite output trees in each step that are attached to the already produced output (see [7] for an introduction to tree transducers).

A *top-down tree transducer* (a ↓TT) is of the form $\mathcal{T} = (Q, \Sigma, \Gamma, q_0, \Delta)$ consisting of a finite set of states Q, a finite input alphabet Σ, a finite output alphabet Γ, an initial state $q_0 \in Q$, and Δ is a finite set of transition rules of the form $q(f(x_1, \ldots, x_i)) \to w[q_1(x_{j_1}), \ldots, q_n(x_{j_n})]$, or $q(x_1) \to w[q_1(x_1), \ldots, q_n(x_1)]$($\varepsilon$-transition), where $f \in \Sigma_i$, w is an n-context over Γ, $q, q_1, \ldots, q_n \in Q$ and variables $x_{j_1}, \ldots, x_{j_n} \in X_i$. A *deterministic* top-down tree transducer (a D↓TT) has no ε-transitions and no two rules with the same left-hand side.

A *configuration* of a top-down tree transducer is a triple $c = (t, t', \varphi)$ of an input tree $t \in T_\Sigma$, an output tree $t' \in T_{\Gamma\cup Q}$ and a function $\varphi : D_{t'} \to dom_t$, where

- $val_{t'}(u) \in \Gamma_i$ for each $u \in dom_{t'}$ with $i > 0$ successors, and
- $val_{t'}(u) \in \Gamma_0$ or $val_{t'}(u) \in Q$ for each leaf $u \in dom_{t'}$, and
- $D_{t'} \subseteq dom_{t'}$ with $D_{t'} = \{u \in dom_{t'} \mid val_{t'}(u) \in Q\}$, i.e., φ maps every node from the

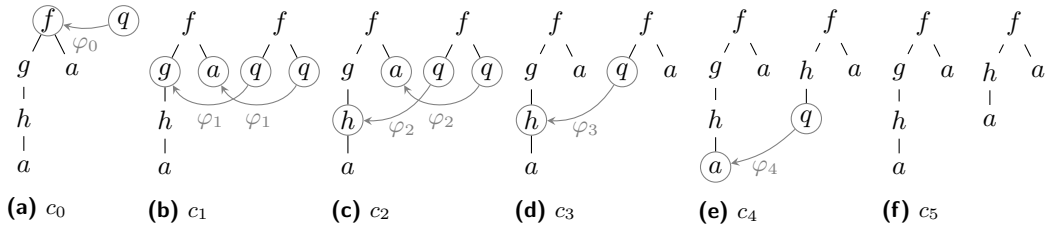


Figure 1 The configuration sequence c_0 to c_5 of $\mathcal{T}$ on t from Example 1.

output tree t' that has a state-label to a node of the input tree t.

Let $c_1 = (t, t_1, \varphi_1)$ and $c_2 = (t, t_2, \varphi_2)$ be configurations of a top-down tree transducer over the same input tree. We define a *successor relation* $\to_{\mathcal{T}}$ on configurations as usual by applying one rule. Figure 1 illustrates a configuration sequence explained in Example 1 below. Formally, for the application of a non-ε-rule, we define $c_1 \to_{\mathcal{T}} c_2 :\Leftrightarrow$

- There is a state-labeled node $u \in D_{t'}$ of the output tree t_1 that is mapped to a node $v \in dom_t$ of the input tree t, i.e., $\varphi_1(u) = v$, and
- there is a rule $val_{t_1}(u)\,(val_t(v)(x_1, \ldots, x_i)) \to w[q_1(x_{j_1}), \ldots, q_n(x_{j_n})] \in \Delta$ such that the output tree is correctly updated, i.e., $t_2 = t_1[\circ/u] \cdot w[q_1, \ldots, q_n]$, and
- the mapping φ_2 is correctly updated, i.e., $\varphi_2(u') = \varphi_1(u')$ if $u' \in D_{t_1} \setminus \{u\}$ and $\varphi_2(u') = v.j_i$ if $u' = u.u_i$ with u_i is the ith hole in w.

Furthermore, let $\to_{\mathcal{T}}^*$ be the reflexive and transitive closure of $\to_{\mathcal{T}}$ and $\to_{\mathcal{T}}^n$ the reachability relation for $\to_{\mathcal{T}}$ in n steps. From here on, let φ_0 always denote the mapping $\varphi_0(\varepsilon) = \varepsilon$. A configuration (t, q_0, φ_0) is called *initial configuration* of $\mathcal{T}$ on t. A configuration $c = (t, t', \varphi)$ is said to be *reachable* in a computation of $\mathcal{T}$ on t, if $c_0 \to_{\mathcal{T}}^* c$, where c_0 is the initial configuration of $\mathcal{T}$ on t. The relation $R(\mathcal{T})$ induced by a top-down tree transducer $\mathcal{T}$ is $R(\mathcal{T}) = \{(t, t') \in T_\Sigma \times T_\Gamma \mid (t, q_0, \varphi_0) \to_{\mathcal{T}}^* (t, t', \varphi)\}$. For a (special) tree $t \in T_\Sigma$ or $t \in S_\Sigma$ let $\mathcal{T}(t) \subseteq T_{\Gamma \cup Q}$ be the set of *final transformed outputs* of a computation of $\mathcal{T}$ on t, that is the set $\{t' \mid (t, q_0, \varphi_0) \to_{\mathcal{T}}^* (t, t', \varphi)$ s.t. there is no successor configuration of $(t, t', \varphi)\}$. Note, we explicitly do not require that the final transformed output is a tree over Γ. In the special case that $\mathcal{T}(t)$ is a singleton set $\{t'\}$, we also write $\mathcal{T}(t) = t'$. The class of relations definable by $\downarrow$TTs is called the class of *top-down tree transformations*.

▶ **Example 1.** Let Σ be a ranked alphabet given by $\Sigma_2 = \{f\}$, $\Sigma_1 = \{g, h\}$, and $\Sigma_0 = \{a\}$. Consider the $\downarrow$TT $\mathcal{T}$ given by $(\{q\}, \Sigma, \Sigma, \{q\}, \Delta)$ with $\Delta = \{\ q(a) \to a,\ q(g(x_1)) \to q(x_1),\ q(h(x_1)) \to h(q(x_1)),\ q(f(x_1, x_2)) \to f(q(x_1), q(x_2))\ \}$. For each $t \in T_\Sigma$ the transducer deletes all occurrences of g in t. Consider $t := f(g(h(a)), a)$. A possible sequence of configurations of $\mathcal{T}$ on t is $c_0 \to_{\mathcal{T}}^5 c_5$ such that $c_0 := (t, q, \varphi_0)$ with $\varphi_0(\varepsilon) = \varepsilon$, $c_1 := (t, f(q, q), \varphi_1)$ with $\varphi_1(1) = 1$, $\varphi_1(2) = 2$, $c_2 := (t, f(q, q), \varphi_2)$ with $\varphi_2(1) = 11$, $\varphi_2(2) = 2$, $c_3 := (t, f(q, a), \varphi_3)$ with $\varphi_3(1) = 11$, $c_4 := (t, f(h(q), a), \varphi_4)$ with $\varphi_4(11) = 111$, and $c_5 := (t, f(h(a), a), \varphi_5)$. A visualization of this sequence is shown in Figure 1.

We consider two restricted types of top-down tree transducers. The first type are *transducers with bounded (output) delay*. Intuitively, delay occurs in a computation of a transducer if there is a difference between the number of produced output symbols and read input symbols. If the output is behind this is called output delay. More formally, in a configuration (t, t', φ) occurs *delay* d w.r.t. a node $u \in D_{t'}$ if the absolute value of $|\varphi(u)| - |u|$ equals d. We speak of *output delay* if $|\varphi(u)| - |u|$ is a positive integer. We say the *delay* (resp. *output delay*) in a $\downarrow$TT $\mathcal{T}$ is *bounded* by k, if there is a $k \in \mathbb{N}$ such that for every reachable configuration c of $\mathcal{T}$ the maximal delay (resp. output delay) that occurs in c is at most k.

We speak of *synchronous* ↓TTs if the delay is bounded by 0. Consider $\mathcal{T}$ from Example 1 and the configuration sequence of $\mathcal{T}$ given in Example 1. In c_2 occurs output delay 1 resp. 0 w.r.t. node 1 resp. 2 of the output tree. It is easy to see that the transducer has unbounded output delay, because it deletes all occurrences of g in an input tree.

The second restricted type of top-down tree transducer concerns the ability to copy and swap subtrees. A ↓TT is *linear* if all the trees in the transitions are linear. In Section 4.1, we consider a special case of linear ↓TTs called *path-preserving*. Intuitively, a ↓TT is said to be path-preserving if in every computation the read input and correspondingly produced output are always on the same path, i.e., every node of the output tree is produced from a node of the input tree that is above or below the output node. More formally, in every reachable configuration (t, t', φ) of the transducer it holds either $u \sqsubseteq \varphi(u)$ or $\varphi(u) \sqsubseteq u$ for every node $u \in D_{t'}$. We refer to this kind of ↓TTs as P↓TTs for short.

3 Undecidability Results

▶ **Theorem 2.** *It is undecidable whether a given tree-automatic relation has a uniformization by a deterministic top-down tree transducer.*

Proof Sketch. We give a reduction from the halting problem for Turing machines (TM). Given a TM M, our goal is to describe a tree-automatic specification R_M which can only be realized by a deterministic top-down tree transducer if M does not halt on the empty input tape. In order to save space, we draw trees from left to right rather than from top to bottom. For explaining the idea, we provide for a given Turing machine a specification S and a uniformizer θ and a D↓TT-realizable transformation θ such that θ uniformizes S if, and only if, M does not halt on the empty tape. For the full proof, the specification and the uniformizer have to be adapted such that θ becomes the only candidate for uniformizer of S, which then implies the undecidability of the existence of a uniformizer.

In the following, we explain the simple versions of S and θ. Let Q_M denote the state set of M, q_0 denote the initial state of M, and Γ_M denote the tape alphabet of M. We can represent a configuration c of M, as a unary tree, i.e., as a string, of the form $u_1 - \cdots - u_k - q - v_1 \cdots - u_\ell$, where $u_1, \ldots, u_k, v_1, \ldots, v_\ell \in \Gamma_M$, $u_1 \ldots u_k v_1 \ldots v_\ell$ is the content of the tape of M, $q \in Q_M$ is the current control state of M, and the head of M is on v_1.

We start with the first step. Concerning the specification S, we are interested in pairs (t, t') of trees over $Q_M \cup \Gamma_M \cup \{f, a\}$ which have the form

$$\left(\begin{array}{ccccccc} f & - & f & - \cdots - & f & - & a \\ | & & | & & | & & \\ c_0 & & c_1 & & c_n & & \end{array} \;,\; \begin{array}{ccccccc} f & - & f & - \cdots - & f & - & a \\ | & & | & & | & & \\ k_1 & & k_2 & & k_m & & \end{array} \right),$$

where $m \geq n$, each c_i (resp. k_i) is a configuration of M, c_0 is the initial configuration of M on the empty tape and c_n is a halting configuration of M. Note that the numbering of the c_i starts with 0 and the numbering of the k_i with 1, this is intended. Such a pair of trees is part of the specification S if it additionally satisfies the following: There is an $i \in \{0, \ldots, n-1\}$ such that $succ(c_i) \neq k_{i+1}$, where $succ(c_i)$ is the successor configuration of c_i.

The specification S is tree-automatic. Note that for a pair (t, t') of the correct form, the configurations c_i and k_{i+1} overlap for each $i \in \{0, \ldots, n-1\}$ in $t \otimes t'$. A tree-automaton can guess a branch and verify that $succ(c_i) \neq k_{i+1}$ holds.

Now, we consider the function $\theta\colon dom(S) \to T_{Q_M \cup \Gamma_M \cup \{f,a\}}$ defined by

$$\begin{array}{ccccccc} f & - & f & - \cdots - & f & - & a \\ | & & | & & | & & \\ c_0 & & c_1 & & c_n & & \end{array} \quad \mapsto \quad \begin{array}{ccccccc} f & - & f & - \cdots - & f & - & a \\ | & & | & & | & & \\ c_1 & & c_2 & & c_n & & \end{array} .$$

Assuming that a transducer is only given input trees that have the desired form, this function is realizable by a deterministic top-down tree transducer, e.g., by some transducer that produces no output in the first step, continues at the right child and then simply copies the rest of the input tree.

Assume that M does not halt on the empty input tape and consider an input tree $t \in dom(S)$, then there are configurations c_i and c_{i+1} such that c_{i+1} is not the successor configuration of c_i. The transformation θ yields $c_{i+1} = k_{i+1}$, it follows that $succ(c_i) \neq k_{i+1}$, i.e., $(t, \theta(t)) \in S$. Conversely, assume that M halts on the empty input tape. Consider an input tree $t \in dom(S)$ such that $c_0 c_1 \cdots c_n$ is the halting configuration sequence. It follows that $k_{i+1} = succ(c_i) = c_{i+1}$ for all $i \in \{0, \ldots, n-1\}$, i.e., $(t, \theta(t)) \notin S$. Clearly, S is uniformized by θ if, and only if, M does not halt on the empty input tape.

However, the specification S does not suffice to enforce that this kind of transformation is the only possible uniformizer. This can be achieved by extending the alphabet and the specification. ◀

From the undecidability proof one can derive that the uniformization problems remain undecidable if we restrict the D↓TTs, as stated in the following two theorems. Together with the decidability result from Section 4.1 this gives an almost complete picture of the frontier between decidability and undecidability (for the case of all tree-automatic relations as specifications, and subclasses of D↓TTs as uniformizers).

▶ **Theorem 3.** *It is undecidable whether a given tree-automatic relation has a uniformization by a linear deterministic top-down tree transducer with delay bounded by 1.*

▶ **Theorem 4.** *It is undecidable whether a given tree-automatic relation has a uniformization by a synchronous deterministic top-down tree transducer.*

4 Decidability Results

In the previous section we have seen that the uniformization problem for general tree-automatic specifications is undecidable. In order to regain decidability of the uniformization problem for non-deterministic top-down specifications we present two approaches. In Section 4.1, we consider general non-deterministic top-down specifications and restrict the uniformizer, whereas in Section 4.2 we consider a restricted class of non-deterministic top-down specifications and ask whether there is a synchronous uniformizer.

4.1 Path-preserving uniformization

In this section, we consider general non-deterministic tree relations and restrict the uniformizer; we are looking for a uniformization by a deterministic path-preserving top-down transducer. We solve the following uniformization problem.

▶ **Theorem 5.** *It is decidable whether a given tree-automatic relation has a uniformization by a deterministic path-preserving top-down tree transducer.*

In the following we show that deciding whether a general non-deterministic top-down tree relation has a path-preserving uniformization reduces to deciding the winner in a safety game between two players. We show that the use of guidable tree automata [17] for the specifications makes it feasible to adapt a decision procedure presented in [18], where the uniformization problem for deterministic top-down tree relations was reduced to deciding the winner in a safety game.

Before we present the decision procedure, we need to fix some notations. Given $\Sigma = \bigcup_{i=0}^{m} \Sigma_i$, let $\mathrm{dir}_\Sigma = \{1, \dots, m\}$ be the set of directions compatible with Σ. For $\Sigma = \bigcup_{i=0}^{m} \Sigma_i$, the set Path_Σ of labeled paths over Σ is defined inductively by:

- ε is a labeled input path and each $f \in \Sigma$ is a labeled input path,
- given a labeled input path $\pi = x \cdot f$ with $f \in \Sigma_i$ $(i > 0)$ over Σ, then $\pi \cdot jg$ with $j \in \{1, \dots, i\}$ and $g \in \Sigma$ is a labeled input path.

For $\pi \in \mathrm{Path}_\Sigma$, we define the path $path(\pi) \in \mathrm{dir}_\Sigma^*$ and the word $labels(\pi) \in \Sigma^*$ induced by π inductively by:

- if $\pi = \varepsilon$ or $\pi = f$, then $path(\varepsilon) = path(f) = \varepsilon$, $labels(\varepsilon) = \varepsilon$ and $labels(f) = f$,
- if $\pi = x \cdot jf$ with $j \in \mathrm{dir}_\Sigma$, $f \in \Sigma$, then $path(\pi) = path(x) \cdot j$, $labels(\pi) = labels(x) \cdot f$.

The length $||\,||$ of a labeled path over Σ is the length of the word induced by its path, i.e., $||\pi|| = |labels(\pi)|$.

For $\pi \in \mathrm{Path}_\Sigma$ let $T_\Sigma^\pi := \{t \in T_\Sigma \mid val_t\big(path(\pi)[1,(i-1)]\big) = labels(\pi)[i] \text{ for } 1 \leq i \leq ||\pi||\}$ be the set of trees t over Σ such that π is a prefix of a labeled path through t. For a tree-automatic relation $R \subseteq T_\Sigma \times T_\Gamma$ recognized by an N↓TA $\mathcal{A}$, $\pi \in \mathrm{Path}_\Sigma$ and $q \in Q_\mathcal{A}$ let $R^\pi := \{(t, t') \in R \mid t \in T_\Sigma^\pi\}$ and $R_q^\pi := \{(t, t') \in R(\mathcal{A}_q) \mid t \in T_\Sigma^\pi\}$.

Since we consider labeled paths through trees, it is convenient to define the notion of convolution also for labeled paths. For a labeled path $x \in \mathrm{Path}_\Sigma$ with $||x|| > 0$, let $dom_x := \{u \in \mathrm{dir}_\Sigma^* \mid u \sqsubseteq path(x)\}$ and $val_x : dom_x \to \Sigma$, where $val_x(u) = labels(x)[i]$ if $u \in dom_x$ with $|u| = i + 1$. Let $x \in \mathrm{Path}_\Sigma$, $y \in \mathrm{Path}_\Gamma$ with $path(y) \sqsubseteq path(x)$ or $path(x) \sqsubseteq path(y)$, then the *convolution* of x and y is $x \otimes y$ defined by $dom_{x\otimes y} = dom_x \cup dom_y$, and $val_{x\otimes y}(u) = (val_x^\perp(u), val_y^\perp(u))$ for all $u \in dom_{x\otimes y}$, where $val_x^\perp(u) = val_x(u)$ if $u \in dom_x$ and $val_x^\perp(u) = \perp$ otherwise, analogously defined for $val_y^\perp(u)$.

Furthermore, it is useful to relax the notion of runs to labeled paths. Let $x \in \mathrm{Path}_\Sigma$, $y \in \mathrm{Path}_\Gamma$ such that $x \otimes y$ is defined, i.e., $path(y) \sqsubseteq path(x)$ or $path(x) \sqsubseteq path(y)$. We define the run of $\mathcal{A}$ on $x \otimes y$ such that it maps all nodes from $dom_{x\otimes y}$ as well as all nodes that are a direct successor of a node from $dom_{x\otimes y}$ to a state of $\mathcal{A}$. Formally, let the (partial) *run* of $\mathcal{A}$ on $x \otimes y$ be the partial function $\rho : \mathrm{dir}_\Sigma^* \to Q_\mathcal{A}$ such that $\rho(\varepsilon) = q_0^\mathcal{A}$, and for each $u \in dom_{x\otimes y}$: if $q := \rho(u)$ is defined and there is a transition $(q, val_{x\otimes y}(u), q_1, \dots, q_i) \in \Delta_\mathcal{A}$, then $\rho(u.j) = q_j$ for all $j \in \{1, \dots, i\}$. Let $path(x \otimes y) = v$ and $i \in \mathrm{dir}_\Sigma$. Shorthand, we write $\mathcal{A} : q_0^\mathcal{A} \xrightarrow{x\otimes y}_i q$, if $q := \rho(vi)$ is defined. We write $\mathcal{A} : q_0^\mathcal{A} \xrightarrow{x\otimes y} Acc$ if $rk(val_{x\otimes y}(v)) = 0$ and $(\rho(v), val_{x\otimes y}(v)) \in \Delta_\mathcal{A}$ to indicate that the (partial) run ρ of $\mathcal{A}$ on $x \otimes y$ is accepting.

We explicitly state a simple lemma that is used in several places.

▶ **Lemma 6** ([18]). *Given a ↓TA $\mathcal{A}$ and a state q of $\mathcal{A}$, the following properties are decidable:* ***1.*** $\forall t \in T_\Sigma : t \otimes \perp \in T(\mathcal{A}_q)$, ***2.*** $\exists t' \in T_\Gamma : \perp \otimes t' \in T(\mathcal{A}_q)$, ***3.*** $\exists t' \in T_\Gamma\, \forall t \in T_\Sigma : t \otimes t' \in T(\mathcal{A}_q)$.

Towards the decision procedure, we consider the relationship between the delay that a transducer introduces and uniformizability. Intuitively, if a specification is uniformized by a transducer such that the uniformizer introduces long delays between outputs, then only one path in an input tree is relevant in order to determine an output tree. We express this property by introducing the term path-recognizable function, meaning that there is a D↓TT that first deterministically reads a path from the root to a leaf in an input tree and then outputs a matching output tree. Note that such a uniformizer is always path-preserving.

Formally, we say a relation $R \subseteq T_\Sigma \times T_\Gamma$ is *uniformizable by a path-recognizable function*, if there exists a D↓TT $\mathcal{T}$ that uniformizes R such that $\Delta_\mathcal{T}$ only contains transitions of the following form: **1.** $q(f(x_1, \dots, x_i)) \to q'(x_j)$, or **2.** $q(a) \to t$, where $f \in \Sigma_i$, $i > 0$, $a \in \Sigma_0$, $q, q' \in Q_\mathcal{T}$ and $j \in \{1, \dots, i\}$ and $t \in T_\Gamma$.

This notion was introduced in [18], where it was shown to be decidable whether a top-down

deterministic relation can be uniformized by a path-recognizable function. Using guidable automata, the result carries over to general tree-automatic relations.

▶ **Theorem 7.** *It is decidable whether a given tree-automatic relation can be uniformized by a path-recognizable function.*

Given a specification, we can show that there exists a computable bound with the following property: If it is necessary for a D↓PTT to introduce delay that exceeds the bound in order to satisfy the specification, then either the remaining specification has a simple uniformization by a path-recognizable function, which is decidable by the above theorem, or is not D↓PTT-uniformizable.

Towards the definition of the game, we need one more notion. We introduce a relation that contains state transformations of a given specification automaton that a labeled path together with some output sequence on this path induces. However, we are only interested in the result of a state transformation if it suffices for a uniformizer to read this labeled path segment in an input tree to (partially) determine a matching output tree. Formally, let $x \in \mathrm{Path}_\Sigma$, $y \in \mathrm{Path}_\Gamma$ and $i \in \mathrm{dir}_\Sigma$ such that $x \otimes y$ is defined. We define the relation $\tau_{xi,y} \subseteq Q_\mathcal{A} \times Q_\mathcal{A}$ such that $(q, q') \in \tau_{xi,y}$ if there is a partial run ρ_q of $\mathcal{A}_q$ on $x \otimes y$ with $\mathcal{A}_q : q \xrightarrow{x \otimes y}_i q'$ and for each uj with $u \in dom_{x\otimes y}, uj \not\sqsubseteq path(x \otimes y)i$, and $j \in \{1, \ldots, rk\big((val_x^\perp(u), val_y^\perp(u))\big)\}$ holds

- if $r := \rho_q(uj)$ and $j \leq rk(val_x^\perp(u)), rk(val_y^\perp(u))$, then there exists $t' \in T_\Gamma$ such that $t \otimes t' \in T(\mathcal{A}_r)$ for all $t \in T_\Sigma$, and
- if $r := \rho_q(uj)$ and $rk(val_y^\perp(u)) < j \leq rk(val_x^\perp(u))$, then $t \otimes \perp \in T(\mathcal{A}_r)$ for all $t \in T_\Sigma$, and
- if $r := \rho_q(uj)$ and $rk(val_x^\perp(u)) < j \leq rk(val_y^\perp(u))$, then there exists $t' \in T_\Gamma$ such that $\perp \otimes t' \in T(\mathcal{A}_r)$.

Lemma 6 implies that it is decidable whether $(q, q') \in \tau_{xi,y}$. Basically, if q is in the domain of $\tau_{xi,y}$, then there exists a fixed (partial) output tree $s' \in S_\Gamma^{yi\circ}$ such that for each input tree $t \in T_\Sigma^x \cap dom(T(\mathcal{A}_q))$ there exists some $t' \in T_\Gamma$ such that $t \otimes (s' \cdot t') \in T(\mathcal{A}_q)$.

Now, we are ready to show that the uniformization problem posed in this section reduces to deciding the winner in a safety game, provided that the specification is given by a guidable automaton. The game is played between In and Out on a game graph parameterized by k, where In can follow any path from the root to a leaf in an input tree such that In plays one input symbol at a time. Out can either react with an output symbol, or delay the output a bounded number of times (at most $2k$ times) and react with a direction in which In should continue with his input sequence. As stated after Theorem 7, when the output delay increases to a computable bound, then uniformization is either impossible or can be realized by a path-recognizable function (Out then wins automatically, see o4. in the construction below). To make the decision procedure sound, the parameter k has to be chosen as this bound.

Given a tree-automatic relation $R \subseteq T_\Sigma \times T_\Gamma$, we assume its domain to be deterministic, otherwise no deterministic ↓TT can recognize the domain. Let R be recognized by a guidable N↓TA $\mathcal{A}$ and let $dom(R)$ be recognized by a D↓TA $\mathcal{B}$. Formally, the game graph $G^k_{\mathcal{A},\mathcal{B}}$ is constructed as follows.

- $V_{\mathsf{In}} = \{(p, q, \pi j) \in Q_\mathcal{B} \times Q_\mathcal{A} \times \mathrm{Path}_\Sigma \cdot \mathrm{dir}_\Sigma \mid \|\pi\| \leq 2k+1, \pi \in \mathrm{Path}_\Sigma, j \in \mathrm{dir}_\Sigma\} \cup 2^{Q_\mathcal{B} \times Q_\mathcal{A}}$ is the set of vertices of player In including the initial vertex $\{(q_0^\mathcal{B}, q_0^\mathcal{A})\}$.
- $V_{\mathsf{Out}} = \{(p, q, \pi) \in Q_\mathcal{B} \times Q_\mathcal{A} \times \mathrm{Path}_\Sigma \mid \|\pi\| \leq 2k+1\}$ is the set of vertices of player Out.
- From a vertex of In the following moves are possible:
  - **i1.** $(p, q, \pi j) \to (p, q, \pi j f)$ for each $f \in \Sigma$ such that $\mathcal{B}\colon p \xrightarrow{\pi}_j p'$ and there exists $(p', f, p_1, \ldots, p_i) \in \Delta_\mathcal{B}$ if $\|\pi\| < 2k+1$ (delay; In chooses the next input symbol)
  - **i2.** $\{(p_1, q_1), \ldots, (p_n, q_n)\} \to (p_j, q_j, f)$ for each $f \in \Sigma$ such that there is $(p_j, f, p_j^1, \ldots, p_j^i) \in \Delta_\mathcal{B}$ and each $j \in \{1, \ldots, n\}$ (no delay; In chooses the next direction and input symbol)

- From a vertex of Out the following moves are possible:
  - **o1.** $(p, q, f) \xrightarrow{r} \{(p_1, q_1), \ldots, (p_i, q_i)\}$ if there is $r = (q, (f, g), q_1, \ldots, q_n) \in \Delta_{\mathcal{A}}$, $(p, f, p_1, \ldots, p_i) \in \Delta_{\mathcal{B}}$, $f \in \Sigma$ is i-ary, $g \in \Sigma_\perp$ is j-ary, $n = max\{i, j\}$, and if $j > i$ there exist trees $t_{i+1}, \ldots, t_j \in T_\Gamma$ such that $\perp \otimes t_\ell \in T(\mathcal{A}_{q_l})$ for all $i < \ell \leq j$.
    (no delay; Out applies a transition; Out can pick output trees for all directions where the input has ended; In can continue from the other directions)
    Note, if $f \in \Sigma_0$, i.e., the input symbol is a leaf, then the next reached vertex is $\emptyset \in V_{\mathsf{In}}$, which is a terminal vertex.
  - **o2.** $(p, q, fj\pi) \xrightarrow{r} (p_j, q_j, \pi)$ if there is $r = (q, (f, g), q_1, \ldots, q_n) \in \Delta_{\mathcal{A}}$ such that $(q, q_j) \in \tau_{fj,g}$ and $(p, f, p_1, \ldots, p_i) \in \Delta_{\mathcal{B}}$.
    (delay; Out applies a transition, removes the leftmost input symbol and advances in direction of the labeled path ahead; Out can pick output trees for all divergent directions)
  - **o3.** $(p, q, \pi j f) \to (p, q, \pi j f j')$ for each $j' \in \{1, \ldots, i\}$ for $f \in \Sigma_i$ if $\|\pi j f\| < k + 1$
    (Out delays and chooses a direction from where In should continue)
  - **o4.** $(p, q, \pi) \to (p, q, \pi)$ if R^π_q is uniformizable by a path-recognizable function.
    (Out stays in this vertex and wins)

Note that the game graph can effectively be constructed, because Lemma 6 and Theorem 7 imply that it is decidable whether the edge constraints are satisfied.

The desired winning condition expresses that player Out loses the game if the input can be extended, but no valid output can be produced. This is represented in the game graph by a set of bad vertices B that contains all vertices of Out with no outgoing edges. If one of these vertices is reached during a play, Out loses the game. Thus, we define $\mathcal{G}^k_{\mathcal{A},\mathcal{B}} = (G^k_{\mathcal{A},\mathcal{B}}, V \setminus B)$ as safety game for Out.

The following two lemmata show that from the existence of a winning strategy a top-down tree transducer that uniformizer the relation can be obtained and vice versa.

▶ **Lemma 8.** *Given k, if Out has a winning strategy in $\mathcal{G}^k_{\mathcal{A},\mathcal{B}}$, then R is D↓PTT-uniformizable.*

The key idea in order to lift the proof in [18] from deterministic to general non-deterministic specifications is, given a guidable automaton for the specification, to turn a uniformizer into a guide for the specification automaton in order to construct a winning strategy.

▶ **Lemma 9.** *If R is D↓PTT-uniformizable, then Out has a winning strategy in $\mathcal{G}^k_{\mathcal{A},\mathcal{B}}$, where k is a number effectively computable from $\mathcal{A}$.*

As a consequence of Lemmata 8 and 9 and the fact that a winning strategy for Out in $\mathcal{G}^k_{\mathcal{A},\mathcal{B}}$ can effectively be computed, together with the fact that for each tree-automatic relation a guidable N↓TA can effectively be constructed, see [17], we immediately obtain Theorem 5.

4.2 Union of top-down deterministic specifications

In this section, we assume that $R \subseteq T_\Sigma \times T_\Gamma$ is given as the union $\bigcup_{i=1}^n R_i$ of n relations with pairwise disjoint domains, where each R_i is recognized by a D↓TA $\mathcal{A}_i$ and its domain is recognized by a D↓TA $\mathcal{B}_i$. Furthermore, we assume that the domain of the relation is D↓TA-recognizable, otherwise there exists no uniformization by a deterministic top-down tree transducer.

▶ **Example 10.** Let Σ be an input alphabet given by $\Sigma_1 = \{h\}$ and $\Sigma_0 = \{c, d\}$ and let Γ be an output alphabet given by $\Gamma_2 = \{f\}$, $\Gamma_1 = \{h\}$ and $\Gamma_0 = \{c, d\}$. We consider the relation $R \subseteq T_\Sigma \times T_\Gamma$ defined by $\{(h(t), f(t, t')) \mid t, t' \in T_\Sigma \text{ such that } t \text{ and } t' \text{ have the same leaf symbol}\}$. This specification can be obtained by the union of two deterministic top-down specifications,

one specification for each leaf symbol. Clearly, a deterministic top-down transducer can realize the specification by producing $f(t,t)$ for a unary input tree $h(t)$, e.g., by starting with $q_0(h(x_1)) \to f(q(x_1), q(x_1))$. However, there is no linear synchronous uniformizer for R, because in the first step a linear D↓TT would have to pick for either the right or the left subtree an output tree with a fixed leaf symbol. As the actual leaf symbol of the input tree is yet unknown it is not possible to fix a correct output tree.

We provide a decision procedure for the following problem.

▶ **Theorem 11.** *It is decidable whether the union of D↓TA-recognizable relations with pairwise disjoint D↓TA-recognizable domains has a uniformization by a synchronous deterministic top-down tree transducer.*

We show that the existence of a synchronous uniformizer for such a relation is a regular property over infinite trees that can be checked by a parity tree automaton. For an introduction to parity tree automata, see e.g. [24]. We define a regular infinite tree, given as the unfolding of a finite graph, such that each vertex of the infinite tree represents a node in an input tree together with a set of output nodes produced from this input node. Since the uniformizer might be non-linear, output at different positions in the output tree can depend on the same position in the input tree. Our construction bounds the number of required output choices by making the choice only depending on the state transformation that the current output sequences together with the input sequence induces.

Before we formally define the finite graph, we describe its components. Recall, $R = \bigcup_{i=1}^{n} R_i$, where R_i is recognized by a D↓TA $\mathcal{A}_i$ and $dom(R)$ is recognized by a D↓TA $\mathcal{D}$. The graph keeps track of the state of $\mathcal{D}$ on the input, and the states of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ on the produced output. For the latter we use vectors with n elements. We define a function λ_ℓ that returns the ℓth element of a vector, for each $1 \leq \ell \leq n$. Let L denote such a vector, then $\lambda_\ell(L)$ stores the information w.r.t. $\mathcal{A}_\ell$. We model that read input and produced output can be on the same or on divergent paths as follows: In case that input and output are on the same path, $\lambda_\ell(L)$ is the state of $\mathcal{A}_\ell$ on the combined input sequence and output sequence. In case that the output is mapped to a divergent path, $\lambda_\ell(L)$ is a set of states of $\mathcal{A}_\ell$ that is obtained by combining all possible input sequences with the produced output sequence. Now we are ready to formally define the graph $\mathcal{G}$:

- From a vertex v of the form $(p, \{L_1, \ldots, L_m\})$, where p is a state of $\mathcal{D}$ and each L_j is a vector of states resp. sets of states over $\mathcal{A}_1, \ldots, \mathcal{A}_n$, the following edges exist:
  - $v \to (v, f)$ if there is $(p, f, p_1 \ldots, p_i) \in \Delta_{\mathcal{D}}$ (edges for every possible input symbol)
- An edge $((p, \{L_1, \ldots, L_m\}), f) \overset{o_1,\ldots,o_m}{\to} [(p_1, Q_1), \ldots, (p_i, Q_i)]$ defining output choices $o_1, \ldots, o_m$ exists if $(p, f, p_1 \ldots, p_i) \in \Delta_{\mathcal{D}}$ and the following conditions hold:
  - $o_j \in \Gamma(X_i)$ for each $1 \leq j \leq m$, and
    (for each L_j an output o_j consisting of one symbol and directions to continue is chosen)
  - the set Q_d is constructed as follows for each $1 \leq d \leq i$:
    if for output $o_j = g(x_{j_1}, \ldots, x_{j_r})$ there is $k \in \{1, \ldots, r\}$ with $j_k = d$,
    (the kth child of the output o_j depends on the dth child of the input)
    we add a vector V_k to Q_d, where the component $\lambda_\ell(V_k)$ referring to $\mathcal{A}_\ell$ is build up from $\lambda_\ell(L_j)$ and o_j as follows:
    * if $\lambda_\ell(L_j) \in Q_{\mathcal{A}_\ell}$, say $q \in Q_{\mathcal{A}_\ell}$, (input and output are at the same position)
      and there is $(q, (f, g), q_1, \ldots, q_{max\{rk(f), rk(g)\}}) \in \Delta_{\mathcal{A}_\ell}$,
      then $\lambda_\ell(V_k) = \begin{cases} q_k & \text{if } d = k, \quad \text{(input and output continue in the same direction)} \\ \{q_k\} & \text{otherwise. (input and output continue in divergent directions)} \end{cases}$

(the corresponding transition in $\mathcal{A}_\ell$ is applied)

- if $\lambda_\ell(L_j) \in 2^{Q_{\mathcal{A}_\ell}}$, (input and output are on divergent paths)
  then set $\lambda_\ell(V_k)$ to $\emptyset$ and for each $q \in \lambda_\ell(L_j)$ and each $f' \in \Sigma$ such that there is $(q, (f', g), q_1, \ldots, q_{max\{rk(f'),rk(g)\}}) \in \Delta_{\mathcal{A}_\ell}$, add q_k to $\lambda_\ell(V_k)$.
  (all possibly reachable states in $\mathcal{A}_\ell$ are collected)

- From $[v_1, \ldots, v_i]$ an edge to v_j exists for all $1 \leq j \leq i$. (edges to all directions)
- The initial vertex is $(p_0, \{L\})$, where $L = [q_0^{\mathcal{A}_1}, \ldots, q_0^{\mathcal{A}_n}]$ and p_0 is the initial state of $\mathcal{D}$.

Now that we have defined $\mathcal{G}$, we consider the unfolding $\mathcal{H}$ of $\mathcal{G}$ which is a regular infinite tree. Consequently, each vertex of $\mathcal{H}$ is associated with a labeled path, interpreted as an input sequence π, and additionally it is associated with a bounded number of labeled paths, interpreted as output sequences produced by a transducer while reading the input sequence π. Note that different vertices of $\mathcal{H}$ may represent the same input sequence, but differ in the associated output sequences. This is a regular infinite tree that has the desired property, namely, each input sequence together with a (sufficiently large) number of possible output sequences is represented in the tree.

Our goal is to construct a parity tree automaton, whose tree language is non-empty iff R has a uniformization by a synchronous deterministic top-down tree transducer. The idea is to annotate $\mathcal{H}$ with an output strategy σ. The strategy selects for each node of the form (v, f) with $f \in \Sigma$ one child, i.e., σ fixes an output choice. Let $\mathcal{H}^\frown\sigma$ denote the tree $\mathcal{H}$ with annotations encoding σ. Given $\mathcal{H}^\frown\sigma$ and some input tree $t \in dom(R)$, the output choices defined by σ identify a unique output tree that a D↓TT can produce while reading t. For an input tree t, let $\sigma(t)$ denote the corresponding output tree. The strategy σ corresponds to a uniformizer if for all $t \in dom(R)$ holds that $(t, \sigma(t)) \in R$. The following lemma shows that the set of trees $\mathcal{H}^\frown\sigma$ such that σ corresponds to a uniformizer is a regular set of trees.

▶ **Lemma 12.** *There exists a parity tree automaton $\mathcal{C}$ that accepts exactly those trees $\mathcal{H}^\frown\sigma$ such that $(t, \sigma(t)) \in R$ for all $t \in dom(R)$.*

The next lemma shows that the uniformization problem posed in this section reduces to deciding the emptiness problem for $\mathcal{C}$. It directly implies Theorem 11 because emptiness of parity tree automata is decidable (see [24]).

▶ **Lemma 13.** *The tree language $T(\mathcal{C})$ is non-empty if, and only if, R has a uniformization by a synchronous deterministic top-down tree transducer.*

5 Conclusion

We have considered uniformization of tree-automatic relations by D↓TTs. Using the subclasses of bounded-delay, linear, and path-preserving D↓TTs, we have obtained an almost complete picture of the frontier between decidability and undecidability. We have also presented a class of tree-automatic relations for which the uniformization problem is decidable but requires, in general, non-linear uniformizers.

As further research questions it would be interesting to extend the class of specifications beyond those of tree-automatic relations. In [6] decidability results for word transformations have been obtained for deterministic rational relations, and for uniformization questions in which the delay of the uniformizer is related to the one of the specification. We plan to study extensions of these ideas from words to trees.

References

1 J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969. doi:10.1090/S0002-9947-1969-0280205-0.

2 Aranud Carayol, Christof Löding, Damian Niwiński, and Igor Walukiewicz. Choice functions and well-orderings over the infinite binary tree. *Central European Journal of Mathematics*, 8(4):662–682, 2010.

3 Alonzo Church. Logic, arithmetic and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35, 1962.

4 Thomas Colcombet and Christof Löding. Transforming structures by set interpretations. *Logical Methods in Computer Science*, 3(2):1–36, 2007. doi:10.2168/LMCS-3(2:4)2007.

5 Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, ICALP 2008*, volume 5126 of *Lecture Notes in Computer Science*, pages 398–409. Springer, 2008.

6 Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In *International Colloquium on Automata, Languages and Programming, ICALP 2016*, 2016. to appear, full version on http://arxiv.org/abs/1602.08565.

7 Hu. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications, 2007. Release October, 12th 2007. URL: http://www.grappa.univ-lille3.fr/tata.

8 R. Diestel. *Graph Theory, 2nd Edition*, volume 173 of *Graduate Texts in Mathematics*. Springer, 2000.

9 C. C. Elgot and J. E. Mezei. On relations defined by generalized finite automata. *IBM J. Res. Dev.*, 9(1):47–68, 1965.

10 Ferenc Gécseg and Magnus Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.

11 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

12 Yuri Gurevich and Saharon Shelah. Rabin's uniformization problem. *J. Symb. Log.*, 48(4):1105–1119, 1983.

13 Michael Holtmann, Łukasz Kaiser, and Wolfgang Thomas. Degrees of lookahead in regular infinite games. In *Foundations of Software Science and Computational Structures*, volume 6014 of *Lecture Notes in Computer Science*, pages 252–266. Springer, 2010. doi:/10.1007/978-3-642-12032-9_18.

14 Frederick A. Hosch and Lawrence H. Landweber. Finite delay solutions for sequential conditions. In *ICALP*, pages 45–60, 1972.

15 Kojiro Kobayashi. Classification of formal languages by functional binary transductions. *Information and Control*, 15(1):95–109, 1969.

16 Dietrich Kuske and Thomas Weidner. Size and computation of injective tree automatic presentations. In *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Proceedings*, volume 6907 of *Lecture Notes in Computer Science*, pages 424–435. Springer, 2011.

17 C. Löding. Logic and automata over infinite trees, 2009. Habilitation Thesis, RWTH Aachen, Germany.

18 Christof Löding and Sarah Winter. Synthesis of deterministic top-down tree transducers from automatic tree relations. In *Proceedings Fifth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2014, Verona, Italy, September 10-12, 2014.*, volume 161 of *EPTCS*, pages 88–101, 2014. doi:10.4204/EPTCS.161.

19 T. Milo, D. Suciu, and V. Vianu. Typechecking for xml transformers. *J. Comput. Syst. Sci.*, 66(1):66–97, 2003. doi:10.1016/S0022-0000(02)00030-2.

20 Yiannis Nichola Moschovakis. *Descriptive Set Theory*, volume 100 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Company, Amsterdam, New York, Oxford, 1980.

21 Michael O. Rabin. *Automata on Infinite Objects and Church's Problem*. American Mathematical Society, Boston, MA, USA, 1972.

22 F.P. Ramsey. On a problem of formal logic. *Proceedings of the London Mathematical Society*, 2(1):264, 1930. doi:10.1007/978-0-8176-4842-8_1.

23 Dirk Siefkes. The recursive sets in certain monadic second order fragments of arithmetic. *Arch. für mat. Logik und Grundlagenforschung*, 17:71–80, 1975.

24 Wolfgang Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 133–192. Elsevier Science Publishers, Amsterdam, 1990.

A Details for Section 3

▶ **Theorem 2.** *It is undecidable whether a given tree-automatic relation has a uniformization by a deterministic top-down tree transducer.*

Proof. We build on the description of the proof idea given in the main part. We describe the desired specification R_M which is a refined version of S such that the following holds: If M halts on the empty tape, then R_M is D↓TT-uniformizable in a θ-like fashion. Otherwise, if M does not halt on the empty tape, then R_M is not D↓TT-uniformizable.

First, we need to adapt the form of the input and output trees as follows. From now on, an input tree over Σ with $\Sigma = Q_M \cup \Gamma_M \cup \{f, g, a, b, A, B\}$ is said to be a correct coding if it is of the form

$$\begin{array}{ccccccc} \$ & - & \$ & - & \cdots & - & \$ - X \\ | & & | & & & & | \\ C_0 & & C_1 & & & & C_n \end{array} ,$$

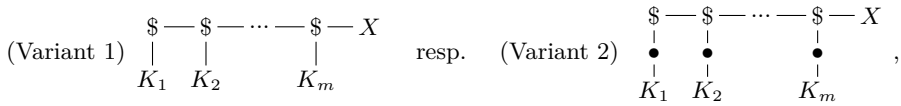
where $\$ \in \{f, g\}$, $X \in \{A, B\}$ is a special leaf symbol, and each C_i is a binary tree of the form

$$\begin{array}{ccccccccccccccc} u_1 & - & u_2 & - & \cdots & - & u_{(k-1)} & - & u_k & - & q & - & v_1 & - & v_2 & - & \cdots & - & v_{(\ell-1)} & - & v_\ell & - & \beta_{\ell+1} \\ | & & | & & & & | & & | & & | & & | & & | & & & & | & & | & & \\ \alpha_1 & & \alpha_2 & & & & \alpha_{(k-1)} & & \alpha_k & & \beta_0 & & \beta_1 & & \beta_2 & & & & \beta_{(\ell-1)} & & \beta_\ell & & \end{array} ,$$

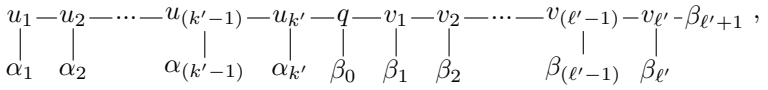
where $u_1, \ldots, u_k, v_1, \ldots, v_\ell \in \Gamma_M$, $q \in Q_M$, $\alpha_1, \ldots, \alpha_k, \beta_0, \beta_1, \ldots, \beta_\ell, \beta_{\ell+1} \in \{a, b\}$ are leaf symbols and $u_1 \ldots u_k q v_1 \ldots v_\ell$ encodes the configuration c_i. Wlog, let $|u_1 \ldots u_k| \geq 2$ and $|v_1 \ldots v_\ell| \geq 1$. Furthermore, c_0 is the initial configuration on the empty input tape and c_n is a halting configuration. The special letter X determines which requirements an output tree (additionally to being a correct coding; the form of a correct coding is specified below) has to satisfy.

We actually can show a more specific statement than the statement given in this theorem. Our goal is two give two variants of R_M such that the following holds. If M halts on the empty input tape, then the first resp. second R_M-variant can be uniformized by a linear D↓TT with delay bounded by 1 resp. by a synchronous D↓TT. Otherwise, if M does not halt on the empty input tape, then non of the variants is D↓TT-uniformizable. Therefore, we distinguish between two (very similar) variants of R_M in the remainder of this proof.

An output tree over Σ resp. over $\Sigma \cup \{\bullet\}$ is said to be a correct coding if it is of the form



where $\$ \in \{f, g\}$, $X \in \{A, B\}$, and each K_i is a binary tree of the form



where $u_1, \ldots, u_{k'}, v_1, \ldots, v_{\ell'} \in \Gamma_M$, $q \in Q_M$, $\alpha_1, \ldots, \alpha_{k'}, \beta_0, \beta_1, \ldots, \beta_{\ell'}, \beta_{\ell'+1} \in \{a, b\}$ are leaf symbols and $u_1 \ldots u_{k'} q v_1 \ldots v_{\ell'}$ encodes the configuration k_i. The occurrences of $\bullet$ are introduced because for the second variant we want to be able to construct a synchronous uniformizer for R_M if M halts on the empty tape.

We are ready to define the (two variants of) R_M. For notational convenience, let $C_i(\alpha_j)$ refer to the α_j-position in the C_i-tree and let $val\big(C_i(\alpha_j)\big)$ refer to its value, and so on. Since the two R_M-variants are very similar, the requirements given below should be satisfied in both variants if not stated explicitly otherwise. The specification R_M contains a pair of trees (t, t') from $T_\Sigma \times T_\Sigma$ resp. $T_\Sigma \times T_{\Sigma \cup \{\bullet\}}$ if conditions 1., 2., A., and B. are satisfied, or condition 3. is satisfied.

1. t is a correct coding and t' is a correct (V1- resp. V2-)coding with $m = n$.
2. Recall, w.r.t. positions, 2^i is short for the node $\underbrace{2 \ldots 2}_{i \text{ times}}$ and 2^0 is short for the root ε.

Depending on the variant, we require that

V1. $val_{t'}(2^i) = val_t(2^{i+1})$ for all $i \in \{0, \ldots, n-1\}$, resp.
V2. $val_{t'}(2^i) = val_t(2^i)$ for all $i \in \{0, \ldots, n-1\}$.

Furthermore, we require that

A. if the right-most leaf of t is A, then there is an $i \in \{0, \ldots, n-1\}$ such that $succ(c_i) \neq k_{i+1}$.

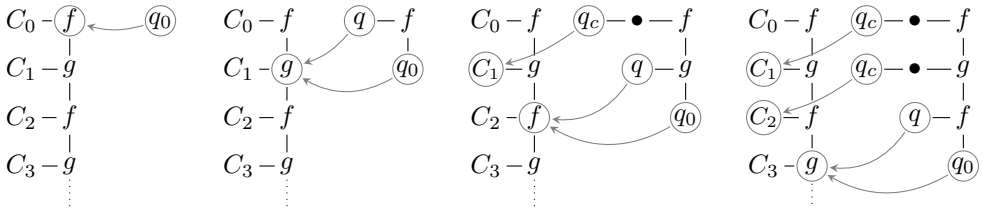
Next, we add additional requirements to enforce the desired θ-like transformation. Recall, we want to ensure that a transformation yields $k_i = c_i$ for all $i \in \{1, \ldots, n\}$. However, k_i and c_i do not overlap in $t \otimes t'$, and we have to make sure that the specification remains tree-automatic. Since the length of configurations can be arbitrarily large, we can only talk about a finite amount of positions.

The principle behind the next requirements is that a D↓TT is not aware of the next symbol in a configuration. Since a D↓TT can make no guesses, it then has to fulfill the requirements for all positions to be safe, because the current position could already be the position in question.

Therefore, in order to guarantee $k_i = c_i$ for all $i \in \{1, \ldots, n\}$, we require that

B. if the right-most leaf of t is B, then for all $i \in \{1, \ldots, n\}$ holds

- $val\big(K_i(\alpha_{k'-1})\big) = val\big(C_i(\alpha_{k-1})\big)$, and
- $val\big(K_i(u_{k'})\big) = val\big(C_i(u_k)\big)$, and
- $val\big(K_i(\alpha_{k'})\big) = val\big(C_i(\alpha_k)\big)$, and
- $val\big(K_i(q)\big) = val\big(C_i(q)\big)$, and
- $val\big(K_i(\beta_0)\big) = val\big(C_i(\beta_0)\big)$, and
- $val\big(K_i(v_1)\big) = val\big(C_i(v_1)\big)$, and
- $val\big(K_i(\beta_{\ell'-1})\big) = val\big(C_i(\beta_{\ell-1})\big)$, and
- $val\big(K_i(v_{\ell'})\big) = val\big(C_i(v_\ell)\big)$.



(a) initial config. c_0 **(b)** config. c_1 **(c)** config. c_2 **(d)** config. c_3

Figure 2 Illustration of the functioning of the D↓TT $\mathcal{T}_2$ as defined in the proof of Theorem 2. A possible configuration sequence c_0 to c_3 of $\mathcal{T}_2$ on an input tree of the form $f(C_0, g(C_1, f(C_2, g(C_3, \dots))))$ is shown. The input tree is always depicted on the left-hand side and the so far produced output tree on the right-hand side of each subfigure. The respective mapping of output positions to read input positions in shown in gray.

Finally, in order to make the specification R_M total, we include also all pairs of trees (t, t') from $T_\Sigma \times T_\Sigma$ resp. $T_\Sigma \times T_{\Sigma\cup\{\bullet\}}$ such that

3. t is not a correct coding.

We argue that R_M defines a tree-automatic specification. First, note that the conditions 1., 2., and 3. are clearly tree-automatic. A specification automaton guesses whether conditions 1., 2., A., and B. are satisfied, or whether condition 3. is satisfied. Depending on its guess, it checks that the respective requirements hold. Since the if-conditions of A. and B. are mutual exclusive, the automaton guesses the right-most leaf symbol in t and verifies that the respective then-condition is satisfied and also verifies that the guess of the leaf-symbol was correct. To verify the then-condition of A., the automaton guesses a c_i-configuration branch and verifies that $succ(c_i) \neq k_{i+1}$ is true. This can be done, because c_i and k_{i+1} overlap. To verify the then-condition of B., the automaton guesses and verifies the correct labeling of the 16 positions in question for each pair of k_i-configuration branch and successive c_i-configuration branch.

We claim that each variant of R_M is uniformizable if, and only if, M does not halt on the empty input tape. If M does not halt, then the first resp. second R_M-variant can be uniformized by a linear D↓TT $\mathcal{T}_1 = (Q_1, \Sigma, \Sigma, q_0, \Delta_1)$ with delay bounded by 1 resp. by a synchronous D↓TT $\mathcal{T}_2 = (Q_2, \Sigma, \Sigma \cup \{\bullet\}, q_0, \Delta_2)$. The part of the transition rules relevant for input trees that are a correct coding is given below. The state q_c stands for "copy"-state. An illustration of the functioning of the latter transducer is shown in Figure 2.

$$\text{Let } \Delta_1 = \{ \quad \begin{aligned} q_0(\$(x_1, x_2)) &\to q_c(x_2) && \text{for all } \$ \in \{f, g\},\\ q_c(\sigma(x_1, \dots, x_i)) &\to \sigma(q_c(x_1), \dots, q_c(x_i)) && \text{for all } \sigma \in \Sigma_i, i \geq 0 \ \}. \end{aligned}$$

$$\text{Let } \Delta_2 = \{ \quad \begin{aligned} q_0(\$(x_1, x_2)) &\to \$(q(x_2), q_0(x_2)) && \text{for all } \$ \in \{f, g\},\\ q_0(X) &\to X && \text{for all } X \in \{A, B\},\\ q(\$(x_1, x_2)) &\to \bullet(q_c(x_1)) && \text{for all } \$ \in \{f, g\},\\ q_c(\sigma(x_1, \dots, x_i)) &\to \sigma(q_c(x_1), \dots, q_c(x_i)) && \text{for all } \sigma \in \Sigma_i, i \geq 0 \ \}. \end{aligned}$$

It is easy to see that $\mathcal{T}_1$ uniformizes the first R_M-variant and $\mathcal{T}_2$ uniformizes the second R_M-variant; the same argumentation as for θ and S can be applied.

Conversely, assume M does halt on the empty input tape. We show that neither the first nor the second R_M-variant is D↓TT-uniformizable. Towards a contradiction, assume there exist uniformizer $\mathcal{U}_1$ and $\mathcal{U}_2$ of the first resp. second R_M-variant.

To begin with, we show that $\mathcal{U}_1$ produces no output in the first computation step. For the first R_M-variant it has to hold that the root symbol of the output tree is equal to the second symbol on the right-most path of the input tree (condition 2.), hence $\mathcal{U}_1$ has to delay at first.

Next we show that (apart from the first step of $\mathcal{U}_1$) both $\mathcal{U}_1$ and $\mathcal{U}_2$ work synchronously and the transformation yields $c_i = k_i$ for all $i \in \{1, \dots, n\}$ for both uniformizers. Since the argumentation is analogous for $\mathcal{U}_1$ and $\mathcal{U}_2$, in the following we only talk about $\mathcal{U}_1$ and the first R_M-variant.

Consider an input tree t that is a valid coding and let t' denote the output tree produced by $\mathcal{U}_1$. First, we show that the right-most path in t' is produced synchronously (after the initial delay of 1). We distinguish two cases. For the first case, assume at some point $\mathcal{U}_1$ produces at once more than one output symbol mapped to the right-most path. Say, $\mathcal{U}_1$ reads the node 2^{i+1} in t and produces output at the nodes 2^i and 2^{i+1} in t'. Let $val_{t'}(2^{i+1}) = f$. If the next input symbol at 2^{i+2} is g, then condition 2. is not satisfied. For the second case, assume $\mathcal{U}_1$ reads the node 2^{i+1} and produces no output. Either $\mathcal{U}_1$ continues to read at node $2^{i+1}1$ or at $2^{i+1}2$. At the left child, the c_i-configuration branch begins. If the input tree ends in B, c_i has to be read in order to make it possible to satisfy the then-condition of B., thus $\mathcal{U}_1$ has to continue reading at $2^{i+1}1$. As a consequence, the right-most path can not be accessed any longer. However, the right-hand path must also be fully read by $\mathcal{U}_1$ in order to satisfy condition 2..

Secondly, we show that $\mathcal{U}_1$ synchronously realizes $k_i = c_i$ for all $i \in \{1, \dots, n\}$. From the previous paragraph we know that output configuration branches are produced independent of the label of the rightmost leaf. Since the leaf could be a B, $\mathcal{U}_1$ must have produced t' such that the then-condition of B. is satisfied. Also we know from the previous paragraph that when $\mathcal{U}_1$ reads the first position of the c_i-configuration branch, then the output produced in this computation step will be mapped to the first position of the k_i-configuration branch.

We now show that in order to satisfy the then-condition of B. the uniformizer $\mathcal{U}_1$ must synchronously copy C_i. Let us build symbol-by-symbol a tree C_i coding a configuration of the form $u_1 \dots u_k q v_1 \dots v_\ell$ meaning we give a symbol in C_i, consider one computation step from $\mathcal{U}_1$ and react with another symbol from C_i and so on. Assume at some point the uniformizer does not exactly copy the input symbol, let us call this a copy mistake. We distinguish whether a mistake happens in the part before the q-position, at the q-position, or after the q-position in c_i.

First, assume a position $C_i(u_j)$ is read and in the corresponding computation step a copy mistake occurs, which means $\mathcal{U}_1$ either produces no output symbol, or more than one and/or the wrong output symbol. Then, we let the next input symbol be the state of the configuration c_i, i.e., we let $C_i(u_j) = C_i(u_k)$. Assume no output was produced, then $\mathcal{U}_1$ can continue to read at either $C_i(\alpha_k)$ or $C_i(q)$, but both positions have to be read in order to satisfy $val\big(K_i(\alpha_{k'})\big) = val\big(C_i(\alpha_k)\big)$ and $val\big(K_i(q)\big) = val\big(C_i(q)\big)$. Now assume, more and/or wrong output has been produced. We illustrate this case in Figure 3. Recall that $val\big(C_i(\alpha_{k-1})\big) = val\big(K_i(\alpha_{k'-1})\big)$ has to be satisfied. Since $C_i(\alpha_{k-1})$ is left of $C_i(u_{k-1})$ and $C_i(\alpha_k)$ is right of $C_i(u_{k-1})$, the $\mathcal{U}_1$-state reached at $C_i(\alpha_k)$ is independent of $val\big(C_i(\alpha_{k-1})\big)$. Note that output at $K_i(\alpha_{k'-1})$ has been produced at the earliest when $\mathcal{U}_1$ reads $C_i(\alpha_k)$, because in this step either more than one symbol has been produced, or if the symbol was wrong, then at some point later another one with $val\big(C_i(v_k)\big)$ is produced to satisfy $val\big(K_i(v_{k'})\big) = val\big(C_i(v_k)\big)$. Hence, the input symbol at $C_i(\alpha_{k-1})$ can be changed without affecting the output symbol that is produced at $K_i(\alpha_{k'-1})$. Thus, making the copy mistake would not guarantee $val\big(K_i(\alpha_{k'-1}))\big) = val\big(C_i(\alpha_{k-1}))\big)$.

Secondly, assume a copy mistake occurs at $C_i(q)$. Assume the copy mistake was to delay the output, then not both $C_i(\beta_0)$ and $C_i(v_1)$ can be read, but knowledge of both values is needed to satisfy the specification. So, assume the copy mistake was to produce more or wrong output. If an output symbol was produced that belongs to the part before the q-position of k_i, then eventually a symbol at $K_i(\alpha_{k'})$ is produced. However, this is done without knowledge of $val\big(C_i(\alpha_k)\big)$, because the output at $K_i(\alpha_{k'})$ is produced after $\mathcal{U}_1$ reads $C_i(q)$. Since $C_i(\alpha_k)$ is left of $C_i(u_k)$ and $C_i(q)$ is right of $C_i(u_k)$, the $\mathcal{U}_1$-state reached at $C_i(q)$ is independent of $val\big(C_i(\alpha_k)\big)$. Hence we can chose $val\big(C_i(\alpha_k)\big)$ to be different from $val\big(K_i(\alpha_{k'})\big)$. Or, if an output symbol was produced that belongs to the part after the q-position of k_i, then $val\big(K_i(v_1)\big)\big)$ is determined before $C_i(v_1)$ is read. Thus, $val\big(K_i(v_1)\big)\big) = val\big(C_i(v_1)\big)\big)$ is not ensured. We have ruled out that a copy mistake occurs at $C_i(q)$.

Lastly, assume a copy mistake occurs at a position $C_i(v_j)$, then let $C_i(v_j) = C_i(v_\ell)$ and the next input symbol is the leaf at $C_i(\beta_{\ell+1})$. With the same argumentation as in the previous cases, we can show that a uniformizer cannot make such a mistake.

Altogether, we can conclude that both $\mathcal{U}_1$ and $\mathcal{U}_2$ always realize $k_i = c_i$ for all $i \in \{1, \dots, n\}$. Now consider an input tree that is a correct coding and ends in A and codes the halting configuration sequence $c_0 c_1 \dots c_n$, then $succ(c_i) = k_{i+1}$ for all $i \in \{0, \dots, n-1\}$. Thus, condition A. is not satisfied. This contradicts that $\mathcal{U}_1$ and $\mathcal{U}_2$ uniformize the first resp. second R_M-variant. ◀

B Details for Section 4.1

This section is concerned with the the proof of Theorem 5.

▶ **Theorem 5.** *It is decidable whether a given tree-automatic relation has a uniformization by a deterministic path-preserving top-down tree transducer.*

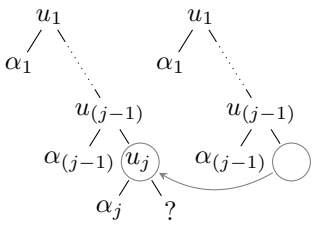
As already stated in Section 4.1, the decision procedure for the above question is an adaption of the decision procedure for deterministic top-down tree-automatic relations presented in [18]. Here, we follow the same proof structure and lift the introduced notions and auxiliary lemmas used in [18] from deterministic tree-automatic relations to general tree-automatic relations.

We start with introducing some additional notations. For some of the proofs, it is convenient to concatenate a special tree with a single letter, instead of a tree. For $t \in S_\Sigma$ and $f \in \Sigma_i, i \geq 0$, let $t \cdot f$ denote the result of replacing $\circ$ with f in t. Formally, if $i > 0$, then the result is not a tree over Σ, but we treat it as such. In particular, for a D↓TT $\mathcal{T}$, we define $\mathcal{T}(t \cdot f)$ as $\mathcal{T}(t) \cdot w[q_1, \dots, q_n]$ if there is $q(f(x_1, \dots, x_i)) \to w[q_1(x_{j_1}), \dots, q_n(x_{j_n})] \in \Delta_\mathcal{T}$ and there is a configuration $c = (t, \mathcal{T}(t), \varphi)$ with $u \in dom_{\mathcal{T}(t)}$ such that $\varphi(u) = v$, $val_t(v) = \circ$ and $val_{\mathcal{T}(t)}(u) = q$. In words, when $\mathcal{T}$ reads this special f output is produced even though f has no child nodes.

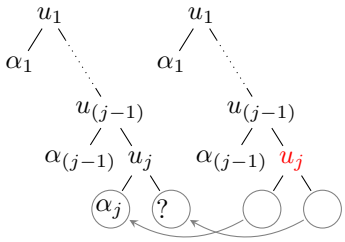
For $t \in T_\Sigma$ and $u \in \mathrm{dir}^*_\Sigma$, let $||t||^u := max\{|v| \mid v \in dom_t \text{ and } (u \sqsubseteq v \text{ or } v \sqsubseteq u)\}$. If $u \in dom_t$, then $||t||^u$ is the length of a maximal path in t through u. Otherwise, if $u \notin dom_t$, then $||t||^u$ is the length of the unique path in t that is a prefix of u.

Sometimes it is sufficient to consider only the output that is mapped to a certain path. For a D↓TT $\mathcal{T}$, an input tree $t \in T_\Sigma$ or $t \in S_\Sigma$ and a path $u \in \mathrm{dir}^*_\Sigma$, we define

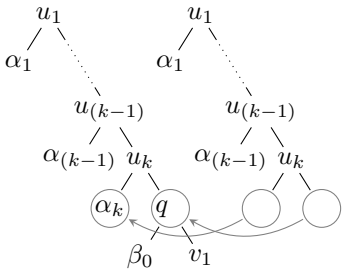
$$out_\mathcal{T}(t,u) := \{\pi \in \mathrm{Path}_\Gamma \mid \mathcal{T}(t) \in T^\pi_\Gamma(X) \text{ and } (path(\pi) \sqsubseteq u \text{ or } u \sqsubseteq path(\pi)) \text{ and } ||\mathcal{T}(t)||^u = |path(\pi)|\}$$



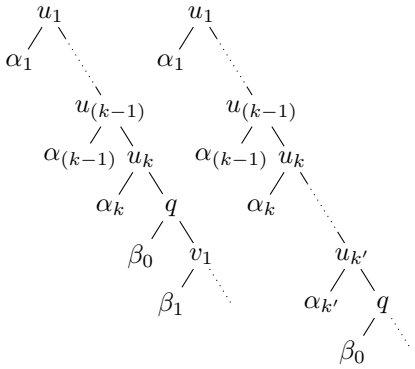
(a) $\mathcal{U}_1$ reads $C_i(u_j)$. Output produced by $\mathcal{U}_1$ from here is independent of $val(C_i(\alpha_{j-1}))$.



(b) $\mathcal{U}_1$ has produced $K_i(u_j)$ with $val(C_i(u_j)) \neq val(K_i(u_j))$.



(c) We chose the next input symbol in C_i to be the state of c_i. That makes $C_i(u_j) = C_i(u_k)$ and $K_i(u_j) \neq K_i(u_{k'})$ because $val(C_i(u_k)) = val(K_i(u_{k'}))$ must be satisfied. Thus $k' > k$.



(d) At some point $\mathcal{U}_1$ produces $K_i(u_{k'})$ with $val(C_i(u_k)) = val(K_i(u_{k'}))$. $K_i(\alpha_{k'-1})$ is a node below $K_i(u_k)$ independently produced of $val(C_i(\alpha_{k-1}))$.

Figure 3 Illustration of a copy mistake made in the part before the q-state in some c_i as described in the proof of Theorem 2. The C_i-tree is always depicted on the left, the K_i-tree on the right. The respective mapping of output positions to read input positions in shown in gray. The subfigures show that if $\mathcal{U}_1$ makes such a copy mistake then $val(C_i(\alpha_{k-1})) = val(K_i(\alpha_{k'-1}))$ can not be guaranteed, because then we can force $\mathcal{U}_1$ to produce $K_i(\alpha_{k'-1})$ independent of $val(C_i(\alpha_{k-1}))$.

to be the set of maximal labeled paths in the output tree $\mathcal{T}(t)$ through u. Note that if $||\mathcal{T}(t)||^u < |u|$, then $out_{\mathcal{T}}(t,u)$ is a singleton set and contains the maximal labeled path π in $\mathcal{T}(t)$ such that $path(\pi)$ is a prefix of u. Then, we identify $out_{\mathcal{T}}(t,u)$ with its single element.

We are ready to prove the following.

▶ **Theorem 7.** *It is decidable whether a given tree-automatic relation can be uniformized by a path-recognizable function.*

Proof. Let $R \subseteq T_\Sigma \times T_\Gamma$ be a tree-automatic relation. We assume that its domain is also top-down deterministic, otherwise no deterministic top-down tree transducer can verify its domain. For such a relation, we show that the existence of a uniformizer that realizes a path-recognizable function is a regular property over infinite trees. The key ingredient of this proof is an MSO-formula over infinite trees that asserts the existence of such a uniformizer. We can then use the equivalence between regular and MSO-definable languages over infinite trees.

The proof is split in two parts. Recall, to describe a path-recognizable function, we have to reason about labeled paths through input trees and corresponding output trees. Hence, first, we define a regular infinite tree that is suitable to specify (via MSO) a set of deterministically readable labeled paths. Secondly, we state an MSO-formula that is satisfiable by this infinite

tree if, and only if, there exists a finite state uniformizer that realizes a path-recognizable function.

For the first part, we construct a regular infinite tree by defining a finite graph and using its unfolding to obtain a regular infinite tree, see e.g. [24]. Intuitively, to be able to specify labeled paths in this tree, the node labels alternate between directions from dir_Σ and possible input symbols from Σ (if the input has not yet ended). A node labeled by $f \in \Sigma_n$, $n > 0$ has exactly n children, where the ith child is labeled by i. A direction labeled node has an f-labeled child for every $f \in \Sigma$ (or $\bot$ if it occurs below a leaf-labeled node). For a node labeled by a leaf symbol $a \in \Sigma_0$, the nodes below form a string with labels alternating between 1 and $\bot$. As a special case, the root is labeled by 0.

Formally, let $\Sigma = \bigcup_{i=0}^{m} \Sigma_i$ and let $\mathcal{B} = (Q_\mathcal{B}, \Sigma, q_0^\mathcal{B}, \Delta_\mathcal{B})$ be a D↓TA that recognizes $dom(R)$. We want to construct the desired infinite tree w.r.t. the domain of the relation. Thus, we define the graph $\mathcal{G} = (V, E)$ with vertices $V \subseteq (Q_\mathcal{B} \uplus \{p_\bot\}) \times (\Sigma_\bot \cup \{0\} \cup \mathrm{dir}_\Sigma)$ and a set of edges E that include the information of the runs of $\mathcal{B}$, and is defined by

$$
\begin{aligned}
&\big\{\big((q_0^\mathcal{B}, 0), (q_0^\mathcal{B}, f)\big) \mid f \in \Sigma\big\} \\
\cup\ &\big\{\big((p, i), (p, f)\big) \mid p \in Q_\mathcal{B}, i \in \mathrm{dir}_\Sigma, f \in \Sigma\big\} \\
\cup\ &\big\{\big((p, f), (p_j, j)\big) \mid (p, f, p_1, \ldots, p_k) \in \Delta_\mathcal{B}, f \in \Sigma_k, k > 0, j \in \{1, \ldots, k\}\big\} \\
\cup\ &\big\{\big((p, a), (p_\bot, 1)\big) \mid (p, a) \in \Delta_\mathcal{B}, a \in \Sigma_0\big\} \\
\cup\ &\big\{\big((p_\bot, 1), (p_\bot, \bot)\big), \big((p_\bot, \bot), (p_\bot, 1)\big)\big\}.
\end{aligned}
$$

Let $(q_0^\mathcal{B}, 0)$ be the initial vertex of $\mathcal{G}$ and let $t_\mathcal{G}$ denote the unfolding of $\mathcal{G}$ from this initial vertex.

Now, we give an MSO-formula that is satisfiable by $t_\mathcal{G}$ if, and only if, R can be uniformized by a finite state transducer that realizes a path-recognizable function. Let T_V^{Inf} denote the set of ranked infinite trees over V, the rank of a symbol $v \in V$ is the outdegree of v in $\mathcal{G}$. As usual, an infinite tree $t \in T_V^{\mathrm{Inf}}$ corresponds to the logical structure $\underline{t} = (dom_t, S_1^t, \ldots, S_m^t, S^t, (P_v^t)_{v \in V})$, where $S_i^t = i$th successor relation on dom_t, $S^t := \bigcup_{i=0}^{m} S_i^t$, and $P_v^t := \{u \in dom_t \mid val_t(u) = v\}$. To begin with, we introduce some simple auxiliary formulas:

$$
P_\lambda(x) := \bigvee_{p \in (Q_\mathcal{B} \uplus \{p_\bot\})} P_{(p,\lambda)}(x) \text{ for all } \lambda \in (\Sigma_\bot \cup \{0\} \cup \mathrm{dir}_\Sigma), \text{ and } P_{\mathrm{dir}}(x) := \bigvee_{i=0}^{m} P_i(x).
$$

Also, formulas for set inclusion, prefix closed sets, and partition are needed:

$$
\begin{aligned}
&X \subseteq Y := \forall x\big(X(x) \to Y(x)\big),\ \phi_{\mathrm{pre}}(X) := \forall x \forall y\big(X(y) \wedge S(x, y) \to X(x)\big), \text{ and} \\
&\phi_{\mathrm{part}}(X_1, \ldots, X_n) := \forall x \left(\bigvee_{i=1}^{n} X_i(x) \wedge \bigwedge_{i \neq j \wedge i, j \in \{1, \ldots, n\}} \neg(X_i(x) \wedge X_j(x)) \right).
\end{aligned}
$$

We give a formula that specifies an infinite labeled path. If such a labeled path contains a $\bot$-labeled node, then this labeled path describes a path through a finite input tree.

$$
\begin{aligned}
\mathrm{Path}(X) := \exists x\big(P_0(x) \wedge X(x)\big) \wedge \forall x \Big(X(x) \to \exists y \big[&S(x, y) \wedge X(y) \wedge \\
&\forall z\big(S(x, z) \wedge X(z) \to y = z\big)\big] \Big).
\end{aligned}
$$

Furthermore, we want to specify a formula that is satisfied by a set of labeled paths such that a transducer is able to deterministically read (along a labeled path from this set)

through every possible input tree. This is satisfied if for a Σ-labeled node that is part of the set, exactly one direction-labeled direct successor is specified, and in addition, for a direction-labeled node that is part of the set, every Σ-labeled direct successor is also part of the set, ensuring that a transducer can react to every input symbol.

$$\begin{aligned}\mathrm{PathSet}(X) := \exists x\big(P_0(x) \wedge X(x)\big) \wedge \forall x\big(X(x) \wedge P_{\mathrm{dir}}(x) \to \forall y(S(x,y) \to X(y))\big) \wedge \\ \forall x\big(X(x) \wedge P_\Sigma(x) \to \exists y\,(S(x,y) \wedge X(y) \wedge \forall z\,(S(x,z) \wedge X(z) \to y = z))\big).\end{aligned}$$

Ultimately, all formulas are designed to be evaluated in $\underline{t_{\mathcal{G}}}$. Thus, if $(\underline{t_{\mathcal{G}}}, X) \models \mathrm{Path}(X)$ resp. $(\underline{t_{\mathcal{G}}}, X) \models \mathrm{PathSet}(X)$, then X indeed describes a labeled path resp. a set of labeled paths that can be part of an input tree from the domain of the relation, because $t_{\mathcal{G}}$ is designed w.r.t. the domain of the relation.

Towards our desired formula, given a labeled path and a prefix of this path, we want to express that there exists an output labeling of the given prefix such that there is an accepting (partial) run of $\mathcal{A}$ on the combination of input labels and output labels on this path. Additionally, since we want to obtain a path-recognizable function in the end, we require that the partial run can be extend to a successful run for each input tree that contains this labeled path, i.e., we require that the run only depends on the given labeled path. Recall, we already have a formal notion for this. Let $\mathcal{A} = (Q_{\mathcal{A}}, \Sigma_\perp \times \Gamma_\perp, q_0^{\mathcal{A}}, \Delta_{\mathcal{A}})$ be a guidable $\downarrow$TA that recognizes R. For $x \in \mathrm{Path}_\Sigma$, $y \in \mathrm{Path}_\Gamma$ and $i \in \mathrm{dir}_\Sigma$ such that $x \otimes y$ is defined, i.e., $path(x) \sqsubseteq path(y)$ or $path(y) \sqsubseteq path(x)$, the relation $\tau_{xi,y} \subseteq Q_{\mathcal{A}} \times Q_{\mathcal{A}}$ includes a pair of states (q, q') of $\mathcal{A}$, if there is a partial run ρ on $x \otimes y$ of $\mathcal{A}_q$ with $\mathcal{A} : q \xrightarrow{x \otimes y}_i q'$ and there exists a fixed partial output tree $s \in S_\Gamma^{yi\circ}$ such that ρ can be extend to a successful run of $\mathcal{A}_q$ on each input tree $t \in T_\Sigma^x \cap dom(T(\mathcal{A}_q))$ together with an output tree $s \cdot t' \in T_\Gamma^y$, that is, an output tree whose first part is always s.

In this proof, we let $\tau_{\perp i,y}$ resp. $\tau_{xi,\perp}$ denote $\tau_{\varepsilon i,y}$ resp. $\tau_{xi,\varepsilon}$. Also, we extend this notion; we define the relation $\tau_{x,y} \subseteq Q_{\mathcal{A}} \times \{Acc\}$ if x is a labeled path that ends in a leaf, such that $(q, Acc) \in \tau_{x,y}$ if there is a partial run ρ on $x \otimes y$ of $\mathcal{A}_q$ with $\mathcal{A} : q \xrightarrow{x \otimes y} Acc$ and there exists a fixed partial output tree $s \in S_\Gamma^y$ as described above.

The following formula expresses the requirements stated above. Let $Q_{\mathcal{A}} = \{q_1, \ldots, q_n\}$, $q_1 = q_0^{\mathcal{A}}$, and $\Gamma = \{g_1, \ldots, g_\ell\}$, we then write

$$\begin{aligned}R_{\mathrm{Path}}(X,Y) := \mathrm{Path}(X) \wedge Y \subseteq X \wedge \phi_{\mathrm{pre}}(Y) \wedge \\ \exists Y_{g_1} \ldots \exists Y_{g_\ell} \exists Y_\perp \exists X_{q_1} \ldots \exists X_{q_n} \Bigg(\phi_{\mathrm{part}}(Y_{g_1}, \ldots, Y_{g_\ell}, Y_\perp) \wedge \phi_{\mathrm{part}}(X_1, \ldots, X_n) \wedge \\ \forall y\big(Y(y) \to \neg Y_\perp(y)\big) \wedge \exists x \exists y \big(P_0(x) \wedge S(x,y) \wedge X(x) \wedge X_{q_1}(y)\big) \wedge \\ \forall x \forall d \forall y \Big[S(x,d) \wedge S(d,y) \wedge P_{\mathrm{dir}}(d) \wedge X(x) \wedge X(d) \wedge X(y) \wedge \neg\big(P_\perp(y) \wedge Y_\perp(y)\big) \to \\ \Big(\bigvee_{\substack{(q,(f,g),q_1,\ldots,q_j) \in \Delta_{\mathcal{A}}, \\ (q,q_i) \in \tau_{fi,g}}} X_q(x) \wedge P_f(x) \wedge Y_g(x) \wedge P_i(d) \wedge X_{q_i}(y) \Big) \Big] \wedge \\ \exists x \Big[X(x) \wedge \neg P_{\mathrm{dir}}(x) \to \Big(\bigvee_{(q,(f,g)) \in \Delta_{\mathcal{A}}} X_q(x) \wedge P_f(x) \wedge Y_g(x) \Big) \Big] \Bigg).\end{aligned}$$

We need one more auxiliary formula. To obtain a finite state transducer that realizes a path-recognizable function, we require that only finitely many different output trees are needed. If only finitely many different output trees are needed, then the length of the output sequences that are mapped to the relevant labeled paths is bounded, that is, the set of needed

output sequences is finite. The next formula is satisfied by a finite set that describes prefixes of labeled paths. If all needed output sequences stay inside the positions specified by the set, then this implies that finitely many different output sequences suffice.

$$\phi_{\text{pre,fin}}(X) := \phi_{\text{pre}}(X) \wedge \forall X_1 \Big(\text{Path}(X_1) \rightarrow \exists x \Big[X_1(x) \wedge X(x) \wedge \neg \big(\exists y \big[S(x,y) \wedge X_1(y) \wedge X(y) \big] \big) \Big] \Big).$$

Now, we are ready to state the desired formula. The formula describes that there is a set of deterministically readable labeled paths such that for each labeled path from this set that describes a path through a finite input tree there is a matching output and additionally the set of used matching outputs (along the relevant paths) is finite.

$$\phi_{\text{unif}} := \exists X \exists Y \Big(\text{PathSet}(X) \wedge Y \subseteq X \wedge \phi_{\text{pre,fin}}(Y) \wedge \forall X_1 \big[X_1 \subseteq X \wedge \text{Path}(X_1) \wedge \exists x \big(X_1(x) \wedge P_\perp(x) \big) \rightarrow \exists Y_1 \big(Y_1 \subseteq Y \wedge R_{\text{Path}}(X_1, Y_1) \big) \big] \Big).$$

Since the equivalence between MSO-definable tree languages and regular tree languages is effective, we can construct an automaton $\mathcal{A}_{\phi_{\text{unif}}}$ such that $T(\mathcal{A}_{\phi_{\text{unif}}})$ is equivalent to $T(\phi_{\text{unif}}) := \{ t \in T_V^{\text{Inf}} \mid \underline{t} \models \phi_{\text{unif}} \}$.

Finally, we claim $t_{\mathcal{G}} \in T(\mathcal{A}_{\phi_{\text{unif}}})$ if, and only if, R is uniformizable by a finite state transducer that realizes a path-recognizable function. If R is uniformized by a path-recognizable function, then there exists an assignment of the variables such that $\underline{t_{\mathcal{G}}} \models \phi_{\text{unif}}$. From a D↓PTT $\mathcal{T}$ that uniformizes R and realizes a path-recognizable function we can obtain a valuation as follows. Recall that a position in $t_{\mathcal{G}}$ can be either interpreted as a labeled path, or as a path resp. a position in a tree if we disregard the labels. Concerning the valuation of X, a position is included in the set X if it describes a (prefix of a) labeled path that the uniformizer reads from the root to a leaf in an input tree in order to produce an output tree. Generally, such a uniformizer specifies a set of finite labeled paths of arbitrary length. However, since ϕ_{unif}, more specifically its subformula $\text{PathSet}(X)$, requires X to describe a set of infinite labeled paths, for each finite labeled path π that the uniformizer reads, X also includes the positions that describe its infinite extension $\pi(1\perp)^\omega$. Then X describes the desired set of infinite labeled paths. Concerning the valuation of Y, we have to consider the output trees that $\mathcal{T}$ produces. These are only finitely many different trees, say $t_1, \ldots, t_k$. We choose Y as the subset of X such that a position from X is included in Y if there is (at least) one t_i such that this position describes a position that occurs in dom_{t_i}. Since the set $dom_{t_1} \cup \cdots \cup dom_{t_k}$ is finite, also Y is finite. Then $\underline{t_{\mathcal{G}}} \models \phi_{\text{unif}}$, because for each finite labeled path π from the set X the set Y describes a prefix of this path such that there is an output sequence o that can be mapped to this prefix (obtained from one of the t_is) such that $\mathcal{A} : q_0^{\mathcal{A}} \xrightarrow{\pi \otimes o} Acc$ and furthermore $(q_0^{\mathcal{A}}, Acc) \in \tau_{\pi,o}$, that is, this partial run can be extended to a successful run of $\mathcal{A}$ on each input tree $t \in T_\Sigma^\pi$. That indeed $(q_0^{\mathcal{A}}, Acc) \in \tau_{\pi,o}$ holds can be seen as follows. Let $T_\Sigma^\pi \times \{t_o\}$ with $t_o \in T_\Gamma^o$ be the relation that $\mathcal{T}$ realizes when π is read. This subset of R can be recognized by a D↓TA $\mathcal{C}$. Since $\mathcal{A}$ is guidable, there is a mapping g such that $(\mathcal{C}, g)$ guides $\mathcal{A}$. Hence, $g(\rho)$ restricted to $\pi \otimes o$ is the same for each run ρ of $\mathcal{C}$ on $t \otimes t_o$ for some $t \in T_\Sigma^\pi$, because $\mathcal{C}$ is deterministic. Thus, $(q_0^{\mathcal{A}}, Acc) \in \tau_{\pi,o}$.

If $t_{\mathcal{G}} \in T(\mathcal{A}_{\phi_{\text{unif}}})$, then there exists a valuation of the variables such that a regular ([21], see also [24]) deterministically readable set of labeled paths can be obtained that captures every input tree. Since this set is regular, it is recognizable by a finite state transducer. Additionally, the valuation of Y yields a global bound such that the size of each output mapped to a relevant path through an input tree remains inside this bound.

Consequently, only finitely many different output trees are needed. Hence, there exists a finite state uniformizer that realizes a path-recognizable function, such a uniformizer can e.g., be constructed in the following way. Clearly, a deterministic finite state top-down tree transducer that reads the relevant path in an input tree can be obtained from a regular set of labeled paths. Also, a global bound of the length of needed output sequences on the relevant paths is known. Since there are only finitely many output sequences that remain inside the bound, the transducer can gradually compute $\tau_{\pi,o}$ for each such output sequence o, where π is the read input sequence. Eventually, when a leaf is reached, for at least one o holds that $(q_0^{\mathcal{A}}, Acc) \in \tau_{\pi,o}$. The transducer then produces an output tree matching to the (chosen) accepting state transformation. ◀

We have seen that it is decidable whether a tree-automatic relation can be uniformized by a path-recognizable function.

Now, we establish a connection between long output delay and path-recognizable functions. Therefore, we first introduce profiles for labeled path segments w.r.t. a given relation automaton. We define the profile of a labeled path segment xi to be the set that contains all possible state transformations $\tau_{xi,y}$ induced by x together with some y of same or smaller length. Formally, let $x \in \mathrm{Path}_\Sigma$ and $i \in \mathrm{dir}_\Sigma$, we define the *profile* of xi to be $P_{xi} = (P_{xi,=}, P_{xi,<}, P_{xi,\varepsilon})$ with

$$P_{xi,=} := \{\tau_{xi,y} \mid |y| = |x|\},\ P_{xi,<} := \{\tau_{xi,y} \mid y \neq \varepsilon \text{ and } |y| < |x|\}, \text{ and } P_{xi,\varepsilon} := \{\tau_{xi,y} \mid y = \varepsilon\}.$$

From $P_{x_1i_1}$ and $P_{x_2i_2}$ the profile $P_{x_1i_1x_2i_2}$ is uniquely determined, i.e.,

$$\begin{aligned} P_{x_1i_1x_2i_2,=} &:= \{\tau_1 \circ \tau_2 \mid \tau_1 \in P_{x_1i_1,=}, \tau_2 \in P_{x_2i_2,=}\}, \\ P_{x_1i_1x_2i_2,<} &:= \{\tau_1 \circ \tau_2 \mid \tau_1 \in P_{x_1i_1,=}, \tau_2 \in P_{x_2i_2,<}\} \cup \{\tau_1 \circ \tau_{x_2i_2,\varepsilon} \mid \tau_1 \in P_{x_1i_1,<}\}, \text{ and} \\ P_{x_1i_1x_2i_2,\varepsilon} &:= \{\tau_{x_1i_1,\varepsilon} \circ \tau_{x_2i_2,\varepsilon}\}. \end{aligned}$$

Thus, the concatenation of $P_{x_1i_1}$, $P_{x_2i_2}$ is naturally defined by $P_{x_1i_1} \cdot P_{x_2i_2} = P_{x_1i_1x_2i_2}$. A segment $xi \in (\Sigma\mathrm{dir}_\Sigma)^*\mathrm{dir}_\Sigma$ of a labeled path is called *idempotent* if $P_{xi} = P_{xixi}$.

As a consequence of Ramsey's Theorem [22], we obtain the next lemma. Note that the lemma refers to idempotent factors w.r.t. N↓TAs , however the proof is the same as in [18] for idempotent factors w.r.t. D↓TAs.

▶ **Lemma 14.** *There exists a bound $K \in \mathbb{N}$ such that each labeled path $\pi \in \mathrm{Path}_\Sigma$ with $||\pi|| \geq K$ contains an idempotent factor.*

Proof. Ramsey's Theorem yields that for any number of colors c and any number r, there exists a number $K \in \mathbb{N}$ such that if the edges of a complete graph with at least K vertices are colored with c colors, then the graph must contain a complete subgraph with r vertices such that all edges have the same color, c.f. [8].

Let $\pi \in \mathrm{Path}_\Sigma$ with the factorization $\pi = f_1j_1 \dots j_{n-1}f_n$, $f_1, \dots, f_n \in \Sigma$ and $j_1, \dots, j_{n-1} \in \mathrm{dir}_\Sigma$. Consider the complete graph $G = (V, E, col)$ with edge-coloring $col : E \to Cols$, where $V := \{1, \dots, n\}$, $E := V \times V$, $Cols$ is the finite set of profiles and $col(e) := P_{f_ij_i \dots f_kj_k}$ if $e = (i, k)$ for all $e \in E$. If there exist $i, j, k \in \mathbb{N}$ with $i < j < k \leq n$ such that the edges (i, k), (i, j) and (j, k) have the same color, i.e., the respective profiles are the same, then π has a factorization that contains an idempotent factor.

As a consequence of Ramsey's Theorem, for $r = 3$ and $c = |Cols|$, if $n \geq K$, then π must contain an idempotent factor. ◀

We introduce some additional notation on how to split a tree in parts and how to repeat a part of a tree that contains an idempotent factor. For a special tree $s \in S_\Sigma$, we inductively

define the special tree $s^n \in S_\Sigma$ by $s^n := s^{n-1} \cdot s$ and $s^0 := \circ$ for $n \in \mathbb{N}$. Let $x, y \in \mathrm{Path}_\Sigma$, $i, j \in \mathrm{dir}_\Sigma$, and $path(x)i = u$, $path(y)j = v$. For a tree $t \in T_\Sigma^{xiy}$, we introduce shorthand notations $t_{[:u]}$, $t_{[u:uv]}$, and $t_{[uv:]}$ to denote $t[\circ/u]$, $t[\circ/uv]|_u$, and $t|_{uv}$, respectively. Note that $t = t_{[:u]} \cdot t_{[u:uv]} \cdot t_{[uv:]}$. Furthermore, let $y \neq \varepsilon$ and yj be an idempotent factor, we fix $t^n_{(u,v)}$ to be the tree that results from *repeating the idempotent factor* n times. More formally, we define

$$t^n_{(u,v)} := t_{[:u]} \cdot t^n_{[u:uv]} \cdot t_{[uv:]} \text{ for } n \in \mathbb{N}.$$

If it is clear from the context to which idempotent factor we refer, then we leave out the subscript.

The following Lemma establishes the connection between long output delay and path-recognizable functions. Basically, the lemma states that if there exists a uniformization by a D↓PTT such that an idempotent path segment can be repeated any number of times and the length of the output on the repetition is bounded, i.e., the output delay is unbounded, then there also exists a uniformization by a path-recognizable function.

▶ **Lemma 15.** *Given a tree-automatic relation R, $x, y \in \mathrm{Path}_\Sigma$, $i, j \in \mathrm{dir}_\Sigma$ with $path(x)i = u$, $path(y)j = v$, $y \neq \varepsilon$ and yj idempotent. If R^{xiy} is uniformized by a D↓PTT $\mathcal{T}$ such that $||out_\mathcal{T}(t_{[:u]} \cdot t^n_{[u:uv]}, uv^n)|| \leq ||x||$ for each $n > 0$ and for each $t \in T_\Sigma^{xiy}$, then R^{xiy} can be uniformized by a path-recognizable function.*

Proof. Recall the proof of Theorem 7, to show the statement of this lemma it suffices to show that the MSO-sentence ϕ_{unif} constructed from a guidable ↓TA $\mathcal{A}$ for R^{xiy} is satisfied by $\underline{t_\mathcal{G}}$ constructed from a D↓TA $\mathcal{B}$ for its domain. We state the sentence again:

$$\phi_{\text{unif}} := \exists X \exists Y \Big(\mathrm{PathSet}(X) \wedge Y \subseteq X \wedge \phi_{\text{pre,fin}}(Y) \wedge \\ \forall X_1 \big[X_1 \subseteq X \wedge \mathrm{Path}(X_1) \wedge \exists x \big(X_1(x) \wedge P_\perp(x) \big) \rightarrow \exists Y_1 \big(Y_1 \subseteq Y \wedge R_{\mathrm{Path}}(X_1, Y_1) \big) \big] \Big).$$

A suitable valuation for X, that is, a valuation describing a deterministically readable regular set of labeled paths, can be obtained as follows. Since $||out_\mathcal{T}(t_{[:u]} \cdot t^n_{[u:uv]}, uv^n)|| \leq ||x||$ for each $n > 0$ and for each $t \in T_\Sigma^{xiy}$, there is a state s of $\mathcal{T}$ that is reached at the node uv^{m_1} and again at uv^{m_2} for some $m_1 < m_2$ in a tree $t \in T_\Sigma^{xiy^{m_2-1}jy}$. As a consequence, $\mathcal{T}_s$ can read only one labeled path in a tree, because $\mathcal{T}$ is a path-preserving uniformizer and since the delay can be arbitrarily large when $\mathcal{T}$ is in s the path-preserving property is only ensured if a single path is read. Let P denote the set of labeled paths read by $\mathcal{T}_s$. Then the valuation of X is chosen to describe $\{xiyj\} \cdot P \cdot \{(1\perp)^\omega\}$. We append $(1\perp)^\omega$ to each path, because formally the set X has to describe infinite paths.

We now show that for each finite labeled path π from the root to a leaf that is described by X there exists a suitable output tree t_o such that for each $t \in T_\Sigma^\pi$ holds $(t, t_o) \in R^{xiy}$. The labeled path π is of the form $xiyjz$, let $path(z) = w$. For a $t \in T_\Sigma^\pi$ let t^n denote the tree obtained from t, where the idempotent factor yj is repeated n times. Since the delay can get arbitrary large when reading the idempotent factor, we can pick some k such that $||out_\mathcal{T}(t^k, uv^k w)|| < |uv^k|$ and in a computation of $\mathcal{T}$ on t^k the uniformizer is in state s at the node uv^k. Since $\mathcal{T}$ is a D↓PTT it follows that $T_\Sigma^{xi(yj)^k z} \times \mathcal{T}(t^k) \subseteq R^{xiy}$. This subset of R^{xiy} is D↓TA-recognizable, say by a D↓TA $\mathcal{C}$. Then $\mathcal{C}$ can guide $\mathcal{A}$. Let g be a mapping such that $(\mathcal{C}, g)$ guides $\mathcal{A}$. Let ρ be the deterministic run of $\mathcal{C}$ on $t^k \otimes \mathcal{T}(t^k)$, and let $g(\rho) = \rho'$ be the run of $\mathcal{A}$ on $t^k \otimes \mathcal{T}(t^k)$ which looks as follows:

$$\mathcal{A} : q_0 \xrightarrow{x \otimes o_1}_i q_1 \xrightarrow{(yj)^{k-1} y \otimes o_2}_j q_2 \xrightarrow{z \otimes \varepsilon} Acc$$

with $||o_2|| \leq ||(yj)^{k-1}y||$. Note since $\mathcal{C}$ is deterministic, for each tree t' from $T_\Sigma^{xi(yj)^k z}$ the accepting guided run of $\mathcal{A}$ on $t' \otimes \mathcal{T}(t^k)$ has the same form along $xi(yj)^k z$. Hence, $(q_0, q_1) \in \tau_{xi,o_1}$, $(q_1, q_2) \in \tau_{(yj)^k,o_2}$ and $(q_2, Acc) \in \tau_{z,\varepsilon}$ Also, since yj is idempotent, there is some $o_3 \in \mathrm{Path}_\Gamma$ such that $\tau_{(yj)^k,o_2} = \tau_{yj,o_3}$ with $||o_3|| \leq ||y||$. Let t_o denote an output tree corresponding to τ_{xiyjz,o_1io_3} Consequently, a run of $\mathcal{A}$ on $t \otimes t_o$ has the following form:

$$\mathcal{A} : q_0 \xrightarrow{x\otimes o_1}_i q_1 \xrightarrow{y\otimes o_3}_j q_2 \xrightarrow{z\otimes\varepsilon} Acc,$$

i.e., $(t, t_o) \in R^{xiy}$ and the corresponding MSO-formula describing a successful run can be satisfied. That is, the formula $R_{\mathrm{Path}}(X_1, Y_1)$, where $X_1 \subseteq X$ corresponds to $\pi = xiyjz$ and $Y_1 \subseteq Y$ corresponds to o_1io_3.

We have seen that for each finite labeled path π from the root to a leaf that is described by X there exists a suitable output tree t_o, furthermore the size of such a t_o is bounded by the length of xiy, thus only finitely many different t_os are needed. A suitable valuation of Y has to be a finite set representing the needed output trees. As argued above, finitely many t_os suffice, and hence Y can be chosen as a finite set representing the needed t_os. ◀

As we have seen, if a transducer that uniformizes a relation introduces long output delay, then the relation can also be uniformized by a path-recognizable function.

We introduce a complexity measure for D↓PTTs w.r.t. the lookahead that the transducer introduces. This will help us reason about the behavior of a uniformizer with minimal complexity. The idea is to measure the complexity of a uniformizer by counting the introduced lookahead. We do not consider lookahead-paths that could be part of a path-recognizable function, because this is generally an infinite set of paths.

For a tree-automatic relation $R \subseteq T_\Sigma \times T_\Gamma$, a D↓PTT $\mathcal{T}$ that uniformizes R, and a limit $l \in \mathbb{N}$, we define the *lookahead-path-language* $\mathrm{LPL}_\mathcal{T}(R, l)$ such that $\pi \in \mathrm{Path}_\Sigma$ is included if the following conditions hold:

- There is a $t \in T_\Sigma^\pi \cap dom(R)$ with $(t, q_0^\mathcal{T}, \varphi_0) \to_\mathcal{T} (t, t_1, \varphi_1) \to_\mathcal{T} \cdots \to_\mathcal{T} (t, t_n, \varphi_n)$ such that there is $u_i \in D_{t_i} \cap dom_\pi$ with $u_i \sqsubseteq \varphi_i(u_i) \sqsubseteq path(\pi)$ and $|u_i| < l$ and $\varphi_n(u_n) = path(\pi)$ and it occurs output delay w.r.t. u_i in (t, t_i, φ_i) for all $i \in \{1, \ldots, n\}$, and
- R^π is not uniformizable by a path-recognizable function.

Note, $\mathrm{LPL}_\mathcal{T}(R, l)$ is a prefix-closed set. Furthermore, we define $value(\mathrm{LPL}_\mathcal{T}(R, l))$ to be the sum of all lengths of all labeled paths, i.e., $\sum_{\pi \in \mathrm{LPL}_\mathcal{T}(R,l)} ||\pi||$.

▶ **Lemma 16.** *The set* $\mathrm{LPL}_\mathcal{T}(R, l)$ *is finite.*

Proof. We prove this statement by contradiction. Assume $\mathrm{LPL}_\mathcal{T}(R, l)$ is infinite. Then $\mathrm{LPL}_\mathcal{T}(R, l)$ contains labeled paths of arbitrary length. We pick some $\pi \in \mathrm{LPL}_\mathcal{T}(R, l)$ such that $||\pi|| > l + l \cdot K|Q_\mathcal{T}|$ with K from Lemma 14. Now, we prove that R^π is uniformizable by a path-recognizable function. This is a contradiction to $\pi \in \mathrm{LPL}_\mathcal{T}(R, l)$.

Pick any $t \in T_\Sigma^\pi \cap dom(R)$ and let $c_0 = (t, q_0^\mathcal{T}, \varphi_0)$. The configuration sequence witnessing membership of π in $\mathrm{LPL}_\mathcal{T}(R, l)$ has length $> l + l \cdot K|Q_\mathcal{T}|$. For each $j \in \{0, \ldots, l-1\}$, we can thus pick a subsequence of configurations $c_1 = (t, t_1, \varphi_1)$, $c_2 = (t, t_2, \varphi_2)$, $\ldots$, $c_K = (t, t_K, \varphi_K)$ with $c_0 \to_\mathcal{T}^* c_1 \to_\mathcal{T}^* \cdots \to_\mathcal{T}^* c_K$ such that there is $s \in Q_\mathcal{T}$ and $u_i \in D_{t_i} \cap dom_\pi$ with $val_{t_i}(u_i) = s$ for all $i \in \{1, \ldots, K\}$ and furthermore $l + j \cdot K|Q_\mathcal{T}| < |\varphi_1(u_1)|$ and $|\varphi_K(u_K)| \leq l + (j+1) \cdot K|Q_\mathcal{T}|$.

Let π_j denote the jth segment of length $K|Q_\mathcal{T}|$ of π starting after the first l letters of π. Together with Lemma 14 it follows that each such segment contains an idempotent factor yj_2 such that $\pi = xj_1yj_2z$ with $path(x_1)j_1 = \varphi_m(u_m)$ and $path(xj_1y)j_2 = \varphi_n(u_n)$ for some

$m < n \leq K$ w.r.t. a suitable subsequence. There are at least l such segments in π, because $||\pi|| > l + l \cdot K|Q_{\mathcal{T}}|$.

Since $\pi \in \mathrm{LPL}_{\mathcal{T}}(R, l)$, there is at least one segment π_j such that $\mathcal{T}$ does not produce output while reading the idempotent factor in π_j. Otherwise, $\mathcal{T}$ would produce at least l output symbols while reading π, which is a contradiction to $\pi \in \mathrm{LPL}_{\mathcal{T}}(R, l)$. Consider a subsequence that yields an idempotent factor yj_2 such that $\pi = xj_1yj_2z$ with $path(x_1)j_1 = \varphi_m(u_m)$ and $path(xj_1y)j_2 = \varphi_n(u_n)$ for some $m < n \leq K$ such that $\mathcal{T}$ produces no output while reading yj_2. Let $\varphi_m(u_m) = v_m$ and $\varphi_n(u_n) = v_n$. Then, since $||out_{\mathcal{T}}(t_{[:v_m]} \cdot t^k_{[v_m:v_n]}, path(\pi))|| < l$ for all $k \in \mathbb{N}$, Lemma 15 implies that R^π can be uniformized by a path-recognizable function. ◀

From the above lemma it follows directly that $value(\mathrm{LPL}_{\mathcal{T}}(R, l)) \in \mathbb{N}$. This allows us to compare uniformizers by comparing their values.

For the following proofs we need an auxiliary lemma that allows us to translate a computation of a uniformizer step-by-step into a run of a non-deterministic specification automaton by using only the part of the input tree that the uniformizer has read so far. In particular this means if the uniformizer reads a lookahead-path, then we require that the already constructed part of the run of the specification automaton depends only on the lookahead. Recall that we use guidable automata for the specifications. The following lemma states that a uniformizer can be turned into a guide for the specification automaton that has the desired property.

▶ **Lemma 17.** *Let R be a tree-automatic relation and let $\mathcal{T}$ be a D↓PTT that uniformizes R. Then, there exists an infinite state N↓TA $\mathcal{A}_{\mathcal{T}}$ such that $R(\mathcal{A}_{\mathcal{T}}) = R(\mathcal{T})$ with the following property:*

Let $t \in dom(R) \cap T_\Sigma^{xiy}$, if $xiy \in \mathrm{LPL}_{\mathcal{T}}(R, ||x||)$ and $\mathcal{T}(t[\circ/path(xiy)] \cdot val_t(path(xiy))) \in T^o_{\Gamma \cup Q_{\mathcal{T}}}$ with $path(x) = path(o)$, then there exists a run of ρ of $\mathcal{A}_{\mathcal{T}}$ on $t \otimes \mathcal{T}(t)$ such that $(\rho(\varepsilon), \rho(path(x)i)) \in \tau_{xi,o}$.

Proof. The relation $R(\mathcal{T})$ is generally not D↓TA-recognizable, thus we have to construct an N↓TA that recognizes $R(\mathcal{T})$. However, from a run of $\mathcal{T}$ on a tree $t \in dom(R)$, it will be possible to deterministically construct an accepting run of $\mathcal{A}_{\mathcal{T}}$ on $t \otimes \mathcal{T}(t)$.

We now present the construction of $\mathcal{A}_{\mathcal{T}} = (Q_{\mathcal{A}_{\mathcal{T}}}, \Sigma_\perp \times \Gamma_\perp, q_0^{\mathcal{A}_{\mathcal{T}}}, \Delta_{\mathcal{A}_{\mathcal{T}}})$ given $\mathcal{T} = (Q_{\mathcal{T}}, \Sigma, \Gamma, q_0^{\mathcal{T}}, \Delta_{\mathcal{T}})$. We define the set of states $Q_{\mathcal{A}_{\mathcal{T}}}$ as $Q_A \cup Q_D \cup Q_T$ with $q_0^{\mathcal{T}}$ as initial state, where

a) $Q_A := \{s \cdot p \mid s \in S_\Gamma, p \in Q_{\mathcal{T}}\}$ is the set of states indicating that output is ahead or one the same level as the input, and
b) $Q_D := \{(\pi, s \cdot p) \mid \pi \in \mathrm{Path}_\Sigma, s \in S_\Gamma, p \in Q_{\mathcal{T}}\} \cup \{(\pi, t) \mid \pi \in \mathrm{Path}_\Sigma, t \in T_\Gamma\}$, is the set of states indicating that there is output delay, and
c) $Q_T := \{t \mid t \in T_\Gamma\}$ is the set of states that are used to recognize $T_\Sigma \times \{t\}$ for each $t \in Q_T$.

Now we present the construction of $\Delta_{\mathcal{A}_{\mathcal{T}}}$, for that we make an observation about the possible configurations of $\mathcal{T}$. Let (t, t', φ) be a configuration of $\mathcal{T}$ such that there occurs delay w.r.t. a node u, for example $\varphi(u) = v$, $v \sqsubset u$ and $vi \sqsubseteq u$, i.e., output is ahead. Assume in the next computation step output is produced, in order to satisfy the restriction that input and output have to be on the same path, the right-hand side of the applied rule has to be of the form $w[p(x_i)]$, where w is a 1-context. Otherwise, for some output of the form $w'[\ldots, p(x_i), \ldots, q(x_j), \ldots]$, we would obtain a successor configuration (t, t'', φ') with $\varphi'(vj) = u'$ and $vj \not\sqsubseteq u \sqsubset u'$, i.e., input and output are no longer on the same path. In case that there is output delay w.r.t. u, i.e., $\varphi(u) = v$, $u \sqsubset v$ and $ui \sqsubseteq v$, in a computation the next rule that is applied which produces output has to be of the form

$g[t_1, \dots, t_{i-1}, w[\dots], t_{i+1}, \dots, t_n]$ with $t_1, \dots, t_{i-1}, t_{i+1}, \dots, t_n \in T_\Gamma$. Thereby, a successor configuration (t, t'', φ') is reached with $\varphi(u') = v'$ such that $u \sqsubset ui \sqsubseteq u'$ and $v \sqsubset v'$, i.e., input and output are on the same path.

First, we describe how $\Delta_{\mathcal{A}_\mathcal{T}}$ is constructed w.r.t. a configuration of $\mathcal{T}$ such that input and output position are the same (in d)), or the output position is ahead of the input position (in e)). The part that is ahead is stored in the states.

d) For $p \in Q_\mathcal{T}$ and $p(f(x_1, \dots, x_i)) \to g(p_1(x_1), \dots, p_i(x_i), t_{i+1}, \dots, t_n) \in \Delta_\mathcal{T}$, we add

$$(p, (f, g), p_1, \dots, p_i, t_{i+1} \dots, t_n) \text{ to } \Delta_{\mathcal{A}_\mathcal{T}},$$

for $p \in Q_A$ and $p(f(x_1, \dots, x_i)) \to g(t_1, \dots, t_{j-1}, w[q(x_j)], t_{j+1}, \dots, t_n) \in \Delta_\mathcal{T}$, we add

$$(p, (f, g), t_1, \dots, t_{j-1}, w[q], t_{j+1} \dots, t_n) \text{ to } \Delta_{\mathcal{A}_\mathcal{T}}, \text{and}$$

e) for $s \cdot p \in Q_A$ and $p(f(x_1, \dots, x_i)) \to w'[q(x_j)] \in \Delta_\mathcal{T}$ such that s is of the form $g(t_1, \dots, t_{j-1}, w[\dots], t_{j+1}, \dots, t_n)$, we add

$$(s \cdot p, (f, g), t_1, \dots, t_{j-1}, w[w'[q]], t_{j+1} \dots, t_n) \text{ to } \Delta_{\mathcal{A}_\mathcal{T}}.$$

Secondly, we describe how $\Delta_{\mathcal{A}_\mathcal{T}}$ is constructed w.r.t. a configuration of $\mathcal{T}$ such that input and output position are the same and $\mathcal{T}$ delays the output (in f)), or the output is already delayed (in g)) and $\mathcal{T}$ might delay further. The uniformizer $\mathcal{T}$ reads deterministically through the input and at some point produces output, the corresponding transitions we construct allow us to guess a possible lookahead-path (in f) resp. g)) and verify the correctness of the guess (in h)). In the state space the lookahead-path as well as the output of the transition that is applied by $\mathcal{T}$ is stored.

f) For $p \in Q_\mathcal{T}$, and $p(f(\dots)) \to p_1(x_j), p_1(f_1(\dots)) \to p_2(x_{j_1}), \dots, p_{n-1}(f_{n-1}(\dots)) \to p_n(x_{j_{n-1}}) \in \Delta_\mathcal{T}$, and $p_n(f_n(\dots)) \to g(t_1, \dots, t_{j-1}, w[\dots], t_{j+1}, \dots, t_m) \in \Delta_\mathcal{T}$, we add

$$(p, (f, g), t_1, \dots, t_{j-1}, (\pi, w[\dots]), t_{j+1}, \dots, t_n) \text{ to } \Delta_{\mathcal{A}_\mathcal{T}},$$

where $\pi = f_1 j_1 f_2 j_2 \dots f_n$,

g) for $(fiyjf_1, p_1) \in Q_D$, $p_1(f_1(\dots)) \to p_2(x_{j_1}), \dots, p_{n-1}(f_{n-1}(\dots)) \to p_n(x_{j_{n-1}}) \in \Delta_\mathcal{T}$, and $p_n(f_n(\dots)) \to g(t_1, \dots, t_{j-1}, w[\dots], t_{j+1}, \dots, t_m) \in \Delta_\mathcal{T}$, we add

$$((fiy, p_1), (f, g), t_1, \dots, t_{i-1}, (yz, w[\dots], t_{i+1}, \dots, t_n) \text{ to } \Delta_{\mathcal{A}_\mathcal{T}},$$

where $z = f_1 j_1 f_2 j_2 \dots f_n$.

Note that the constructions from f) and g) cause the non-determinism of $\mathcal{A}_\mathcal{T}$. The guess that was made in f) resp. g) is then verified, meaning the input symbol must match the first symbol in the stored lookahead-path and the output symbol must match the root symbol of the stored output.

h) For $(fiy, w[\dots]) \in Q_D$ such that $w[\dots]$ is of the form $g(t_1, \dots, t_{j-1}, w'[\dots], t_{j+1}, \dots, t_n)$, we add

$$((fiy, w[\dots]), (f, g), t_1, \dots, t_{i-1}, (y, w'[\dots]), t_{i+1} \dots, t_n) \text{ to } \Delta_{\mathcal{A}_\mathcal{T}}.$$

It follows directly from the construction that $T(\mathcal{A}_\mathcal{T})$ recognizes $R(\mathcal{T})$. We still have to show the statement of the lemma concerning the runs of $\mathcal{A}_\mathcal{T}$. For every $t \in dom(R)$, we can show by induction on the height of a node $v \in dom_t$ that there is a run of $\mathcal{A}_\mathcal{T}$ on $t \otimes \mathcal{T}(t)$ with the following properties:

1. If there exists no configuration $c = (t, t', \varphi)$ of $\mathcal{T}$ on t such that there is $u \in D_{t'}$ with $\varphi(u) = v$, then $\rho(v) = \mathcal{T}(t)|_v$.
2. If there exists a configuration $c = (t, t', \varphi)$ of $\mathcal{T}$ on t such that there is $u \in D_{t'}$ with $\varphi(u) = v$ and $v \sqsubseteq u$, then $\rho(v) = \mathcal{T}(t[\circ/v])|_v$.
3. If there exists a configuration $c = (t, t', \varphi)$ of $\mathcal{T}$ on t such that there is $u \in D_{t'}$ with $\varphi(u) = v$ and $u \sqsubset v$, then $\rho(v) = (y, \mathcal{T}(t[\circ/v'] \cdot val_t(v'))|_v)$, where $xiy \in \mathrm{Path}_\Sigma$ s.t. $t \in T_\Sigma^{xiy}$ with $path(x)i = v$, $path(xiy) = v'$ and $||\mathcal{T}(t[\circ/v'])||^v < |v|$, $||\mathcal{T}(t[\circ/v'] \cdot val_t(v'))||^v \geq |v|$. In words, when the input symbol at node v' is read by $\mathcal{T}$, then output is produced that is mapped to v.

Recall the statement given in the lemma: Let $t \in dom(R) \cap T_\Sigma^{xiy}$, if $xiy \in \mathrm{LPL}_\mathcal{T}(R, ||x||)$ and $\mathcal{T}(t[\circ/path(xiy)] \cdot val_t(path(xiy))) \in T^o_{\Gamma \cup Q_\mathcal{T}}$ with $path(x) = path(o)$, then there exists a run of ρ of $\mathcal{A}_\mathcal{T}$ on $t \otimes \mathcal{T}(t)$ such that $(\rho(\varepsilon), \rho(path(x)i)) \in \tau_{xi,o}$.

From the above induction it follows that $\mathcal{A}_\mathcal{T}$ has a run ρ on $t \otimes \mathcal{T}(t)$ such that for each $v \sqsubseteq path(x)i$ with $v \neq \varepsilon$ a state $\rho(v)$ as described in 3. is reached. For $v = \varepsilon$ the run yields $\rho(\varepsilon) = q_0^\mathcal{T}$ (item 2. applies). Note that the form of the states only depend on $\mathcal{T}(t[\circ/path(xiy)] \cdot val_t(path(xiy)))$. Furthermore, since xiy is a lookahead-path, we have $\mathcal{T}(t[\circ/path(xiy)] \cdot val_t(path(xiy))) = \mathcal{T}(s[\circ/path(xiy)] \cdot val_s(path(xiy)))$ for each $s \in T_\Sigma^{xiy}$. Consequently, by construction of $\mathcal{A}_\mathcal{T}$, there exists a run ρ' on $s \otimes \mathcal{T}(s)$ such that $\rho(v) = \rho'(v)$ for each $v \sqsubseteq path(x)i$. Hence, there is partial run of the form $\mathcal{A}_\mathcal{T} \colon \rho(\varepsilon) \xrightarrow{x \otimes o}_i \rho(path(x)i)$ such that this run can be extended to a successful run for each tree from $dom(R) \cap T_\Sigma^{xiy}$, i.e., $(\rho(\varepsilon), \rho(path(x)i)) \in \tau_{xi,o}$. ◀

The next lemma basically states that for a uniformizer with minimal complexity it is not necessary to further delay the output if the read lookahead-path already contains an idempotent factor, unless the relation has to be uniformized by a path-recognizable function.

▶ **Lemma 18.** *Let R be a tree-automatic relation, let $x \in \mathrm{Path}_\Sigma$, let $l \leq ||x||$ and let $\mathcal{T}$ be a D↓PTT with $x \in \mathrm{LPL}_\mathcal{T}(R, l)$ that uniformizes R such that $value(\mathrm{LPL}_\mathcal{T}(R, l)) \leq value(\mathrm{LPL}_\mathcal{U}(R, l))$ for each D↓PTT $\mathcal{U}$ with $x \in \mathrm{LPL}_\mathcal{U}(R, l)$ that uniformizes R.*

Let $y, y_1, y_2 \in \mathrm{Path}_\Sigma$, $i, d, j \in \mathrm{dir}_\Sigma$ with $y = y_1 d y_2$, $y_2 j$ is idempotent, $P_{y_1 d} = P_{y_2 j}$. If $xiy \in \mathrm{LPL}_\mathcal{T}(R, l)$, then $xiyj\pi \notin \mathrm{LPL}_\mathcal{T}(R, l)$ for each $\pi \in \mathrm{Path}_\Sigma$.

Proof. Proof by contradiction. Assume $xiy \in \mathrm{LPL}_\mathcal{T}(R, l)$ and there is $xiyj\pi \in \mathrm{LPL}_\mathcal{T}(R, l)$ for some $\pi \in \mathrm{Path}_\Sigma$. Towards a contradiction, first, we show that there is a uniformizer $\mathcal{U}$ such that $xiy_1 d y_2$ is never a prefix of a labeled path in $\mathrm{LPL}_\mathcal{U}(R, l)$, but xiy_1 is a prefix of some labeled paths in $\mathrm{LPL}_\mathcal{U}(R, l)$. Intuitively, this means, it is in fact not necessary for a uniformizer to read the idempotent factor $y_2 j$. Consequently, the lookahead-paths can be shortened. Secondly, we show that $value(\mathrm{LPL}_\mathcal{U}(R, l)) < value(\mathrm{LPL}_\mathcal{T}(R, l))$.

For the first part of the proof, we construct a uniformizer $\mathcal{U}$ based on $\mathcal{T}$. Let $path(xiy_1) = u$, $path(y_2) = v$ and we fix t_{y_2} to be a special tree from $T_\Sigma^{y_2 j \cdot \circ}$. For a tree $t \in T_\Sigma^{xiy_1}$, we denote by t' the tree $t_{[:ud]} \cdot t_{y_2} \cdot t_{[ud:]} \in T_\Sigma^{xiy_1 d y_2} = T_\Sigma^{xiy}$ that is obtained from t by inserting t_{y_2}. The idea behind the construction of $\mathcal{U}$ is that for an input tree $t \notin T_\Sigma^{xiy_1}$, $\mathcal{U}$ works like $\mathcal{T}$. Otherwise, for an input tree $t \in T_\Sigma^{xiy_1}$, the behavior of $\mathcal{U}$ is based on the computation of $\mathcal{T}$ on t'.

For this, we have to verify whether a labeled path that has xiy_1 as prefix is read. This can be done by copying for the first $||xiy_1||$ computation steps the behavior of $\mathcal{T}$ and additionally storing the so far read labeled path in the state space. If some $(s_1, z_1 d_1) \in Q_\mathcal{T} \times \mathrm{Path}_\Sigma \mathrm{dir}_\Sigma$ at a σ_1-labeled node with $z_1 d_1 \sigma_1 \not\sqsubseteq xiy_1$ is reached, then a labeled path that does not have xiy_1 as prefix is read and $\mathcal{U}$ switches to $\mathcal{T}_{s_1}$. Otherwise, a state $(s_2, z_2 d_2) \in Q_\mathcal{T} \times \mathrm{Path}_\Sigma$ at

a σ_2-labeled node with $xiy_1 = z_2d_2\sigma_2$ is reached, from then on $\mathcal{U}$ continues differently than $\mathcal{T}_{s_2}$.

Since by assumption $xiyj\pi \in \mathrm{LPL}_{\mathcal{T}}(R,l)$, also its prefix $xiy \in \mathrm{LPL}_{\mathcal{T}}(R,l)$. Thus, in a computation of $\mathcal{T}$ on $t' \in T_\Sigma^{xiy}$, a configuration $c = (t', t'', \varphi)$ is reached such that there is $\alpha \in D_{t''}$ with $\alpha \sqsubseteq \varphi(\alpha) = path(xiy)j = udvj$. Let $val_{t''}(\alpha) = s \in Q_{\mathcal{T}}$. Then $\mathcal{U}$ continues to read at the dth child, that is, at the node $ud \in dom_t$ and simulates the computation of $\mathcal{T}_s$ on $t'|_{udvj}$. This can be done because $t'|_{udvj} = t|_{ud}$ by definition of t'. In the process, $\mathcal{U}$ reads the labeled path through $t|_{ud}$ that $\mathcal{T}_s$ chooses to read in $t'|_{udvj}$ to produce output that is mapped to xiy_1dy_2, i.e., output that is mapped to the path through $udvj$ in t'.

Concerning the produced output, $\mathcal{U}$ behaves as follows. Assume $\mathcal{U}$ reads the node udw in t, then in the simulation $\mathcal{T}$ reads the node $udvjw$ in t'. Let $udvjw = \alpha$. We distinguish three cases. If $||out_{\mathcal{T}}(t'_{[:\alpha]} \cdot val_{t'}(\alpha), udv)|| < ||x||$, then $\mathcal{U}$ produces the same output as $\mathcal{T}$ would have and continues to read. Otherwise, if $||x|| \leq ||out_{\mathcal{T}}(t'_{[:\alpha]} \cdot val_{t'}(\alpha), udv)|| < ||xiy_1dy_2||$, then $\mathcal{U}$ does not produce output and just continues to read. Eventually, if $||out_{\mathcal{T}}(t'_{[:\alpha]} \cdot val_{t'}(\alpha), udv)|| = ||xiy_1dy_2||$, then $\mathcal{U}$ produces an output tree t_o compatible to a labeled path o, where o is chosen as described below.

In order to choose output suitable for t, we consider the computation of $\mathcal{T}$ on t'. Let $\mathcal{T}(t') \in T_\Gamma^{o_1io_2do_3}$ with $||x|| = ||o_1||$, $||y_1|| = ||o_2||$ and $||y_2|| = ||o_3||$. From $\mathcal{T}$ we can construct a (possibly infinite state) $\downarrow$TA $\mathcal{A}_{\mathcal{T}}$ as described in Lemma 17 that recognizes $R(\mathcal{T})$. By construction, there is a run of $\mathcal{A}_{\mathcal{T}}$ on $t' \otimes \mathcal{T}(t')$ such that $(\rho(\varepsilon), \rho(udvj)) \in \tau_{xiy_1dy_2j, o_1io_2do_3}$. Since $\mathcal{A}$ is guidable there exists a mapping g such that $(\mathcal{A}_{\mathcal{T}}, g)$ guides $\mathcal{A}$. Hence, there exists a partial run $g(\rho)$ of $\mathcal{A}$ on $xiy_1dy_2 \otimes o_1io_2do_3$ of the form

$$\mathcal{A} : q_0^{\mathcal{A}} \xrightarrow{x \otimes o_1}_i q_1 \xrightarrow{y_1 \otimes o_2}_d q_2 \xrightarrow{y_2 \otimes o_3}_j q_3,$$

such that this partial run can be extended to a successful run of $\mathcal{A}$ on $t' \otimes \mathcal{T}(t')$. In other words, $(q_0^{\mathcal{A}}, q_1) \in \tau_{xi,o_1}$, $(q_1, q_2) \in \tau_{y_1d,o_2}$ and $(q_2, q_3) \in \tau_{y_2j,o_3}$. Since y_2j is idempotent and $P_{y_1d} = P_{y_2j}$, we obtain $P_{y_1d} = P_{y_1dy_2j}$. Thus we can pick an output $o \in \mathrm{Path}_\Gamma$ such that $||o|| = ||y_1||$ and $y_1 \otimes o$ induces the same state transformation as $y_1dy_2 \otimes o_2do_3$ on $\mathcal{A}$ w.r.t. direction d, and let t_o be the tree compatible to o.

After producing t_o, the transducer $\mathcal{U}$ switches to $\mathcal{T}_{s_k}$ at the kth child of udw in t, if s_k is the state that $\mathcal{T}$ would be in when $\mathcal{T}$ reads the node $udvjwk$ in t'.

Now we show that $t \otimes \mathcal{U}(t) \in R$. From above it follows that there is a run of $\mathcal{A}$ on $t \otimes \mathcal{U}(t)$ of the form

$$\mathcal{A} : q_0^{\mathcal{A}} \xrightarrow{x \otimes o_1}_i q_1 \xrightarrow{y_1 \otimes o}_d q_3.$$

By construction of $\mathcal{U}$ we have $\mathcal{T}(t')|_{udvj} = \mathcal{U}(t)|_{ud}$ and also $t'|_{udvj} = t|_{ud}$. Since there is a successful run ρ of $\mathcal{A}$ on $t' \otimes \mathcal{T}(t')$ with $\rho(udvj) = q_3$ and since $t'|_{udvj} \otimes \mathcal{T}(t')|_{udvj}$ is accepted by $A_{\mathcal{T}}$ from q_3, we obtain $t \otimes \mathcal{U}(t) \in R$. Hence, $\mathcal{U}$ uniformizes R.

For the second part, we now show $value(\mathrm{LPL}_{\mathcal{U}}(R,l)) < value(\mathrm{LPL}_{\mathcal{T}}(R,l))$. We first remark, that if $z \in \mathrm{LPL}_{\mathcal{U}}(R,l)$ with $xiy_1 \not\sqsubseteq z$, then also $z \in \mathrm{LPL}_{\mathcal{T}}(R,l)$, because on paths that do not have xiy_1 as prefix, $\mathcal{U}$ works like $\mathcal{T}$. Also, if $z \in \mathrm{LPL}_{\mathcal{U}}(R,l)$ with $z \sqsubseteq xiy_1$, then $z \in \mathrm{LPL}_{\mathcal{T}}(R,l)$, because for the first $||xiy_1||$ computation steps, $\mathcal{U}$ works like $\mathcal{T}$.

Otherwise, if $z \in \mathrm{LPL}_{\mathcal{U}}(R,l)$ with $xiy_1j \sqsubseteq z$, let z be of the form xiy_1jz_1 for some $z_1 \in \mathrm{Path}_\Sigma$ and let $path(z_1) = w$. We show $xiy_1dy_2jz_1 \in \mathrm{LPL}_{\mathcal{T}}(R,l)$. Consider an input tree $t' \in T_\Sigma^{xiy_1dy_2jz_1}$ that is obtained from a tree $t \in T_\Sigma^{xiy_1jz_1}$ by inserting t_{y_2}. Then, from the construction of $\mathcal{U}$ follows that $\mathcal{T}$ reads $xiy_1dy_2jz_1$ in t'. Since $xiy_1dz_1 \in \mathrm{LPL}_{\mathcal{U}}(R,l)$, we know $||out_{\mathcal{U}}(t_{[:udw]}, u)|| < l$. Moreover, it follows from the construction of $\mathcal{U}$ that if $||out_{\mathcal{U}}(t_{[:udw]}, u)|| < l$, then also $||out_{\mathcal{T}}(t'_{[:udvjw]}, u)|| < l$, because $l \leq ||x||$ and by construction $out_{\mathcal{T}}(t'_{[:udvjw]}, udv) = out_{\mathcal{U}}(t_{[:udw]}, u)$ as long as $||out_{\mathcal{T}}(t'_{[:udvjw]}, udv)|| \leq ||x||$. We

can conclude that $xiy_1dy_2jz_1 \in \mathrm{LPL}_{\mathcal{T}}(R,l)$ if $R^{xiy_1dy_2jz_1}$ is not uniformizable by a path-recognizable function. Since $xiy_1jz_1 \in \mathrm{LPL}_{\mathcal{U}}(R,l)$, we know $R^{xiy_1jz_1}$ is not uniformizable by a path-recognizable function. Towards a contradiction, assume $R^{xiy_1dy_2jz_1}$ is uniformizable by a path-recognizable function. Then, from a uniformizer for $R^{xiy_1dy_2jz_1}$ that realizes a path-recognizable function, we can easily obtain a uniformizer for $R^{xiy_1jz_1}$ that realizes a path-recognizable function, because $Py_1dy_2j = P_{y_1d}$. This is a contradiction to $xiy_1jz \in \mathrm{LPL}_{\mathcal{U}}(R,l)$.

Altogether, for each labeled path $z \in \mathrm{LPL}_{\mathcal{U}}(R,l)$, there is either $z \in \mathrm{LPL}_{\mathcal{T}}(R,l)$ if $z \not\sqsubseteq xiy_1$ or $z \sqsubseteq xiy_1$, or $xiy_1dy_2jz_1 \in \mathrm{LPL}_{\mathcal{T}}(R,l)$ if z is of the form xiy_1jz_1 for some $z_1 \in \mathrm{Path}_\Sigma$. That means, $value(\mathrm{LPL}_{\mathcal{U}}(R,l)) \leq value(\mathrm{LPL}_{\mathcal{T}}(R,l))$. However, by assumption there is some $\pi \in \mathrm{Path}_\Sigma$ such that $xiyj\pi \in \mathrm{LPL}_{\mathcal{T}}(R,l)$ and by construction we obtain $xiy_1j\pi \in \mathrm{LPL}_{\mathcal{U}}(R,l)$. Thus, $value(\mathrm{LPL}_{\mathcal{U}}(R,l)) < value(\mathrm{LPL}_{\mathcal{T}}(R,l))$. This is a contradiction to $\mathcal{T}$ being a uniformizer for R with $x \in \mathrm{LPL}_{\mathcal{T}}(R,l)$ that has minimal complexity w.r.t. l. ◀

Now, we are ready to show that deciding whether a tree-automatic relation R is D↓PTT-uniformizable reduces to deciding the winner in the safety game $\mathcal{G}^k_{\mathcal{A},\mathcal{B}}$ between In and Out presented in Section 4.1 for a suitable k.

▶ **Lemma 8.** *Given k, if Out has a winning strategy in $\mathcal{G}^k_{\mathcal{A},\mathcal{B}}$, then R is D↓PTT-uniformizable.*

Proof. Assume that Out has a winning strategy in the safety game $\mathcal{G}_{\mathcal{A},\mathcal{B}}$, then there is also a positional one cf. [11]. We can represent a positional winning strategy by a function $\sigma\colon V_{\mathsf{Out}} \to V$. We construct a deterministic P↓TT $\mathcal{T} = (Q, \Sigma, \Gamma, q_0^{\mathcal{A}}, \Delta)$ from such a positional winning strategy σ as follows:

First, we describe how to translate moves of type $o1.$ and type $o3.$ to transition rules, that is, moves of Out that lead to a vertex of In.

1. For each $\sigma\colon (p,q,f) \overset{r}{\mapsto} \{(p_1,q_1),\dots,(p_i,q_i)\}$ with $r = (q,(f,g),q_1,\dots,q_n) \in \Delta_{\mathcal{A}}$:
   - add $q(f(x_1,\dots,x_i)) \to g(q_1(x_1),\dots,q_j(x_j))$ to Δ if $j \leq i$, or
   - add $q(f(x_1,\dots,x_i)) \to g(q_1(x_1),\dots,q_i(x_i),t_{i+1},\dots,t_j)$ to Δ if $j > i$

   where $f \in \Sigma_i$, $g \in \Gamma_j$ and $t_{i+1},\dots,t_j \in T_\Gamma$ are chosen according to the r-edge constraints in (p,q,f).
2. For each $\sigma\colon (p,q,\pi jf) \mapsto (p,q,\pi jfj')$ add $(q,\pi j)(f(x_1,\dots,x_i)) \to (q,\pi jfj')(x_{j'})$ to Δ.

If the strategy σ defines a sequence of moves of Out of type $o2.$, then this corresponds to an output sequence that is produced without reading further input. Each output of these moves can be represented by a special tree s as follows. A move of type $o2.$ has the form $(p,q,fj\pi) \overset{r}{\to} (p',q',\pi)$ with $r = (q,(f,g),q_1,\dots,q_n)$ and $q' = q_j$. Then, let $s = g(t_1,\dots,t_{j-1},\circ,t_{j+1},\dots,t_n) \in S_\Sigma$ be the special tree, where each $t_\ell \in T_\Gamma$ is chosen according to the r-edge constraints in $(p,q,fj\pi)$ for $\ell \neq j, 1 \leq \ell \leq n$. Eventually, the strategy defines a move of Out of type $o1.$, $o3.$, or $o4.$, otherwise σ is not a winning strategy. These parts of the strategy are transformed as follows:

3. For each $\sigma\colon (p,q,\pi jf) \overset{r_1}{\mapsto} \dots \overset{r_{\ell-1}}{\mapsto} (p',q',\pi' jf) \mapsto (p',q',\pi' jfj')$ add

$$(q,\pi j)(f(x_1,\dots,x_i)) \to s_1 \cdot \ldots \cdot s_{\ell-1} \cdot (q',\pi' jfj')(x_{j'}) \text{ to } \Delta,$$

   where each $s_i \in S_\Gamma$ is a special tree corresponding to the r_i-edge in the ith move.

4. For each $\sigma\colon (p, q, \pi j f) \overset{r_1}{\mapsto} \dots \overset{r_{\ell-1}}{\mapsto} (p', q', f) \overset{r_\ell}{\mapsto} \{(p_1, q_1), \dots, (p_i, q_i)\}$ add

$$(q, \pi j)(f(x_1, \dots, x_i)) \to s_1 \cdot \ldots \cdot s_{\ell-1} \cdot s \text{ to } \Delta,$$

where each $s_i \in S_\Gamma$ is a special tree corresponding to the r_i-edge in the ith move and s is an output corresponding to r_ℓ constructed as described in step 1.

Lastly, if such a (possibly empty) sequence ends with a move of Out of type $o4.$, that is, a move that indicate uniformizability by a path-recognizable function, then this is transformed as follows:

5. For each $\sigma\colon (p, q, \pi j f) \overset{r_1}{\mapsto} \dots \overset{r_{\ell-1}}{\mapsto} (p', q', \pi' j f) \mapsto (p', q', \pi' j f)$ add

$$(q, \pi j)(f(x_1, \dots, x_i)) \to s_1 \cdot \ldots \cdot s_{\ell-1} \cdot q_u(\dots) \text{ to } \Delta,$$

where each $s_i \in S_\Gamma$ is a special tree corresponding to the r_i-edge in the ith move and construct a D↓PTT $\mathcal{U}$ that uniformizes $R_{q'}^{\pi' j f}$ as described in the proof of Theorem 7 and switch to $\mathcal{U}$ in $s_{\ell-1}$, where $q_u(\dots)$ is the right-hand side of the transition that $\mathcal{U}$ takes after reading $\pi' j f$.

We now verify that $\mathcal{T}$ defines a uniformization of R. Clearly, $R(\mathcal{T}) \subseteq R$ because every rule of $\mathcal{T}$ either corresponds to a rule of $\mathcal{A}$ or is a switch to a uniformizer that realizes a path-recognizable function for some subset of R. We have to show $dom(R(\mathcal{T})) = dom(R)$. Let $t \in dom(R)$. We can show by induction on the number of steps needed to reach a configuration from the initial configuration $(t, q_0^{\mathcal{A}}, \varphi_0)$ that for each configuration $c = (t, t', \varphi)$ such that $D_{t'} \neq \emptyset$, in other words $t' \notin T_\Gamma$, there exists a successor configuration c'. The induction hypothesis states that in c, if $u \in D_{t'}$ with $\varphi(u) = v$, $val_t(v) = f \in \Sigma$, $val_{t'}(u) = (q, yj) \in Q_{\mathcal{A}} \times \mathrm{Path}_\Sigma \cdot \mathrm{dir}_\Sigma$, then the following conditions are satisfied:

- $t \in T_\Sigma^{xiyjf}$ for some $x \in \mathrm{Path}_\Sigma$ and $i \in \mathrm{dir}_\Sigma$ with $path(x)i = u$ and $path(xiy)j = v$, and
- in a play according to σ, the vertex $(p, q, yjf) \in V_{\mathsf{Out}}$ is reached after a sequence of moves corresponding to the labeled path $xiyjf$ with $p \in Q_{\mathcal{B}}$ such that $\mathcal{B}\colon q_0^{\mathcal{B}} \xrightarrow{x}_i p$.

Since σ is a winning strategy, σ defines the next move of Out from (p, q, yjf). Consequently, there exists a corresponding transition with left-hand side $(q, yj)(f(x_1, \dots, x_{rk(f)}))$ in $\Delta_{\mathcal{T}}$ that is applicable at u in t'. This guarantees the existence of a successor configuration c'.

If $val_t(v) \in \Sigma_0$, i.e., if $\mathcal{T}$ reads a leaf, then by construction the transition that is applicable at u in t' is a rule from step 1., or step 4., where the right-hand side is a tree over Γ.

If the transition that is applicable at u in t' is a rule from step 5., then $\mathcal{T}$ switches to a uniformizer that realizes a path-recognizable function on the remainder of the input tree. Hence, eventually a tree over Γ is produced.

From the above induction it follows that $(t, q_0^{\mathcal{A}}, \varphi_0) \to_{\mathcal{T}}^* (t, \mathcal{T}(t), \varphi)$ with $D_{\mathcal{T}(t)} = \emptyset$, i.e., $\mathcal{T}(t) \in T_\Gamma$, because in each computation step one input symbol is read and eventually, a leaf is reached and the output is a tree over Γ. ◀

Now we show the other direction.

▶ **Lemma 9.** *If R is D↓PTT-uniformizable, then Out has a winning strategy in $\mathcal{G}^k_{\mathcal{A},\mathcal{B}}$, where k is a number effectively computable from $\mathcal{A}$.*

Proof. We choose k as K from Lemma 14. Assume that R is uniformizable by a D↓PTT. We show by induction on the number of moves played by Out that the strategy in $\mathcal{G}_{\mathcal{A}}$ can be chosen such that in every play according to the strategy the following induction

hypothesis is satisfied. Let (p, q, π) denote a vertex of Out that is reached after a sequence of moves in a play. W.l.o.g., we make the assumption that in this sequence until now no vertex of Out was reached that has a self-loop. If a vertex with self-loop was reached, then Out can stay in this vertex and wins. We claim that there is some $l \leq K$ such that π can be split into $xiyjf$ for some $x, y \in \mathrm{Path}_\Sigma$, $i, j \in \mathrm{dir}_\Sigma$ and $f \in \Sigma$ with $||x|| = l$ and there exists a D↓PTT $\mathcal{T}$ that uniformizes R_q^x such that the following holds: $xiy \in \mathrm{LPL}_\mathcal{T}(R_q^x, l)$ if $||xiy|| \geq 1$, and for every D↓PTT $\mathcal{U}$ that uniformizes R_q^x with $x \in \mathrm{LPL}_\mathcal{U}(R_q^x, l)$ holds $value(\mathrm{LPL}_\mathcal{T}(R_q^x, l)) \leq value(\mathrm{LPL}_\mathcal{U}(R_q^x, l))$.

In words, if in a play $(p, q, \pi) \in V_{\mathsf{Out}}$ is reached, then there exists a factorization of π into $xiyjf$ such that there is a uniformizer $\mathcal{T}$ of R_q^x with xiy as lookahead-path and $\mathcal{T}$ has minimal complexity w.r.t. $||x||$ compared to every uniformizer of R_q^x that has x as lookahead-path.

Note, we allow to choose x or y resp. x and y as ε, then we identify $xiyjf$ with yjf or xif resp. f. First, we show that the induction hypothesis is true at the first reached vertex of Out in a play. Such a vertex is of the form $(q_0^\mathcal{B}, q_0^\mathcal{A}, f) \in V_{\mathsf{Out}}$ for some $f \in \Sigma$. For $l = 0$, i.e., $x = \varepsilon$ and $y = \varepsilon$, the induction hypothesis can be satisfied. Since R is uniformizable, there exists a D↓PTT $\mathcal{T}$ with minimal complexity w.r.t. limit 0 that uniformizes $R_{q_0}^\varepsilon = R$.

Now we define the strategy. Assume the play is in a vertex $(p, q, \pi) \in V_{\mathsf{Out}}$ and the induction hypothesis is true for some $l \leq K$ and π can be split accordingly into $xiyjf$ with $||x|| = l$ and let $\mathcal{T}$ by a uniformizer that satisfies the claim. To define the next move of Out, we consider the computation of $\mathcal{T}$ on some $t \in T_\Sigma^{xiyjf}$ and check whether the output produced while reading $xiyjf$ exceeds the limit l. Note, the induction hypothesis states that $xiy \in \mathrm{LPL}_\mathcal{T}(R_q^x, l)$, which means the output produced while reading xiy does not exceed the limit l.

If $||\mathcal{T}(t[\circ/path(xiy)j] \cdot f)||^{path(x)} \geq l$, that is, $\mathcal{T}$ produces at least l output symbols while reading $xiyjf$, then the strategy defines output moves. We pick an arbitrary $z \in out_\mathcal{T}(t[\circ/path(xiy)j] \cdot f, path(x))$, then Out makes l output moves according to the prefix of length l of z as follows. Let o denote this prefix. Note that o is the greatest common prefix of every labeled path from $out_\mathcal{T}(t[\circ/path(xiy)j] \cdot f, path(x))$. By Lemma 17 there exists an infinite state N↓TA $\mathcal{A}_\mathcal{T}$ that recognizes $R(\mathcal{T})$, then there exists a run ρ of $\mathcal{A}_\mathcal{T}$ on $t \otimes T(t)$ such that $(\rho(\varepsilon), \rho(path(x)i)) \in \tau_{xi,o}$. Since $\mathcal{A}$ is guidable, there is a mapping g such that $(\mathcal{A}_\mathcal{T}, g)$ guides $\mathcal{A}$. Note that it is not necessary to be able to compute g because it suffices to show the existence of moves for Out. The existence of a winning strategy for Out implies that there is also a positional one as used in Lemma 8. Thus, $g(\rho)$ is a run of $\mathcal{A}$ on $t \otimes T(t)$ such that $(q, q') \in \tau_{xi,o}$, where $q' = g(\rho(path(xi)))$. Out takes the moves leading to the vertex $(p', q', yjf) \in V_{\mathsf{Out}}$ with $\mathcal{A}\colon q \xrightarrow{x \otimes o}_i q'$ and $\mathcal{B}\colon p \xrightarrow{x} p'$. Since $\mathcal{T}$ is a uniformizer for R_q^x that reads $xiyjf$ and $(q, q') \in \tau_{xi,o}$, there exists also a uniformizer for $R_{q'}^{yjf}$. Moreover, from the induction hypothesis follows $||out_\mathcal{T}(t[\circ/path(xiy)j], path(x))|| < l$. This implies that there exists a uniformizer for $R_{q'}^{yjf}$ such that yif is a lookahead-path of this uniformizer w.r.t. limit $||yjf||$. Let $\mathcal{T}'$ denote such a uniformizer that has minimal complexity w.r.t. $||yjf||$ compared to every uniformizer of $R_{q'}^{yjf}$ that also has yif as lookahead-path. For $(q', yjf) \in V_{\mathsf{Out}}$, the induction hypothesis is then satisfied by choosing a new limit $l = ||yjf||$ and $\mathcal{T}'$ as described.

Otherwise, if $||\mathcal{T}(t[\circ/path(xiy)j] \cdot f)||^{path(x)} < l$, that is, $\mathcal{T}$ produces at less than l output symbols while reading $xiyjf$, we distinguish two cases. For the first case assume $||\pi|| < 2K$, then Out delays and picks direction d chosen as follows. Since $xiy \in \mathrm{LPL}_\mathcal{T}(R_q^x, l)$ and $||out_\mathcal{T}(t[\circ/path(xiy)j] \cdot f, path(x))|| < l$, it follows that for every $s \in T_\Sigma^{xiyjf}$ there is a configuration $c = (s, s', \varphi)$ of $\mathcal{T}$ reachable such that there is $u \in D_{s'}$ with $\varphi(u) = path(xiyjf)$ and $u \sqsubseteq \varphi(u)$. Then, there exists a configuration $c' = (s, s'', \varphi')$ with $c \to_\mathcal{T} c'$ such that there is $u' \in D_{s''}$ with $u \sqsubseteq u' \sqsubseteq path(xiyjf)$ and $\varphi'(u') = path(xiyif)d$ for some direction d.

Out moves to $(q, xiyjfd)$. Then, the next reached vertex of Out is of the form $(q, xiyjfdg)$ for some $g \in \Sigma$. The induction hypothesis is satisfied for the same limit $||x||$ and $\mathcal{T}$ as before: Clearly, we obtain $xiyjf \in \mathrm{LPL}_{\mathcal{T}}(R_q^x, l)$, because $||out_{\mathcal{T}}(t[\circ/path(xiy)j] \cdot f, path(x))|| < l$. Also, as before, $\mathcal{T}$ has minimal complexity w.r.t. $||x||$ compared to every uniformizer that also has x as lookahead-path.

For the second case assume $||\pi|| = 2K$. Since l is at most K, $||y|| \geq K$. By Lemma 14 it follows that there is a factorization $y_1j_1y_2j_2y_3j_3y_4$ of yjf such that y_3j_3 is idempotent and $P_{y_2j_2} = P_{y_3j_3}$. By induction hypothesis, we are guaranteed that $\mathcal{T}$ is a uniformizer for R_q^x with minimal complexity w.r.t. l compared to every uniformizer that also has x as lookahead-path. Together with Lemma 18, this implies that $xiyjf \notin \mathrm{LPL}_{\mathcal{T}}(R_q^x, l)$. However, since $\mathcal{T}$ uniformizes R_q^x and reads $xiyjf$ this means that R_q^{xiyjf} is uniformizable by a path-recognizable function. Consequently, the vertex $(p, q, \pi) \in V_{\mathsf{Out}}$ has a self-loop and Out stays in this vertex from then on.

The strategy is winning because it ensures that Out can always make a move. ◀

As a consequence of Lemmata 8 and 9 and the fact that a winning strategy for Out in $\mathcal{G}^k_{\mathcal{A},\mathcal{B}}$ can effectively be computed, together with the fact that for each tree-automatic relation a guidable N↓TA $\mathcal{A}$ can effectively be constructed, see [17], we immediately obtain Theorem 5.

C Details for Section 4.2

▶ **Lemma 12.** *There exists a parity tree automaton $\mathcal{C}$ that accepts exactly those trees $\mathcal{H}^\frown\sigma$ such that $(t, \sigma(t)) \in R$ for all $t \in dom(R)$.*

Proof. We show that there exists a parity tree automaton that accepts exactly those trees $\mathcal{H}^\frown\sigma$ for which σ is a strategy such that there exists an input tree $t \in dom(R)$ and it holds $(t, \sigma(t)) \notin R$. Then, by closure properties of regular tree languages, we can obtain a parity tree automaton that accepts exactly those trees $\mathcal{H}^\frown\sigma$ such that $(t, \sigma(t)) \in R$ for all $t \in dom(R)$. That is, $\mathcal{H}^\frown\sigma$ is accepted if, and only if, σ corresponds to a synchronous uniformizer.

A parity tree automaton, say $\bar{\mathcal{C}}$, that accepts exactly those trees $\mathcal{H}^\frown\sigma$ for which there is $t \in dom(R)$ such that $(t, \sigma(t)) \notin R$, i.e., $(t, \sigma(t)) \notin R_i$ for all i, is constructed as follows. Recall that $dom(R)$ is the union of pairwise disjoint domains. This means, in order to show that σ does not correspond to a uniformizer, it is sufficient to show for some i that there is $t \in dom(R_i)$ such that $(t, \sigma(t)) \notin R_i$. Hence, $\bar{\mathcal{C}}$ first guesses an i for which there is a tree $t \in dom(R_i)$ such that $\mathcal{A}_i$ does not accept $t \otimes \sigma(t)$. Since $\mathcal{A}_i$ is deterministic, we can prove the existence of such a tree by guessing a path where the deterministic run of $\mathcal{A}_i$ on the combined input and output specified by σ fails.

However, since σ may be chosen such that the paths of input and output sequence diverge, to show that the run fails, $\bar{\mathcal{C}}$ operates as follows. The parity tree automaton wants to simulate the run of $\mathcal{A}_i$ on the output choices defined by σ that lead to a position where the run fails. Note that although σ fixes the output choices, more than one output sequence might be associated with the same input sequence. Thus, $\bar{\mathcal{C}}$ has to guess the input sequence that leads to the output choices fixed by σ causing a non-accepting run, and also, it has to pick the output sequence among the associated output sequences that will be used to simulate a run of $\mathcal{A}_i$. For this purpose, $\bar{\mathcal{C}}$ guesses an input sequence and follows the vertices that correspond to choosing this labeled path in $\mathcal{H}^\frown\sigma$. As long as the path of the guessed input sequence is the same path as the output sequence under consideration, $\bar{\mathcal{C}}$ simulates the run of $\mathcal{A}_i$ on the

combined input and output on that path. The idea is explained a bit more formally below. Let x resp. y denote the input resp. output sequence up to last position where both share the same path, let $path(x) = path(y) = u$. Thus, $\bar{\mathcal{C}}$ is in a vertex in $\mathcal{H}^\frown\sigma$ that corresponds to the input sequence x and output sequence y and has simulated the run of $\mathcal{A}_i$ on $x \otimes y$. Now, we describe how we proceed if input and output diverge, i.e., $\bar{\mathcal{C}}$ continues the input sequence in direction i and the associated output sequence under consideration is continued in direction j with $i \neq j$. Let x' resp. y' denote the continuation of the input resp. output sequence. Furthermore, let $path(x') = iv$ and $path(y') = jv'$. Then, in addition to following x' in $\mathcal{H}^\frown\sigma$, $\bar{\mathcal{C}}$ chooses the input labels along ujv' beginning from uj. Let x'' denote the resulting labeled path with $path(x'') = jv'$. The labels have to be chosen such that both xx' and xx'' can be part of a tree from $dom(R_i)$. Let $\mathcal{A}_i : q_0^{\mathcal{A}_i} \xrightarrow{x\otimes y}_j q$, that is, q is the result of the simulation of the run of $\mathcal{A}_i$ so far. In the process of following x' in $\mathcal{H}^\frown\sigma$, $\bar{\mathcal{C}}$ simulates the run of $\mathcal{A}_i$ on $x'' \otimes y'$ starting from q. Then, $\bar{\mathcal{C}}$ accepts if the run fails, i.e., $\mathcal{A}_i : q_0^{\mathcal{A}_i} \xrightarrow{x\otimes y}_j q \xrightarrow{x''\otimes y'} Q_{\mathcal{A}_i} \setminus F_{\mathcal{A}_i}$. This means, there exists some $t \in dom(R_i) \cap T_\Sigma^{xx'} \cap T_\Sigma^{xx''}$ such that $(t, \sigma(t)) \notin R_i$, and consequently, $(t, \sigma(t)) \notin R$.

It follows that $\bar{\mathcal{C}}$ accepts exactly those trees $\mathcal{H}^\frown\sigma$ for which there is $t \in dom(R)$ such that $(t, \sigma(t)) \notin R$. A formal construction of $\bar{\mathcal{C}}$ is omitted. The desired parity tree automaton $\mathcal{C}$ can then be obtained from $\bar{\mathcal{C}}$ using complementation and intersection. ◀

▶ **Lemma 13.** *The tree language $T(\mathcal{C})$ is non-empty if, and only if, R has a uniformization by a synchronous deterministic top-down tree transducer.*

Proof. If the tree language $T(\mathcal{C})$ is non-empty, then there exists at least one regular tree $\mathcal{H}^\frown\sigma$ that is accepted by $\mathcal{C}$. From such an $\mathcal{H}^\frown\sigma$ we can construct a synchronous D↓TT that uniformizes R as follows. Given a vertex from $\mathcal{H}^\frown\sigma$ of the form (p, Q) with $L \in Q$, then (p, L) is used as a state of the uniformizer. The transition rules have to be constructed such that they correspond to the output choices specified by σ. This is, if $((p, \{L_1, \ldots, L_m\}), f) \xrightarrow{o_1,\ldots,o_m} [(p_1, Q_1), \ldots, (p_i, Q_i)]$ is selected by σ in $\mathcal{H}^\frown\sigma$, then for each $1 \leq j \leq m$ we add the rule $(p, L_j)(f(x_1, \ldots, x_i)) \to g((p_{j_1}, V_1)(x_{j_1}), \ldots, (p_{j_r}, V_r)(x_{j_r}))$ to the transitions, where $o_j = g(x_{j_1}, \ldots, x_{j_r})$ and $V_1, \ldots, V_r$ as described in the construction of $\mathcal{G}$ in Section 4.2.

Conversely, assume that R has a uniformization by a synchronous D↓TT $\mathcal{T}$. We have to show that $T(\mathcal{C})$ is non-empty. Therefore, it is sufficient to define σ inductively in correspondence to $\mathcal{T}$, which is done as follows. A node $((p, \{L_1, \ldots, L_m\}), f)$ in $\mathcal{H}$ corresponds to exactly one input sequence $xi_1 f \in \mathrm{Path}_\Sigma$ such that $\mathcal{D}\colon p_0 \xrightarrow{x}_{i_1} p$. Furthermore, each vector $L_j \in \{L_1, \ldots, L_m\}$ represents a set of output sequences from $\mathrm{Path}_\Gamma \cdot \mathrm{dir}_\Gamma$ computable from the output choices. This is a set because different output choices made along this input sequence eventually induce the same vector. Note that if a vector consists of single states, then this vector corresponds to exactly one output sequence that is on the same path as the input sequence. If a vector consists of state sets, then it can be the case that the vector represents more than one output sequence.

For a node $((p, \{L_1, \ldots, L_m\}), f)$ reachable in $\mathcal{H}$ (w.r.t. the already defined part of σ), we can show by induction on the length of the corresponding $xi_1 f$ that the following holds for $1 \leq j \leq m$:

1. There is some $yi_2 \in \mathrm{Path}_\Gamma \cdot \mathrm{dir}_\Gamma$ represented by L_j such that $\mathcal{T}(t) \in T_\Gamma^y$ and there exists a configuration $c = (t, t', \varphi)$ of $\mathcal{T}$ such that $u \in D_{t'}$ with $\varphi(u) = v$, where $path(x)i_1 = v$ and $path(y)i_2 = u$, for all $t \in T_\Sigma^{xi_1 f} \cap dom(R)$, and
2. if some $zi_3 \in \mathrm{Path}_\Gamma \cdot \mathrm{dir}_\Gamma$ is represented by L_j and if there is $\tilde{t} \in T_\Sigma^{xi_1 f} \cap dom(R_\ell) \cap T_\Sigma^{x'}$ such that $path(x') = path(z)$ with $\mathcal{A}_\ell\colon q_0^{\mathcal{A}_\ell} \xrightarrow{x'\otimes z}_{i_3} q$, then $q \in \lambda_\ell(L_j)$ and there is

$\hat{t} \in T_\Sigma^{xi_1 f} \cap dom(R_\ell) \cap T_\Sigma^{x''}$ such that $path(x'') = path(y)$ with $\mathcal{A}_\ell \colon q_0^{\mathcal{A}_\ell} \xrightarrow{x'' \otimes y}_{i_2} q$.

The first property states that one represented output sequence y is a output sequence produced by $\mathcal{T}$ while reading x. The second property states that for all other represented output sequences z, a state transformation induced by z together with some valid input can also be induced by y and some valid input.

We define the strategy σ. Consider a configuration $c = (t, t', \varphi)$ as in 1., let $val_{t'}(u) = s \in Q_\mathcal{T}$, then there exists a rule $s(f(x_1, \ldots, x_i)) \to g(s_1(x_{j_1}), \ldots, s_r(x_{j_r})) \in \Delta_\mathcal{T}$, because $\mathcal{T}$ is a uniformizer and $t \in dom(R)$. We pick $o_j = g(x_{j_1}, \ldots, x_{j_r})$ as output choice for L_j. Then σ selects the child that is reached via $o_1, \ldots, o_m$.

Recall the definition of $\bar{\mathcal{C}}$ from the proof of Lemma 12, $\bar{\mathcal{C}}$ accepts exactly those trees $\mathcal{H}^\frown\sigma$ for which there is $t \in dom(R)$ such that $(t, \sigma(t)) \notin R$. We show that $\bar{\mathcal{C}}$ rejects $\mathcal{H}^\frown\sigma$, then it follows that $\mathcal{C}$ accepts $\mathcal{H}^\frown\sigma$. Pick a $t \in dom(R)$, wlog, let $t \in dom(R_\ell)$. The automaton $\bar{\mathcal{C}}$ accepts if there exists a path in the computation of $\mathcal{A}_\ell$ on $t \otimes \sigma(t)$ where the run fails. However, from the above induction using property 2. follows that the run is defined on every path, i.e., $\bar{\mathcal{C}}$ rejects. ◀