Uniformization Problems for Tree-Automatic Relations and Top-Down Tree Transducers*

Christof Löding[1] and Sarah Winter[2]

1 **Lehrstuhl für Informatik 7, RWTH Aachen University, Aachen, Germany**
2 **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.2016.65

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 [18] 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 [8], also called rational relations, were first shown to have rational uniformizations in [13, 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 [19]. Over infinite trees, the uniformization property fails for MSO definable relations (corresponding to synchronous tree automata) [10, 2], while it has been shown recently that uniformization is possible for synchronous

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

41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016).
Editors: Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier; Article No. 65; pp. 65:1–65:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

relations over finite trees [14, 4]. These relations defined by synchronous automata are usually 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 [12, 11]. 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 [17]. As class for the uniformizations we consider deterministic top-down tree transducers (D$\downarrow$TTs), which are a natural extension of sequential transducers on words. A first result in this setting was obtained in [16], where we show that it is decidable whether a tree-automatic relation that is defined by a deterministic top-down tree automaton (D$\downarrow$TA) can be uniformized by a D$\downarrow$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$\downarrow$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$\downarrow$TTs is decidable. Intuitively, we call a D$\downarrow$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$\downarrow$TT is in particular linear). For this class of uniformizers we can adapt the game theoretic approach by using guidable automata [5, 15] instead of deterministic automata for the specification.
2. If we restrict the specifications to unions of D$\downarrow$TAs with disjoint domain, we obtain decidability for uniformizations by synchronous D$\downarrow$TTs. We call a D$\downarrow$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 $\Sigma \uplus \{\circ\}$ 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. [9] 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 $(\rho(u), val_t(u), \rho(u1), \ldots, \rho(ui)) \in \Delta$. 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 [9].

In Section 4.1 we use *guidable tree automata* [15]. 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

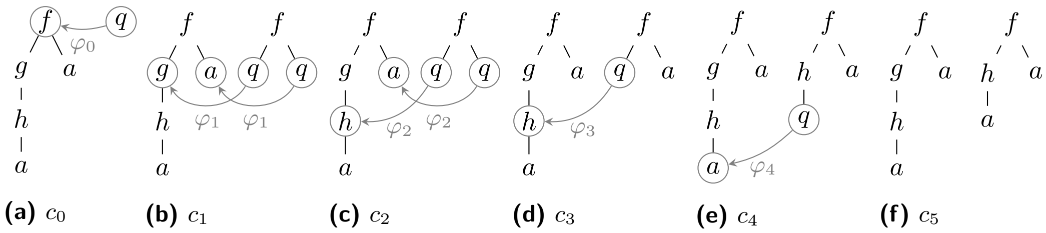

Figure 1 The configuration sequence c_0 to c_5 of $\mathcal{T}$ on t from Example 1.

- $val_{t'}(u) \in \Gamma_0$ 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 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 $val_{t_1}(u)\,(val_t(v)(x_1, \dots, x_i)) \to w[q_1(x_{j_1}), \dots, q_n(x_{j_n})] \in \Delta$ such that the output tree is correctly updated, i.e., $t_2 = t_1[\circ/u] \cdot w[q_1, \dots, q_n]$, 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 ↓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 ↓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* $\downarrow$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 $\downarrow$TT is *linear* if all the trees in the transitions are linear. In Section 4.1, we consider a special case of linear $\downarrow$TTs called *path-preserving*. Intuitively, a $\downarrow$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 $\downarrow$TTs as P$\downarrow$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$\downarrow$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

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

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

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

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 [15] for the specifications makes it feasible to adapt a decision procedure presented in [16], 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 $\mathrm{dir}_\Sigma = \{1, \ldots, m\}$ 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 $j \in \{1, \ldots, i\}$ and $g \in \Sigma$ is a labeled input path.

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

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

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

For $\pi \in \mathrm{Path}_\Sigma$ let $T_\Sigma^\pi := \{t \in T_\Sigma \mid \mathit{val}_t\big(\mathit{path}(\pi)[1,(i-1)]\big) = \mathit{labels}(\pi)[i] \text{ for } 1 \leq i \leq ||\pi||\}$ 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 $\mathit{dom}_x := \{u \in \mathrm{dir}_\Sigma^* \mid u \sqsubseteq \mathit{path}(x)\}$ and $\mathit{val}_x : \mathit{dom}_x \to \Sigma$, where $\mathit{val}_x(u) = \mathit{labels}(x)[i]$ if $u \in \mathit{dom}_x$ with $|u| = i + 1$. Let $x \in \mathrm{Path}_\Sigma$, $y \in \mathrm{Path}_\Gamma$ with $\mathit{path}(y) \sqsubseteq \mathit{path}(x)$ or $\mathit{path}(x) \sqsubseteq \mathit{path}(y)$, then the *convolution* of x and y is $x \otimes y$ defined by $\mathit{dom}_{x\otimes y} = \mathit{dom}_x \cup \mathit{dom}_y$, and $\mathit{val}_{x\otimes y}(u) = (\mathit{val}_x^\perp(u), \mathit{val}_y^\perp(u))$ for all $u \in \mathit{dom}_{x\otimes y}$, where $\mathit{val}_x^\perp(u) = \mathit{val}_x(u)$ if $u \in \mathit{dom}_x$ and $\mathit{val}_x^\perp(u) = \perp$ otherwise, analogously defined for $\mathit{val}_y^\perp(u)$.

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., $\mathit{path}(y) \sqsubseteq \mathit{path}(x)$ or $\mathit{path}(x) \sqsubseteq \mathit{path}(y)$. We define the run of $\mathcal{A}$ on $x \otimes y$ such that it maps all nodes from $\mathit{dom}_{x\otimes y}$ as well as all nodes that are a direct successor of a node from $\mathit{dom}_{x\otimes y}$ 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 $u \in \mathit{dom}_{x\otimes y}$: if $q := \rho(u)$ is defined and there is a transition $(q, \mathit{val}_{x\otimes y}(u), q_1, \ldots, q_i) \in \Delta_\mathcal{A}$, then $\rho(u.j) = q_j$ for all $j \in \{1, \ldots, i\}$. Let $\mathit{path}(x \otimes y) = v$ 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 $\mathcal{A} : q_0^\mathcal{A} \xrightarrow{x\otimes y} \mathit{Acc}$ if $\mathit{rk}(\mathit{val}_{x\otimes y}(v)) = 0$ and $(\rho(v), \mathit{val}_{x\otimes y}(v)) \in \Delta_\mathcal{A}$ 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** ([16]). *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. $q(f(x_1,\ldots,x_i)) \to q'(x_j)$, or
2. $q(a) \to t$,

where $f \in \Sigma_i$, $i > 0$, $a \in \Sigma_0$, $q, q' \in Q_\mathcal{T}$ and $j \in \{1,\ldots,i\}$ and $t \in T_\Gamma$.

This notion was introduced in [16], 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 $x \in \text{Path}_\Sigma$, $y \in \text{Path}_\Gamma$ and $i \in \text{dir}_\Sigma$ 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 $j \in \{1,\ldots,rk\big((val^\perp_x(u), val^\perp_y(u))\big)\}$ holds

- if $r := \rho_q(uj)$ and $j \le rk(val^\perp_x(u)), rk(val^\perp_y(u))$, 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 $rk(val^\perp_y(u)) < j \le rk(val^\perp_x(u))$, then $t \otimes \perp \in T(\mathcal{A}_r)$ for all $t \in T_\Sigma$, and
- if $r := \rho_q(uj)$ and $rk(val^\perp_x(u)) < j \le rk(val^\perp_y(u))$, 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 $s' \in S^{yi\circ}_\Gamma$ such that for each input tree $t \in T^x_\Sigma \cap dom(T(\mathcal{A}_q))$ 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. $(p, q, \pi j) \rightarrow (p, q, \pi j f)$ for each $f \in \Sigma$ such that $\mathcal{B}\colon p \xrightarrow{\pi}_j p'$ and there exists $(p', f, p_1, \dots, p_i) \in \Delta_{\mathcal{B}}$ if $\|\pi\| < 2k+1$ (delay; In chooses the next input symbol)
  i2. $\{(p_1, q_1), \dots, (p_n, q_n)\} \rightarrow (p_j, q_j, f)$ for each $f \in \Sigma$ such that there is $(p_j, f, p_j^1, \dots, p_j^i) \in \Delta_{\mathcal{B}}$ and each $j \in \{1, \dots, n\}$ (no delay; In chooses the next direction and input symbol)
- From a vertex of Out the following moves are possible:
  o1. $(p, q, f) \xrightarrow{r} \{(p_1, q_1), \dots, (p_i, q_i)\}$ if there is $r = (q, (f, g), q_1, \dots, q_n) \in \Delta_{\mathcal{A}}$, $(p, f, p_1, \dots, p_i) \in \Delta_{\mathcal{B}}$, $f \in \Sigma$ is i-ary, $g \in \Sigma_\perp$ is j-ary, $n = max\{i, j\}$, and if $j > i$ there exist trees $t_{i+1}, \dots, t_j \in T_\Gamma$ 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. $(p, q, f j \pi) \xrightarrow{r} (p_j, q_j, \pi)$ if there is $r = (q, (f, g), q_1, \dots, q_n) \in \Delta_{\mathcal{A}}$ such that $(q, q_j) \in \tau_{fj,g}$ and $(p, f, p_1, \dots, p_i) \in \Delta_{\mathcal{B}}$.
  (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. $(p, q, \pi j f) \rightarrow (p, q, \pi j f j')$ for each $j' \in \{1, \dots, i\}$ for $f \in \Sigma_i$ if $\|\pi j f\| < k+1$ (Out delays and chooses a direction from where In should continue)
  o4. $(p, q, \pi) \rightarrow (p, q, \pi)$ if R_q^π 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 [16] 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 [15], 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 $\{(h(t), f(t, t')) \mid t, t' \in T_\Sigma$ such that t and t' 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. [20]. 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 $\mathcal{A}_1, \dots, \mathcal{A}_n$ 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 $(p, \{L_1, \dots, L_m\})$, where p is a state of $\mathcal{D}$ and each L_j is a vector of states resp. sets of states over $\mathcal{A}_1, \dots, \mathcal{A}_n$, the following edges exist:
  - $v \to (v, f)$ if there is $(p, f, p_1 \dots, p_i) \in \Delta_\mathcal{D}$ (edges for every possible input symbol)
- An edge $((p, \{L_1, \dots, L_m\}), f) \xrightarrow{o_1,\dots,o_m} [(p_1, Q_1), \dots, (p_i, Q_i)]$ defining output choices $o_1, \dots, o_m$ exists if $(p, f, p_1 \dots, p_i) \in \Delta_\mathcal{D}$ 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 $o_j = g(x_{j_1}, \dots, x_{j_r})$ there is $k \in \{1, \dots, r\}$ 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 $(q, (f, g), q_1, \dots, q_{max\{rk(f), rk(g)\}}) \in \Delta_{\mathcal{A}_\ell}$,
    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 $(q, (f', g), q_1, \dots, q_{max\{rk(f'), rk(g)\}}) \in \Delta_{\mathcal{A}_\ell}$, add q_k to $\lambda_\ell(V_k)$.
    (all possibly reachable states in $\mathcal{A}_\ell$ are collected)
- From $[v_1, \dots, v_i]$ 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 $L = [q_0^{\mathcal{A}_1}, \dots, q_0^{\mathcal{A}_n}]$ 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 [20]).

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

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

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

11 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.

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

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

14 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.

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

16 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.

17 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.

18 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.

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

20 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.