Decision Problems for Origin-Close Top-Down Tree Transducers

Sarah Winter
Université libre de Bruxelles, Brussels, Belgium

Abstract

Tree transductions are binary relations of finite trees. For tree transductions defined by non-deterministic top-down tree transducers, inclusion, equivalence and synthesis problems are known to be undecidable. Adding origin semantics to tree transductions, i.e., tagging each output node with the input node it originates from, is a known way to recover decidability for inclusion and equivalence. The origin semantics is rather rigid, in this work, we introduce a similarity measure for transducers with origin semantics and show that we can decide inclusion, equivalence and synthesis problems for origin-close non-deterministic top-down tree transducers.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Transducers

Keywords and phrases tree tranducers, equivalence, uniformization, synthesis, origin semantics

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.90

Related Version *Full Version*: https://arxiv.org/abs/2107.02591

Funding This research was supported by the DFG grant LO 1174/3.

Acknowledgements The author wants to thank Emmanuel Filiot for valuable feedback on the presentation.

1 Introduction

In this paper we study decision problems for top-down tree transducers over finite trees with origin semantics. Rounds [30] and Thatcher [31] independently invented tree transducers (their model is known today as top-down tree transducer) as a generalization of finite state word transducers in the context of natural language processing and compilers in the beginning of the 1970s. Nowadays, there is a rich landscape of various tree transducer models used in many fields, for example, syntax-directed translation [18], databases [29, 20], linguistics [27, 5], programming languages [33, 28], and security analysis [23].

Unlike tree automata, tree transducers have undecidable inclusion and equivalence problems [13]. This is already the case for word transducers [19, 17]. The intractability of, e.g., the equivalence problem for transducers (whether two given transducers recognize the same transduction, that is, the same relation) mainly stems from the fact that two transducers recognizing the same transduction may produce their outputs very differently. One transducer may produce its output fast and be ahead of the other. In general, there is an infinite number of transducers for a single transduction. To overcome this difficulty Bojanczyk [1] has introduced origin semantics, that is, additionally, there is an origin function that maps output positions to their originating input positions. The main result of [1] is a machine-independent characterization of transductions defined by deterministic two-way transducers with origin semantics. Word transducers with origin semantics where further investigated in [2], and properties of subclasses of transductions with origin semantics definable by one-way word transducers have been studied in [14, 9]. Under origin semantics, many interesting problems become decidable, e.g., equivalence of one-way word transducers. This is not surprising as a transduction now incorporates *how* it translates an input word into an output word providing much more information.

In [16], the authors have initiated a study of several decision problems for different tree transducer models on finite trees with origin semantics. More concretely, they studied inclusion, equivalence, injectivity and query determinacy problems for top-down tree transducers, tree transducers definable in monadic second order logic, and top-down tree-to-word transducers. They showed (amongst other results) that inclusion and equivalence become decidable for all models except tree-to-string transducers with origin semantics.

In general, there has been an interest to incorporate some kind of origin information (i.e., *how* a transduction works) into tree transductions, in order to gain more insight on different tree transductions, see, e.g., [32, 11, 26].

However, the origin semantics is rather rigid. To mitigate this, in [15], the authors have introduced a similarity measure between (one-way) word transducers with origin semantics which amounts to a measure that compares the difference between produced outputs on the same input prefix, in short, the measure compares their output delays. They show that inclusion, equivalence, and sequential uniformization (see next paragraph) problems become decidable for transducers that have bounded output delay. These problem are undecidable for word transducers in general, see [19, 17, 7]. The introduction of this similarity measure has triggered similar works on two-way word transducers, see [4, 3].

In order to obtain decidability results (in a less rigid setting than origin semantics), we initiate the study of inclusion, equivalence, and uniformization problems for top-down tree transducers under similarity measures which are based on the behavior of the transducers.

A uniformization of a binary relation is a function that selects for each element of the domain of the relation an element in its image. Synthesis problems are closely related to *effective* uniformization problems; algorithmic synthesis of specifications (i.e., relations) asks for effective uniformization by functions that can be implemented in a specific way. The classical setting is Church's synthesis problem [8], where logical specifications over infinite words are considered. Büchi and Landweber [6] showed that for specifications in monadic second order logic, that is, specifications that can be translated into synchronous finite automata, it is decidable whether they can be realized by a synchronous sequential transducer. Later, decidability has been extended to asynchronous sequential transducers [22, 21]. Detailed studies of the synthesis of sequential transducers from synchronous and asynchronous finite automata on finite words are provided in [15, 34], for an overview see [7].

Uniformization questions in this spirit have been first studied for relations over finite trees in [25, 24]. The authors have considered tree-automatic relations, that is, relations definable by tree automata over a product alphabet. They have shown that for tree-automatic relations definable by deterministic top-down tree automata uniformization by deterministic top-down tree transducers (which are a natural extension of sequential transducer on words) is decidable. However, for non-deterministic top-down tree automata it becomes undecidable.

Our contribution is the introduction of two similarity measures for top-down tree transducers. The first measure is an extension of the output delay measure introduced for word transducers in [15] to tree transducers. Comparing top-down tree transducers based on their output delay has also been done in e.g., [12], we use the same notion of delay to define our measure. Unfortunately, while decidability for major decision problems is regained in the setting of word transducers, we show that it is not in the setting of tree transducers. The second similarity measure is more closely connected to the origin semantics. We define two transducers as origin-close if there is a bound on the distance of two positions which are origins of the same output node by the two transducers. Our main result is that inclusion, equivalence and uniformization by deterministic top-down tree transducers is decidable for origin-close top-down tree transducers.

The paper is structured as follows. In Section 2 we provide definitions and terminology used throughout the paper. In Section 3 we present two similarity measures for (top-down tree) transducers and provide a comparison of their expressiveness, and in Section 4 we consider decision problems for origin-close top-down tree transducers.

2 Preliminaries

Words, trees, and contexts. An *alphabet* Σ is a finite non-empty set of *letters* or *symbols*. 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. 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 $i \in \mathbb{N} \setminus \{0\}$, and the $\sqsubseteq$-maximal elements of *dom* are called *leaves*.

A (finite Σ-labeled) *tree* is a mapping $t : dom_t \to \Sigma$ such that for each node $u \in dom_t$ the number of successors of u is a rank of $t(u)$. 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 $t|_u(v) = t(uv)$ 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 $\circ$ has rank zero and 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 transductions, origin mappings, and uniformizations. Let Σ, Γ be ranked alphabets. A *tree transduction (from T_Σ to T_Γ)* is a relation $R \subseteq T_\Sigma \times T_\Gamma$. Its *domain*, denoted $\mathrm{dom}(R)$, is the projection of R on its first component. Given trees t_1, t_2, an *origin mapping* of t_2 in t_1 is a function $o\colon dom_{t_2} \to dom_{t_1}$. Given $v \in dom_{t_2}$, $u \in dom_{t_1}$, we say v has origin u if $o(v) = u$. Examples are depicted in Figures 1g and 2. A *uniformization* of a tree transduction $R \subseteq T_\Sigma \times T_\Gamma$ is a function $f\colon \mathrm{dom}(R) \to T_\Gamma$ such that $(t, f(t)) \in R$ for all $t \in \mathrm{dom}(R)$.

Top-down tree transducers. We consider top-down tree transducers, which read the tree from the root to the leaves in a parallel fashion and produce finite output trees in each step that are attached to the already produced output.

A *top-down tree transducer* (a TDTT) 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

$$q(f(x_1, \ldots, x_i)) \to w[q_1(x_{j_1}), \ldots, q_n(x_{j_n})],$$

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* TDTT (a DTDTT) has no two rules with the same left-hand side.

We now introduce a non-standard notion of configurations which is more suitable to prove our results. Usually, a configuration is a partially transformed input tree; the upper part is the already produced output, the lower parts are remainders of the input tree. Here, we keep the input and output tree separate and introduce a mapping from nodes of the output tree to nodes of the input tree from where the transducer continues to read. A visualization of several configurations is given in Figure 1.

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

- $t'(u) \in \Gamma_i$ for each $u \in dom_{t'}$ with $i > 0$ successors, and
- $t'(u) \in \Gamma_0$ or $t'(u) \in Q$ for each leaf $u \in dom_{t'}$, and
- $D_{t'} \subseteq dom_{t'}$ with $D_{t'} = \{u \in dom_{t'} \mid t'(u) \in Q\}$, 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 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 $t_1(u)\,(t(v)(x_1, \ldots, x_i)) \to w[q_1(x_{j_1}), \ldots, q_n(x_{j_n})] \in \Delta$ 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 $\to_{\mathcal{T}}$. 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 sequence starting with an initial configuration where each configuration is a successor of the previous one is called a *run*. For a tree $t \in T_\Sigma$ 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 *transduction* $R(\mathcal{T})$ induced by a TDTT $\mathcal{T}$ is $R(\mathcal{T}) = \{(t, t') \mid t' \in \mathcal{T}(t) \cap T_\Gamma\}$. The class of relations definable by TDTTs is called the class of *top-down tree transductions*, conveniently denoted by TDTT.

Let $\mathcal{T}$ be a TDTT, and let $\rho = c_0 \ldots c_n$ be a run of $\mathcal{T}$ on an input tree $t \in T_\Sigma$ that results in an output tree $s \in T_\Gamma$. The *origin function* o of ρ maps a node u of the output tree to the node v of the input tree that was read while producing u, formally $o : dom_s \to dom_t$ with $o(u) = v$ if there is some i, such that $c_i = (t, t_i, \varphi_i)$, $c_{i+1} = (t, t_{i+1}, \varphi_{i+1})$ and $\varphi_i(u) = v$ and $t_{i+1}(u) = s(u)$, see Figure 1. We define $R_o(\mathcal{T})$ to be the set

$$\{(t, s, o) \mid t \in T_\Sigma, s \in T_\Gamma \text{ and } \exists\, \rho : (t, q_0, \varphi) \to_{\mathcal{T}}^* (t, s, \varphi') \text{ with origin } o\}.$$

▶ **Example 1.** Let Σ be a ranked alphabet given by $\Sigma_2 = \{f\}$, $\Sigma_1 = \{g, h\}$, and $\Sigma_0 = \{a\}$. Consider the TDTT $\mathcal{T}$ 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

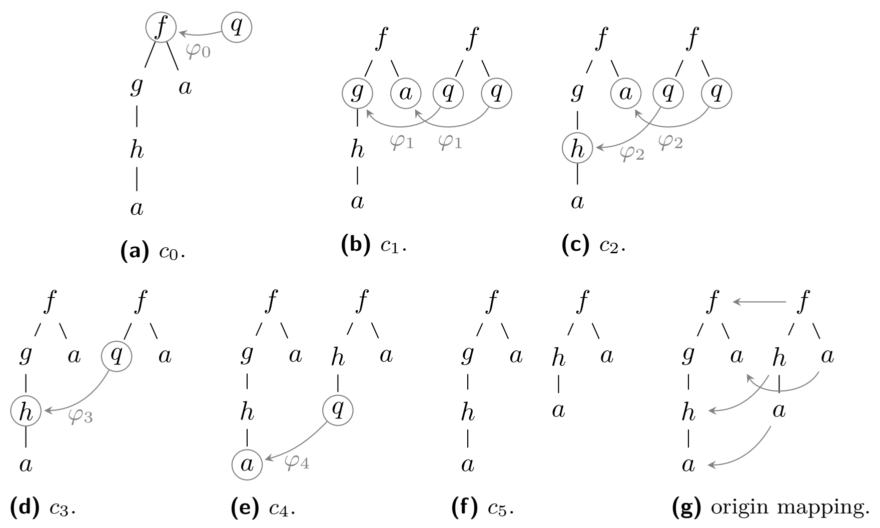

(a) c_0. **(b)** c_1. **(c)** c_2. **(d)** c_3. **(e)** c_4. **(f)** c_5. **(g)** origin mapping.

Figure 1 The configuration sequence c_0 to c_5 of $\mathcal{T}$ on $f(g(h(a)), a)$ and resulting origin mapping from Example 1.

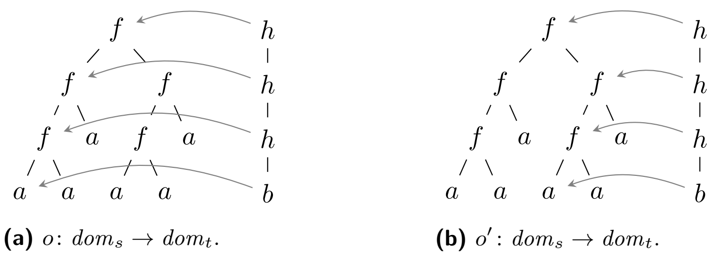

(a) $o\colon \mathit{dom}_s \to \mathit{dom}_t$. **(b)** $o'\colon \mathit{dom}_s \to \mathit{dom}_t$.

Figure 2 Origin mappings o, o'. We have that $\mathit{dist}(o(111), o'(111))$, that is, the distance of the origins of the leaf node, is the length of the shortest path from node 111 to node 211 which is 6.

all occurrences of g in t. Consider $t := f(g(h(a)), a)$. A possible sequence of configurations of $\mathcal{T}$ on t is $c_0 \to^5_{\mathcal{T}} c_5$ 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 and resulting origin mapping is shown in Figure 1.

▶ **Example 2.** Let Σ, Γ be given by $\Sigma_2 = \{f\}$, $\Sigma_0 = \{a\}$, $\Gamma_1 = \{h\}$, and $\Gamma_0 = \{b\}$. Consider the TDTT $\mathcal{T}$ given by $(\{q\}, \Sigma, \Gamma, \{q\}, \Delta)$ with $\Delta = \{\ q(a) \to b,\ q(f(x_1, x_2)) \to h(q(x_1)),\ q(f(x_1, x_2)) \to h(q(x_2))\ \}$. Basically, when reading an f-labeled node, the TDTT can non-deterministically decide whether to continue reading in left or the right subtree. In Figure 2 two origin mappings $o\colon \mathit{dom}_s \to \mathit{dom}_t$ and $o'\colon \mathit{dom}_s \to \mathit{dom}_t$ are given that are result of runs of $\mathcal{T}$ on $t = f(f(f(a, a), a), f(f(a, a), a), a))$ with final output $s = h(h(h(b)))$.

In this work, we focus on decision problems for transducers with origin semantics. To begin with, we introduce some notations and state relevant known results in this context.

Shorthand notations. Let $\mathcal{C}$ denote a class of transducers with origin semantics, e.g., TDTT or DTDTT. Given a class $\mathcal{C}$ and $\mathcal{T}_1, \mathcal{T}_2 \in \mathcal{C}$, if $R(\mathcal{T}_1) \subseteq R(\mathcal{T}_2)$ (resp. $R_o(\mathcal{T}_1) \subseteq R_o(\mathcal{T}_2)$), we write $\mathcal{T}_1 \subseteq \mathcal{T}_2$ (resp. $\mathcal{T}_1 \subseteq_o \mathcal{T}_2$). Furthermore, if $R(\mathcal{T}_1) = R(\mathcal{T}_2)$ (resp. $R_o(\mathcal{T}_1) = R_o(\mathcal{T}_2)$), we write $\mathcal{T}_1 = \mathcal{T}_2$ (resp. $\mathcal{T}_1 =_o \mathcal{T}_2$). Given classes $\mathcal{C}_1, \mathcal{C}_2$, $\mathcal{T}_1 \in \mathcal{C}_1$, and $\mathcal{T}_2 \in \mathcal{C}_2$, if $\mathcal{T}_1$ defines a function f that is a uniformization of $R(\mathcal{T}_2)$, we say $\mathcal{T}_1$ uniformizes $\mathcal{T}_2$, if additionally $\mathcal{T}_1 \subseteq_o \mathcal{T}_2$, we say $\mathcal{T}_1$ origin uniformizes $\mathcal{T}_2$.

Decision problems. The *inclusion* resp. *origin inclusion problem* for a class $\mathcal{C}$ asks, given $\mathcal{T}_1, \mathcal{T}_2 \in \mathcal{C}$, whether $\mathcal{T}_1 \subseteq \mathcal{T}_2$ resp. $\mathcal{T}_1 \subseteq_o \mathcal{T}_2$. The *equivalence* resp. *origin equivalence problem* for a class $\mathcal{C}$ asks, given $\mathcal{T}_1, \mathcal{T}_2 \in \mathcal{C}$, whether $\mathcal{T}_1 = \mathcal{T}_2$ resp. $\mathcal{T}_1 =_o \mathcal{T}_2$. Lastly, the *uniformization* resp. *origin uniformization problem* for classes $\mathcal{C}_1, \mathcal{C}_2$ asks, given $\mathcal{T}_2 \in \mathcal{C}_2$, whether there exists $\mathcal{T}_1 \in \mathcal{C}_1$ such that $\mathcal{T}_1$ uniformizes (resp. origin uniformizes) $\mathcal{T}_2$.

As mentioned in the introduction, generally, a transduction can be defined by several transducers behaving very differently, making many problems intractable. Adding origin semantics to transducers, i.e., seeing the transducer behavior as part of the transduction, allows to recover decidability. The following is known for the class TDTT.

▶ **Theorem 3** ([13]). *Inclusion and equivalence are undecidable for the class* TDTT.

▶ **Theorem 4** ([16]). *Origin inclusion and origin equivalence are decidable for the class* TDTT.

Turning to uniformization problems, it is known that every TDTT is uniformizable by a DTDTT with regular lookahead (a DTDTT^{R}), that is, the transducer can check membership of the subtrees of a node in regular tree-languages before processing the node.

▶ **Theorem 5** ([10]). *Every* TDTT *has a* DTDTT^{R}*-uniformization.*

However, when requiring that the input should be transformed on-the-fly (without regular lookahead), the uniformization problem becomes undecidable. In [7], it was shown that it is undecidable whether a one-way (non-deterministic) word transducer has a uniformization by a sequential transducer (that is, basically, a one-way deterministic transducer). So, we get undecidability in the tree setting for free (as stated in Theorem 6). This problem has not been investigated with origin semantics so far. We show decidability (also for more relaxed versions), see Theorem 16.

▶ **Theorem 6.** DTDTT*-uniformization is undecidable for the class* TDTT.

Since the origin semantics is rather rigid, in the next section, we introduce two similarity measures between transducers which are based on their behavior and re-investigate the introduced decision problems for transducers with "similar" behavior.

3 Similarity measures for transducers

An idea that naturally comes to mind is to say that two transducers behave similarly if for two computations over the same input that yield the same output their respective origin mappings are "similar".

The other idea is to say that two computations are similar if their output delay is small, roughly meaning that for the same prefix (for an adequate notion of prefix for trees) of the input the so-far produced output is of similar size. Decision problems using this measure have already been investigated for (one-way) word transducers [15], we lift the measure to top-down tree transducers.

Origin distance. Given a tree t, let the distance between two nodes $u, v \in dom_t$, written $dist(u, v)$, be the shortest path between u and v (ignoring the edge directions), an example is given in Figure 2.

Given $\mathcal{T}, \mathcal{T}_1, \mathcal{T}_2 \in \mathcal{C}$, where $\mathcal{C}$ is a class of transducers with origin semantics. We say (t, s, o) is *k-origin included* in $R_o(\mathcal{T})$, written $(t, s, o) \in_k R_o(\mathcal{T})$, if there is $(t, s, o') \in R_o(\mathcal{T})$ such that $dist(o(i), o'(i)) \leq k$ for all $i \in dom_s$. We say $\mathcal{T}_1$ is *k-origin included* in $\mathcal{T}_2$, written $\mathcal{T}_1 \subseteq_k \mathcal{T}_2$, if $(s, t, o) \in_k R_o(\mathcal{T}_2)$ for all $(s, t, o) \in R_o(\mathcal{T}_1)$. We say $\mathcal{T}_1$ and $\mathcal{T}_2$ are *k-origin equivalent*, written $\mathcal{T}_1 =_k \mathcal{T}_2$, if $\mathcal{T}_1 \subseteq_k \mathcal{T}_2$ and $\mathcal{T}_2 \subseteq_k \mathcal{T}_1$. We say $\mathcal{T}_1$ *k-origin uniformizes* $\mathcal{T}_2$ if $\mathcal{T}_1 \subseteq_k \mathcal{T}_2$ and $\mathcal{T}_1$ uniformizes $\mathcal{T}_2$. The k-origin decision problems are defined as expected.

We need some additional notations, before we can introduce the concept of delay.

Partial and prefix trees. Let N_Σ be the set of all trees over Σ which can have symbols from Σ, that is, symbols with rank ≥ 0, at their leaves. The set N_Σ is the set of all *partial trees* over Σ. Note that N_Σ includes T_Σ. We say a tree $t' \in N_\Sigma$ is a *prefix tree* of a tree $t \in N_\Sigma$, written $t' \sqsubseteq t$, if $dom_{t'} \subseteq dom_t$, and $t'(u) = t(u)$ for all $u \in dom_{t'}$. Given $t_1, t_2 \in N_\Sigma$, its *greatest common prefix*, written $t_1 \wedge t_2$, is the tree $t \in N_\Sigma$ such that dom_t is the largest subset of $dom_{t_1} \cap dom_{t_2}$ such that $t \sqsubseteq t_1$ and $t \sqsubseteq t_2$. Removing $t_1 \wedge t_2$ from t_1 and t_2 naturally yields a set of partial trees (we omit a formal definition) called *difference trees*. These notions are visualized in Figure 3.

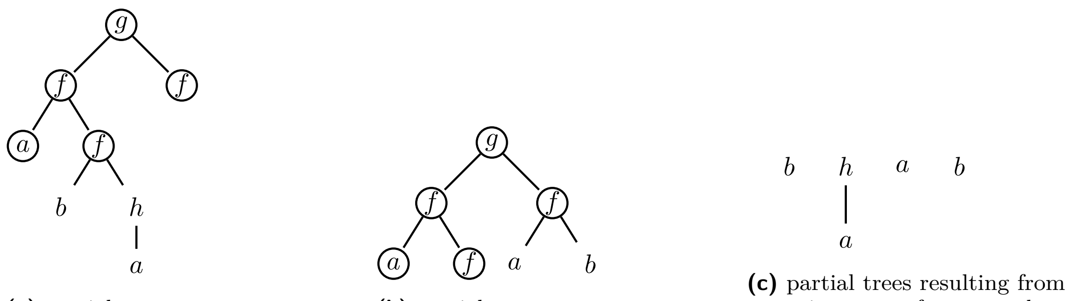
Delay. Given words w_1, w_2, to compute their delay, we remove their greatest common prefix $w = w_1 \wedge w_2$, say $w_1 = wv_1$ and $w_2 = wv_2$, and their delay is the maximum of the length of their respective reminders, i.e., $\max\{|v_1|, |v_2|\}$. We lift this to trees, given (partial) trees t_1, t_2, we remove their greatest common prefix $t_1 \wedge t_2$ from t_1 and t_2 which yields a set S of partial trees, we define their delay as $delay(t_1, t_2) = \max\{h(t) + 1 \mid t \in S\}$. An example is given in Figure 3. Note that for trees over unary and leaf symbols (a way to see words) the definitions for words and trees are equal. Recall that the length of the word a is one, but the height of the tree a is zero.

In order to define a similarity measure between transducers using delay, we take two transducer runs on the same input and compute the delay between their produced outputs throughout their runs. Although we have defined delay between words and trees, we only provide a formal definition for top-down tree transducers. However, for word transducers, examples are given in Example 17, and a formal definition can be found in [15].

Now, let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TDTTs, and ρ_1 and ρ_2 be runs of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, over the same input tree $t \in T_\Sigma$ such that $\rho_1 \colon (t, q_0^{\mathcal{T}_1}, \varphi_0) \rightarrow^*_{\mathcal{T}_1} (t, t_1, \varphi)$ with $t_1 \in T_\Gamma$, and $\rho_2 \colon (t, q_0^{\mathcal{T}_2}, \varphi_0) \rightarrow^*_{\mathcal{T}_2} (t, t_2, \varphi')$ with $t_2 \in T_\Gamma$.

Basically, we take a look at all configurations that occur in the runs and compute the delay between the output trees of compatible configurations where compatible means in both configurations the same prefix (level-wise, see below) of the input tree has been processed.

Let us be a bit more clear what we mean with compatible. Note that when comparing two configuration sequences (i.e., runs) of word transducers the notion of "have processed the same input so far" is clear. For tree transducers, in one configuration sequence, a left-hand subtree might be processed before the right-hand subtree, and in another configuration sequence vice versa. Since these computation steps are done in a parallel fashion (just written down in an arbitrary order in the configuration sequence), we need to make sure to compare configurations where the subtrees have been processed equally far (we call this level-wise). Also, a tree transducer might not even read the whole input tree, as, e.g., in Example 2. We also (implicitly) take care of this in our definition.



(a) partial tree t_1. **(b)** partial tree t_2. **(c)** partial trees resulting from removing $t_1 \wedge t_2$ from t_1 and t_2.

Figure 3 The greatest common prefix of the partial trees t_1 and t_2, $t_1 \wedge t_2$, is marked with circles in t_1 and t_2. The delay between t_1 and t_2 is computed from their non-common parts as $\mathit{delay}(t_1, t_2) = \max\{h(t) + 1 \mid t \in \{b, h(a), a\}\} = 2$.

The result is the maximum of the delay between output trees of compatible configurations. Given $t \in T_\Sigma$, let $\mathit{Prefs}_{\mathit{level}}(t)$ denote the set of all prefix trees of t such that if a node at level i is kept, then all other nodes at level i are kept, i.e., for $t = f(h(a), h(a))$, $\mathit{Prefs}_{\mathit{level}}(t)$ contains $f(h, h)$, but not $f(h(a), h)$. Given an intermediate configuration (t, t_i', φ') of the run ρ_i, we recall that $t_i' \in T_{\Gamma \cup Q_{\mathcal{T}_i}}$ meaning t_i' contains states of $\mathcal{T}_i$ as leaves. Let $t_i'|\Gamma$ denote the partial tree obtained from t_i by removing all non-Γ-labelled nodes. We define $\mathit{delay}(\rho_1, \rho_2)$ as

$$\max\{\mathit{delay}(t_1'|\Gamma, t_2'|\Gamma) \mid \text{there is } t' \in \mathit{Prefs}_{\mathit{level}}(t), \text{ there is } (t, t_1', \varphi') \text{ in } \rho_1 \text{ with } t_1' \in \mathcal{T}_1(t'),$$
$$\text{and, there is } (t, t_2', \varphi'') \text{ in } \rho_2 \text{ with } t_2' \in \mathcal{T}_2(t') \ \}.$$

The conditions $t_1' \in \mathcal{T}_1(t')$ and $t_2' \in \mathcal{T}_2(t')$ are introduced to make sure that all input nodes that can be processed from t' are processed in the selected configurations.

We introduce (shorthand) notations. Let $\mathcal{T}_1, \mathcal{T}_2 \in \mathcal{C}$, where $\mathcal{C}$ is a class of transducers. Given $(t, s) \in R(\mathcal{T}_1)$, we say (t, s) is *k-delay included* in $R(\mathcal{T}_2)$, written $(t, s) \in_{\mathbb{D}_k} R(\mathcal{T}_2)$, if there are runs ρ and ρ' of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, with input t and output s such that $\mathit{delay}(\rho, \rho') \leq k$. We say $\mathcal{T}_1$ is *k-delay included* in $\mathcal{T}_2$, written $\mathcal{T}_1 \subseteq_{\mathbb{D}_k} \mathcal{T}_2$, if $(t, s) \in_{\mathbb{D}_k} R(\mathcal{T}_2)$ for all $(t, s) \in R(\mathcal{T}_1)$. We say $\mathcal{T}_1$ and $\mathcal{T}_2$ are *k-delay equivalent*, written $\mathcal{T}_1 =_{\mathbb{D}_k} \mathcal{T}_2$, if $\mathcal{T}_1 \subseteq_{\mathbb{D}_k} \mathcal{T}_2$ and $\mathcal{T}_2 \subseteq_{\mathbb{D}_k} \mathcal{T}_1$. We say $\mathcal{T}_1$ *k-delay uniformizes* $\mathcal{T}_2$ if $\mathcal{T}_1 \subseteq_{\mathbb{D}_k} \mathcal{T}_2$ and $\mathcal{T}_1$ uniformizes $\mathcal{T}_2$. The k-delay decision problems are defined as expected.

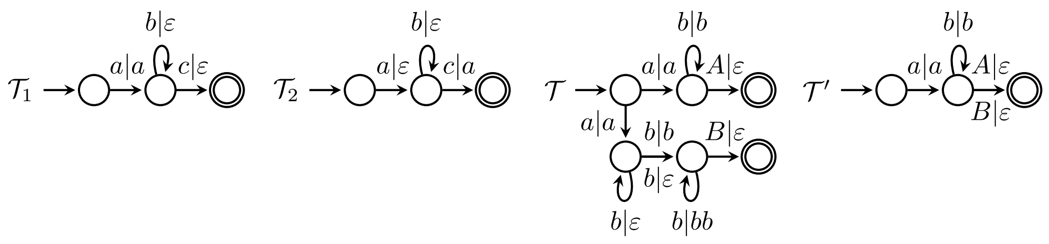
In order to get a better understanding of the expressiveness and differences between the two similarity measures, we first explore their properties on word transductions since words are a particular case of trees (i.e., monadic trees).

Word transducers. We denote by FST a *finite state transducer*, the class of word transductions recognized by FSTs is the class of *rational transductions*, conveniently also denoted by FST. We omit a formal definition of FSTs, because they are not considered outside of this section. An FST is *sequential* if its transitions are input-deterministic, more formally, a DTDTT over ranked alphabets with only unary and leaf symbols can be seen as an FST.

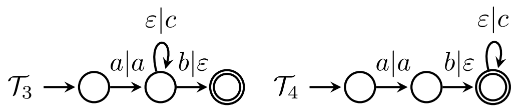
The results below concern origin distance, the same results were proven for delay in [15].

▶ **Proposition 7.**

1. *There exist* FSTs *$\mathcal{T}_1$, $\mathcal{T}_2$ such that $\mathcal{T}_1 \subseteq \mathcal{T}_2$, but $\mathcal{T}_1 \not\subseteq_k \mathcal{T}_2$ for all $k \geq 0$.*
2. *There exist* FSTs *$\mathcal{T}_1$, $\mathcal{T}_2$ such that $\mathcal{T}_1 = \mathcal{T}_2$, but $\mathcal{T}_1 \neq_k \mathcal{T}_2$ for all $k \geq 0$.*
3. *There exists an* FST *$\mathcal{T}$ such that $\mathcal{T}$ is sequentially uniformizable, but $\mathcal{T}$ is not k-origin sequentially uniformizable for all $k \geq 0$.*



(a) We have $\mathcal{T}_1 = \mathcal{T}_2$, and $\mathcal{T}_1 =_{\mathbb{D}_1} \mathcal{T}_2$, but $\mathcal{T}_1 \neq_k \mathcal{T}_2$ for all $k \geq 0$; $\mathcal{T}'$ is a sequential uniformizer of $\mathcal{T}$.



(b) We have $\mathcal{T}_3 = \mathcal{T}_4$, and $\mathcal{T}_3 =_1 \mathcal{T}_4$, but $\mathcal{T}_3 \neq_{\mathbb{D}_k} \mathcal{T}_4$ for all $k \geq 0$.

Figure 4 Comparing origin distance and delay for word transducers, see the proof of Proposition 7 and Example 17.

Proof. First, consider the FSTs $\mathcal{T}_1, \mathcal{T}_2$ depicted in Figure 4a. Both recognize the same function $f\colon \{a,b,c\}^* \to \{a\}^*$ defined as $f(ab^*c) = a$. Clearly, $\mathcal{T}_1 \subseteq \mathcal{T}_2$ and $\mathcal{T}_1 = \mathcal{T}_2$. However, the origin distance between $\mathcal{T}_1$ and $\mathcal{T}_2$ is unbounded. In $\mathcal{T}_1$, the origin of the single output letter a is always the first input letter. In contrast, in $\mathcal{T}_2$, the origin of the output letter a is always the last input letter. Secondly, consider the FSTs $\mathcal{T}, \mathcal{T}'$ depicted in Figure 4a. The recognized relation $\mathcal{R}(\mathcal{T}) \subseteq \{a,b,A,B\}^* \times \{a,b\}^*$ consists of $\{(ab^nA, ab^n) \mid n \in \mathbb{N}\}$ and $\{(ab^nB, ab^m) \mid 0 \leq m \leq 2n-1, n \in \mathbb{N}\}$. The sequential transducer $\mathcal{T}'$ recognizes the function $f\colon \{a,b,A,B\}^* \to \{a,b\}^*$ defined by $f(ab^nX) = ab^n$ for $X \in \{A,B\}$ and all $n \in \mathbb{N}$. Clearly, $\mathcal{T}'$ is a sequential uniformization of $\mathcal{T}$. However, no sequential uniformization with bounded origin distance exists, see Appendix B. ◀

We give an example (depicted in Figure 4 and described in detail in Example 17) that shows that the two notions are orthogonal to each other. However, if we restrict the class FST to *real-time*[1] FST, that is, word transducers such that in every transition exactly one input symbol is read, the notion of delay is more powerful than origin distance, see below. It is important to note that we have proven Proposition 7 for real-time FSTs which are equivalent to TDTTs on monadic trees, i.e., Proposition 7 is true for the class TDTT.

▶ **Proposition 8.** *Let $\mathcal{T}_1, \mathcal{T}_2$ be real-time FSTs, if $\mathcal{T}_1 \subseteq_i \mathcal{T}_2$ for some $i \geq 0$, then $\mathcal{T}_1 \subseteq_{\mathbb{D}_j} \mathcal{T}_2$ for some $j \geq 0$.*

The notion of bounded delay is suitable to regain decidability.

▶ **Theorem 9** ([15]). *Given $k \geq 0$, k-delay inclusion, k-delay equivalence and k-delay sequential uniformization are decidable for the class FST.*

Ideally, we would like to lift Theorem 9 from word to tree transducers, but it turns out that the notion of delay is too expressive to yield decidability results for tree transducers as shown in the next paragraph.

[1] ε-transitions (as, e.g., the loop in $\mathcal{T}_3$ from Figure 4, not be confused with non-producing transitions) are standard for FST, and non-standard for TDTT in the literature. We consider "real-time" TDTT by default.

Tree transducers. It is undecidable whether a given tree-automatic relation has a uniformization by a synchronous DTDTT [24]. A TDTT is called synchronous if for one processed input node one output node is produced as, e.g., in Example 2. Tree-automatic relations are a subclass of the relations that are recognizable by synchronous TDTT. To prove the result, the authors showed that

▶ **Lemma 10** ([24]). *There exists a synchronous* TDTT $\mathcal{T}_M$*, based on a Turing machine* M*, that is 0-delay* DTDTT*-uniformizable iff* M *halts on the empty input.*

In the proof, for a TM M, a DTDTT $\mathcal{T}'_M$ is constructed such that $\mathcal{T}'_M$ 0-delay uniformizes $\mathcal{T}_M$ iff M halts on the empty input. Recall that this implies that $\mathcal{T}'_M \subseteq_{\mathbb{D}_0} \mathcal{T}_M$ iff M halts on the empty input. Consequently, we obtain that

▶ **Theorem 11.** *Given $k \geq 0$, k-delay inclusion and k-delay* DTDTT*-uniformization are undecidable for the class* TDTT *(even for $k = 0$).*

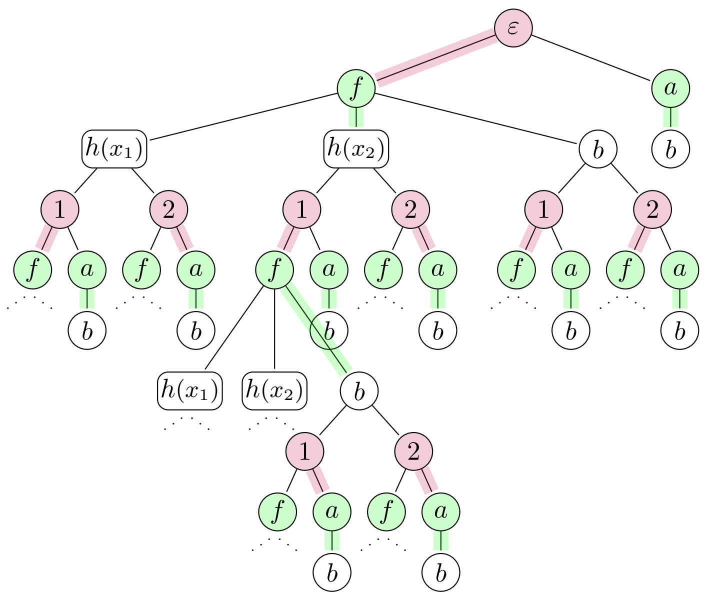
We do not know whether k-equivalence is decidable for a given $k \geq 0$. Note that Theorem 11 does not imply that 0-origin inclusion and 0-origin DTDTT-uniformization is undecidable for the class TDTT. For the class FST, the notions of 0-origin and 0-delay fall together, but for TDTT this is no longer the case. Recall the TDTT given in Example 2 and its unique runs that yield the origin mappings depicted in Figure 2. The delay between these runs is zero, but their origin mappings are different. An analysis of the (un)decidability proof(s) in [24] pins the problems down to the fact that in the specification and in the possible implementations the origins for the same output node lie on different paths in the input tree. For trees, this fact has no influence when measuring the delay between computations (as seen in Example 2). However, it is recognizable using the origin distance as measure. Since the notion of delay is so powerful that the decision problems under bounded delay become undecidable for tree transducers (see Theorem 11) in contrast to word transducers (see Theorem 9), in the next section, we focus on bounded origin distance.

4 Decision problems for origin-close transducers

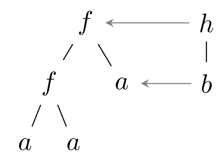
We show that the decision problems become decidable for top-down tree transducers with bounded origin distance, see Theorem 16. The next part is devoted to explaining our proof ideas and introducing our main technical lemma (Lemma 13) which is used in all proofs.

Origin-close transductions are representable as regular tree languages. Given $k \geq 0$, and a TDTT $\mathcal{T}$, we construct an infinite tree $\mathsf{H}_{\mathcal{T},k}$, given as the unfolding of a finite graph $\mathsf{G}_{\mathcal{T},k}$, such that a node in this infinite tree represents an input sequence from a finite input tree and an output sequence (where the intuition is that this output sequence was produced while processing this input sequence). The idea is that in $\mathsf{H}_{\mathcal{T},k}$, we define choices (aka. strategies) of two so-called players In and Out, where a strategy t of In together with a strategy s of Out defines an input tree t, an output tree s, and an origin mapping $o \colon dom_s \to dom_t$ of s in t. We can annotate the tree $\mathsf{H}_{\mathcal{T},k}$ with the strategies t and s which yields a tree $\mathsf{H}_{\mathcal{T},k}{}^\frown\mathsf{t}^\frown\mathsf{s}$. We use this game-like view for all considered decision problems. We illustrate this view.

▶ **Example 12.** Recall the TDTT $\mathcal{T}$ over Σ and Γ given in Example 2. First, we explain how the graph $\mathsf{G}_{\mathcal{T},0}$ looks like. Its unfolding is the infinite tree $\mathsf{H}_{\mathcal{T},0}$ (with annotations t and s) depicted in Figure 5. We have three types of nodes: $\{\varepsilon, 1, 2\}$ to indicate that the current node is the root, a first or a second child. The maximum rank of Σ is two, hence $\{\varepsilon, 1, 2\}$. These nodes belong to In who can choose the input label, represented by nodes $\{f, a\}$. Then



(a) $\mathsf{H}_{\mathcal{T},0}{}^\frown\mathsf{t}^\frown\mathsf{s}$.



(b) (t,s,o).

Figure 5 Infinite tree $\mathsf{H}_{\mathcal{T},0}$ based on $\mathcal{T}$ from Example 2. On red nodes In must make a choice, on green nodes Out must make a choice. Their respective strategies t and s which define their choices are highlighted on the edges in red and green, respectively. Together, t and s encode the input tree $t = f(f(a,a),a)$, the output tree $s = h(b)$ and origin mapping $o\colon dom_s \to dom_t$ as depicted. Note that since t and s are strategies, choices are made whatever the other player does, that is why in $\mathsf{H}_{\mathcal{T},0}{}^\frown\mathsf{t}^\frown\mathsf{s}$, we also have, e.g., a green annotation at node 1112 even though In picked node 1111.

Out chooses which output (from $T_\Gamma(X)$) should be produced while processing a node. Since $k = 0$, and all right-hand sides of rules in $\mathcal{T}$ have height at most one, only outputs of height at most one are suitable to maintain origin distance $k = 0$. For input f possible choices are $h(x_1)$ and $h(x_2)$, indicating whether to continue to process the left or the right subtree, or b. For input a only output b is possible. After the output, edges to $\{1, \cdots, rk(\sigma)\}$ exist, where σ is the last seen input letter. Further explanation is given in Figure 5.

We present our main technical lemma which states that origin-close transductions are representable as tree language recognizable by a parity tree automaton (a PTA).

▶ **Lemma 13.** *Given $k \geq 0$ and a TDTT $\mathcal{T}$, there exists a PTA that recognizes the tree language $\{\mathsf{H}_{\mathcal{T},k}{}^\frown\mathsf{t}^\frown\mathsf{s} \mid (t,s,o) \in_k R_o(\mathcal{T})\}$.*

Proof sketch. The infinite tree $\mathsf{H}_{\mathcal{T},k}{}^\frown\mathsf{t}^\frown\mathsf{s}$ encodes a triple (t,s,o). We construct a PTA (which has in fact a safety acceptance condition) that guesses a run of $\mathcal{T}$ over the input tree t with output tree s that yields an origin mapping o' such that $(t,s,o') \in R_o(\mathcal{T})$ and $dist(o(i),o'(i)) \leq k$ which implies that $(t,s,o) \in_k R_o(\mathcal{T})$.

Checking whether $dist(o(i),o'(i)) \leq k$ can be done on-the-fly because the origin distance is bounded which implies that the difference trees of so-far produced output by the guessed run and the productions encoded by the annotations are of bounded size. Thus, they can be stored in the state space of the PTA. Even tough the construction idea is rather simple, the implementation and correctness proof are non-trivial. We face two difficulties. Firstly, we have to account for the fact that in o and o' origins for the same output node can lie on different paths of the input tree. However, since their distance is bounded, the amount of shared information that the PTA has to check on different paths is also bounded. Secondly, it is possible to have non-linear transformation rules (that is, rules with copy, e.g., $q(f(x_1,x_2)) \to f(q_1(x_2),q_2(x_2)))$ which adds another layer of complication. This causes that an unbounded number of output nodes can have the same input node as origin. We require that $\mathsf{H}_{\mathcal{T},k}{}^\frown\mathsf{t}^\frown\mathsf{s}$ is a tree over a ranked alphabet, hence we have to bound the number of output choices that can be made at an input node. We show that it suffices to only make a bounded number of output choices for each input node. The main insight is that when two continuations of the output tree depend on the same continuation of the input tree, then it suffices to only consider one of them (because the other one can be continued in the same way) if they share the same relevant information where relevant basically means that the state that $\mathcal{T}$ has reached (guessed by the PTA) at these two output nodes and the output difference trees compared to Out's choices (given by s) are the same. ◀

Solving decision problems for origin-close transducers. We show that deciding k-origin inclusion and equivalence for TDTTs reduces to deciding language inclusion for PTAs.

▶ **Proposition 14.** *Given $k \geq 0$, k-origin inclusion and k-origin equivalence are decidable for the class* TDTT.

Proof. Let $\mathcal{T}_1, \mathcal{T}_2$ be TDTTs over the same input and output alphabet. If $\mathcal{T}_1 \subseteq_k \mathcal{T}_2$, then $(t,s,o) \in R_o(\mathcal{T}_1)$ implies that $(t,s,o) \in_k R_o(\mathcal{T}_1)$ for all $(t,s,o) \in R_o(\mathcal{T}_1)$. Lemma 13 yields that there are PTAs $\mathcal{A}_1, \mathcal{A}_2$ that recognize $\{\mathsf{H}_{\mathcal{T}_1,0}{}^\frown\mathsf{t}^\frown\mathsf{s} \mid (t,s,o) \in R_o(\mathcal{T})\}$ and $\{\mathsf{H}_{\mathcal{T}_2,k}{}^\frown\mathsf{t}^\frown\mathsf{s} \mid (t,s,o) \in_k R_o(\mathcal{T}_2)\}$, respectively. Basically, we want to check that $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$. However, we have to overcome a slight technical difficulty. If there are trees $\mathsf{H}_{\mathcal{T}_1,0}{}^\frown\mathsf{t}_1{}^\frown\mathsf{s}_1 \in L(\mathcal{A}_1)$ and $\mathsf{H}_{\mathcal{T}_2,k}{}^\frown\mathsf{t}_2{}^\frown\mathsf{s}_2 \in L(\mathcal{A}_2)$ such that for their encoded triples (t_1,s_1,o_1) and (t_2,s_2,o_2) holds that $t_1 = t_2$, $s_1 = s_2$ and o_1 and o_2 have an origin difference of at most k, i.e., $(t_1,s_1,o_1) \in_k \mathcal{R}_0(\mathcal{T}_2)$, it not necessarily holds that $\mathsf{H}_{\mathcal{T}_1,0}{}^\frown\mathsf{t}_1{}^\frown\mathsf{s}_1 \in L(\mathcal{A}_2)$. This is due to the fact that the base trees $\mathsf{H}_{\mathcal{T}_1,0}$ and $\mathsf{H}_{\mathcal{T}_2,k}$ look different in general because choices for Out in the first tree are based on the rules of $\mathcal{T}_1$ and without origin distance and in the latter tree based on the rules of $\mathcal{T}_2$ with k-origin distance. We only care whether the paths reachable by following the annotations t_1 and s_1 through $\mathsf{H}_{\mathcal{T}_1,0}$ and the paths reachability by following the annotations t_2 and s_2 through $\mathsf{H}_{\mathcal{T}_2,k}$ are the same. Thus, we introduce the operation *purge* which applied to a tree annotated with strategies of In and Out removes all non-strategy paths. It is not difficulty to see that the sets $L_1 := \{\mathit{purge}(\mathsf{H}_{\mathcal{T}_1,0}{}^\frown\mathsf{t}^\frown\mathsf{s}) \mid (t,s,o) \in R_o(\mathcal{T})\}$ and $L_2 := \{\mathit{purge}(\mathsf{H}_{\mathcal{T}_2,k}{}^\frown\mathsf{t}^\frown\mathsf{s}) \mid (t,s,o) \in_k R_o(\mathcal{T}_2)\}$ are also PTA-recognizable. Hence, in order to check whether $\mathcal{T}_1 \subseteq_k \mathcal{T}_2$, we have to check whether $L_1 \subseteq L_2$, which is decidable. We have shown that k-origin inclusion for TDTTs is decidable, consequently, k-origin equivalence for TDTTs is decidable for all $k \geq 0$. ◀

We show that checking whether a TDTT is k-origin DTDTT-uniformizable reduces to deciding emptiness of PTAs.

▶ **Proposition 15.** *Given $k \geq 0$, k-origin* DTDTT*-uniformization is decidable for the class* TDTT.

Proof. Given a TDTT $\mathcal{T}$, by Lemma 13, there is a PTA that recognizes

$\{\mathsf{H}_{\mathcal{T},k}{}^\frown\mathsf{t}^\frown\mathsf{s} \mid (t,s,o) \in_k R_o(\mathcal{T})\}$.

By closure under complementation and intersection, there is a PTA that recognizes

$\{\mathsf{H}_{\mathcal{T},k}{}^\frown\mathsf{t}^\frown\mathsf{s} \mid (t,s,o) \notin_k R_o(\mathcal{T})\}$.

By closure under projection, there is a PTA that recognizes
$\{\mathsf{H}_{\mathcal{T},k}{}^\frown \mathsf{s} \mid \exists \mathsf{t} : (t,s,o) \notin_k R_o(\mathcal{T})\}$.
By closure under complementation and intersection, there is a PTA that recognizes
$\{\mathsf{H}_{\mathcal{T},k}{}^\frown \mathsf{s} \mid \forall \mathsf{t} : (t,s,o) \in_k R_o(\mathcal{T})\}$.
By closure under projection, there is a PTA that recognizes
$\{\mathsf{H}_{\mathcal{T},k} \mid \exists \mathsf{s} : \forall \mathsf{t} : (t,s,o) \in_k R_o(\mathcal{T})\}$.

Let $\mathcal{A}$ denote the PTA obtained in the last construction step. We show that $\mathcal{T}$ is k-origin DTDTT-uniformizable iff $L(\mathcal{A}) \neq \emptyset$. We have that $L(\mathcal{A}) = \{\mathsf{H}_{\mathcal{T},k} \mid \exists \mathsf{s} : \forall \mathsf{t} : (t,s,o) \in_k R_o(\mathcal{T})\}$. Colloquially, this means that we can fix output choices that only depend on the previously seen input choices, which exactly describes DTDTT-uniformizability.

Assume $\mathcal{T}$ is k-origin DTDTT-uniformizable, say by a DTDTT $\mathcal{T}'$. There exists a strategy of Out in $\mathsf{H}_{\mathcal{T},k}$ that copies the computations of $\mathcal{T}'$. Clearly, since $\mathcal{T}'$ is deterministic, we obtain that $\exists \mathsf{s} : \forall \mathsf{t} : (t,s,o) \in_k R_o(\mathcal{T})$, s can be chosen to be the strategy that copies $\mathcal{T}'$. Thus, $L(\mathcal{A}) \neq \emptyset$.

For the other direction, assume that $L(\mathcal{A}) \neq \emptyset$. This implies that also the set $\{\mathsf{H}_{\mathcal{T},k}{}^\frown \mathsf{s} \mid \forall \mathsf{t} : (t,s,o) \in_k R_o(\mathcal{T})\}$ is non-empty and PTA recognizable. Since the set is PTA recognizable, it contains a regular infinite tree (meaning the tree has a finite representation). This tree implicitly contains a finite representation of some strategy s such that $\forall \mathsf{t} : (t,s,o) \in_k R_o(\mathcal{T})$. Hence, the strategy s can be translated into a finite-state DTDTT that k-origin uniformizes $\mathcal{T}$. ◀

Finally, combining Propositions 14 and 15, we obtain our main result.

▶ **Theorem 16.** *Given $k \geq 0$, k-origin inclusion, k-origin equivalence, and k-origin* DTDTT-*uniformization are decidable for the class* TDTT.

5 Conclusion

We introduced two similarity measures for TDTTs based on their behavior and studied decision problems for similar TDTTs. For TDTTs with bounded delay, the decision problems remain undecidable. For origin-close TDTTs they become decidable. For future work, we plan to consider other tree transducer models. In [16], it was shown that origin inclusion and origin equivalence are decidable for MSO tree transducers and macro tree transducers.

References

1 Mikolaj Bojanczyk. Transducers with origin information. In *ICALP (2)*, volume 8573 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2014.

2 Mikolaj Bojanczyk, Laure Daviaud, Bruno Guillon, and Vincent Penelle. Which classes of origin graphs are generated by transducers. In *ICALP*, volume 80 of *LIPIcs*, pages 114:1–114:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.

3 Sougata Bose, Shankara Narayanan Krishna, Anca Muscholl, Vincent Penelle, and Gabriele Puppis. On synthesis of resynchronizers for transducers. In *MFCS*, volume 138 of *LIPIcs*, pages 69:1–69:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

4 Sougata Bose, Anca Muscholl, Vincent Penelle, and Gabriele Puppis. Origin-equivalence of two-way word transducers is in PSPACE. In *FSTTCS*, volume 122 of *LIPIcs*, pages 22:1–22:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

5 Fabienne Braune, Nina Seemann, Daniel Quernheim, and Andreas Maletti. Shallow local multi-bottom-up tree transducers in statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 811–821, 2013.

6 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.

7 Arnaud Carayol and Christof Löding. Uniformization in Automata Theory. In *Proceedings of the 14th Congress of Logic, Methodology and Philosophy of Science Nancy, July 19-26, 2011*, pages 153–178. London: College Publications, 2014.

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

9 María Emilia Descotte, Diego Figueira, and Santiago Figueira. Closure properties of synchronized relations. In *STACS*, volume 126 of *LIPIcs*, pages 22:1–22:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

10 Joost Engelfriet. Top-down tree transducers with regular look-ahead. *Math. Syst. Theory*, 10:289–303, 1977.

11 Joost Engelfriet and Sebastian Maneth. Macro tree translations of linear size increase are mso definable. *SIAM Journal on Computing*, 32(4):950–1006, 2003.

12 Joost Engelfriet, Sebastian Maneth, and Helmut Seidl. Look-ahead removal for total deterministic top-down tree transducers. *Theor. Comput. Sci.*, 616:18–58, 2016.

13 Zoltán Ésik. Decidability results concerning tree transducers i. *Acta Cybernetica*, 5(1):1–20, 1980.

14 Diego Figueira and Leonid Libkin. Synchronizing relations on words. *Theory Comput. Syst.*, 57(2):287–318, 2015.

15 Emmanuel Filiot, Ismaël Jecker, Christof Löding, and Sarah Winter. On equivalence and uniformisation problems for finite transducers. In *ICALP*, volume 55 of *LIPIcs*, pages 125:1–125:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.

16 Emmanuel Filiot, Sebastian Maneth, Pierre-Alain Reynier, and Jean-Marc Talbot. Decision problems of tree transducers with origin. *Inf. Comput.*, 261:311–335, 2018.

17 Patrick C. Fischer and Arnold L. Rosenberg. Multitape one-way nonwriting automata. *Journal of Computer and System Sciences*, 2(1):88–101, 1968. doi:10.1016/S0022-0000(68)80006-6.

18 Zoltán Fülöp and Heiko Vogler. *Syntax-directed semantics: Formal models based on tree transducers*. Springer Science & Business Media, 2012.

19 Timothy V. Griffiths. The unsolvability of the equivalence problem for lambda-free nondeterministic generalized machines. *Journal of the ACM*, 15(3):409–413, 1968.

20 Shizuya Hakuta, Sebastian Maneth, Keisuke Nakano, and Hideya Iwasaki. Xquery streaming by forest transducers. In *2014 IEEE 30th International Conference on Data Engineering*, pages 952–963. IEEE, 2014.

21 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.

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

23 Ralf Küsters and Thomas Wilke. Transducer-based analysis of cryptographic protocols. *Information and Computation*, 205(12):1741–1776, 2007.

24 Christof Löding and Sarah Winter. Uniformization problems for tree-automatic relations and top-down tree transducers. In *MFCS*, volume 58 of *LIPIcs*, pages 65:1–65:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.

25 Christof Löding and Sarah Winter. Synthesis of deterministic top-down tree transducers from automatic tree relations. *Inf. Comput.*, 253:336–354, 2017.

26 Andreas Maletti. Tree transformations and dependencies. In *Conference on Mathematics of Language*, pages 1–20. Springer, 2011.

27 Andreas Maletti, Jonathan Graehl, Mark Hopkins, and Kevin Knight. The power of extended top-down tree transducers. *SIAM Journal on Computing*, 39(2):410–430, 2009.

28 Kazutaka Matsuda, Kazuhiro Inaba, and Keisuke Nakano. Polynomial-time inverse computation for accumulative functions with multiple data traversals. *Higher-Order and Symbolic Computation*, 25(1):3–38, 2012.

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

30 William C Rounds. Mappings and grammars on trees. *Mathematical systems theory*, 4(3):257–287, 1970.

31 James W. Thatcher. Generalized² sequential machine maps. *Journal of Computer and System Sciences*, 4(4):339–367, 1970.

32 Arie Van Deursen, Paul Klint, and Frank Tip. Origin tracking. *Journal of Symbolic Computation*, 15(5-6):523–545, 1993.

33 Janis Voigtländer and Armin Kühnemann. Composition of functions with accumulating parameters. *Journal of functional programming*, 14(3):317, 2004.

34 Sarah Winter. Uniformization problems for synchronizations of automatic relations on words. In *ICALP*, volume 107 of *LIPIcs*, pages 142:1–142:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

A Missing example

▶ **Example 17.** To begin with, consider the FSTs $\mathcal{T}_1, \mathcal{T}_2$ depicted in Figure 4a, we already explained in the proof of Proposition 7 that $\mathcal{T}_1 = \mathcal{T}_2$, but $\mathcal{T}_1 \neq_k \mathcal{T}_2$ for all $k \geq 0$, i.e., their origin distance is unbounded. However, their delay is bounded by 1. It is easy to see that $\mathcal{T}_1 =_{\mathbb{D}_1} \mathcal{T}_2$, because their difference in the length of their outputs for the same input is at most one letter. Now, consider the FSTs $\mathcal{T}_3, \mathcal{T}_4$ depicted in Figure 4b. Both recognize the relation $\{(ab, c^n) \mid n \in \mathbb{N}\}$, hence, $\mathcal{T}_3 = \mathcal{T}_4$. Clearly, their origin distance is bounded by 1. The whole output either has the first or the second letter as origin. However, $\mathcal{T}_3 \neq_{\mathbb{D}_k} \mathcal{T}_4$ for all $k \geq 0$, i.e., their delay is unbounded. For any k, take the consider the unique runs that admit output c^{k+1} in $\mathcal{T}_3$ and $\mathcal{T}_4$, respectively. We compare these runs for the input prefix a, $\mathcal{T}_3$, already has produced c^{k+1}, and $\mathcal{T}_4$ no output so far. Their delay is $k + 1$.

B Missing proofs of Propositions 7 and 8

▶ **Proposition 7.**

1. *There exist* FSTs $\mathcal{T}_1$, $\mathcal{T}_2$ *such that* $\mathcal{T}_1 \subseteq \mathcal{T}_2$, *but* $\mathcal{T}_1 \not\subseteq_k \mathcal{T}_2$ *for all* $k \geq 0$.
2. *There exist* FSTs $\mathcal{T}_1$, $\mathcal{T}_2$ *such that* $\mathcal{T}_1 = \mathcal{T}_2$, *but* $\mathcal{T}_1 \neq_k \mathcal{T}_2$ *for all* $k \geq 0$.
3. *There exists an* FST $\mathcal{T}$ *such that* $\mathcal{T}$ *is sequentially uniformizable, but* $\mathcal{T}$ *is not k-origin sequentially uniformizable for all* $k \geq 0$.

Proof. Secondly, consider the FSTs $\mathcal{T}, \mathcal{T}'$ depicted in Figure 4a. The recognized relation $\mathcal{R}(\mathcal{T}) \subseteq \{a, b, A, B\}^* \times \{a, b\}^*$ consists of $\{(ab^nA, ab^n) \mid n \in \mathbb{N}\}$ and $\{(ab^nB, ab^m) \mid 0 \leq m \leq 2n-1, n \in \mathbb{N}\}$. The sequential transducer $\mathcal{T}'$ recognizes the function $f\colon \{a, b, A, B\}^* \to \{a, b\}^*$ defined by $f(ab^nX) = ab^n$ for $X \in \{A, B\}$ and all $n \in \mathbb{N}$. Clearly, $\mathcal{T}'$ is a sequential uniformization of $\mathcal{T}$. However, no sequential uniformization with bounded origin distance exists. Towards a contradiction, assume there is sequential transducer $\mathcal{T}''$ that uniformizes $\mathcal{T}$ such that $\mathcal{T}'' \subseteq_k \mathcal{T}$ for some $k \geq 0$. Consider the input word $ab^{2k}A$, in $\mathcal{T}$ there is only one run with the input which yields the output ab^{2k} and the origin of the ith output letter is the ith input letter for all i. Since $\mathcal{T}'' \subseteq_k \mathcal{T}$ there exists a run of $\mathcal{T}''$ on $ab^{2k}A$ that yields ab^{2k} and the origin of the first b in the output is at latest the kth b in the input. Now, consider the input $ab^{6k}B$, the output of $\mathcal{T}''$ on $ab^{6k}B$ is ab^{6k}. Since $\mathcal{T}''$ is sequential, the the runs of $\mathcal{T}''$ on $ab^{2k}A$ and $ab^{6k}B$ are the same up to the input ab^{2k}, thus, also for the output ab^{6k}

the origin first output b is at latest the kth b in the input. Now we compare this with all possible runs in $\mathcal{T}$ on $ab^{6k}B$ that also yield ab^{6k}. Note that $\mathcal{T}$ (after producing the first b) must always produce two b at once, thus in order to produce ab^{6k} for the input $ab^{6k}B$, the production of b can only start after while reading the second half of the input. This implies that the first output has an origin in the second half of input which has a distance of more than k (at least $2k$) to the kth b in the input. ◀

▶ **Proposition 8.** *Let $\mathcal{T}_1, \mathcal{T}_2$ be real-time* FSTs*, if $\mathcal{T}_1 \subseteq_i \mathcal{T}_2$ for some $i \geq 0$, then $\mathcal{T}_1 \subseteq_{\mathbb{D}_j} \mathcal{T}_2$ for some $j \geq 0$.*

Proof. Let $\mathcal{T}_1, \mathcal{T}_2$ be real-time FSTs such that $\mathcal{T}_1 \subseteq_i \mathcal{T}_2$ for some $i \geq 0$. Let ℓ be the maximum number of output letters that $\mathcal{T}_1$ produces in a computation step. Consider any $(u, v, o_1) \in R_o(\mathcal{T}_1)$, since $\mathcal{T}_1 \subseteq_i \mathcal{T}_2$, there is $(u, v, o_2) \in R_o(\mathcal{T}_2)$ such that $dist(o_1(d), o_2(d)) \leq i$ for all $d \in dom_v$. Let ρ_1 and ρ_2 be the corresponding runs of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. We show that $delay(\rho_1, \rho_2) \leq \ell \cdot i$ which implies that $\mathcal{T}_1 \subseteq_{\mathbb{D}_{\ell \cdot i}} \mathcal{T}_2$. Let $u = a_1 \cdots a_n$ and $v = b_1 \cdots b_m$. Pick any prefix of u, say $a_1 \cdots a_k$, and consider the prefixes of the runs ρ_1 and ρ_2 such that the input $a_1 \cdots a_k$ has been processed. Let $b_1 \cdots b_{k_1}$ and $b_1 \cdots b_{k_2}$ be the respective produced outputs. Wlog., let $k_1 \leq k_2$. If $k_1 = k_1$, then the output delay for the prefix $a_1 \cdots a_k$ is zero. So assume $k_1 < k_2$. We have to show that $|b_{k_1+1} \cdots b_{k_2}|$ is less than $\ell \cdot i$. Since the origin mappings of ρ_1 and ρ_2, that is, o_1 and o_2, have a distance of at most i, we know that the origin of $b_{k_1+1} \cdots b_{k_2}$ in ρ_1 is no later than at the letter a_{k+i}. On $a_{k+1} \cdots a_{k+1}$, $\mathcal{T}_1$ can produce at most $\ell \cdot i$ output letters. Consequently, $|b_{k_1+1} \cdots b_{k_2}| \leq \ell \cdot i$. ◀