Decision Problems of Tree Transducers with Origin

Emmanuel Filiot[a], Sebastian Maneth[b], Pierre-Alain Reynier[1], Jean-Marc Talbot[1]

[a] *Université Libre de Bruxelles & FNRS*
[b] *University of Edinburgh*
[c] *Aix-Marseille University & CNRS*

Abstract

A tree transducer with origin translates an input tree into a pair of output tree and origin information. The origin information maps each node in the output tree to the unique node in the input tree that created it. In this way, the implementation of the transducer becomes part of its semantics. We show that the landscape of decidable properties changes drastically when origin information is added. For instance, equivalence of nondeterministic top-down and MSO transducers with origin becomes decidable. Both problems are undecidable without origin. The equivalence of deterministic top-down tree-to-string transducers is decidable with origin, while without origin it has (until very recently) been a long standing open problem. With origin, we can decide if a deterministic macro tree transducer can be realized by a deterministic top-down tree transducer; without origin this is an open problem.

Keywords: top-down and MSO tree transducers, origin, decidability, equivalence, injectivity, query determinacy

1. Introduction

Tree transducers were invented in the early 1970's as a formal model for compilers and linguistics by Thatcher [37] and Rounds [35]. They are being applied in many fields of computer science, such as syntax-directed translation [23], databases [33, 25], linguistics [30, 6], programming languages [40, 32], and security analysis [27]. The most essential feature of tree transducers is their good balance between expressive power and decidability.

	top-down tree-to-tree		top-down tree-to-string		MSO tree-to-string	
	det	nd	det	nd	det	nd
no origin	+ [19]	− [24]	[36]	− [24]	+ [15]	−
with origin	+	+	+	−	+	+

Table 1: Decidability of equivalence

Bojańczyk [5] introduces (string) transducers with origin. For "regular" string-to-string transducers with origin he presents a machine independent characterization which admits Angluin-style learning and the decidability of natural subclasses. These results indicate that classes of translations with origin are mathematically even better behaved than their origin-less counter parts.

We initiate a rigorous study of tree transducers with origin by investigating the decidability of *equivalence*, *injectivity*, and *query determinacy* on the following models:

- top-down tree-to-tree transducers [37, 35],
- top-down tree-to-string transducers [17], and
- MSO definable tree-to-string transducers, see, *e.g.*, [15].

Unlike the string transducers of Bojańczyk [5], we will see that equivalent models of tree-to-string transducers do *not* remain equivalent in the presence of origin. This motivates the study of *subclass definability* problems (definability of a translation from a class in a subclass) when considering the origin semantics.

Table 1 summarizes our results on equivalence; the words non-/deterministic are abbreviated by nd/det and decidable/undecidable by +/−. The first change from − to + is the equivalence of nondeterministic top-down tree transducers. In the non-origin case this problem is already undecidable for restricted string-to-string transducers, as shown by Griffith [24]. In the presence of origin it becomes decidable for tree transducers, because origin implies that any connected region of output nodes with the same origin is generated by one single rule. Hence, the problem reduces to letter-to-letter transducers as studied by Andre and Bossut [1]. What about nondeterministic top-down *tree-to-string* transducers (column four in Table 1)? Here output patterns cannot be treated as letters. By deferring output generation to a leaf they can simulate non-origin translations with undecidable equivalence (*i.e.*, finite-state string transducers) [24]. Finally, let us discuss column three. Here the origin information induces a structure on the output strings: recursive calls of origin-equivalent transducers must occur in similar "blocks", so that the same children of the current input node are visited in the same order (but possibly with differing numbers of recursive calls). This block structure allows to reason over single input paths, and to reduce the problem to deterministic tree-to-string transducers with monadic input. The latter can be reduced [31] to the famous HDT0L sequence equivalence problem.

Injectivity for deterministic transducers is undecidable for all origin-free models of Table 1. With origin, we prove undecidability in the tree-to-string case and decidability in the MSO and top-down tree cases. The latter is again due to the rigid structure implied by origins. We can track if two different inputs, over the *same* input nodes, produce the same output tree. We use the convenient framework of recognizable relations to show that the set of trees for which a transducer with origin produces the same output can be recognized by a tree automaton.

We present two results on subclass definability. Recall from [14] that a deterministic top-down tree-to-string translation is MSO definable if and only if it is of linear size increase. As mentioned above, a top-down tree-to-string transducer can defer its output generation to the leaves of the input tree. For instance, it can realize the identity on monadic input trees, by producing a state call for each input node; each such state call ignores the rest of the tree until it reaches the unique input leaf. Thus, even for a translation of linear size increase, the number of output nodes with same origin can be unbounded. In contrast, for an MSO transducer the number of nodes with same origin is always bounded ("bounded origin property"). Thus, origin translations of deterministic top-down tree-to-string transducers form a strict superclass of those of MSO transducers (the same holds for the string-to-string "regular" translations studied by Bojańczyk [5]). We prove that for a given deterministic top-down tree-to-string transducer it is decidable whether an origin-equivalent MSO transducer exists; this is done by deciding the bounded origin property.

The second subclass definability result concerns a more powerful model of tree transducer: the *macro tree transducer* [18]. This model can be seen as a generalization of top-down tree transducers by adding context-parameters (of type output tree) to the states. We show that, under origin semantics, it is decidable for a given total deterministic macro tree transducer, whether it can be realized by a top-down tree transducer. This is an open problem in the non-origin setting. The proof relies on two properties: (1) the origin translation must be order-preserving, *i.e.*, the origins of output descendant nodes must be descendants, and (2) on each path of the output tree, the number of nodes with same origin must be bounded.

Motivation

Clearly, the more information we include in a translation, the more properties become decidable. Consider invertability: on the one extreme, if all reads and writes are recorded (under ACID), then any computation becomes invertible. The question then arises, how much information needs to be included in order to be invertible. This problem has recently deserved much attention in the programming language community, see, *e.g.*, [39]. Our work here was inspired by the very similar view/query determinacy problem. This problem asks for a given view and query, whether the query can be answered on the output of the view. It was shown decidable by Benedikt, Engelfriet, and Maneth [2] for views that are linear extended tree transducers, and queries that are deterministic MSO or top-down transducers. For views that include copying, the problem quickly becomes undecidable [2]. Our results show that such views *can* be supported, if origin is included. Consider for instance a view that regroups a list of publications into sublists of books, articles, etc. A tree transducer realizing this view *needs copying* (*i.e.*, needs to process the original list multiple times). Without origin, we do not know a procedure that decides determinacy for such a view. With origin, We prove that determinacy is decidable for views with origin and (origin-less) queries, where the view and query are either given by deterministic top-down transducers or by deterministic MSO transducers. As expected: the world becomes safer with origin, but more restrictive (*e.g.*, the query "is book X before article Y in the original list?" becomes determined when origin is added to the above view).

The tracking of origin information was studied in the programming language community, see the *e.g.* Deursen, Klint, and Tip [38]. As a technical tool it was used by Engelfriet and Maneth [14] to characterize the MSO definable macro tree translations, and, by Lemay, Maneth, and Niehren [28] to give a Myhill-Nerode theorem for deterministic top-down tree transducers. From a linguistic point of view, origin mappings on their own are subject of interest and are called "dependencies" or "links". Maletti [29] shows that dependencies (*i.e.*, origins) give "surprising insights" into the structure of tree translations: many separation results concerning expressive power can be obtained on the level of dependencies. Further such linking theorems are given by Fülöp and Maletti [22] and are used to prove that certain tree relations cannot be computed by multi bottom-up tree transducers.

A preliminary version of this paper was presented at ICALP [20].

2. Preliminaries

For a nonnegative integer k we denote by $[k]$ the set $\{1, \ldots, k\}$.

2.1. Strings, ranked trees, and contexts

Let A be an alphabet. We denote by A^* the set of strings over A, and by ε the empty string. Let $w \in A^*$. The length of a w is denoted by $|w|$. The set of positions of w is the set $V(w) = [|w|]$. For $j \in [|w|]$, $w[j]$ denotes the j-th symbol of w.

A *ranked alphabet* Σ is a finite set of symbols σ each with an associated natural number k called its rank. We write $\sigma^{(k)}$ to denote that σ has rank k, and denote by $\Sigma^{(k)}$ the set of all symbols in Σ of rank k.

The *set T_Σ of trees over* Σ is the smallest set T so that if $k \geq 0$, $t_1, \ldots, t_k \in T$, and $\sigma \in \Sigma^{(k)}$, then $\sigma(t_1, \ldots, t_k) \in T$. For the tree $\sigma()$ we simply write σ. The set $V(t)$ of nodes of tree $t \in T_\Sigma$ is the subset of $\mathbb{N}^*$ defined as $\{\varepsilon\} \cup \{iu \mid i \in [k], u \in V(t_i)\}$ if $t = \sigma(t_1, \ldots, t_k)$ and $\sigma \in \Sigma^{(k)}$. Thus ε denotes the root node, and ui denotes the i-th child of a node u. For clarity we often write $u.i$ instead of ui. The height of t, denoted $\mathsf{height}(t)$, is defined as one plus the maximum length of any node in $V(t)$. Two nodes u, v are *incomparable* if there does not exists w such that either $u = vw$ or $v = uw$.

For a tree t and $u \in V(t)$ we denote by $t[u]$ the label of node u in t, and by t/u the subtree of t rooted at u. For a tree t', we denote by $t[u \leftarrow t']$ the tree obtained from t by replacing the subtree rooted at node u by the tree t'. We extend this notation to parallel replacement: let $t_1, \ldots, t_n$ be trees and $u_1, \ldots, u_n$ be nodes from t that are pairwise incomparable. Then $t[u_1 \leftarrow t_1, \ldots, u_n \leftarrow t_n]$ stands for $(\ldots((t[u_1 \leftarrow t_1])[u_2 \leftarrow t_2]) \ldots [u_n \leftarrow t_n])$. Given $\Delta \subseteq \Sigma$, $V_\Delta(t)$ denotes the set of nodes $u \in V(t)$ such that $t[u] \in \Delta$.

A Σ-*context* is a tree C over $\Sigma \cup \{\bot^{(0)}\}$ where $\bot$ is a special symbol not in Σ. Let $u_1, \ldots, u_k$ be all $\bot$-nodes of C in preorder. Given trees $t_1, \ldots, t_k$, we denote by $C[t_1, \dots, t_k]$ the tree $C[u_i \leftarrow t_i \mid i \in [k]]$ obtained from C by replacing u_i by t_i for $i \in [k]$.

2.2. Translations

Let Σ, Δ be two ranked alphabets. A *tree translation* (*from* T_Σ *to* T_Δ) is a relation $R \subseteq T_\Sigma \times T_\Delta$. Let A be an alphabet. A *tree-to-string translation* is a relation $R \subseteq T_\Sigma \times A^*$. The *domain* of a translation R, denoted $\mathsf{dom}(R)$, is defined as the projection of R on its first component. A translation R is *functional* if R is a function.

2.3. Origin translations

Let s_1, s_2 be two structures (strings or trees). An *origin mapping* of s_2 in s_1 is a mapping $o : V(s_2) \to V(s_1)$. An *origin translation* is a set of pairs $(s_1, (s_2, o))$ such that o is an origin mapping of s_2 in s_1. Given $v \in V(s_2)$ and $u \in V(s_1)$, if $o(v) = u$ then we say that "v has origin u" or that "the origin of v is u".

3. Tree Translations with Origin

3.1. Top-down Tree Transducers

A top-down tree transducer (TOP transducer for short) is a rule-based finite-state device that translates ranked trees to ranked trees.

Definition 1. *A* top-down tree transducer *is a tuple* $(Q, \Sigma, \Delta, q_0, R)$ *where* Q *is a finite set of states,* Σ *is a ranked alphabet of input symbols,* Δ *is a ranked alphabet of output symbols,* $q_0 \in Q$ *is the initial state, and* R *is a set of rules of the form* $q(\sigma(x_1, \dots, x_k)) \to \zeta$*, where* $q \in Q$*,* $\sigma \in \Sigma^{(k)}$*, and* ζ *is a tree over* $\Delta \cup \{q'(x_i) \mid q' \in Q, i \in [k]\}$*, where each symbol* $q'(x_i)$ *has rank* 0*.*

Intuitively, applying the rule $q(\sigma(x_1, \dots, x_k)) \to \zeta$ to an input tree tree $s = \sigma(s_1, \dots, s_k)$ produces the output tree t obtained by replacing in ζ all symbols $q'(x_i)$ by trees over Δ, themselves obtained by evaluating s_i in state q'. The origin of all Δ-nodes in ζ is the root node of s. Since rules are non-deterministic, there might be several different ways of replacing the subtrees $q'(x_i)$ and therefore several output trees t can be associated with a single input tree s.

Formally, every state $q \in Q$ realizes an origin translation $[\![q]\!]_{\mathsf{o}}$ defined recursively as follows. Let $s = \sigma(s_1, \dots, s_k)$ be a tree and let $q(\sigma(x_1, \dots, x_k)) \to$

ζ be a rule of M. Let $V = V(\zeta) \setminus V_\Delta(\zeta)$ be the set of nodes in ζ with a label of the form $q'(x_i)$. For each $v \in V$, let $\zeta[v] = q_v(x_{i_v})$ where $q_v \in Q$ and $i_v \in [k]$. For all $v \in V$ and $(s_{i_v}, (t_v, o_v)) \in [\![q_v]\!]_\circ$, it holds that $(s, (t, o)) \in [\![q]\!]_\circ$ where

- $t = \zeta[v \leftarrow t_v \mid v \in V]$,
- $o(v') = \varepsilon$ for $v' \in V_\Delta(\zeta)$ and $o(vv') = i_v o_v(v')$ for $v \in V$ and $v' \in V(t_v)$.

The tree translation realized by q is defined as $[\![q]\!] = \{(s, t) \mid \exists o : (s, (t, o)) \in [\![q]\!]_\circ\}$. The *origin* (*tree*) *translation realized by* M is $[\![M]\!]_\circ = [\![q_0]\!]_\circ$, and the (*tree*) *translation realized by* M is $[\![M]\!] = [\![q_0]\!]$. For a TOP transducer M, we often write $\mathsf{dom}(M)$ for $\mathsf{dom}([\![M]\!])$.

Example 2. *Consider the* TOP *transducer* $M_1 = (Q, \Sigma, \Delta, q_0, R)$ *with* $Q = \{q\}$*,* $\Sigma = \{h^{(1)}, a^{(0)}\}$*,* $\Delta = \{f^{(2)}, a^{(0)}\}$*, and* R *consisting of the following rules.*

$$\begin{array}{lcl} q(h(x_1)) & \to & f(q(x_1), q(x_1)) \\ q(a) & \to & a. \end{array}$$

The transducer M_1 *translates a monadic input tree of height* n *into a balanced binary tree of height* n*. The origin of a node* u *in the output tree is the unique node of the input tree at the same depth as* u*. Hence, the origin of* u *is the input node* $1^{|u|}$*. This translation is illustrated in Figure 1 for input tree* $h(h(h(a)))$*.*

Example 3. *Consider the* TOP *transducer* $M_2 = (Q, \Sigma, \Delta, q_0, R)$ *with* $Q = \{q_0, q\}$*,* $\Sigma = \{h^{(1)}, a^{(0)}\}$*,* $\Delta = \{f^{(2)}, h^{(1)}, a^{(0)}\}$*, and* R *consisting of the following rules.*

$$\begin{array}{lcl} q_0(h(x_1)) & \to & f(q_0(x_1), q(x_1)) \\ q_0(a) & \to & a \\ q(h(x_1)) & \to & h(q(x_1)) \\ q(a) & \to & a. \end{array}$$

This transducer translates a monadic input tree of height n *into a left-comb of monadic subtrees of decreasing height. Thus,* $h(h(h(a)))$ *is translated into* $f(f(f(a, a), h(a)), h(h(a)))$*. Again, the origin of node* u *is the node* $1^{|u|}$*. This translation is illustrated in Figure 2.*

Let M be a top-down tree transducer. Then M is *deterministic* if for any two of its rules $q(\sigma(x_1, \ldots, x_k)) \to \zeta$ and $q(\sigma(x_1, \ldots, x_k)) \to \zeta'$ from M,

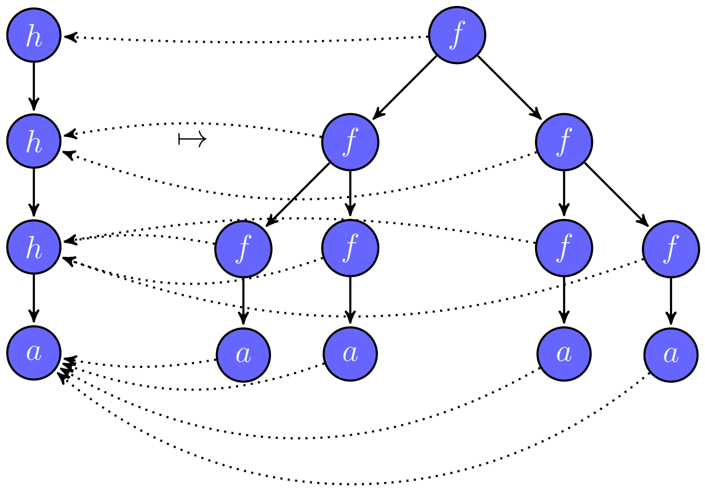

Figure 1: Translation of input tree $h(h(h(a)))$ by the transducer M_1. Dashed arrows depict the origin translation.

it holds that $\zeta = \zeta'$. The class of deterministic top-down tree transducers is denoted by DTOP. Transducers from DTOP realize functional translations and functional origin translations. The transducer M is *non-deleting* if each x_i that occurs in the left-hand side of a rule also appears in the right-hand side of that rule. It is *non-erasing* if no rule has a right-hand side of the form $q'(x_i)$, and thus, each rule contains at least one symbol from Δ. Finally, it is *trimmed* if for any transducer M' obtained from M by removing one of its rules, it holds that $[\![M']\!]_\circ \neq [\![M]\!]_\circ$.

Example 4. *Consider the* TOP *transducer* $M_3 = (Q, \Sigma, \Delta, q_0, R)$ *with* $\Sigma = \Delta = \{f^{(2)}, g^{(3)}, a^{(0)}, b^{(0)}\}$, $Q = \{q_0, q_a, q_b\}$, *and* R *consisting of the following rules.*

$$\begin{aligned}
q_0(f(x_1, x_2)) &\rightarrow f(q_a(x_1), q_0(x_2))\\
q_0(f(x_1, x_2)) &\rightarrow g(q_0(x_2), q_b(x_1), q_0(x_2))\\
q_0(a) &\rightarrow a\\
q_0(b) &\rightarrow b\\
q_a(a) &\rightarrow a\\
q_b(b) &\rightarrow b
\end{aligned}$$

The domain $\mathsf{dom}(M)$ *of* $[\![M]\!]$ *are trees of the form*

$$f(c_1, f(c_2, \ldots f(c_{n-1}, c_n) \ldots))$$

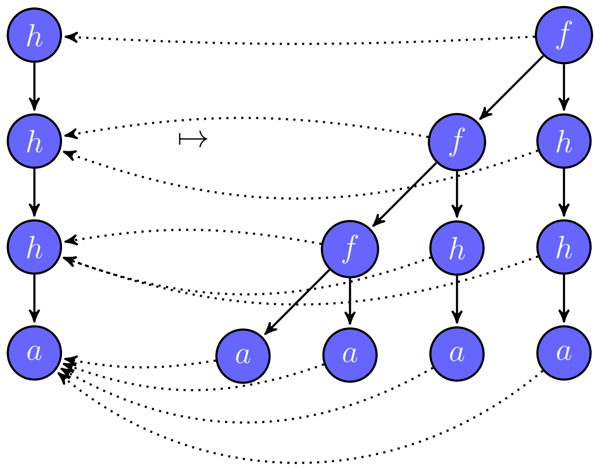

Figure 2: Translation of the input tree $h(h(h(a)))$ by the transducer M_2. Dashed arrows depict the origin translation.

where $c_1 \ldots, c_n \in \{a, b\}$*. Note that* $(a, (t, o))$ *(resp.* $(b, (t, o))$*) does not belong to* $[\![q_b]\!]_\circ$ *(resp.* $[\![q_a]\!]_\circ$*) for any* t *and* o*. Figure 3 shows the unique output tree for the input tree* $s = f(a, f(b, a))$*. Note that* M *is non-deterministic, but at the same time* $[\![M]\!]$ *is a function. Indeed, it can be shown that* $[\![M]\!]$ *cannot be realized by a* DTOP *transducer. This is because at an* f*-input node* u *the transducer has to output either an* f *or a* g*, depending on the first subtree of* u*. Since the transducer has no way of checking the first subtree of an* f*-input node, it has to guess non-deterministically and then verify the guess using the states* q_a *and* q_b*. Note that in the presence of look-ahead, such a situation cannot occur: every functional non-deterministic top-down tree translation can (effectively) be realized by a* DTOP *transducer with look-ahead, as proven by Engelfriet [11].*

3.2. Top-Down Transducers with Look-Ahead

Note that TOP transducers are forced to produce at least one output symbol when they translate the leaf of an input tree. To inspect parts of the input tree without producing output, a TOP transducer can be equipped with *regular look-ahead*, leading to top-down tree transducers with regular look-ahead (TOP$^{\mathrm{R}}$ transducers for short). This is achieved by changing the rules so that each left-hand side is of the form $q(\sigma(x_1, \ldots, x_k) : L)$ where L

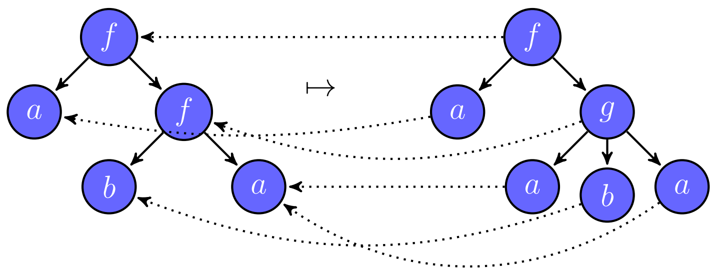

Figure 3: Translation of input tree $f(a, f(b, a))$ by the transducer M_3.

is a regular tree language. Such a rule can be applied only if the input tree $\sigma(s_1, \dots, s_k)$ is in L. A TOP$^{\mathrm{R}}$ transducer M is *deterministic* if for any two rules with left-hand sides $q(\sigma(x_1, \dots, x_k) : L_1)$ and $q(\sigma(x_1, \dots, x_k) : L_2)$, we have $L_1 \cap L_2 = \emptyset$. Note that any TOP transducer M can be transformed into a TOP$^{\mathrm{R}}$ transducer M^R by adding universal look-ahead languages. The classes of deterministic top-down tree transducers with regular look-ahead is denoted by DTOP$^{\mathrm{R}}$. As for DTOP, transducers from DTOP$^{\mathrm{R}}$ realize only functional translations and functional origin translations.

Example 5. *Recall the non-deterministic top-down tree transducer M_3 from Example 4. Its input and output alphabets are $\Sigma = \Delta = \{f^{(2)}, g^{(3)}, a^{(0)}, b^{(0)}\}$. Let L_a be the regular tree language consisting of all trees $f(a, t)$ with $t \in T_\Sigma$, and let $L_b = \{f(b, t) \mid t \in T_\Sigma\}$. Then $[\![M_3]\!]$ is realized by a deterministic top-down tree transducer with look-ahead M_4. The transducer M_4 has the unique state q_0 and the following rules.*

$$\begin{array}{rcl}
q_0(f(x_1, x_2) : L_a) & \to & f(a, q_0(x_2)) \\
q_0(f(x_1, x_2) : L_b) & \to & g(q_0(x_2), b, q_0(x_2)) \\
q_0(a) & \to & a \\
q_0(b) & \to & b
\end{array}$$

A (top-down) tree automaton is a triple (P, Σ, δ) where P is a finite set of states, Σ is a ranked alphabet, and δ is the transition function. The rules in δ are of the form $(p, a) \to (p_1, \dots, p_k)$ where $a \in \Sigma^{(k)}$ and $p, p_1, \dots, p_k \in P$. The tree language $L(p)$ induced by state p is defined recursively as

$$L(p) = \bigcup_{(p,a)\to(p_1,\dots,p_k)} \{a(t_1, \dots, t_k) \mid t_1 \in L(p_1), \dots, t_k \in L(p_k)\}.$$

To define look-ahead of a top-down transducer M with input alphabet Σ, one can assume a tree automaton (P, Σ, δ) and replace in the left-hand side of transitions $q(\sigma(x_1, \ldots, x_k) : L)$ of M the language L by some state p such that $L(p) = L$. Equivalently, one can replace the left-hand side by $q(\sigma(x_1 : p_1, \ldots, x_k : p_k))$ where $p_1, \ldots, p_k \in P$ so that $\sigma(L(p_1), \ldots, L(p_k)) = L$. We will later make use of these different ways of defining top-down tree transducers with look-ahead.

Example 6. *As an example of the notation of look-ahead just defined, we present an equivalent* DTOP^R *for the transducer* M_4 *of Example 5. Let* $\Sigma = \{f^{(2)}, g^{(3)}, a^{(0)}, b^{(0)}\}$*. We define the look-ahead automaton* (P, Σ, δ) *with* $P = \{p_a, p_b, p\}$ *and* δ *consisting of these rules:*

$$\begin{array}{lcl}
(a) & \rightarrow & p_a \\
(b) & \rightarrow & p_b \\
(a) & \rightarrow & p \\
(b) & \rightarrow & p \\
(z, f) & \rightarrow & p \text{ for every } z \in P.
\end{array}$$

The rules of the transducer are now written as follows.

$$\begin{array}{rcl}
q_0(f(x_1 : p_a, x_2 : p)) & \rightarrow & f(a, q_0(x_2)) \\
q_0(f(x_1 : p_b, x_2 : p)) & \rightarrow & g(q_0(x_2), b, q_0(x_2)) \\
q_0(a) & \rightarrow & a \\
q_0(b) & \rightarrow & b
\end{array}$$

3.3. MSO *Transducers*

In this section we define deterministic MSO tree transducers as a restriction of deterministic MSO graph transducers, a logic-based transducer model defined using monadic second-order logic (MSO).

Let Σ, Γ be two disjoint alphabets, and let $S_{\Sigma,\Gamma} = \{(\sigma(x))_{\sigma\in\Sigma}, (\gamma(x, y))_{\gamma\in\Gamma}\}$ be a signature with unary and binary predicate symbols. A *labeled graph* g is a structure over $S_{\Sigma,\Gamma}$, *i.e.* a domain $V(g)$ of vertices, together with an interpretation of the predicates σ and γ (denoted σ^g and γ^g) such that every vertex is labeled by a unique symbol from Σ, and every edge is labeled by a unique symbol from Γ. We denote by $\mathrm{Gr}(\Sigma, \Gamma)$ the set of labeled graphs over Σ and Γ.

If Σ is a ranked alphabet of maximal rank k, any ranked tree t over Σ can be identified with a labeled graph over Σ and $\Gamma = [k]$, such that any vertex labeled $\sigma \in \Sigma^{(r)}$ has exactly r outgoing edges labeled respectively 1 to r. The predicate $i(x, y)$, $1 \leq i \leq r$, relates any node x with its i-th child y.

By $\textsc{mso}[\Sigma, \Gamma]$ we denote all monadic second-order formulas φ over the signature $S_{\Sigma,\Gamma}$, that is

$$\varphi \ ::= \ \mathsf{true} \mid \sigma(x) \mid \gamma(x,y) \mid \varphi \vee \varphi \mid \neg\varphi \mid \exists x\varphi \mid \exists X\varphi$$ [1]

for $\sigma \in \Sigma$ and $\gamma \in \Gamma$. The semantics is as usual, in particular, x, y, z range over nodes and X, Y, Z over sets of nodes. We write $g \models \phi$ whenever a graph g satisfies a formula $\phi \in \textsc{mso}[\Sigma, \Gamma]$. A set of graphs G is $\textsc{mso}[\Sigma, \Gamma]$-definable if there exists an $\textsc{mso}[\Sigma, \Gamma]$-formula ϕ such that $G = \{g \mid g \models \phi\}$. When Σ is a ranked alphabet of maximal rank k, we simply write $\textsc{mso}[\Sigma]$ (or just $\textsc{mso}$ when it is clear from the context) to denote $\textsc{mso}[\Sigma, [k]]$.

To define the output graph g' of an input graph g, an MSO transducer uses $\textsc{mso}[\Sigma, \Delta]$ formulas with one or two free variables, interpreted over a fixed number of copies of g, to define the predicates of g'. Formally,

Definition 7. *Let (Σ_1, Γ_1) and (Σ_2, Γ_2) be two pairs of disjoint alphabets. A* deterministic $\textsc{mso}$ graph transducer *from $Gr(\Sigma_1, \Gamma_1)$ to $Gr(\Sigma_2, \Gamma_2)$ is a tuple*

$$M = (C, \phi_{dom}, (\phi^c_\sigma(x))_{\sigma\in\Sigma_2, c\in C}, (\phi^{c,d}_\gamma(x,y))_{\gamma\in\Gamma_2, c,d\in C})$$

such that C is a finite set of copy indices, ϕ_{dom} is an $\textsc{mso}[\Sigma_1, \Gamma_1]$-sentence, ϕ^c_σ are $\textsc{mso}[\Sigma_1, \Gamma_1]$-formulas with one free variable x, and $\phi^{c,d}_\gamma$ are $\textsc{mso}[\Sigma_1, \Gamma_1]$-formulas with two free variables x and y.

For a graph $g \in \mathrm{Gr}(\Sigma_1, \Gamma_1)$, the output graph g' by M is defined if $g \models \phi_{dom}$, and has domain $V(g') \subseteq V(g) \times C$ of pairs (u, c) (denoted u^c) such that there exists a unique $\sigma \in \Sigma_2$ such that $g \models \phi^c_\sigma(u)$ (the node (u, c) is then labeled σ in g'), and for all $u^c, v^d \in V(g')$, the edge (u^c, v^d) exists if there is a unique $\gamma \in \Gamma_2$ such that $g \models \phi^{c,d}_\gamma(u, v)$ (this edge is then labeled γ in g').

The origin mapping o of the translation of g into g' is the mapping defined by $o(u^c) = u$, for all $u^c \in V(g')$. Note that, for any input node u, there are at most $|C|$ nodes in the output with u as origin. We denote by $[\![M]\!]$ the

[1] As usual, we use false, $\varphi \wedge \varphi'$, $\varphi' \rightarrow \varphi$, $\forall x\varphi$ and $\forall X\varphi$ as shortcuts for $\neg\mathsf{true}$, $\neg(\varphi \vee \varphi')$, $\neg\varphi \vee \varphi'$, $\forall x\varphi$, $\forall X\varphi$ respectively.

translation defined by M, and by $[\![M]\!]_{\mathsf{o}}$ the translation with origin defined by M. We simply write $\mathsf{dom}(M)$ for $\mathsf{dom}([\![M]\!])$.

Let Σ, Δ be two ranked alphabets. We have seen that any ranked tree can be identified with a labeled graph. A *deterministic* MSO *tree transducer* M (DMSOT transducer for short) from T_Σ to T_Δ is a deterministic graph transducer such that $[\![M]\!] \subseteq T_\Sigma \times T_\Delta$. Note that it is easily enforcible that every input graph of an MSO transducer is a tree, by adding the MSO-definable property of being a tree to the domain formula. For an MSO transducer M such that $\mathsf{dom}(M) \subseteq T_\Sigma$, it is decidable whether every output graph of M is a tree: inverse MSO translations preserve MSO definability, thus, we can construct an MSO sentence of input trees so that the corresponding output graphs by M are not trees, and then test emptiness of this regular tree language.

Example 8. *Consider the translation τ (definable by any transducer from* DTOP^R*) over the ranked (input and output) alphabet $\Sigma = \{f^{(2)}, a^{(1)}, b^{(1)}, e^{(0)}\}$, which maps any tree $f(s_1, s_2)$, where s_1 is a monadic tree over $\Sigma \setminus \{f\}$, to $f(s_1, rev(s_1))$, for $rev(s_1)$ the reverse of s_1 (s_1 put upside down). E.g., $f(a(a(b(e))), e)$ is mapped to $f(a(a(b(e))), b(a(a(e))))$, as illustrated on Figure 4. The translation τ is definable by the following* DMSOT *transducer the formulas of which are precisely given below:*

$$M_\tau = ([2], \phi_{dom}, (\phi^c_\sigma(x))_{c\in[2],\sigma\in\Sigma}, (\phi^{c,d}_i(x,y))_{i,c,d\in[2]})$$

Since we copy s_1 twice, one copy of s_1 being reversed, one needs a copy set $[2]$. The formula ϕ_{dom} is an MSO$[\Sigma]$*-sentence expressing that there is only one node labeled f and it is the root:*

$$\begin{array}{lcl} \phi_{dom} & = & \exists x \ (root(x) \wedge f(x) \wedge \forall y \neq x \ (\neg f(y)) \\ root(x) & = & \forall y \ \bigwedge_{i\in[2]} \neg i(y,x) \end{array}$$

In the first copy, we keep only input nodes in 1^* (the root node and s_1), and in the second one, only input nodes in 1^+ (the subtree s_1). Both properties – being in 1^* and in 1^+ for a node x – are definable in MSO$[\Sigma]$ *by two formulas $\phi_{1^*}(x)$ and $\phi_{1^+}(x)$ respectively, which express the fact that every set containing the root (respectively the second-child of the root) which is closed by the first-child relation also contains x:*

$$\begin{array}{lcl} \phi_{1^*}(x) & = & \forall X \ (\exists y \ (root(y) \wedge y \in X) \wedge \forall y{\in}X \forall z \ (1(y,z) \rightarrow z{\in}X)) \rightarrow x{\in}X \\ \phi_{1^+}(x) & = & \phi_{1^*}(x) \wedge \neg root(x) \end{array}$$

where $root_1(y) = \exists x\ (root(x) \wedge 2(x,y))$ denotes the first-child y of the root.

In the two copies, nothing is changed with respect to labels:

$$\forall \sigma \in \Sigma : \quad \phi^1_\sigma(x) = \phi_{1^*}(x) \wedge \sigma(x) \qquad \phi^2_\sigma(x) = \phi_{1^+}(x) \wedge \sigma(x)$$

For the successor predicates, in the first copy, nothing is changed:

$$\phi^{1,1}_1(x,y) = 1(x,y) \qquad \phi^{1,1}_2(x,y) = 2(x,y)$$

Note that by definition of the nodes that are kept in the first copy, there is no pairs of nodes x, y such that $2(x,y)$, and therefore we could have equivalently set $\phi^{1,1}_2(x,y)$ to false.

In the second copy, that correspond to $rev(s_1)$, first-child predicates are reversed for all inner nodes, and the root of s_1 in the second copy is connected to the leaf of s_1 (this is required because $rev(s_1)$ reverses all symbols of s_1 except the leaf, which bear the only constant symbols and hence cannot be used as the label of a root):

$$\phi^{2,2}_1(x,y) = (\neg leaf(x) \wedge 1(y,x)) \vee leaf(y) \wedge root_1(x)$$

where $leaf(y) = \forall z \bigwedge_{i\in[2]} \neg i(y,z)$ holds true if y is a leaf.

Then, it remains to connect the first copy to the second copy, i.e. the root node labelled f to the root of $rev(s_1)$, using

$$\phi^{1,2}_2(x,y) = root(x) \wedge \exists z\, (1(y,z) \wedge leaf(z))$$

Other formulas are put to false:

$$\phi^{1,1}_2(x,y) = \phi^{2,1}_2(x,y) = \phi^{1,2}_1(x,y) = \phi^{2,1}_1(x,y) = \mathsf{false}$$

Let us describe the origin mappings defined by M_τ. It is illustrated on Figure 4. Let $(t,o) = [\![M_\tau]\!]_{\mathsf{o}}(s)$ for some input tree $s \in T_\Sigma$. Necessarily, s is of the form $f(s_1,s_2)$ and $t = f(s_1, rev(s_1))$. Let us describe the origin mapping o:

$$\begin{array}{lllll} o & : & V(t) & \to & V(s) \\ & & u & \mapsto & \begin{cases} 1^j & \text{if } u = 1^j \text{ for some } j{\geq}0 \\ 1^{j+1} & \text{if } u = 2.1^j \text{ for some } j{\geq}0 \text{ and } u \text{ is a leaf} \\ 1^{height(s_1)-j} & \text{if } u = 2.1^j \text{ for some } j{\geq}0 \text{ and } u \text{ is not a leaf} \end{cases} \end{array}$$

Finally in this example, if we restrict Σ to $\Sigma' = \{f^{(2)}, a^{(1)}, e^{(0)}\}$, then the restriction $\tau' = \tau|_{T_{\Sigma'}}$ becomes DTOP^R*-definable (reversing a monadic tree*

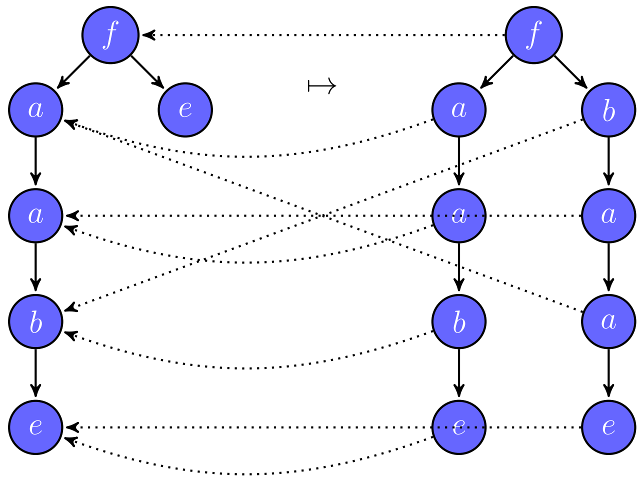

Figure 4: Translation of input tree $f(a(a(b(e))), e)$ by the transducer M_τ.

over a one-letter alphabet is like doing the identity). Let M' be the previous DMSOT *transducer M whose domain is restricted to $T_{\Sigma'}$, i.e. $[\![M']\!] = \tau'$. Even though there is only one unary symbol $a^{(1)}$, the transducer M' still reverses the origins, something that is not doable by a* DTOP^R *transducer. Therefore we have that $[\![M']\!]$ can be realized by a* DTOP^R *transducer, but we have that $[\![M']\!]_\circ$ cannot be realized by any* DTOP^R *transducer.*

MSO transducers can be extended with non-determinism [4]. The ability of defining non-deterministic translations originates only from (free) second-order variables and their possible valuations. Hence, all formulas ϕ_{dom}, ϕ_σ^c and $\phi_\gamma^{c,c'}$ can use a fixed additional finite set of free second-order variables $\overline{X}$. Once an assignment ν of each variable of $\overline{X}$ by a set of nodes of an input graph g is fixed, the previous formulas can be interpreted as before with respect to this assignment, thus defining an output pair (g_ν, o_ν) (if the domain formula holds true). The set of outputs associated with g is the set of all such pairs (g_ν, o_ν), for all assignments ν of $\overline{X}$. We denote by MSOT the class of (non-deterministic) MSO transducers from trees to trees.

Example 9. *One can modify Example 8 to define the translation which maps any tree of the form $f(s_1, s_2)$ to $f(s_1', rev(s_1'))$ where s_1' is a subword of s_1 (i.e. s_1' has been obtained by erasing some symbols of s_1, except the last one). It suffices to use a free variable X intended to define all the erased* *positions of s_1. Only nodes in X are kept, by letting $\phi^1_\sigma(x) = \neg leaf(x) \to (x \in X \wedge \phi_{1^*}(x) \wedge \sigma(x))$, and $\phi^2_\sigma(x) = \neg leaf(x) \to (x \in X \wedge \phi_{1^+}(x) \wedge \sigma(x))$.*

3.4. Origin-Equivalence Problem

In this section, we prove that origin-equivalence is decidable for MSO tree transducers and for DTOP$^{\mathrm{R}}$ transducers. Two transducers M_1, M_2 are origin-equivalent if $[\![M_1]\!]_o = [\![M_2]\!]_o$. The origin-equivalence problem for a class $\mathcal{C}$ of transducers asks whether any two given transducers $M_1, M_2 \in \mathcal{C}$ are origin-equivalent or not.

Theorem 10. *Origin-equivalence is decidable for the classes* TOP^R *and* MSOT.

The proof of Theorem 10 (for MSOT) relies on the following set:

$$E(\tau_1, \tau_2) = \{t \in \mathsf{dom}(\tau_1) \cap \mathsf{dom}(\tau_2) \mid \tau_1(t) = \tau_2(t)\}$$

where, for $i \in \{1, 2\}$, $\tau_i(t) = \{t' \mid (t, t') \in \tau_i\}$.

We will show that this set is regular if τ_1, τ_2 are origin translations defined by (1) MSOT transducers, and (2) TOP$^{\mathrm{R}}$ transducers. Note that without origins, even simple TOP$^{\mathrm{R}}$ and MSOT translations yield a non-regular set: *e.g.*, the translations $R_1 : f(s_1, s_2) \mapsto s_1$ and $R_2 : f(s_1, s_2) \mapsto s_2$ are both MSOT definable but their (origin-free) equality set $\{f(s, s) \mid s \in T_\Sigma\}$, is not regular. We first show (1), which holds more generally for graph-to-tree MSO transducers.

Proposition 11. *Let Σ, Γ be two disjoint alphabets, and Δ be a ranked alphabet. Let T_1, T_2 be non-deterministic* MSO *graph-to-tree transducers from $Gr(\Sigma, \Gamma)$ to T_Δ. Then $E([\![T_1]\!]_o, [\![T_2]\!]_o)$ is (effectively)* MSO$[\Sigma_1, \Gamma_1]$*-definable.*

Proof. We first assume that T_1 and T_2 are deterministic, the non-deterministic case is treated afterwards. We let ℓ be the maximal rank of Δ-symbols. Let

$$\begin{array}{rcl} T_1 & = & (C_1, \phi_{dom}, (\phi^c_\delta(x))_{\delta \in \Delta, c \in C_1}, (\phi^{c,c'}_i(x, y))_{i \in [\ell], c, c' \in C_1}) \\ T_2 & = & (C_2, \psi_{dom}, (\psi^c_\delta(x))_{\delta \in \Delta, c \in C_2}, (\psi^{c,c'}_i(x, y))_{i \in [\ell], c, c' \in C_2}) \end{array}$$

We assume wlog that $C = C_1 = C_2$: if T_1 has less copies than T_2 then we simply add those copies and define the corresponding formulas as false.

Intuitively, we consider, for some input graph g, $[\![T_1]\!]_o(g)$ and $[\![T_2]\!]_o(g)$ in a single structure obtained from the by overlap of the two output trees. We are going to built an MSO formula which is valid for g iff $[\![T_1]\!]_o(g)$ and

$[\![T_1]\!]_{\mathsf{o}}(g)$ are actually equal. The fact that outputs are trees together with origins is crucial for this equality test to be possible.

First, we shall make clear how the nodes of the output trees are represented. Let g be a graph and $[\![T_i]\!]_{\mathsf{o}}(g) = (t_i, o_i)$ its output, $i = 1, 2$. Then, $t \in T_\Delta$ is a tree whose set of nodes $V(t_i)$ is a subset of $\mathbb{N}^*$. By definition of MSO transducers, any node of t_i can be identified with a pair (v, c) such that $v \in V(g)$ and $c \in C$. We denote by addr_i the partial function that associates with any pair (v, c) the node of $V(t_i)$ (if it exists) as a result of this identification.

We will also need the notion of *path labels* and that of *path origins*. Let $u \in V(t_i)$ be a node of t_i. By $\text{p-labels}_{t_i}(u)$ (resp. $\text{p-origins}_{t_i}(u)$) we denote the sequence $t_i[u_0]t_i[u_1]\ldots t_i[u_n] \in \Delta^+$ (resp. $o_i(u_0)o_i(u_1)\ldots o_i(u_n) \in V(g)^+$) where $u_0u_1\ldots u_n \in V(t_i)^+$ is sequence of nodes from the root ($u_0 = \epsilon$) to u ($u_n = u$).

We start by a key observation saying that if $[\![T_1]\!]_{\mathsf{o}}(g) = [\![T_2]\!]_{\mathsf{o}}(g) = (t, o)$, for an input graph $g \in \mathrm{Gr}(\Sigma, \Gamma)$, then whenever there is a "synchronized" root-to-node path π in t then for any one-step continuation of this path by T_1 there exists a one-step continuation by T_2, and vice versa.

Claim: Let $g{\in}\mathsf{dom}(T_1){\cap}\mathsf{dom}(T_2)$, and $[\![T_1]\!]_{\mathsf{o}}(g){=}(t_1, o_1)$, $[\![T_2]\!]_{\mathsf{o}}(g){=}(t_2, o_2)$. Then $(t_1, o_1){=}(t_2, o_2)$ iff

1. $t_1[\varepsilon] = t_2[\varepsilon]$ and $o_1(\varepsilon) = o_2(\varepsilon)$, and
2. for all $u \in V(t_1) \cap V(t_2)$ such that $\text{p-labels}_{t_1}(u) = \text{p-labels}_{t_2}(u)$ and $\text{p-origins}_{t_1}(u) = \text{p-origins}_{t_2}(u)$, for all $\alpha, i \in [2]$, if $ui \in V(t_\alpha)$, then $ui \in V(t_{3-\alpha})$, $t_1[ui] = t_2[ui]$, and $o_1(ui) = o_2(ui)$.

The proof of this claim is straightforward. The formula we are going to build is based on this claim and expresses for instance that any sequence of input nodes $u_1, \ldots, u_n$ that are connected with successor formulas $\phi_i^{c,c'}(x, y)$ by T_1 are also connected with successor formulas $\psi_i^{d,d'}$ of T_2 with the same sequence of indices i, and conversely. It also expresses that the sequence of label formulas $\phi_\delta^c(x)$ of T_1 and $\psi_\delta^{c'}(x)$ of T_2 that hold on $u_1, \ldots, u_n$ respectively, carry the same respective output symbols δ.
Condition 1 of the claim can be expressed by the sentence χ_{root} that expresses that there are a node x^{c_1} in t_1 and a node x^{c_2} in t_2 which are the root of t_1

and t_2 respectively, have the same origin x, and the same label:

$$\begin{array}{lcl} \chi_{root} & \equiv & \exists x \bigvee_{c_1,c_2 \in C} \mathrm{root}_{c_1,c_2}(x) \\ \mathrm{root}_{c_1,c_2}(x) & \equiv & \bigwedge_{c \in C, i \in [\ell]} \forall z (\neg \phi_i^{c,c_1}(z,x) \wedge \neg \psi_i^{c,c_2}(z,x)) \wedge \bigvee_{a \in \Delta} (\phi_a^{c_1}(x) \wedge \psi_a^{c_2}(x)) \end{array}$$

In order to express the Condition (2) of the claim, we first define, for all $c_1, c_2 \in C$, a formula $\mathrm{Path}_{c_1,c_2}(x)$, such that $g \models \mathrm{Path}_{c_1,c_2}(v)$ iff $\mathrm{addr}_1(v^{c_1})$ and $\mathrm{addr}_2(v^{c_2})$ are both defined and equal, and:

$$\begin{array}{lcl} \text{p-labels}_{t_1}(\mathrm{addr}(v^{c_1})) & = & \text{p-labels}_{t_2}(\mathrm{addr}(v^{c_2})) \\ \text{p-origins}_{t_1}(\mathrm{addr}(v^{c_1})) & = & \text{p-origins}_{t_2}(\mathrm{addr}(v^{c_2})) \end{array}$$

It is obtained as the transitive closure of the formulas $R_{(c_1,c_1'),(c_2,c_2')}(x,y)$ defined for all $c_1, c_2, c_1', c_2' \in C$, and which holds true whenever there exists $i \in [\ell]$ such that $y^{c_1'}$ is the i-th child of $\mathrm{addr}_1(x^{c_1})$ in t_1, $y^{c_2'}$ is the i-th child of $\mathrm{addr}_2(x^{c_2})$ in t_2, the nodes $\mathrm{addr}_1(x^{c_1})$ and $\mathrm{addr}_2(x^{c_2})$ have the same label, as well as the nodes $\mathrm{addr}_1(y^{c_1'})$ and $\mathrm{addr}_2(y^{c_2'})$. Formally:

$$R_{(c_1,c_1'),(c_2,c_2')}(x,y) \equiv \bigvee_{i \in [\ell], a,b \in \Delta} \phi_i^{c_1,c_1'}(x,y) \wedge \psi_i^{c_2,c_2'}(x,y) \wedge \phi_a^{c_1}(x) \wedge \psi_a^{c_2}(x) \wedge \phi_b^{c_1'}(y) \wedge \psi_b^{c_2'}(y)$$

Again, due to origin semantics, the sources (resp. the targets) of the considered edges from the output must orginate from the same input node.

Based on the latter formula, we define $\mathrm{Path}_{c_1,c_2}(x)$ as follows:

$$\mathrm{Path}_{c_1,c_2}(x) \equiv \exists r \bigvee_{e_1,e_2 \in C} \mathrm{root}_{e_1,e_2}(r) \wedge \forall X_{1,1} \forall X_{1,2} \dots \forall X_{|C|,|C|} (r \in X_{e_1,e_2} \wedge$$
$$\bigwedge_{d_1,d_1',d_2,d_2' \in C} \forall z \forall z' (z \in X_{d_1,d_2} \wedge R_{(d_1,d_1'),(d_2,d_2')}(z,z') \to z' \in X_{d_1',d_2'})) \to x \in X_{c_1,c_2}$$

We can finally express conditions 1 and 2 of the claim by a sentence $\Phi_{E(T_1,T_2)}$ defined as:

$$\begin{array}{l} \phi_{dom} \wedge \psi_{dom} \wedge \chi_{root} \wedge \\ \quad (\forall x \bigwedge_{c_1,c_2 \in C} \mathrm{Path}_{c_1,c_2}(x) \to \\ \quad\quad (\forall x' \bigwedge_{c_1' \in C, i \in [\ell], a \in \Delta} \phi_i^{c_1,c_1'}(x,x') \wedge \phi_a^{c_1'}(x') \to \bigvee_{c_2' \in C} \psi_i^{c_2,c_2'}(x,x') \wedge \psi_a^{c_2'}(x')) \\ \quad \wedge (\forall x' \bigwedge_{c_2' \in C, i \in [\ell], a \in \Delta} \psi_i^{c_2,c_2'}(x,x') \wedge \psi_a^{c_2'}(x') \to \bigvee_{c_1' \in C} \phi_i^{c_1,c_1'}(x,x') \wedge \phi_a^{c_1'}(x'))) \end{array}$$

It should be clear that $\Phi_{E(T_1,T_2)}$ defines the set $E(\llbracket T_1 \rrbracket_\circ, \llbracket T_2 \rrbracket_\circ)$.

Extension of the proof to non-deterministic transducers To extend this proof to non-deterministic MSO transducers, we now assume that all formulas of T_1 and T_2 are parameterized respectively by a tuple of second-order variables $\overline{X}$ and $\overline{Y}$. We can construct a formula $\Phi_{E(T_1,T_2)}(\overline{X}, \overline{Y})$ exactly as before, except that all formulas of T_1 and T_2 used in the definition of $\Phi_{E(T_1,T_2)}$ are, as well, parameterized respectively by $\overline{X}$ and $\overline{Y}$. The set $E(\llbracket T_1 \rrbracket_\circ, \llbracket T_2 \rrbracket_\circ)$ is then definable by the sentence

$$\Phi^{ND}_{E(T_1,T_2)} \equiv \forall \overline{X} \exists \overline{Y} \cdot \Phi_{E(T_1,T_2)}(\overline{X}, \overline{Y}) \wedge \forall \overline{Y} \exists \overline{X} \cdot \Phi_{E(T_1,T_2)}(\overline{X}, \overline{Y}).$$

Indeed, $g \models \Phi^{ND}_{E(T_1,T_2)}$ iff for all interpretations of $\overline{U} \in (2^{V(g)})^{|\overline{X}|}$, there exists an interpretation $\overline{V} \in (2^{V(g)})^{|\overline{Y}|}$ such that $g \models \Phi^{E(T_1,T_2)}(\overline{U}, \overline{V})$ and symmetrically, iff for all $\overline{U} \in (2^{V(g)})^{|\overline{X}|}$, there exists $\overline{V} \in (2^{V(g)})^{|\overline{Y}|}$ such that $\llbracket T_1[\overline{U}] \rrbracket_\circ(g) = \llbracket T_2[\overline{V}] \rrbracket_\circ(g)$ and symmetrically, iff $\llbracket T_1 \rrbracket_\circ(g) \subseteq \llbracket T_2 \rrbracket_\circ(g)$ and $\llbracket T_2 \rrbracket_\circ(g) \subseteq \llbracket T_1 \rrbracket_\circ(g)$, iff $\llbracket T_1 \rrbracket_\circ(g) = \llbracket T_2 \rrbracket_\circ(g)$. □

A similar result as Proposition 11 holds for DTOP$^{\mathrm{R}}$ transducers, as shown below. While it immediately yields a procedure for deciding origin-equivalence of DTOP$^{\mathrm{R}}$ transducers, the result of Theorem 10 also holds for (non-deterministic) TOP$^{\mathrm{R}}$ transducers with a different proof. However, we give the following result, as it is interesting in its own. We leave open however whether it still holds for TOP$^{\mathrm{R}}$ transducers.

Proposition 12. *Let M_1, M_2 be* DTOP^R *transducers. Then $E(\llbracket M_1 \rrbracket_\circ, \llbracket M_2 \rrbracket_\circ)$ is a regular tree language.*

Proof. For $i \in [2]$, let $M_i = (Q_i, \Sigma_i, \Delta_i, q_{0i}, R_i)$. We make the following assumptions:

- we assume M_1, M_2 to be trimmed is the sense that each of their transition rule appears in some computation.
- If $q(f(x_1, \ldots, x_k) : L) \to C[q_1(x_{i_1}), \ldots, q_k(x_{i_n})]$ is a rule of M_i (where C is a Δ-context), then $s/i_j \in \mathsf{dom}(\llbracket q_j \rrbracket)$ for every $s \in L$ and every $j \in [n]$.

The second condition can be achieved by changing L in each rule that does not have the property into the set of all trees $\sigma(s_1, \ldots, s_k)$ such that s_i is in the intersection of all $\mathsf{dom}(\llbracket q_j \rrbracket)$ for $j \in [n]$ with $i_j = i$, intersected with L. This can be achieved rule by rule as it does not change the domains $\mathsf{dom}(\llbracket q \rrbracket)$.

The semantics with origin imposes strong constraints on the shapes of the right-hand sides of rules of M_1 and M_2 that can be applied in parallel: roughly speaking when M_1 and M_2 process the same input node, they must produce an output context of the same shape. Indeed, if they produce a node with different labels then the equivalence fails obviously. But even more, if one, say M_1, produces a node that is not yet present in the output produced by M_2 then the equivalence fails also due to the semantics with origin. This node will have for M_1 the current input u node as origin whereas for M_2 this node will originate from a descendant of u.

We define the relation $\overset{\circ}{\sim}$ over $Q_1 \times Q_2$ as $p \overset{\circ}{\sim} q$ if for all f in Σ, either none of p, q is a left-hand side of an f-rule or for any of such two rules whose left-hand side is p and q respectively, their respective form is

$$\begin{array}{rcl} p(f(x_1, \ldots, x_k) : L_1) & \to & C_1[p_1(x_{i_1}), \ldots, p_m(x_{i_m})] \\ q(f(x_1, \ldots, x_k) : L_2) & \to & C_2[q_1(x_{j_1}), \ldots, q_n(x_{j_n})], \end{array}$$

with

- $C_1 = C_2$, $m = n$,
- $i_\nu = j_\nu$ for all $\nu \in [m]$,
- $p_\nu \overset{\circ}{\sim} q_\nu$ for all $\nu \in [m]$.

One can prove that for all $p \in Q_1$, $q \in Q_2$, for all $s \in \mathsf{dom}(\llbracket p \rrbracket_\circ) \cap \mathsf{dom}(\llbracket q \rrbracket_\circ)$, $\llbracket p \rrbracket_\circ(s) = \llbracket q \rrbracket_\circ(s)$ iff $p \overset{\circ}{\sim} q$.

To show the only-if direction, assume that $\llbracket p \rrbracket_\circ(s) = \llbracket q \rrbracket_\circ(s) = (t, o)$ and let $r_1 \in R_1$ and $r_2 \in R_2$ be the rules for p and q that are applicable to s, *i.e.* $s \in L_1 \cap L_2$. Then $V_\Delta(C_1) = V_\Delta(C_2)$, because they both equal $\{v \in V(t) \mid o(v) = \epsilon\}$ by the semantics of M_1 and M_2. This implies that $C_1 = C_2$ and hence $m = n$. Let $t_1, \ldots, t_m$ such that $t = C_1[t_1, \ldots, t_m]$. Let $\nu \in [m]$, and $v \in V(t)$ the node that corresponds to the root of the ν-th subtree t_ν of t. By definition of the origin semantics, $o(v) = i$ for some $i \in [k]$, and hence $i_\nu = j_\nu = i$. The third item above holds by induction hypothesis. For the if direction, let us assume that $p \overset{\circ}{\sim} q$ and consider some $s \in \mathsf{dom}(\llbracket p \rrbracket_\circ) \cap \mathsf{dom}(\llbracket q \rrbracket_\circ)$. Depending on the symbol at the root of s, there must be two unique rules r_1 and r_2 for p and q respectively. To preserve to origin semantics, the contexts C_1, C_2 must be the same and hence $m = n$.

Now, by definition of $\overset{\circ}{\sim}$, we have $p_\nu \overset{\circ}{\sim} q_\nu$ for all $\nu \in [m]$. We can then conclude by induction.

Let us denote by P the set of pairs (r_1, r_2) of rules whose left-hand sides p, q satisfy $p \overset{\circ}{\sim} q$. We construct a TOP^{R} transducer M such that $\mathsf{dom}(M) = E(\llbracket M_1 \rrbracket_{\mathsf{o}}, \llbracket M_2 \rrbracket_{\mathsf{o}})$. The set of states of M is $Q_1 \times Q_2$, with initial states $\langle q_{01}, q_{02} \rangle$, and for all pairs of rules $(r_1, r_2) \in P$, we add to the rules of M the following rule:

$$\langle p, q \rangle (a(x_1, \ldots, x_k) : L_1 \cap L_2) \to C_1[\langle p_1, q_1 \rangle (x_{i_1}), \ldots, \langle p_m, q_m \rangle (x_{i_m})]$$

It is well-known that TOP^{R} transducers have regular domains, hence the result follows. □

Proof of Theorem 10. Let τ_1, τ_2 be two origin translations defined respectively by two transducers from MSOT. They are equivalent if and only if they have the same domain D and $D = E(\tau_1, \tau_2)$. Since MSOT translations have effective regular domains, and $E(\tau_1, \tau_2)$ is effectively regular by Proposition 11, we get the result. The proof goes the same way for τ_1, τ_2 defined by DTOP^{R} transducers using Proposition 12.

However, when τ_1, τ_2 are origin translations defined by TOP^{R} transducers an alternative and somehow more direct proof exists. As a first step, an equality test of the domains is performed. This is obviously decidable due to the effective regularity of domains of TOP^{R} transducers. We then proceed by reduction to the equivalence problem of letter-to-letter transducers as follows. For non-deleting non-erasing top-down tree transducers **without** regular look-ahead, the decidability of origin-equivalence follows directly from the decidability of equivalence for non-deterministic non-deleting top-down letter-to-letter tree transducers [1].

Indeed, the origin semantics forces for all input trees $s \in \mathsf{dom}(M_1)$, for all $(t, o) \in \llbracket M_1 \rrbracket_{\mathsf{o}}(s)$, M_2 must produce, in a successful execution, the symbols of t exactly at the same moment as M_1, and conversely for all $(t, o) \in \llbracket M_2 \rrbracket_{\mathsf{o}}(s)$. When considering non-erasing TOP, this property has a nice consequence: both M_1 and M_2 can be seen as symbol-to-symbol transducers: each – necessarily non-empty – Δ-context in the right-hand side of a rule can be considered as one letter having for rank the number of occurrences of $q_j(x_{i_j})$ in this right-hand side. For instance, if $\zeta = f(q_1(x_1), h(q_2(x_1)), q_3(x_2))$, then $f(., h(.), .)$ is seen as a single symbol of rank 3.

We now show how to reduce the equivalence problem of origin translations realized by DTOP$^{\mathrm{R}}$ transducers to the one of non-deleting non-erasing top-down tree transducers **without** regular look-ahead.

Consider a TOP$^{\mathrm{R}}$ tranducer M. Here we assume that the look-ahead is given as states of a single non-deterministic top-down tree automaton. For each rule of the following form

$$q(a(x_1, \dots, x_k) : p) \to C[q_1(x_{i_1}), \dots, q_k(x_{i_n})]$$

and for each rule of the look-ahead automaton $(p, a) \to (p_1, \dots, p_k)$ we construct the following rule:

$$q(a(x_1, \dots, x_k)) \to C'[p_1(x_1), \dots, p_k(x_k), q_1(x_{i_1}), \dots, q_k(x_{i_n})]$$

where $C' = \$(a(\perp, \dots, \perp), C)$ and \$ is a new binary symbol.

In addition, we have to define rules for look-ahead states. For each rule $(p, a) \to (p_1, \dots, p_k)$ of the look-ahead automaton (with $a \in \Sigma^{(k)}$), we construct the following rule:

$$p(a(x_1, \dots, x_k)) \ \to \ a(p_1(x_1), \dots, p_k(x_k))$$

Intuitively, for each transition applied in the original transducer, a new thread is started which checks the look-ahead, and produces as output a copy of the input subtree. Observe that this does not depend on M. We systematically add such an output. Observe that the new transducer is non-deleting and non-erasing.

Let us point out that regarding origins, this reduction yields a conservative extension: nodes in the output shared by the old and new transducers have the same origin. Now let us assume that we apply this reduction to two transducers T_1 and T_2, yielding T_1' and T_2'. It is important to notice that when processing the same input, the new nodes in the output of T_1' (those obtained from look-ahead elimination) and the new ones in the output of T_2' will be by construction the same and moreover, will share the same origin.

Given two TOP$^{\mathrm{R}}$ transducers M_1, M_2, we denote by M_1', M_2' the non-deleting and non-erasing TOP transducers (without look-ahead) obtained by this construction. It is easy to show that $[\![M_1]\!]_\circ = [\![M_2]\!]_\circ$ iff $[\![M_1']\!]_\circ = [\![M_2']\!]_\circ$. $\square$

Note that based on Proposition 11, the decidability result of Theorem 10 can be extended to any (non-deterministic) MSO graph-to-tree transducers over a class of graphs with decidable MSO theory. More precisely, given a class of graphs $\mathcal{C}$ with decidable MSO theory, the following problem is decidable: given two MSO graph to tree transducers M_1, M_2, decide whether $\mathsf{dom}(M_1) \cap \mathcal{C} = \mathsf{dom}(M_2) \cap \mathcal{C}$ and for all graphs $g \in \mathcal{C} \cap \mathsf{dom}(M_1)$, $[\![M_1]\!]_{\mathsf{o}}(t) = [\![M_2]\!]_{\mathsf{o}}(t)$. For instance, by taking $\mathcal{C}$ the class of graphs of treewidth k, the latter problem is decidable.

3.5. Origin-Injectivity Problem

A function $\tau : A \to B$ is injective if and only if for all $x, y \in A$, $f(x) = f(y)$ implies that $x = y$. Thus, an injective function maps distinct arguments to distinct images. It is well-known that injectivity is undecidable for DTOP transducers, in fact, it is undecidable even already for tree homomorphism (= *DT* transducers with only one state) [21]. However, tree homomorphism are not definable by DMSOT transducers in general therefore, we give a simple proof of undecidability for DTOP$^{\mathrm{R}}$ and DMSOT transducers, based on a reduction from the Post correspondence problem (PCP). In this reduction, the fact that the transducers can copy subtrees is crucial.

Proposition 13. *Injectivity is undecidable for tree translations defined by* DTOP^{R} *and* DMSOT *transducers.*

Proof. The proof is by reduction from PCP. Let $\Delta = \{a, b\}$ be an alphabet. An instance of PCP is a finite sequence of pairs $\mathcal{I} = (u_1, v_1) \dots (u_n, v_n)$ such that $u_i, v_i \in \Delta^*$. It has a solution if there exists a sequence of indices $i_1, \dots, i_k \in \{1, \dots, n\}$ such that $u_{i_1} \dots u_{i_k} = v_{i_1} \dots v_{i_k}$. For all instances $\mathcal{I}$ of PCP we define a tree translation $\tau_{\mathcal{I}}$ from T_Σ to T_Σ, where $\Sigma = \{U^{(1)}, V^{(1)}, \#^{(2)}, a^{(1)}, b^{(1)}, 1^{(1)}, \dots, n^{(1)}, e^{(0)}\}$. For a word $u = \sigma_1 \dots \sigma_k \in (\Delta \cup \{1, \dots, n\})^*$ and a tree $t \in T_\Sigma$, we write $u(t)$ for the tree $\sigma_1(\sigma_2(\dots \sigma_k(t) \dots))$.

The translation $\tau_{\mathcal{I}}$ is defined on trees of the form $Z(i_1 i_2 \dots i_k(e))$ where $Z \in \{U, V\}$, as follows:

$$\begin{array}{cccc} \tau_{\mathcal{I}} : & T_\Sigma & \to & T_\Sigma \\ & Z(i_1 i_2 \dots i_k(e)) & \mapsto & \left\{ \begin{array}{ll} \#(i_1 i_2 \dots i_k(e), u_{i_1} u_{i_2} \dots u_{i_k}(e)) & \text{if } Z = U \\ \#(i_1 i_2 \dots i_k(e), v_{i_1} v_{i_2} \dots v_{i_k}(e)) & \text{if } Z = V \end{array} \right. \end{array}$$

We claim that $\tau_{\mathcal{I}}$ is not injective iff $\mathcal{I}$ has a solution. Indeed, if $\mathcal{I}$ has a solution $i_1 \dots i_k$, then let $u = U(i_1 \dots i_k(e))$ and $v = V(i_1 \dots i_k(e))$. Then clearly $\tau_{\mathcal{I}}(u) = \tau_{\mathcal{I}}(v)$ but $u \neq v$. Conversely, if $\tau_{\mathcal{I}}$ is not injective, then there exist $u \neq v \in T_\Sigma$ such that $\tau_{\mathcal{I}}(u) = \tau_{\mathcal{I}}(v)$. The trees u and v are necessarily of the form $u = Z_1(i_1 \dots i_k(e))$ and $v = Z_2(j_1 \dots j_p(e))$ for some

$Z_1 \neq Z_2$, say $Z_1 = U$ and $Z_2 = V$. Then the equality $\tau_{\mathcal{I}}(u) = \tau_{\mathcal{I}}(v)$ implies $u_{i_1} \dots u_{i_k} = v_{j_1} \dots v_{j_p}$, but also $i_1 \dots i_k = j_1 \dots j_p$, and hence there is a solution to PCP.

The translation $\tau_{\mathcal{I}}$ is easily implemented by DTOP$^{\mathrm{R}}$ and DMSOT transducers, but they need to copy the input tree, in order to produce, on the output, the subtree $i_1 \dots i_k(e)$, and the subtree $u_{i_1} \dots u_{i_k}(e)$ (or $v_{i_1} \dots v_{i_k}(e)$). $\square$

Now, we show that in presence of origin, the injectivity problem becomes decidable. Origin-injectivity asks whether the origin translation $[\![M]\!]_{\mathsf{o}}$ of a given transducer M is injective, *i.e.* for all trees s_1, s_2, t_1, t_2 and origin mappings o_1, o_2 such that $[\![M]\!]_{\mathsf{o}}(s_1) = (t_1, o_1)$ and $[\![M]\!]_{\mathsf{o}}(s_2) = (t_2, o_2)$, if $t_1 = t_2$ and $o_1 = o_2$, then $s_1 = s_2$. We now prove that origin-injectivity is decidable for DMSO and DTOP$^{\mathrm{R}}$ transducers.

Theorem 14. *Origin-injectivity is decidable for* DMSOT *transducers and for* DTOP^{R} *transducers.*

The proof of Theorem 14 relies on the framework of recognizable translations that we recall first: let Σ, Σ' be ranked alphabets and let $\bot^{(0)}$ be a special symbol not in $\Sigma \cup \Sigma'$. We define $\Sigma \otimes \Sigma'$ as

$$\{(f,g)^{(\max(m,n))} \mid f \in (\Sigma \cup \{\bot\})^{(m)}, g \in (\Sigma' \cup \{\bot\})^{(n)}\} - \{(\bot,\bot)^{(0)}\}$$

Given a tree $t_1 \in T_\Sigma$ and a tree $t_2 \in T_{\Sigma'}$, the *overlap* of t_1 and t_2 is a tree $t_1 \otimes t_2 \in T_{\Sigma\otimes\Sigma'}$ such that $V(t_1 \otimes t_2) = V(t_1) \cup V(t_2)$ and for all nodes $u \in V(t_1 \otimes t_2)$,

$$(t_1 \otimes t_2)[u] = \begin{cases} (t_1[u], t_2[u]) & \text{if } u \in V(t_1) \cap V(t_2) \\ (t_1[u], \bot) & \text{if } u \in V(t_1) \setminus V(t_2) \\ (\bot, t_2[u]) & \text{otherwise.} \end{cases}$$

Let $L_1 \subseteq T_\Sigma$ and $L_2 \subseteq T_{\Sigma'}$. We define $L_1 \otimes L_2 = \{t_1 \otimes t_2 \mid t_1 \in L_1, t_2 \in L_2\}$. A tree translation $\tau \subseteq T_\Sigma \times T_{\Sigma'}$ is *recognizable* if $\{t_1 \otimes t_2 \mid (t_1, t_2) \in \tau\}$ is regular [7]. Note that if both L_1 and L_2 are regular, then so is $L_1 \otimes L_2$.

Definition 15. *Let τ_1, τ_2 be binary relations We define the set*

$$R(\tau_1, \tau_2) = \{(t_1, t_2) \in \mathsf{dom}(\tau_1) \times \mathsf{dom}(\tau_2) \mid \tau_1(t_1) = \tau_2(t_2)\}$$

We will show that these sets are recognizable if τ_1, τ_2 are origin translations of (1) DTOP$^{\mathrm{R}}$ transducers, and (2) MSOT transducers.

Lemma 16. *Let M_1, M_2 be* DTOP^R *transducers. Then $R(\llbracket M_1 \rrbracket_\circ, \llbracket M_2 \rrbracket_\circ)$ is recognizable.*

Proof. For $i = 1, 2$ we construct the DTOP$^\mathrm{R}$ transducer $\tilde{M}_i$ so that for all input trees t_1, t_2 of M_1, M_2: $\llbracket \tilde{M}_i \rrbracket_\circ(t_1 \otimes t_2) = \llbracket M_i \rrbracket_\circ(t_i)$. The transducer $\tilde{M}_i$ has the same states as M_i. The input alphabet of $\tilde{M}_i$ is $\Sigma = \Sigma_1 \otimes \Sigma_2$ where Σ_i is the input alphabet of M_i. If $q(f(x_1, \ldots, x_k) : L) \to C[q_1(x_{i_1}), \ldots, q_k(x_{i_n})]$ is a rule of M_1 then for every g, k' such that $(f, g) \in \Sigma^{(k')}$ the transducer $\tilde{M}_1$ has the rule $q((f, g)(x_1, \ldots, x_{k'}) : \tilde{L}) \to C[q_1(x_{i_1}), \ldots, q_k(x_{i_n})]$ where $\tilde{L} = L \otimes T_{\Sigma_2}$. Remind that $L \otimes T_{\Sigma_2}$ is regular. The rules for $\tilde{M}_2$ are defined similarly. It now holds $R(\llbracket M_1 \rrbracket_\circ, \llbracket M_2 \rrbracket_\circ) = E(\llbracket \tilde{M}_1 \rrbracket_\circ, \llbracket \tilde{M}_2 \rrbracket_\circ)$ and hence the result follows by Lemma 12. $\square$

Lemma 17. *Let T_1, T_2 be* DMSOT *transducers. Then $R(\llbracket T_1 \rrbracket_\circ, \llbracket T_2 \rrbracket_\circ)$ is recognizable.*

Proof. It follows from Lemma 11 that $E(\llbracket T_1 \rrbracket_\circ, \llbracket T_2 \rrbracket_\circ)$ is regular. Then, the result can then be shown similarly to Lemma 16. $\square$

Proof of Theorem 14. An origin translation τ is injective if and only if $R_{\neq} \cap R(\tau, \tau) = \varnothing$ where $R_{\neq} = \{(t_1, t_2) \mid t_1 \neq t_2\}$. Clearly, $R_{\neq}$ is a recognizable translation, and by Lemmas 16 and 17, $R(\tau, \tau)$ is recognizable for the origin translations realized by DMSOT and DTOP$^\mathrm{R}$ transducers. $\square$

3.6. Query Determinacy

Let τ_Q be a functional tree translation and let τ_V be a functional origin tree translation. We will call τ_Q the "query", and τ_V the "view". We say that τ_Q is *determined* by τ_V if for all trees $s_1, s_2 \in \mathsf{dom}(\tau_V)$, if $\tau_V(s_1) = \tau_V(s_2)$ then $\tau_Q(s_1) = \tau_Q(s_2)$. Clearly, the notion of being determined is a generalization of injectivity: the identity tree translation is determined by a view τ_V iff τ_V is injective. The query-determinacy problem, *i.e.*, to decide for given τ_Q, τ_V whether or not τ_Q is determined by τ_V has been studied in the database community [34]. It was recently considered for tree transducers [2]; as mentioned in the Introduction, they can only decide determinacy for very limited classes of views, in particular, these classes do not allow copying. We now show that in the presence of origin, this restriction can be lifted and indeed determinacy can be decided for views that do allow copying.

Corollary 18. *Let τ_Q be a tree translation and let τ_V be an origin tree translation so that both τ_Q, τ_V are defined by two given* DTOP^R *transducers or* *two given* DMSOT *transducers. It is decidable whether or not τ_Q is determined by τ_V.*

Proof. Let τ_{Q_1}, τ_{Q_2} be tree translations with ranked alphabets of input and output symbols $\Sigma \otimes \Sigma$ and Δ, respectively defined by $\tau_{Q_i}(s_1 \otimes s_2) = \tau_Q(s_i)$, for all $s_1, s_2 \in T_\Sigma$. Note that the τ_{Q_i} are effectively definable by DTOP$^{\mathrm{R}}$ transducers (resp. DMSOT transducers) if τ_Q is. Let $r(\tau_V)$ be the set of trees $s_1 \otimes s_2$ such that $s_1, s_2 \in \mathsf{dom}(\tau_V)$ and $\tau_V(s_1) = \tau_V(s_2)$. Then τ_Q is determined by τ_V iff $\tau_{Q_1}(s) = \tau_{Q_2}(s)$ for all s in $r(\tau_V)$, iff τ_{Q_1} and τ_{Q_2} are equivalent on $r(\tau_V)$. As seen before to solve the injectivity problem, we show that the pairs (s_1, s_2) such that $\tau_V(s_1) = \tau_V(s_2)$ form a recognizable relation, for transducers in DTOP$^{\mathrm{R}}$ or DMSOT. So, $r(\tau_V)$ is regular. The result follows because equivalence is decidable for DTOP$^{\mathrm{R}}$ transducers and for DMSOT transducers, see [31]. □

For instance, query determinacy is decidable for queries defined by DMSOT or DTOP$^{\mathrm{R}}$ transducers, and views with origin defined by DMSOT or DTOP$^{\mathrm{R}}$ transducers.

4. Tree-to-String Translations with Origin

In this section we consider top-down tree-to-string transducers. Such transducers generalize top-down tree transducers by outputting arbitrary strings by a rule (instead of well-bracketed strings, in the case of tree transducer). Naturally, decision problems become harder for tree-to-string transducers: origin-equivalence of non-deterministic such transducers is undecidable, and origin-injectivity for deterministic such transducers is undecidable. This is in contrast to top-down tree transducers, where both problems are decidable (Theorems 10 and 14). On the positive side, the main result of this section is that origin-equivalence is decidable for deterministic top-down tree-to-string transducers. The proof uses a reduction to the HDT0L sequence equivalence problem. Without origin, this equivalence problem had been a famous open problem, already stated by Engelfriet [12]; it was solved very recently by Seidl, Maneth, and Kemper [36]

4.1. Top-down tree-to-string transducers

A *top-down tree-to-string transducer* (yTOP transducer for short) M is a tuple $(Q, \Sigma, \Delta, q_0, R)$ where Q is a finite set of states, Σ is a ranked alphabet

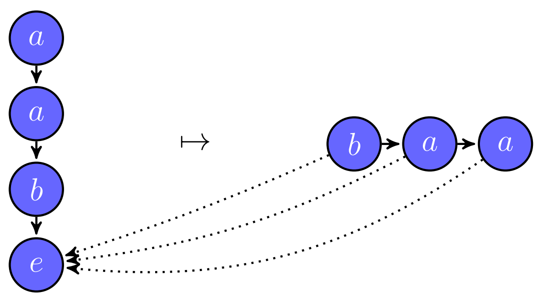

Figure 5: Translation of the input tree $a(a(b(e)))$ by the transducer M_5.

of input symbols, Δ is an alphabet of output symbols, $q_0 \in Q$ is the initial state, and R is a finite set of rules of the form

$$q(\sigma(x_1, \ldots, x_k)) \to w,$$

where w is a (possibly empty) *string* over Δ and symbols $q'(x_i)$ with $q' \in Q$ and $i \in [k]$. The definition of $[\![q]\!]_\circ$ is similar as for TOP transducers (only that the trees t and t_v are now strings over Δ). In this way we obtain $[\![M]\!]_\circ$ and $[\![M]\!]$. yTOP transducers generalize TOP transducers, as right-hand sides of rules of a TOP transducer can be encoded as well-bracketed strings. The converse is false: even considering strings as monadic trees, a yTOP transducer can for instance easily implement string reversal (see Example 19), which is not possible using TOP transducers. As for TOP transducers, we can equip this model with regular look-ahead, and consider deterministic machines. This defines the classes of deterministic top-down tree-to-string transducers (with regular look-ahead): yDTOP (yDTOP$^{\mathrm{R}}$).

Example 19. *Let $M_5 = (Q, \Sigma, \Delta, q_0, R)$ by the yDTOP transducer with $Q = \{q, q_a, q_b\}$, $\Sigma = \{a^{(1)}, b^{(1)}, e^{(0)}\}$, $\Delta = \{a, b\}$, and R consisting of the following rules.*

$$\begin{array}{lcll}
q(\sigma(x_1)) & \to & q(x_1)q_\sigma(x_1) & \\
q(e) & \to & \varepsilon & \\
q_\sigma(\sigma'(x_1)) & \to & q_\sigma(x_1) & \textit{for } \sigma, \sigma' \in \Sigma^{(1)} \\
q_\sigma(e) & \to & \varepsilon & \textit{for } \sigma \in \Sigma^{(1)}.
\end{array}$$

The transducer M_5 implements string reversal with respect to its monadic input tree. For instance, the input tree $s = a(a(b(e)))$ is translated to the string $[\![M]\!](s) = baa = w$, as illustrated in Figure 5. Note that the origin *of each letter of w is the unique leaf of s (here, the node* 111*). Hence, the origin translation $[\![M]\!]_\mathsf{o}$ is* not MSO *definable: there may be unboundedly many letters having such a leaf as origin. In contrast, the translation $[\![M]\!]$ is* MSO *definable, as tree-to-string* MSO *transducers are equivalent to y*DTOP^R *transducers of linear size increase [14, 13]. In Section 5 we show how to decide whether the origin translation defined by a y*DTOP^R *transducer is* MSO *definable.*

4.2. Undecidability Results

A finite-state string transducer (also known as "generalized sequential machine" or as "rational string relation") can be seen as a linear yTOP transducer with monadic input, *i.e.*, a yTOP transducer with rules of the form $q(ax) \to vq'(x)$, where $v \in \Delta^*$. Similar to Example 19, instead of outputting in a rule the string $v = a_1 \cdots a_n$ with $a_1, \ldots, a_n \in \Delta$, we may replace each a_i by a state call $q_{a_i}(x)$. These states ignore the unary input nodes via rules of the form $q_a(bx) \to q_a(x)$, and produce their output symbol a the unique input leaf. This shows that origin-equivalence of yTOP transducers can be reduced to equivalence of finite-state string transducers, a problem well-known to be undecidable [24, 26]. In a similar way it can be shown that origin-injectivity of yDTOP transducers is also undecidable, by giving a direct encoding of the Post Correspondence Problem. This contrasts with the positive results presented in Section 3. There, decidability of origin-equivalence relied on the regularity of the set $E([\![M_1]\!]_\mathsf{o}, [\![M_2]\!]_\mathsf{o})$ of input trees s for which $[\![M_1]\!]_\mathsf{o}(s)$ equals $[\![M_2]\!]_\mathsf{o}(s)$. As the reader may verify, it is easy to come up with two yDTOP transducers M_1, M_2 such that $E([\![M_1]\!]_\mathsf{o}, [\![M_2]\!]_\mathsf{o})$ is not regular. For instance, take the transducer $M_1 = M$ for string reversal of before, and M_2 the identity with rules $q(\sigma(x_1)) \to q_\sigma(x_1)q(x_1)$ and $q_\sigma(\sigma'(x_1)) \to q_\sigma(x_1)$ for $\sigma, \sigma' \in \Sigma^{(1)}$, and leaf rules as for M. Now $E([\![M_1]\!]_\mathsf{o}, [\![M_2]\!]_\mathsf{o})$ is the set of palindromes on Δ^* (seen as monadic trees), which is not regular.

Theorem 20. *Origin-equivalence is undecidable for y*TOP *transducers.*

Proof. The proof proceeds by a reduction from undecidability of equivalence of non-deterministic string-to-string transducers [24, 26]. As explained above, one can simulate such a transducer by forcing the non-deterministic top-down monadic tree-to-string transducer to produce output at the unique leaf only. □

Theorem 21. *Origin-injectivity is undecidable for y*DTOP *transducers.*

Proof. Let A be a finite alphabet such that $A \cap \mathbb{N} = \varnothing$. We consider an instance of the Post Correspondence Problem (PCP)

$$\mathcal{I} = \{(u_1, v_1), \ldots, (u_n, v_n)\},$$

where $u_i, v_i \in A^*$. Recall that $\mathcal{I}$ has a solution iff there exist $i_1, \ldots, i_k \in \{1, \ldots, n\}$ such that $u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$. We construct a yDTOP transducer M such that $[\![M]\!]_{\mathsf{o}}$ is not injective iff $\mathcal{I}$ has a solution.

We define Σ a ranked alphabet as follows: $\Sigma^{(0)} = \{e\}$, and $\Sigma^{(1)} = \{1, \ldots, n\} \cup \{\$_1, \$_2\}$. The output alphabet is $\Delta = A$. The transducer M reads monadic trees of the form $\$(i_1(i_2(\ldots(i_k(e))\ldots)))$, where $\$ \in \{\$_1, \$_2\}$. Its semantics is:

$$[\![M]\!]_{\mathsf{o}}(\$(i_1(i_2(\ldots(i_k(e))\ldots)))) = \begin{cases} (i_1 \ldots i_k u_{i_k} \ldots u_{i_1}, o) & \text{if } \$ = \$_1 \\ (i_1 \ldots i_k v_{i_k} \ldots v_{i_1}, o) & \text{if } \$ = \$_2 \end{cases}$$

where o is the constant function that maps any output position to the input node 1^{k+1}, *i.e.*, all output positions have the same origin, which is the input position labeled e.

Before showing how to construct M, let us show that $[\![M]\!]_{\mathsf{o}}$ is not injective iff $\mathcal{I}$ has a solution. Suppose that $[\![M]\!]_{\mathsf{o}}$ is not injective. Then here exist two input trees $t_1 = \$_a(i_1(\ldots(i_k(e))\ldots))$ and $t_2 = \$_b(j_1(\ldots(j_\ell(e))\ldots))$ such that $t_1 \neq t_2$ and $[\![M]\!]_{\mathsf{o}}(t_1) = [\![M]\!]_{\mathsf{o}}(t_2)$. By definition of $[\![M]\!]_{\mathsf{o}}$, since $[\![M]\!]_{\mathsf{o}}(t_1) = [\![M]\!]_{\mathsf{o}}(t_2)$ and $A \cap \mathbb{N} = \varnothing$, we necessarily get that $i_1 \ldots i_k = j_1 \ldots j_\ell$. Since $t_1 \neq t_2$, we get that $a \neq b$. If $a = 1$ and $b = 2$, then $u_{i_k} \ldots u_{i_1} = v_{i_k} \ldots v_{i_1}$, and therefore $\mathcal{I}$ has a solution (and similarly if $a = 2$ and $b = 1$). Conversely, if $\mathcal{I}$ as a solution $i_1, \ldots, i_k$, then by taking $t_1 = \$_1(i_1(\ldots(i_k(e))\ldots))$ and $t_2 = \$_2(i_1(\ldots(i_k(e))\ldots))$, we get $[\![M]\!]_{\mathsf{o}}(t_1) = [\![M]\!]_{\mathsf{o}}(t_2)$ and $t_1 \neq t_2$.

The transducer M is constructed as follows: its set of states is $Q = \{q_0, q_1, q_2\} \cup \{p_j \mid 1 \leq j \leq n\} \cup Q'$ with $Q' = \{p_w \mid \exists 1 \leq i \leq n,\ w = u_i \vee w =$

$v_i\}$, initial state q_0, and set of rules consisting of the following.

$$\begin{array}{lcll}
q_0(\$_1(x)) & \to & q_1(x) & \\
q_0(\$_2(x)) & \to & q_2(x) & \\
q_1(i(x)) & \to & p_i(x)q_1(x)p_{u_i}(x) & \\
q_1(e) & \to & \epsilon & \\
q_2(i(x)) & \to & p_i(x)q_2(x)p_{v_i}(x) & \\
q_2(e) & \to & \epsilon & \\
p_j(i(x)) & \to & p_j(x) & \text{for } 3 \leq j \leq n \\
p_j(e) & \to & j & \text{for } 3 \leq j \leq n \\
p_w(i(x)) & \to & p_w(x) & \text{for } p_w \in Q' \\
p_w(e) & \to & w & \text{for } p_w \in Q'
\end{array}$$

□

*4.3. Equivalence of y*DTOP^R *transducers*

Though it is not possible to obtain decidability results by regular sets (or recognizable relations), we can prove, using more involved techniques, that origin-equivalence of yDTOP$^{\mathrm{R}}$ transducers is decidable.

Theorem 22. *Origin-equivalence is decidable for* yDTOP^R *transducers.*

Proof. Let M_1, M_2 be two yDTOP$^{\mathrm{R}}$ transducers for which we want to test whether $[\![M_1]\!]_\circ = [\![M_2]\!]_\circ$. The procedure is divided into several steps and based on several intermediate lemmas that are proved later on.

Step 1. We first check whether M_1 and M_2 have the same domain D. If not we output "not equivalent". Otherwise, we build yDTOP transducers *without* look-ahead M_1', M_2', and a regular tree language D' such that $[\![M_1]\!]_\circ = [\![M_2]\!]_\circ$ iff M_1' and M_2' are origin-equivalent on D'. To this end, we extend the input alphabet with look-ahead labeling of both transducers, and compute a language of interest D' from D (Lemma 23).

Step 2. Then, as shown in Lemma 27, we reduce the origin-equivalence problem of M_1' and M_2' on D' to the (origin-free) equivalence problem of monadic yDTOP transducrs, where 'monadic' means that every input symbol has rank 0 or 1. The latter problem is known to be decidable, see [31], based on the HDT0L sequence equivalence problem [9]. The reduction of origin-equivalence of M_1' and M_2' on D', to equivalence of yDTOP transducers on monadic trees, explained in Lemma 27, is based on the following idea. To check origin-equivalence of M'_1 and M'_2 on D', one only needs to consider their behaviours on paths, *i.e.*, the partial output strings produced on root-to-node paths of the input tree (called string abstractions). This latter observation is formalised in Lemma 26. $\square$

4.4. Lemmas used in the proof of Theorem 22

The proof of Theorem 22 is based on several key lemmas.

Lemma 23. *Let M_1, M_2 be two yDTOP^R transducers. One can compute two yDTOP transducers M'_1 and M'_2 and a regular tree language E, such that M_1 and M_2 are origin-equivalent iff M'_1 and M'_2 are origin-equivalent on E.*

Proof. Let P_i be the set of states of the look-ahead automaton A_i of M_i (assumed to be non-deterministic top-down tree automata). Without loss of generality we assume that every rule $q(a(\dots) : p) \to w$ of M_i is such that the language of A_i from initial state p is non-empty (otherwise we remove such a useless rule).

Then, the idea is to extend the alphabet of the transducers M_i with look-ahead states, and delegate the look-ahead tests to the regular language E. More precisely, if M_1 has a rule $q(a(\dots) : p_1) \to w$ then M'_1 has, for every $p_2 \in P_2$, the rule $q(\langle a, p_1, p_2\rangle(\dots)) \to w$. Similarly, if M_2 has a rule $q(a(\dots) : p_2) \to w$ then M'_2 has for every $p_1 \in P_1$ the rule $q(\langle a, p_1, p_2\rangle(\dots)) \to w$. The resulting transducers M'_1 and M'_2 are both deterministic. Suppose it is not the case, then it implies that there are two rules of the form $q(\langle a, p_1, p_2\rangle(\dots)) \to w$ and $q(\langle a, p_1, p_2\rangle(\dots)) \to w'$ with $w \neq w'$, say in M'_1. By definition of M'_1, we get that there are two rules of the form $q(a(\dots) : p_1) \to w$ and $q(a(\dots) : p_1) \to w'$ in M_1. Since M_1 is deterministic, it implies that the language of A_1 from initial state p_1 is empty. This contradicts the fact that M_1 does not contain such rules by our initial assumption.

The language E accepts a tree t over the extended alphabet iff for all nodes u of t with label $\langle a, p_1, p_2\rangle$, for all $i \in \{1, 2\}$, the projection of the subtree t/u on the first component is accepted by A_i from initial state p_i. The language E can be defined by an alternating top-down tree automaton: when reading a label $\langle a, p_1, p_2\rangle$, two universal transitions are triggered to two automata A'_1 and A'_2 respectively, which simulate A_1 and A_2 on the first projection. Alternating top-down tree automata are no more expressive than regular tree languages [7], hence the result follows. $\square$

String abstractions. Towards our goal (prove Step 2), we will need the notion of string abstraction. Let A, B two alphabets (not necessarily finite). Let $w \in (A \times B)^*$ and $U \subseteq B$. The *U-abstraction of* w is the word obtained from U by keeping only those letters (a, b) such that $b \in U$. Formally, it is the image $\mu_U(w)$ of w by the morphism μ_U defined by $\mu_U((a, b)) = \epsilon$ if $b \notin U$ and $\mu_U((a, b)) = (a, b)$ if $b \in U$. The next lemma shows that a word is uniquely determined by its U-abstractions such that U has cardinality one or two.

Lemma 24. *For all $w_1, w_2 \in (A \times B)^*$, $w_1 = w_2$ iff for all $U \subseteq B$ such that $|U| \in \{1, 2\}$, $\mu_U(w_1) = \mu_U(w_2)$.*

Proof. The *only if* direction is trivial. Let us show the *if* direction. Suppose that $w_1 \neq w_2$. We consider two cases:

- if $|w_1| = |w_2|$. Let i be the first position such that $w_1[i] \neq w_2[i]$. Let $(a_1, b_1) = w_1[i]$ and $(a_2, b_2) = w_2[i]$. By definition of i, there exist $\alpha, \beta_1, \beta_2 \in (A \times \{b_1, b_2\})^*$ such that $\mu_{\{b_1,b_2\}}(w_1) = \alpha(a_1, b_1)\beta_1$ and $\mu_{\{b_1,b_2\}}(w_2) = \alpha(a_2, b_2)\beta_2$. Since $(a_1, b_1) \neq (a_2, b_2)$, we get $\mu_{\{b_1,b_2\}}(w_1) \neq \mu_{\{b_1,b_2\}}(w_1)$. Note that we may have $b_1 = b_2$ therefore we also need to consider singleton sets U in the lemma.

- if $|w_1| \neq |w_2|$. Observe that $|w_j| = \sum_{b \in B} |\mu_{\{b\}}(w_j)|$ for $j = 1, 2$. Since $|w_1| \neq |w_2|$, there exists $b \in B$ such that $|\mu_{\{b\}}(w_1)| \neq |\mu_{\{b\}}(w_2)|$, and so $\mu_{\{b\}}(w_1) \neq \mu_{\{b\}}(w_2)$.

□

String abstractions modulo origins. Let s be a tree, $w = a_1 \dots a_n \in \Delta^*$ a string with $a_i \in \Delta$ for all i, and $o : V(w) \to V(s)$ an origin mapping. We denote by $w \otimes o \in (\Delta \times V(s))^*$ the word w whose labels are extended with origins, *i.e.* $|w \otimes o| = |w|$ and for all $i \in \{1, \dots, |w|\}$, $(w \otimes o)[i] = (w[i], o(i))$. Let $U \subseteq V(s)$. The U-abstraction of w *modulo* o is defined as $\Pi_U(w, o) = \mu_U(w \otimes o)$. A direct consequence of Lemma 24 is that a word with origins is uniquely determined by the set of its U-abstractions modulo origin, with $|U| \in \{1, 2\}$. The following example shows that considering sets U of cardinality 2 is necessary.

Example 25. *Consider the string $w = aaaa$ and the two origin mappings*

$$\begin{aligned} o_1 &= \{(1, u), (2, v), (3, x), (4, u)\} \\ o_2 &= \{(1, u), (2, x), (3, v), (4, u)\} \end{aligned}$$

where u, v, x refer to three distinct nodes in some input tree. By putting origin information in w, one gets

$$\begin{array}{rcl} w \otimes o_1 & = & (a,u)(a,v)(a,x)(a,u) \\ w \otimes o_2 & = & (a,u)(a,x)(a,v)(a,u) \end{array}$$

Then we have:

$$\begin{array}{rcl} \Pi_{\{u\}}(w,o_1) & = & (a,u)(a,u) \\ \Pi_{\{u\}}(w,o_2) & = & (a,u)(a,u) \\ \Pi_{\{v\}}(w,o_1) & = & (a,v) \\ \Pi_{\{v\}}(w,o_2) & = & (a,v) \\ \Pi_{\{x\}}(w,o_1) & = & (a,x) \\ \Pi_{\{x\}}(w,o_2) & = & (a,x) \\ \Pi_{\{v,x\}}(w,o_1) & = & (a,v)(a,x) \\ \Pi_{\{v,x\}}(w,o_2) & = & (a,x)(a,v) \end{array}$$

Note that the U-abstractions of $w \otimes o_1$ and $w \otimes o_2$ are equal for $|U| = 1$, while $w \otimes o_1 \neq w \otimes o_2$. They are distinguished by $\{v, x\}$, $|\{v, x\}| = 2$.

As a direct consequence of Lemma 24 we obtain the following lemma.

Lemma 26. *Let τ_1, τ_2 be two functional tree-to-string origin translations. Let $s \in \mathsf{dom}(\tau_1) \cap \mathsf{dom}(\tau_2)$. Then $\tau_1(s) = \tau_2(s)$ iff $\Pi_U(\tau_1(s)) = \Pi_U(\tau_2(s))$ for all $U \subseteq V(s)$ such that $|U| \in \{1, 2\}$.*

Based on Lemma 26, the next lemma shows origin-equivalence can be decided for yDTOP transducers.

Lemma 27. *Let M_1, M_2 be two yDTOP transducers and let D be a regular tree language. It is decidable whether or not $[\![M_1]\!]_o(s) = [\![M_2]\!]_o(s)$ for all $s \in D$.*

Proof. We may assume that the input and output alphabets of M_1 and M_2 coincide. Thus $M_i = (Q_i, \Sigma, \Delta, r_i, R_i)$. In order to prove this decidability result, we apply Lemma 26 to $\tau_i = [\![M_i]\!]_o$ for $i = 1, 2$. Without loss of generality, we may assume that $D \subseteq \mathsf{dom}(\tau_1) \cap \mathsf{dom}(\tau_2)$.

We now explain how to decide the characterization of Lemma 26 for singleton sets U. Note that for singleton sets, since the symbols of the U-abstraction of a word always have the same second component, we can ignore it. More precisely, if one denotes by π_1 the first projection on pairs, we have

$\Pi_U(\tau_1(s)) = \Pi_U(\tau_2(s))$ iff $\pi_1(\Pi_U(\tau_1(s))) = \pi_1(\Pi_U(\tau_2(s)))$ for all $s \in D$. We show how to check the latter equality.

To this end, we build deterministic top-down monadic tree-to-string transducers $\hat{M}_i$, $i = 1, 2$ that will run on paths of trees and will simulate M_i on this path: only the symbols whose origin correspond to the leaf of the considered path are produced. Their input ranked alphabet is defined as $\pi(\Sigma) = \Sigma^{(0)} \cup \bigcup_{k \geq 1} \{\sigma_\ell^{(1)} \mid \sigma \in \Sigma^{(k)}, \ell \in [k], k \geq 1\}$. That is, all symbols from Σ are in $\pi(\Sigma)$ and their arity is set to 0, and all symbols from Σ which are not constant are augmented with a direction ℓ and their arity is set to 1.

Given a tree s on Σ, and a node $u \in V(s)$, we introduce the notion of label path $\mathsf{lpath}_s(u)$, as the monadic tree representing the labeled path in s from the root to the leaf u. Formally:

- if $s = \sigma(s_1, \ldots, s_k)$, $k \geq 0$ and $u = \epsilon$, then $\mathsf{lpath}_s(u) = \sigma^{(0)}$,
- if $s = \sigma(s_1, \ldots, s_k)$, $k \geq 1$, and $u = \ell u'$, $\ell \in [k]$, then $\mathsf{lpath}_s(u) = \sigma_\ell^{(1)}(\mathsf{lpath}_{s_\ell}(u'))$.

First define the intermediate deterministic top-down monadic tree-to-string transducer $\overline{M_i} = (Q_i, \pi(\Sigma), \Delta, r_i, \overline{R_i})$. If $q(\sigma(x_1, \ldots, x_k)) \to w$ is a rule of M_i with $k \geq 0$, then $\overline{M_i}$ has the following rules for every $\ell \in [k]$:

$$q(\sigma^{(0)}) \to w[q'(x_j) \leftarrow \varepsilon \mid q' \in Q_i, j \in [k]]$$
$$q(\sigma_\ell^{(1)}(x_1)) \to w[q'(x_\ell) \leftarrow q'(x_1),\ q'(x_j) \leftarrow \varepsilon \mid q' \in Q_i, j \in [k] \setminus \{\ell\}] \text{ if } k \geq 1$$

Roughly speaking, this new transducer processes paths of the input tree and produces for a given path only elements whose origin lies on this path.

Claim. For every $s \in \mathsf{dom}(\tau_i)$ and every $u \in V(s)$, $[\![\overline{M_i}]\!](\mathsf{lpath}_s(u)) = \pi_1(\Pi_{\{u\}}(\tau_i(s)))$.

We define now $\mathsf{LPath}(D) = \{\mathsf{lpath}_s(u) \mid s \in D, u \in V(s)\}$. As D is a regular tree language on Σ, $\mathsf{LPath}(D)$ is a regular monadic tree language on $\pi(\Sigma)$. It is thus recognized by a deterministic top-down finite tree automaton. The transducer $\hat{M}_i$ is defined as the product of $\overline{M_i}$ with this finite tree automaton.

Claim. $\Pi_{\{u\}}(\tau_1(s)) = \Pi_{\{u\}}(\tau_2(s))$ for all $s \in D$ and $u \in V(s)$ iff $[\![\hat{M}_1]\!] = [\![\hat{M}_2]\!]$.

As the equivalence of deterministic top-down monadic tree-to-string transducers is decidable (see [31]), so is the characterization of Lemma 26 for singleton sets U.

For sets $U = \{u_1, u_2\}$ of cardinality 2, the equalities $\Pi_{\{u_1,u_2\}}(\tau_1(s)) = \Pi_{\{u_1,u_2\}}(\tau_2(s))$ can be checked similarly with two additional modifications: (i) instead of running on paths, the transducers $\hat{M}_i$ run on products of two paths that can be embedded in a same tree of D, (ii) one cannot consider the first projection π_1 any more, and instead one colours the symbols whose origin is the leaf of the first path by 1, and those whose origin is the leaf of the second path by 2. More precisely, inputs of $\hat{M}_i$ are monadic trees of the form $p_1 \otimes p_2$, where $p_j = \mathsf{lpath}_s(u_j)$ for $j \in [2]$ for some tree $s \in \mathsf{dom}(\tau_i)$ and two nodes $u_1 \neq u_2 \in V(s)$. The tree $p_1 \otimes p_2$ is obtained by overlapping the paths p_1 and p_2, with a product alphabet and a padding symbol in case they do not have the same length, as defined in Section 3.4. The transducer $\hat{M}_i$ then simulates M_i on the two paths (with a construction similar to the singleton case) and produce only the symbols produced by M_i at the respective leaves of these two paths (coloured respectively by 1 or 2 to distinguish their origin). Moreover, the order in which those symbols appear in the output string $\tau_i(s)$ is kept. More precisely, the set of states of $\hat{M}_i$ is $Q_i \cup Q_i \times \{1, 2\}$: as long as the two paths coincide, we stay in Q_i, and when they diverge, one goes to $Q_i \times \{1\}$ or to $Q_i \times \{2\}$ depending on whether one evaluates the first the second path. Let us describe some of the important rules of $\hat{M}_i$ that are created from a rule of the form $q(\sigma(x_1, \ldots, x_k)) \to w$ in M_i:

1. As long as the two paths coincide, it works as in the singleton case. We create the following rules in $\hat{M}_i$:

   $$q((\sigma_\ell, \sigma_\ell)(x_1)) \to w[a \in \Delta \leftarrow \epsilon, q'(x_{j\neq\ell}) \leftarrow \epsilon, q'(x_\ell) \leftarrow q'(x_\ell)] \text{where } \ell \in [k]$$

2. The first time the two paths diverge, *i.e.* when a symbol of the form $(\sigma_\ell, \sigma_{\ell'})$ with $\ell \neq \ell'$ is met, $\hat{M}_i$ sends copies to $Q_i \times \{1\}$ and $Q_i \times \{2\}$ respectively, by rules of the form, for $\ell, \ell' \in [k]$: $q((\sigma_\ell, \sigma_{\ell'})(x_1)) \to$

   $$w[a \in \Delta \leftarrow \epsilon, q'(x_{\ell' \neq j \neq \ell}) \leftarrow \epsilon, q'(x_\ell) \leftarrow (q', 1)(x_\ell), q'(x_{\ell'}) \leftarrow (q', 2)(x_{\ell'})]$$

3. In a copy $Q_i \times \{c\}$, for $c \in \{1, 2\}$, the c'-th path is ignored (it becomes inactive), where $c \neq c' \in \{1, 2\}$. This is for instance achieved by rules of the form:

   $(q, 1)((\sigma_\ell, \beta_{\ell'})(x_1)) \to w[a \in \Delta \leftarrow \epsilon, q'(x_{j\neq\ell}) \leftarrow \epsilon, q'(x_\ell) \leftarrow (q', 1)(x_\ell)]$ or

   $(q, 2)((\beta_{\ell'}, \sigma_\ell)(x_1)) \to w[a \in \Delta \leftarrow \epsilon, q'(x_{j\neq\ell}) \leftarrow \epsilon, q'(x_\ell) \leftarrow (q', 2)(x_\ell)]$.

4. when some active path reaches a leaf, the rules output symbols that are produced in the rhs w, augmented with the colour 1 or 2 depending on whether the first or second path is active. For instance, one may add the following rules:

   $(q,1)((\sigma^{(0)},z)(x_1)) \to w[a \in \Delta \leftarrow (a,1), q'(x_j) \leftarrow \epsilon, j \in [k]]$ or

   $(q,2)((z,\sigma^{(0)})(x_1)) \to w[a \in \Delta \leftarrow (a,2), q'(x_j) \leftarrow \epsilon, j \in [k]]$

   where z can be any symbol of the form $\beta_{\ell'}$ or $\beta \in \Sigma$.

We have presented the essential rules of this construction. As for the singleton case, we also need to restrict the domain of $\hat{M}_i$ to path products that can be embedded into trees s of D. This is again a regular property.

Finally, checking whether or not $\Pi_{\{u_1,u_2\}}(\tau_1(s)) = \Pi_{\{u_1,u_2\}}(\tau_2(s))$ for all $s \in D$ and all $u_1 \neq u_2 \in V(s)$ reduces to checking equivalence of $\hat{M}_1$ and $\hat{M}_2$, which is decidable. $\square$

5. Subclass Definability Problems

5.1. From yDTOP^R Transducers to MSO Transducers

Deterministic MSO tree-to-string transducers (DMSOTS transducers for short) can be defined as a particular case of DMSOT transducers (their origin-equivalence is thus decidable by Theorem 10). While DMSOTS transducers are equivalent to yDTOP$^{\mathrm{R}}$ transducers of linear size increase [14, 13], this is not true in the presence of origin; there is a yDTOP$^{\mathrm{R}}$ transducer for which no origin-equivalent DMSOTS transducer exists (*e.g.* Example 19 of string reversal). However, every DMSOTS transducer effectively has an origin-equivalent yDTOP$^{\mathrm{R}}$ transducer (obtained by following the respective constructions for origin-less transducers). It raises the question whether one can decide for a given yDTOP$^{\mathrm{R}}$ transducer whether its origin translation is definable by a DMSOTS transducer. In Theorem 30 we give an affirmative answer to this question.

The proof of the theorem is based on the notion of "bounded origin". An origin translation τ is of *bounded origin* if there exists a number k such that for every $(s,(w,o)) \in \tau$ and $u \in V(s)$: $|\{v \in V(w) \mid o(v) = u\}| \leq k$, *i.e.*, every input node can be the origin of only a bounded number of output positions. By their definition, origin translations of MSO transducers have bounded origin. We will prove in Lemma 29 that the yDTOP$^{\mathrm{R}}$ transducers of bounded origin are effectively MSO definable. First, let us prove in Lemma 28 that the bounded origin property is decidable for $y\text{DTOP}^{\text{R}}$ transducers.

The proof of Lemma 28 is similar to the proof of Lemma 4.10 in [14], where it is proved that the finite copying restriction is decidable for macro tree transducers (this property means that every input node is translated only a bounded number of times).

Lemma 28. *Let M by a $y\text{DTOP}^{R}$ transducer. It is decidable whether or not M has bounded origin; if it does, then a bound can be computed.*

Proof. Let $M = (Q, \Sigma, \Delta, q_0, R)$ and let $A = (P, \Sigma, \delta)$ be its look-ahead automaton. Let $\hat{\Sigma} = \{\hat{\sigma}^{(k)} \mid \sigma \in \Sigma^{(k)}, k \geq 0\}$. We construct the $y\text{DTOP}^{\text{R}}$ transducer $\hat{M}$ that mimics the state behavior of M, but does not produce any output on input trees in T_Σ. The new transducer $\hat{M}$ produces outputs only on hatted symbols in $\hat{\Sigma}$. Let $\hat{M} = (Q, \Sigma \cup \hat{\Sigma}, \Delta, q_0, R')$ and $\hat{A} = (P, \Sigma \cup \hat{\Sigma}, \delta \cup \hat{\delta})$. For every rule $q(\sigma(x_1 : p_1, \ldots, x_k : p_k)) \to w$ in R we let the rules

$$\begin{array}{lcl} q(\hat{\sigma}(x_1 : p_1, \ldots, x_k : p_k)) & \to & w[q'(x_i) \leftarrow \varepsilon \mid q' \in Q, i \in [k]] \\ q(\sigma(x_1 : p_1, \ldots, x_k : p_k)) & \to & w[d \leftarrow \varepsilon \mid d \in \Delta] \end{array}$$

be in R'. For every look-ahead transition $\delta(\sigma, p_1, \ldots, p_k) = p$ we define $\hat{\delta}(\hat{\sigma}, p_1, \ldots, p_k) = p$. We now consider input trees over $\Sigma \cup \hat{\Sigma}$ with exactly one node labeled by a hatted symbol. Let $(t, (\xi, o)) \in [\![M]\!]_{\text{o}}$ and let $u \in V(t)$ with $t[u] = \sigma$. Let $\hat{t}$ be the tree obtained from t by changing the label at node u to $\hat{\sigma}$. Then

$$|[\![\hat{M}]\!](\hat{t})| = |\{v \in V(\xi) \mid o(v) = u\}|.$$

Thus, $[\![M]\!]$ has bounded origin if and only if $[\![\hat{M}]\!](\hat{D})$ is finite, where

$$\hat{D} = \{t[u \leftarrow \hat{\sigma}(t_1, \ldots, t_k)] \mid t \in D, \exists u \in V(t), t/u = \sigma(t_1, \ldots, t_k)\}$$

and D is the domain of M. Since the domains of $y\text{DTOP}^{\text{R}}$ transducers are effectively regular, so is $\hat{D}$. Finiteness of $[\![\hat{M}]\!](\hat{D})$ is decidable by Theorem 6.2 of [10]. In case of finiteness, the theorem provides an algorithm that computes the elements of $[\![\hat{M}]\!](\hat{D})$. Thus, to compute an origin bound, we determine the maximal size of the elements in $[\![\hat{M}]\!](\hat{D})$. □

In the next lemma it is shown that origin translations of (total) $y\text{DTOP}^{\text{R}}$ transducers with bounded origin are effectively MSO definable. This is done by verifying that various constructions of the literature (that are proved for ordinary semantics) also hold true with respect to origin semantics. Note that the proof of Lemma 29 makes use of macro tree transducers, which are formally defined and explained only in Section 5.2.

Lemma 29. *Let M be a total yDTOP^R transducer that has bounded origin. There effectively exists a* DMSOTS *transducer N that is origin-equivalent to M.*

Proof. Let $M = (Q, \Sigma, \Delta, q_0, R)$ with look-ahead automaton $A = (P, \Sigma, \delta)$, and let k be an origin bound of M. In this proof, we assume that right-hand sides of rules of M are non-empty words. Observe that this assumption can be made without loss of generality as inverse images of regular languages by yDTOP$^{\mathrm{R}}$ transducers are regular, and thus input trees that are translated into ε can be filtered out using regular look-aheads.

We first transform M into the total MAC transducer M'; see Section 5.2 for the formal definition of MAC transducers. Let $M' = (Q', \Sigma, \Delta', q_0, R')$ with same look-ahead automaton A and with $Q' = \{q^{(1)} \mid q \in Q\} \cup \{(q')^{(2)} \mid q \in Q\}$ and $\Delta' = \{\perp^{(0)}\} \cup \{a^{(1)} \mid a \in \Delta\}$. For every rule $q(\sigma(x_1 : p_1, \ldots, x_k : p_k)) \to w$ of M, the transducer M' has rules

$$\begin{array}{lcl} q(\sigma(x_1 : p_1, \ldots, x_k : p_k)) & \to & \mathsf{mon}(w) \\ q'(\sigma(x_1 : p_2, \ldots, x_k : p_k), y_1) & \to & \mathsf{mon}(wy_1). \end{array}$$

For a non-empty word w, $a \in \Delta$, $q \in Q$, and $i \in \mathbb{N}$, the tree $\mathsf{mon}(w)$ is defined as: $\mathsf{mon}(a) = a(\perp)$, $\mathsf{mon}(aw) = a(\mathsf{mon}(w))$, $\mathsf{mon}(y_1) = y_1$, $\mathsf{mon}(q(x_i)) = q(x_i)$, and $\mathsf{mon}(q(x_i)w) = q'(x_i, \mathsf{mon}(w))$. Note that the parameter y_1 appears exactly once in the right-hand side of every rule of a state of rank two of M'. A MAC transducer with this property is called "linear and nondeleting (in the parameters)". Moreover, since every right-hand side of the rules of M is a non-empty word, M' does not have a right-hand side of the form y_1. A MAC transducer with this property is called "nonerasing".

The construction from M to M' preserves origin in the following sense. If $[\![M]\!]_{\mathrm{o}}(s)$ equals (w, o) with $V(w) = [n]$, then $[\![M']\!]_{\mathrm{o}}(s)$ equals $(\mathsf{mon}(w), o')$ where $o'(1^{i-1}) = o(i)$ for every $i \in [n]$, and $o'(1^n) = o(n)$. Hence $k + 1$ is an origin bound for M'. Note that if M' has an origin-equivalent DMSO transducer N, then so does M, because it is straightforward to remove $\perp$ from the output of the transducer.

For a MAC$^{\mathrm{R}}$ transducer that is linear, nondeleting, nonerasing, and "finite copying in the input" (fci), it is shown in Lemma 6.10 of [13] that an equivalent “single use restricted” (sur) MAC^{R} transducer can be constructed. It can be verified that the corresponding construction produces a transducer that is origin-equivalent. In order to apply this construction we need to show that M' is fci.

Intuitively, fci means that there is a bound on the number of times that the transducer translates any given input subtree. Technically this is achieved by marking an arbitrary input node and having the transducer “stop” at this node and output as a terminal nodes the occurrences of the particular states that translate this input node. These numbers of state occurrences need to be globally bounded. Formally, fci means that there is a number β such that for every input tree s and node u of s: the number of occurrences of states in the tree $[\![\widehat{M'}]\!](s[u \leftarrow \delta(s/u)])$ is at most β, where $\widehat{M'}$ is the *extension* of M' to input trees over $\Sigma \cup \{p^{(0)} \mid p \in P\}$. The transducer $\widehat{M'}$ extends the rules of M' by $q(p) \to p$ for every state q of M' and look-ahead state p of M', and extends the transition function δ of the look-ahead automaton of M' by $\delta(p) = p$ for all $p \in P$. Assume now by contradiction that M' is not fci. For every $p \in P$, let $s_p \in T_\Sigma$ be a fixed (but arbitrary) tree such that $\delta(s_p) = p$. Let the integer m be $\max\{|V(s_p)| \mid p \in P\}$. Note that in particular we can choose s_p to be a tree of minimal size such that $\delta(s_p) = p$. We claim that M' is fci with bound $\beta = (k+1)\cdot m$. Suppose not. Then there exist s and u such that the number of occurrences of states in $\zeta = [\![\widehat{M'}]\!](s[u \leftarrow \delta(s/u)])$ is larger than $(k+1)\cdot m$. Let $p = \delta(s/u)$. Since M' is total, $t = [\![\widehat{M'}]\!](s[u \leftarrow s_p])$ is defined. Since M' is non-erasing, every occurrence of a state in ζ contributes at least one output symbol to t. Thus, t contains $> (k+1)\cdot m$ nodes with their origin in the subtree s_p (*i.e.*, such that u is a prefix of their origin). Since $m > |V(s_p)|$, there is a node uv in $s[u \leftarrow s_p]$ such that t contains strictly more than $k+1$ nodes with origin uv. This contradicts the bounded origin property of M'.

In [13] it is shown how to construct for a total sur MAC^{R} transducer an equivalent sur attribute tree transducer with look-ahead (ATT^{R} transducer for short). The construction consists of three steps. First in Lemma 5.9 of [13], the sur MAC^{R} transducer M is decomposed into a particular relabeling ρ, followed by a more restricted MAC transducer M', namely one that is “strongly single use restricted” (ssur for short). The rules of M' are essentially the same as those of M, only that the input symbols have changed (intuitively, the relabeled input symbols contain more information, such as the look-ahead states, and information related to the sur property). Thus,

M' generates output at precisely the same input nodes as M, and hence the composition of ρ followed by $[\![M']\!]$ is origin-equivalent to $[\![M]\!]$. In the second step (Lemma 5.12 of [13]) the ssur MAC M' transducer is transformed into an equivalent sur ATT$^{\mathrm{R}}$ transducer A. This is done in such a way that parts of a rule of M' are represented by several rules of A. Intuitively, the trees t_j of a state call $q'(x_i, t_1, \ldots, t_m)$ in a rule are realized by inherited attributes of A, which are evaluated at the i-th child of the current node. As a consequence, the origin (by A) of output nodes generated in a t_j is *different* than the origin with respect to M': they are now at node $u.j$ instead of node u. It is straightforward to alter the construction in such a way that origin is preserved. This is done by adding new synthesized attributes at node u which obtain the right-hand sides of the rules of those inherited attributes. The values of these new synthesized attributes are then simply copied over to new extra inherited attributes (by rules of the parent node of u) and are passed down to the inherited attributes of the j-th child node of u (which are the ones of construction of Lemma 5.12 of [13]). In the third step, the relabeling ρ is represented as a "attributed relabeling" (Lemma 4.1 of [13]), which gives the desired ATT$^{\mathrm{R}}$ transducer A' that is origin-equivalent to M.

Next, we transform A' into an origin equivalent MSO tree transducer. This construction is given in Section 5 of [3]. Technically speaking, the transducer is first changed into "operator normal form", by adding additionally special output symbols (which represent identity). The new ATT$^{\mathrm{R}}$ transducer A'' followed by the pruning of the inserted new symbols is origin equivalent to A'. Now an MSO tree transducer N is constructed; its copy names are the attributes of A'', and it produces output in the same way as A', *i.e.*, it is clear that N is origin-equivalent to A''. Finally, by simply removing the inserted new symbols from before in the rules of N, an MSO tree transducer is obtained that is origin-equivalent to M. $\square$

Theorem 30. *For a given y*DTOP^R *transducer M, it is decidable whether or not there exists an origin-equivalent* DMSOTS *transducer. If so, then such a* DMSOTS *transducer can be constructed.*

Proof. We first check if M has bounded origin, using Lemma 28. If not, then we return "no", because by their definition, all translations of DMSOTS transducers have bounded origin. Otherwise let m be the origin bound computed according to Lemma 28. Let $M = (Q, \Sigma, \Delta, q_0, R)$ with look-ahead automaton $A = (P, \Sigma, \delta)$. We now construct a total version M' of M with the following properties. For every $t \in D = \mathsf{dom}(M)$, $[\![M']\!]_{\mathsf{o}}(t) = [\![M]\!]_{\mathsf{o}}(t)$.

For every $t \in T_\Sigma \setminus D$, $[\![M']\!]_\circ(t) = (\#, o)$ where o maps the unique output position to the root node of t, and $\#$ is a new symbol not in Δ. Thus, M' has bounded origin with bound m. The domain D is effectively given by a non-deterministic tree automaton $A' = (P', \Sigma, \delta', P_f)$. We construct the look-ahead automaton of M' as the product of A and A' (where we forget about the final states). Let $q(\sigma(x_1 : p_1, \ldots, x_k : p_k)) \to w$ be in R and let $p'_1, \ldots, p'_k \in P'$. If $q \neq q_0$ or $\delta'(\sigma, p'_1, \ldots, p'_k) \notin P_f$ then we let the rule $q(\sigma(x_1 : (p_1, p'_1), \ldots, x_k : (p_k, p'_k))) \to w$ be in R'. Otherwise, we let the rule $q_0(\sigma(x_1 : (p_1, p'_1), \ldots, x_k : (p_k, p'_k))) \to \#$ be in R'.

We apply Lemma 29 and construct a DMSOTS transducer N' that is origin equivalent to M'. Finally, we construct an MSO formula ϕ which holds for a tree $t \in T_\Sigma$ if and only if $t \in D$. The DMSOTS transducer N is obtained from N' by changing its domain formula to ϕ. □

*5.2. From Macro Tree Transducers to y*DTOP^R *Transducers*

At last we consider a type of transducer that is expressively more powerful than the models considered so far: the macro tree transducer (MAC transducer for short) [18]. For simplicity, we only consider total deterministic MAC transducers.

Definition of Macro Tree Transducers

A MAC transducer extends a top-down tree transducer by *nesting* of recursive state calls. Thus, a state q is now of rank $m+1$ and takes, besides the input tree (as its first argument), m arguments of type output tree. In the rules, these arguments are denoted by *parameters* $y_1, \ldots, y_m$. Thus, a rule of a MAC transducer is of the form

$$q(\sigma(x_1, \ldots, x_k), y_1, \ldots, y_m) \to \zeta,$$

where ζ is a tree over (nested) states, output symbols, and the parameters $y_1, \ldots, y_m$ which may occur at leaves. As an example, consider a MAC transducer with these rules:

$$\begin{array}{lcl} q_0(h(x_1)) & \to & q(x_1, a) \\ q(h(x_1), y_1) & \to & q(x_1, q(x_1, y_1)) \\ q(a, y_1) & \to & b(y_1, y_1) \end{array}$$

For a monadic input tree $h(\ldots h(a) \ldots)$ with n occurrences of h, it produces a full binary tree of height 2^n. Thus, MAC transducers can have *double-exponential* size increase (and exponential height increase). The models discussed so far have at most exponential size increase (and at most linear height increase). Formally, a (*total deterministic*) *macro tree transducer* is a tuple $M = (Q, \Sigma, \Delta, q_0, R)$ where Σ, Δ are ranked alphabets of input and output symbols, Q is a ranked alphabet of states with $Q^{(0)} = \emptyset$, $q_0 \in Q^{(1)}$ is the initial state, and R is the set of rules. For every $q \in Q^{(m+1)}$, $m \geq 0$, $\sigma \in \Sigma^{(k)}$, and $k \geq 0$, the set R contains exactly one rule of the form $q(\sigma(x_1, \dots, x_k), y_1, \dots, y_m) \to \zeta$, where ζ is a tree over $Q \cup \Delta \cup \{x_1, \dots, x_k, y_1, \dots, y_m\}$ such that x_i occurs in ζ at a node u if and only if u is a leaf and is the first child of a Q-labeled node (and symbols y_j are of rank zero). We denote ζ by $\mathsf{rhs}(q, \sigma)$. Every state $q \in Q^{(m+1)}$ of M induces a total function $[\![q]\!] : T_\Sigma \times T_\Delta^m \to T_\Delta$. Let $s = \sigma(s_1, \dots, s_k) \in T_\Sigma$ and $t_1, \dots, t_m \in T_\Delta$. Then $[\![q]\!](s, t_1, \dots, t_m) = [\zeta]$ where $\zeta = \mathsf{rhs}(q, \sigma)$ and $[\zeta]$ is defined recursively as follows. If $\zeta = y_j$ then $[\zeta] = t_j$. If $\zeta = d(\zeta_1, \dots, \zeta_\ell)$ with $d \in \Delta^{(\ell)}$, then $[\zeta] = d([\zeta_1], \dots, [\zeta_\ell])$. If $\zeta = q'(x_i, \zeta_1, \dots, \zeta_\ell)$ with $q' \in Q^{(\ell+1)}$ and $i \in [k]$, then $[\zeta] = [\![q']\!](s_i, [\zeta_1], \dots, [\zeta_\ell])$.

A MAC transducer can be equipped with regular look-ahead in a similar way as a top-down tree transducer. A MAC^{R} transducer consists of a MAC transducer $M = (Q, \Sigma, \Delta, q_0, R)$ and look-ahead automaton $A = (P, \Sigma, \delta)$. For every $q \in Q$, $\sigma \in \Sigma^{(k)}$, and $p_1, \dots, p_k \in P$, R contains a rule of the form $q(\sigma(x_1 : p_1, \dots, x_k : p_k), y_1, \dots, y_m) \to \zeta$. Such a rule is applicable to an input tree $\sigma(s_1, \dots, s_k)$ if $\delta(s_i) = p_i$ for $i \in [k]$. We denote ζ by $\mathsf{rhs}(q, \sigma, p_1, \dots, p_k)$.

Origin Semantics of Macro Tree Transducers

We define the origin semantics of a MAC^{R} transducer M for a particular input tree s, using the MAC^{R} transducer M^s and the *decorated version* $\mathsf{dec}(s)$ of s (see Definition 4.15 of [14]). The tree $\mathsf{dec}(s)$ is obtained from s by relabeling every node u by $\langle s[u], u\rangle$. For a state q and input symbol $\langle \sigma, u\rangle$ (of rank k), the MAC^{R} transducer M^s applies the $(q, \sigma, \delta(s/u.1), \dots, \delta(s/u.k))$-rule of M, but with every output symbol d replaced by $\langle d, u\rangle$. The origin of an output node then simply is the second component of the label of that node. Intuitively, when a MAC^{R} transducer applies a rule at input node u and generates output inside of parameter positions, then all these outputs have origin u. Note that such nodes may be duplicated later and appear unboundedly often (at arbitrary positions of the output tree).

Example 31. *We consider an origin translation of a* MAC *transducer that cannot be defined by the previous models. On the other hand, the corresponding origin-less translation can be defined by the previous models: it is merely the identity on trees over* $\Sigma = \{f^{(2)}, a^{(0)}\}$. *Let* M *be the* MAC *transducer with* *the following rules:*

$$\begin{array}{lcl} q_0(a) & \to & a \\ q_0(f(x_1,x_2)) & \to & f(q(x_1,a),q(x_2,a)) \\ q(f(x_1,x_2),y_1) & \to & f(q(x_1,y_1),q(x_2,y_1)) \\ q(a,y_1) & \to & y_1. \end{array}$$

For every tree $s \in T_\Sigma$, $[\![M]\!]_\mathsf{o}(s) = (s,o)$ where $o(u) = u$ if u is an internal node, and $o(u) = \varepsilon$ if u is a leaf. Thus, all leaves of an output tree have the input root node as origin. Clearly, the previous models can not *realize this origin translation: for* MSO *transducers this follows because $[\![M]\!]_\mathsf{o}$ is not of bounded origin. For top-down tree transducer it follows because the origin of a leaf node u is not a descendant of the origin of the parent of u (see the "order-preserving" property, discussed in this Section, after Lemma 32).*

Definability Problem

Deciding whether the translation of a MAC transducer can be defined by a DTOP$^\mathrm{R}$ transducer is a difficult open problem: a MAC transducer can use its parameters in complex ways, but still be definable by a DTOP$^\mathrm{R}$ transducer. In contrast, in the presence of origin we are able to prove decidability. Before we dwell on this proof, let us extend a useful normal form, from the originless semantics of MAC$^\mathrm{R}$ transducers to the semantics with origin. Let M be a MAC$^\mathrm{R}$ transducer. Then M is *nondeleting*, if for every $q \in Q^{(m+1)}$ and $j \in [m]$, y_j occurs in the right-hand side of every q-rule. A q-rule is a rule with q in its left-hand side. Further, M is *nonerasing*, if the right-hand side ζ of every q-rule with $q \in Q^{(2)}$ is not equal to y_1, *i.e.*, $\zeta \neq y_1$.

It is shown in Lemma 6.6 of [13] how to construct for a given MAC$^\mathrm{R}$ transducer M, a MAC$^\mathrm{R}$ transducer M' such that $[\![M']\!] = [\![M]\!]$ and M' is nondeleting. The idea is to determine via look-ahead the parameters of a state that are deleted; in the corresponding rules, the corresponding arguments of the state are removed, and the state is changed to one of smaller rank (and appropriate rules are added). It should be clear that this construction does not alter the generation of output symbols. Thus, the origin semantics of M' and M coincide: $[\![M']\!]_\mathsf{o} = [\![M]\!]_\mathsf{o}$.

In Lemma 7.11 of [13] it is shown how to construct for a given nondeleting MAC$^\mathrm{R}$ transducer M, a MAC$^\mathrm{R}$ transducer M' such that $[\![M']\!] = [\![M]\!]$ and M' is nondeleting and nonerasing. The idea is to determine via look-ahead the states q for which $[\![q]\!](s_i) = y_1$, and to accordingly replace in right-hand sides all occurrences of such states by their first parameter argument. It should be clear that this construction does not influence the generation of output symbols. Thus, $[\![M']\!]_{\circ} = [\![M]\!]_{\circ}$.

Lemma 32. *Let M be a* MAC^R *transducer. An origin-equivalent* MAC^R *transducer M' can be constructed so that M' is nondeleting and nonerasing. Let q be a state of M' of rank $m+1$, $s \in T_\Sigma$, and $\xi = [\![M']\!](s)$. Then (i) y_j occurs in ξ for every $j \in [m]$, and (ii) $\xi[\varepsilon] \in \Delta$.*

Note that Point (ii) in Lemma 32 implies that the origin of $\xi[\varepsilon]$ is necessarily a node of the input tree s.

Let us now turn to the proof that in the presence of origin, DTOP$^{\mathrm{R}}$-definability of a given MAC transducer is decidable. The proof is based on the two notions "order-preserving" and "path-wise bounded origin". Let τ be a functional tree translation with origin. The translation τ is *order-preserving*, if for all nodes u, v in an output tree of τ such that v is a descendant of u, it holds that the origin of v is a descendant of the origin of u. It is well-known that the origin translations of top-down tree transducers are order-preserving, see Lemma 19 of [28]. The origin translation τ is *path-wise bounded origin* if there exists a number β such that for every $(s,(t,o)) \in \tau$, every $u \in V(s)$, and every leaf x in $V(t)$, $|\{v \in V(t,x) \mid o(v) = u\}| \leq \beta$, where $V(t,x)$ is the set of ancestors of x in t. The property says that there are at most β output nodes with the same origin on each path of the output tree. It should be clear that origin translations of top-down tree transducers are path-wise bounded origin: a bound is given by the maximal height of the right-hand side of any rule of the transducer.

Lemma 33. *Let M be a* MAC^R *transducer. It is decidable (i) whether or not $[\![M]\!]_{\circ}$ is order-preserving and (ii) whether or not $[\![M]\!]_{\circ}$ is path-wise bounded origin. If $[\![M]\!]_{\circ}$ is path-wise bounded origin, then a bound can be computed.*

Proof. Let $M = (Q, \Sigma, \Delta, q_0, R)$ with look-ahead automaton $A = (P, \Sigma, \delta)$. (i) The order-preserving property is decided by considering input trees with two marked nodes, marked 1 and 2, such that node 2 is not a descendant of node 1. Let $\Sigma' = \Sigma \cup \Sigma_1 \cup \Sigma_2$ and $\Delta' = \Delta \cup \Delta_1 \cup \Delta_2$ where, for $i \in \{1,2\}$, $\Sigma_i = \{\sigma_i^{(k)} \mid \sigma \in \Sigma^{(k)}\}$ and $\Delta_i = \{d_i^{(k)} \mid d \in \Delta^{(k)}\}$. We construct a tree automaton A_1 that recognizes the trees in $T_{\Sigma'}$ that contain exactly one node u with label in Σ_1 and exactly one node v with label in Σ_2, so that v is not a descendant of u. We now change the MAC$^{\mathrm{R}}$ transducer M so that for an i-marked input node it produces i-marked output nodes: let $M' =$

$(Q, \Sigma', \Delta', q_0, R \cup R')$ with look-ahead automaton $A' = (P, \Sigma', \delta \cup \delta')$. For every transition $\delta(\sigma, p_1, \ldots, p_k) = p$ and $i \in \{1, 2\}$ we let $\delta'(\sigma_i, p_1, \ldots, p_k) = p$. For every rule $q(\sigma(x_1 : p_1, \ldots, x_k : p_k)) \to \zeta$ in R and $i \in \{1, 2\}$, we let the rules

$$q(\sigma_i(x_1 : p_1, \ldots, x_k : p_k)) \quad \to \quad \zeta_i$$

be in R', where ζ_i is obtained from ζ by changing the label of every $d \in \Delta$ into d_i. The transducer M is order-preserving if and only if there does not exist an input tree $s \in L(A_1)$ such that $[\![M]\!](s)$ contains a 2-marked descendant of a 1-marked node. We construct a tree automaton A_2 that recognizes the trees in $T_{\Delta'}$ that contain a 2-marked descendant of a 1-marked node. It now holds that M is order-preserving if and only if $[\![M']\!]^{-1}(L(A_2)) \cap L(A_1) = \emptyset$. The latter is decidable because inverse MAC$^{\mathrm{R}}$ translations effectively preserve regularity (Theorem 7.4(1) of [18]).

For (ii), let A_3 be a tree automaton for all trees in $T_{\Sigma \cup \Sigma_1}$ with exactly one node labeled by a symbol in Σ_1 and all other nodes labeled by symbols in Σ. Let $s \in L(A_3) = L$, $s[u] \in \Sigma_1$, and $t = [\![M']\!](s)$. The Δ_1-nodes in t are exactly those nodes that have origin u. Thus, we need to check if there are only boundedly many such nodes on any path in t. For this, we construct a yTOP$^{\mathrm{R}}$ transducer T, that on input t nondeterministically chooses a root-to-leaf path in t, and outputs all Δ_1-nodes on this path. Let Γ be the alphabet of all symbols in Δ. We define $T = (\{q\}, \Delta \cup \Delta_1, \Gamma, q, U)$. For every $d \in \Delta^{(0)}$ we let the rules $q(d) \to \varepsilon$ and $q(d_1) \to d$ be in U. For all $d \in \Delta^{(k)}$, $k \geq 1$, and $i \in [k]$ let the rules

$$\begin{array}{rcl} q(d(x_1, \ldots, x_k)) & \to & q(x_i) \\ q(d_1(x_1, \ldots, x_k)) & \to & d(q(x_i)) \end{array}$$

be in U. It now holds that $O = [\![T]\!]([\![M']\!](L))$ is finite if and only if M is path-wise bounded origin. Finiteness of O is decidable by Theorem 6.2 of [10], and in case of finiteness the elements can be computed by that theorem, thus giving us a path-wise origin bound. $\square$

We know that the origin translation of a DTOP$^{\mathrm{R}}$ transducer is order-preserving and path-wise bounded origin. In the next lemma we show that if the origin translation $[\![M]\!]_{\mathsf{o}}$ of a MAC$^{\mathrm{R}}$ transducer M is order-preserving and path-wise bounded origin, then $[\![M]\!]_{\mathsf{o}}$ is effectively definable by a DTOP$^{\mathrm{R}}$ transducer. By Lemma 32 we may assume that M is nondeleting and non-erasing. Let $M = (Q, \Sigma, \Delta, q_0, R)$ and let (P, Σ, δ) be its look-ahead automaton. As it turns out, the fact that $[\![M]\!]_{\mathsf{o}}$ is order-preserving and path-wise bounded origin severely restricts the way in which M can use its parameters. A rule $q(\sigma(x_1 : p_1, \dots, x_k : p_k), y_1, \dots, y_m) \to \zeta$ of M is *useful* if there exist $s \in T_\Sigma$ and $u \in V(s)$ such that q occurs in $[\![\widehat{M}]\!](s[u \leftarrow \delta(s/u)])$ and $s[u] = \sigma$; here $\widehat{M}$ is the extension of M, as defined in the proof of Lemma 29. Consider now the right-hand side ζ of a useful rule r of M. Let $q(x_i, \zeta_1, \dots, \zeta_m)$ be a subtree of ζ and let v be a node of a ζ_i, $i \in [m]$. We claim that

(P1) $\zeta_i[v]$ is not in Δ

(P2) $\zeta_i[v]$ is not in $\{x_1, \dots, x_k\} - \{x_i\}$

To prove P1, assume $\zeta_i[v] \in \Delta$. Since r is useful, q occurs in $[\![\widehat{M}]\!](s[u \leftarrow \delta(s/u)])$ for some $s \in T_\Sigma$ and $u \in V(s)$ with $s[u] = \sigma$. Thus, $[\![M]\!](s)$ contains the subtree $[\![q]\!](s/ui)[y_j \leftarrow [\![\zeta_j]\!] \mid j \in [m]]$. By Lemma 32, the origin of the root node of this subtree is of the form $u.i.u'$. Since M is nondeleting, $[\![M]\!](s)$ contains, as a descendant of this root node, a node corresponding to the node $\zeta_i[v]$. This latter node has origin u, which is *not* a descendant of $u.i.u'$. This contradicts the order-preserving property of M. Similarly for P2 we assume that $t_i[v] = x_j$ with $j \neq i$. This implies the existence of a node in $[\![M]\!](s)$ with origin $u.i.u'$ having a descendant with origin $u.j.u''$. Again the order-preserving property is contradicted.

Properties P1 and P2 say that in a parameter tree ζ_j of a state call $q(x_i, \zeta_1, \dots, \zeta_m)$ appearing in the right-hand side of a useful rule, *only* other state calls to x_i may appear and parameters y_j. Thus, ζ_j is of the form

$$q_1(x_i, q_2(x_i, q_3(x_i, \dots), \dots), q'(x_i, \dots), \dots),$$

where parameters may appear at the leaves. Consider now the application of such a rule where an input tree s_i is substituted for x_i. Let σ be the root symbol of s_i, and let us apply the σ-rule of each state appearing. For the same reasons as before, any subtree of state calls in the resulting tree must contain state calls that are all on the same input node. What is the maximal depth of any such subtree of state calls? Clearly, this equals the path-wise origin bound β of $[\![M]\!]_\mathrm{o}$.

The origin-equivalent DTOP$^\mathrm{R}$ transducer T that we construct keeps in its states trees of states of M, of depth at most β. Whenever such a state tree becomes deeper than β, then we know that a non-useful rule has been applied; therefore we may define the corresponding rule of the DTOP$^\mathrm{R}$ transducer as a dummy-rule. Similarly, whenever an output symbol or different input variable x_j appears under a state call on x_i (with $i \neq j$), then we know that a non-useful rule has been applied and define a dummy rule for T.

Example 34. *For $n \geq 1$ we define a* MAC *transducer M_n such that M_n is nondeleting, nonerasing, and $[\![M]\!]_\circ$ is order-preserving and path-wise bounded origin. The transducer takes as input monadic trees over $\Sigma = \{a^{(1)}, e^{(0)}\}$ and produces ternary trees over $\Delta = \{f^{(3)}, e^{(0)}\}$. The rules of M_n are as follows.*

$$\begin{array}{lcll}
q_0(a(x_1)) & \to & q_1(x_1, q_1(x_1, q(x_1))) & \\
q_0(e) & \to & e & \\
q_i(a(x_1), y_1) & \to & q_{i+1}(x_1, q_{i+1}(x_1, y_1)) & \textit{for } i \in [n-1] \\
q_i(e, y_1) & \to & f(y_1, e, y_1) & \textit{for } i \in [n] \\
q_n(a(x_1), y_1) & \to & f(y_1, q_0(x_1), y_1) & \\
q_n(e, y_1) & \to & f(y_1, e, y_1) & \\
q(a(x_1)) & \to & q(x_1) & \\
q(e) & \to & e &
\end{array}$$

As an example, let $n = 10$ and consider the input tree $s = a^{10}(a(s'))$ with $s' \in T_\Sigma$. We have

$$\begin{array}{rl}
& [\![M]\!](s) = [\![q_1]\!](a^9(a(s')), [\![q_1]\!](a^9(a(s')), [\![q]\!](a^9(a(s'))))) \\
= & [\![q_2]\!](a^8(a(s')), [\![q_2]\!](a^8(a(s')), [\![q_2]\!](a^8(a(s')), [\![q_2]\!](a^8(a(s')), [\![q]\!](a^9(a(s'))))))) \\
= & [\![q_{10}]\!](a(s'), [\![q_{10}]\!](a(s'), \ldots, [\![q_{10}]\!](a(s'), [\![q]\!](a(s'))))) = \xi
\end{array}$$

where the tree ξ has $2^n = 1024$ occurrences of $[\![q_{10}]\!]$. We now apply the fifth rule to each of these occurrences. We obtain a tree of the form

$$f(f(\ldots), [\![q]\!](s'), f(\ldots)).$$

If we ignore the second subtree of each f-node, then this is a full (complete) binary tree of height 2^n. If $s' = e$, then the second subtree of each f-node is equal to e, otherwise, these subtrees equal $[\![M]\!](s')$. Thus, each sequence of n-many input a-nodes is translated to such a full tree of height 2^n, where the second subtrees of each internal node contain the translation of the following sequence of input a-nodes.

Lemma 35. *Let M be a nondeleting nonerasing* MAC^R *transducer. If $[\![M]\!]$ is order-preserving and path-wise bounded origin then an origin-equivalent* DTOP^R *transducer can be constructed.*

Proof. Assume that $\tau = [\![M]\!]_\circ$ is path-wise bounded origin with bound β. Let $M = (Q, \Sigma, \Delta, q_0, R)$ and let (P, Σ, δ) be its look-ahead automaton. We construct the DTOP$^{\mathrm{R}}$ transducer $M' = (Q', \Sigma, \Delta, \langle q_0\rangle, R')$ with look-ahead automaton (P, Σ, δ). Let U be the ranked alphabet $\{q^{(m-1)} \mid q \in Q^{(m)}, m \geq 1\}$. The set of states Q' of M' is defined as

$$Q' = \{\langle\zeta\rangle^{(1)} \mid \zeta \in T_U, \mathsf{height}(\zeta) \leq \beta\}.$$

Let $\langle\zeta\rangle \in Q'$ and $\sigma \in \Sigma^{(k)}$. We let the rule $\langle\zeta\rangle(\sigma(x_1 : p_1, \dots, x_k : p_k)) \to t$ be in R', where the right-hand side t is defined as follows. Let ζ' be the tree obtained from ζ by changing every subtree $q(\zeta_1, \dots, \zeta_m)$ into the tree $\mathsf{rhs}_M(q, \sigma, p_1, \dots, p_k)[y_j \leftarrow \zeta_j \mid j \in \{1, \dots, m\}]$.

Let v be a top-most node in ζ' so that $\zeta'[v] \in Q$, *i.e.*, $\zeta'/v = q'(x_i, \zeta'_1, \dots, \zeta'_n)$. We check the properties P1, P2, and the path-wise bounded origin property: if ζ' contains a Δ-node, or, if ζ' contains an occurrence of x_j with $j \neq i$, or if $\mathsf{height}(\zeta') > \beta$, then let $t = e$ where e is an arbitrary symbol in $\Delta^{(0)}$. Otherwise, replace v in ζ' by the tree $\langle\zeta''\rangle(x_i)$ where ζ'' is obtained from ζ' by removing the first subtree of every Q-labeled node (*i.e.*, by removing all subtrees consisting of the single node x_i). If we are able to proceed until all top-most Q-labeled nodes in ζ' are replaced by state calls of M', then the resulting tree ζ' defines the right-hand side t of the rule of M'.

It should be clear that every useful rule of M gives rise to a well-defined (non-dummy) rule of M' and that $[\![M']\!]_\circ(s) = [\![M]\!]_\circ(s)$ for every $s \in T_\Sigma$. □

Theorem 36. *For a given* MAC^R *transducer M, it is decidable whether or not there exists an origin-equivalent* DTOP^R *transducer. If so, then such a* DTOP^R *transducer can be constructed.*

Proof. Using Lemma 33 we check if $[\![M]\!]$ is order-preserving. If it is not, then we return "no", because by Lemma 18 of [28], every translation realized by a top-down tree transducer is order-preserving. Otherwise, we check with Lemma 33 whether $[\![M]\!]$ is path-wise bounded origin. If it is not, then we return "no", because it is easily seen that every translation realized by a top-down tree transducer T is path-wise bounded origin with constant β, where β is the maximal size of the right-hand side of any rule of T. We make M nondeleting and nonerasing, according to Lemma 32. According to Lemma 35 we construct the DTOP$^{\mathrm{R}}$ transducer T such that $[\![T]\!]_\circ = [\![M]\!]_\circ$. □

Example 37. *We apply the construction of Lemma 35 to the* MAC *transducer M_n of Example 34. Note that M_n is path-wise bounded origin with bound*

$k = 2^n$. The DTOP^R *transducer M'_n has states $\langle\zeta\rangle$ of rank one, where ζ is a monadic tree of height $\leq 2^n$ with internal nodes labeled by states $q_1, \ldots, q_n$ and with a leaf labeled q or labeled q_0. The initial state of M'_n is $\langle q_0\rangle$. Let us consider the rules of M'_n, for the case that $n = 10$:*

$$\begin{array}{lll}
\langle q_0\rangle(a(x_1)) & \to & \langle q_1(q_1(q))\rangle(x_1) \\
\langle q_0\rangle(e) & \to & e \\
\langle q_1(q_1(q))\rangle(a(x_1)) & \to & \langle q_2^4(q)\rangle(x_1) \\
\langle q_1(q_1(q))\rangle(e) & \to & f(f(e,e,e),e,f(e,e,e)) = t_1 \\
\langle q_2^4(q)\rangle(a(x_1)) & \to & \langle q_3^8(q)\rangle(x_1) \\
\langle q_2^4(q)\rangle(e) & \to & f(f(t_1,e,t_1),e,f(t_1,e,t_1)) = t_2 \\
 & \vdots & \\
\langle q_{10}^{1024}(q)\rangle(a(x_1)) & \to & f(t_9, \langle q_0\rangle(x_1), t_9) \\
\langle q_{10}^{1024}(q)\rangle(e) & \to & f(t_9, e, t_9) \\
\langle q\rangle(a(x_1)) & \to & \langle q\rangle(x_1) \\
\langle q\rangle(a(x_1)) & \to & e
\end{array}$$

It should be clear that $[\![M'_n]\!]_\circ = [\![M_n]\!]_\circ$. The MAC *transducers M_n form a family of transducers of size $O(n)$, with a size of $O(2^{2^n})$ for the* DTOP^R *transducer constructed in the proof of Lemma 35.*

6. Conclusions and Future Work

The origin semantics of a tree transducer contains the information which input node is responsible for generating any given output node. Thus, for the same tree translation (without origin) there are, in general, infinitely many tree translations with origin, corresponding to the distinct transducers realizing the translation.

The status of decidability for many important problems changes drastically, when moving from conventional to the origin semantics. For instance, equivalence is undecidable for nondeterministic top-down and MSO tree transducers. With origin, however, both of these problems become decidable. The reason for this is that origin semantics adds a rigid synchronicity between output and input nodes. Using this synchronicity we proved decidability of equivalence for deterministic top-down tree-to-string transducers under origin semantics. In the conventional semantics, decidability of equivalence for this class has been a long standing open problem. For nondeterministic top-down tree-to-string transducers, equivalence remains undecidable, even in the presence of origin, and, even for monadic input trees. Such transducers can simulate string transducers by producing output only at the unique input leaf, thus loosing all origin synchronicity. In the future we would like to generalize these results to more powerful models of tree transducers. For instance, for both, attributed [8] and deterministic macro tree transducers [12], it is an open question whether equivalence is decidable. Can equivalence be shown decidable for these transducers, in the presence of origin? Note that in the meantime (during the writing of this paper), equivalence for deterministic macro tree-to-string transducers over one-letter output alphabets has been shown decidable [36]; using that result they prove decidability of equivalence for deterministic top-down tree-to-string transducers thus solving the long standing open problem.

Besides equivalence, we also considered injectivity and query determinacy. Our main result is that with origin semantics, injectivity is decidable for deterministic MSO transducers and for deterministic top-down tree transducers. Both problems are undecidable in the conventional non-origin setting. For deterministic top-down tree-to-string transducers the problem remains undecidable even in the presence of origin. The injectivity results can be generalized to query determinacy: for views with origin and queries without origin (in both above decidable classes), query determinacy is decidable. This poses several interesting questions for future work. Is it possible to reduce the amount of retained origin information, without losing query determinacy? Possible reductions of origins are to represent them by Dewey paths (without input node labels), or to represent them by pre-order numbers of input nodes.

Finally, we also considered two subclass definability questions: is it decidable (1) for a given deterministic top-down tree-to-string transducer with regular look-ahead whether there exists an origin-equivalent MSO tree-to-string transducer, and (2) for a given total deterministic macro tree transducer, whether there exists an origin-equivalent deterministic top-down tree transducer with regular look-ahead. Both problems are decidable, and transducers in the subclass can be constructed if they exist. These problems are not known to be decidable in the origin-less setting. There are many interesting definability questions to be considered in the future. For instance, how about the non-deterministic version of (1) and (2) above? Is it decidable for a (deterministic) top-down tree transducer, under origin semantics, whether it can be realized by a (deterministic) bottom-up tree transducer, and, vice versa? Can it be decided for deterministic top-down tree transducer with regular look-ahead, under origin semantics, whether it can be realized by a deterministic top-down tree transducer (without look-ahead)? The latter question is still open in the origin-less setting, but has recently been shown decidable for a restricted case [16].

Acknowledgements. The authors are grateful to Joost Engelfriet for his remarks for improvements and corrections on a preliminary version of this work. We would like to thank the anonymous referees of this and the ICALP'2015 submission for their valuable comments. This work has been carried out thanks to the support of the ARCHIMEDE Labex (ANR-11-LABX-0033) and the A*MIDEX project (ANR-11-IDEX-0001-02) funded by the "Investissements d'Avenir" French Government program, managed by the French National Research Agency (ANR) and by the PEPS project "Synthesis of Stream Processors" funded by CNRS. It was also supported by ARC project TRANSFORM funded by the French-speaking community of Belgium (Fédération Wallonie-Bruxelles), the FNRS CDR project Flare, and the PHC project VAST (35961QJ) funded by Campus France and WBI. Emmanuel Filiot is an FNRS research associate. The work of Sebastian Maneth has been supported by the VADA Programme Grant from the UK Engineering and Physical Sciences Research Council (EPSRC).

References

[1] Y. Andre and F. Bossut. On the equivalence problem for letter-to-letter top-down tree transducers. *Theor. Comput. Sci.*, 205(1-2):207–229, 1998.

[2] M. Benedikt, J. Engelfriet, and S. Maneth. Determinacy and rewriting of top-down and MSO tree transformations. In *MFCS*, pages 146–158, 2013.

[3] R. Bloem and J. Engelfriet. A comparison of tree transductions defined by monadic second order logic and by attribute grammars. *J. Comput. Syst. Sci.*, 61(1):1–50, 2000.

[4] A. Blumensath and B. Courcelle. On the monadic second-order transduction hierarchy. *Logical Methods in Computer Science*, 6(2), 2010.

[5] M. Bojanczyk. Transducers with origin information. In *ICALP*, pages 26–37, 2014.

[6] F. Braune, N. Seemann, D. Quernheim, and A. Maletti. Shallow local multi-bottom-up tree transducers in statistical machine translation. In *ACL*, pages 811–821, 2013.

[7] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, and M. Tommasi. Tree automata techniques and applications. `http://www.grappa.univ-lille3.fr/tata`, 2007.

[8] B. Courcelle and P. Franchi-Zannettacci. On the equivalence problem for attribute systems. *Information and Control*, 52(3):275–305, 1982.

[9] K. Culik II and J. Karhumäki. The equivalence of finite valued transducers (on HDT0L languages) is decidable. *Theor. Comput. Sci.*, 47(3):71–84, 1986.

[10] F. Drewes and J. Engelfriet. Decidability of the finiteness of ranges of tree transductions. *Inf. Comput.*, 145(1):1–50, 1998.

[11] J. Engelfriet. On tree transducers for partial functions. *Inf. Process. Lett.*, 7(4):170–172, 1978.

[12] J. Engelfriet. Some open questions and recent results on tree transducers and tree languages. In R. Book, editor, *Formal language theory; perspectives and open problems*, pages 241–286. Academic Press, New York, 1980.

[13] J. Engelfriet and S. Maneth. Macro tree transducers, attribute grammars, and MSO definable tree translations. *Inf. Comput.*, 154(1):34–91, 1999.

[14] J. Engelfriet and S. Maneth. Macro tree translations of linear size increase are MSO definable. *SIAM J. Comput.*, 32(4):950–1006, 2003.

[15] J. Engelfriet and S. Maneth. The equivalence problem for deterministic MSO tree transducers is decidable. *Inf. Process. Lett.*, 100(5):206–212, 2006.

[16] J. Engelfriet, S. Maneth, and H. Seidl. Look-ahead removal for total deterministic top-down tree transducers. *Theor. Comput. Sci.*, 616:18–58, 2016.

[17] J. Engelfriet, G. Rozenberg, and G. Slutzki. Tree transducers, L systems, and two-way machines. *J. Comput. Syst. Sci.*, 20(2):150–202, 1980.

[18] J. Engelfriet and H. Vogler. Macro tree transducers. *J. Comput. Syst. Sci.*, 31(1):71–146, 1985.

[19] Z. Ésik. Decidability results concerning tree transducers I. *Acta Cybern.*, 5(1):1–20, 1980.

[20] E. Filiot, S. Maneth, P. Reynier, and J. Talbot. Decision problems of tree transducers with origin. In *ICALP*, pages 209–221, 2015.

[21] Z. Fülöp and P. Gyenizse. On injectivity of deterministic top-down tree transducers. *Inf. Process. Lett.*, 48(4):183–188, 1993.

[22] Z. Fülöp and A. Maletti. Linking theorems for tree transducers. In *TTATT*, 2015.

[23] Z. Fülöp and H. Vogler. *Syntax-Directed Semantics - Formal Models Based on Tree Transducers*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 1998.

[24] T. V. Griffiths. The unsolvability of the equivalence problem for lambda-free nondeterministic generalized machines. *J. ACM*, 15(3):409–413, 1968.

[25] S. Hakuta, S. Maneth, K. Nakano, and H. Iwasaki. XQuery streaming by forest transducers. In *ICDE*, pages 952–963, 2014.

[26] O. H. Ibarra. The unsolvability of the equivalence problem for ϵ-free NGSM's with unary input (output) alphabet and applications. *SIAM J. Comput.*, 7(4):524–532, Nov. 1978.

[27] R. Küsters and T. Wilke. Transducer-based analysis of cryptographic protocols. *Inf. Comput.*, 205(12):1741–1776, 2007.

[28] A. Lemay, S. Maneth, and J. Niehren. A learning algorithm for top-down XML transformations. In *PODS*, pages 285–296, 2010.

[29] A. Maletti. Tree transformations and dependencies. In *MOL*, pages 1–20, 2011.

[30] A. Maletti, J. Graehl, M. Hopkins, and K. Knight. The power of extended top-down tree transducers. *SIAM J. Comput.*, 39(2):410–430, 2009.

[31] S. Maneth. Equivalence problems for tree transducers (survey). In *AFL*, pages 74–93, 2014.

[32] K. Matsuda, K. Inaba, and K. Nakano. Polynomial-time inverse computation for accumulative functions with multiple data traversals. In *PEPM*, pages 5–14, 2012.

[33] T. Milo, D. Suciu, and V. Vianu. Typechecking for XML transformers. *J. Comput. Syst. Sci.*, 66(1):66–97, 2003.

[34] A. Nash, L. Segoufin, and V. Vianu. Views and queries: Determinacy and rewriting. *ACM Trans. Database Syst.*, 35(3), 2010.

[35] W. C. Rounds. Mappings and grammars on trees. *Math. Syst. Th.*, 4(3):257–287, 1970.

[36] H. Seidl, S. Maneth, and G. Kemper. Equivalence of deterministic top-down tree-to-string transducers is decidable. In *FOCS*, pages 943–962, 2015.

[37] J. W. Thatcher. Generalized sequential machine maps. *J. Comput. Syst. Sci.*, 4(4):339–367, 1970.

[38] A. van Deursen, P. Klint, and F. Tip. Origin tracking. *J. Symb. Comput.*, 15(5/6):523–545, 1993.

[39] J. Voigtländer, Z. Hu, K. Matsuda, and M. Wang. Enhancing semantic bidirectionalization via shape bidirectionalizer plug-ins. *J. Funct. Program.*, 23(5):515–551, 2013.

[40] J. Voigtländer and A. Kühnemann. Composition of functions with accumulating parameters. *J. Funct. Program.*, 14(3):317–363, 2004.