

# Deciding whether an Attributed Translation can be realized by a Top-Down Transducer

Sebastian Maneth and Martin Vu

Universität Bremen, Germany  
`{maneth,martin.vu}@uni-bremen.de`

**Abstract.** We prove that for a given partial functional attributed tree transducer with monadic output, it is decidable whether or not an equivalent top-down transducer (with or without look-ahead) exists. We present a procedure that constructs an equivalent top-down transducer (with or without look-ahead) if it exists.

## 1 Introduction

It is well known that *two-way (string) transducers* are strictly more expressive than *one-way (string) transducers*. For instance, a two-way transducer can compute the reverse of an input string, which cannot be achieved by any one-way transducer. For a given (functional) two-way transducer, it is a natural question to ask: Can its translation be realized by a one-way transducer? This question was recently shown to be decidable [8], see also [1]. Decision procedures of this kind have several advantages; for instance, the smaller class of transducers may be more efficient to evaluate (i.e., may use less resources), or the smaller class may enjoy better closure properties than the larger class.

One possible pair of respective counterparts of two-way (string) transducers and one-way (string) transducers in the context of trees are *attributed tree transducers* and *top-down tree transducers*. As the name suggests, states of the latter process an input tree strictly in a top-down fashion while the former can, analogously to two-way transducers, change direction as well. As for their string counterparts, attributed tree transducers are strictly more expressive than top-down tree transducers [10]. Hence, for a (functional) attributed tree transducer, it is a natural question to ask: Can its translation be realized by a deterministic top-down transducer?

In this paper, we address this problem for a subclass of attributed tree transducers. In particular, we consider attributed tree transducer with *monadic output* meaning that all output trees that the transducer produces are *monadic*, i.e., “strings”. We show that the question whether or not for a given attributed tree transducer  $A$  with monadic output an equivalent top-down transducer (with or without look-ahead)  $T$  exists can be reduced to the question whether or not a given two-way transducer can be defined by a one-way transducer.

First, we test whether  $A$  can be equipped with look-ahead so that  $A$  has the *single-path property*, which means that attributes of  $A$  only process a single

input path. Intuitively, this means that given an input tree, attributes of  $A$  only process nodes occurring in a node sequence  $v_1, \dots, v_n$  where  $v_i$  is the parent of  $v_{i+1}$  while equipping  $A$  with look-ahead means that input trees of  $A$  are preprocessed by a deterministic bottom-up relabeling. The single-path property is not surprising given that if  $A$  is equivalent to some top-down tree transducer  $T$  (with look-ahead) then states of  $T$  process the input tree in exactly that fashion. Assume that  $A$  extended with look-ahead satisfies this condition. We then show that  $A$  can essentially be converted into a two-way (string) transducer. We now apply the procedure of [1]. It can be shown that the procedure of [1] yields a one-way (string) transducer if and only if a (nondeterministic) top-down tree transducer  $T$  exists which uses the same look-ahead as  $A$ . We show that once we have obtained a one-way transducer  $T_O$  equivalent to the two-way transducer converted from  $A$ , we can compute  $T$  from  $T_O$ , thus obtaining our result. It is well-known that for a functional top-down tree transducer (with look-ahead) an equivalent deterministic top-down tree transducer with look-ahead can be constructed [5]. Note that for the latter kind of transducer, it can be decided whether or not an equivalent transducer *without* look-ahead exists [12] (this is because transducers with monadic output are linear by default).

We show that the above results are also obtainable for attributed tree transducers with look-around. This model was introduced by Bloem and Engelfriet [2] due to its better closure properties. For this we generalize the result from [5], and show that every functional partial attributed tree transducer (with look-around) is equivalent to a deterministic attributed tree transducer with look-around.

Note that in the presence of origin, it is well known that even for (non-deterministic) macro tree transducers (which are strictly more expressive than attributed tree transducers) it is decidable whether or not an origin-equivalent deterministic top-down tree transducer with look-ahead exists [9]. In the absence of origin, the only definability result for attributed transducers that we are aware of is, that it is decidable for such transducers (and even for macro tree transducers) whether or not they are of linear size increase [7]; and if so an equivalent single-use restricted attributed tree transducer can be constructed (see [6]).

## 2 Attributed Tree Transducers

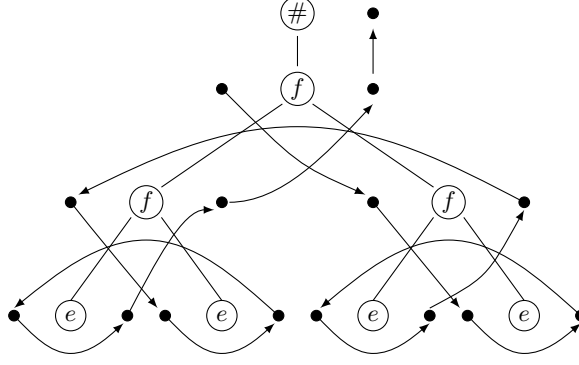
For  $k \in \mathbb{N}$ , we denote by  $[k]$  the set  $\{1, \dots, k\}$ . Let  $\Sigma = \{e_1^{k_1}, \dots, e_n^{k_n}\}$  be a *ranked alphabet*, where  $e_j^{k_j}$  means that the symbol  $e_j$  has *rank*  $k_j$ . By  $\Sigma_k$  we denote the set of all symbols of  $\Sigma$  which have rank  $k$ . The set  $T_\Sigma$  of *trees over*  $\Sigma$  consists of all strings of the form  $a(t_1, \dots, t_k)$ , where  $a \in \Sigma_k$ ,  $k \geq 0$ , and  $t_1, \dots, t_k \in T_\Sigma$ . Instead of  $a()$  we simply write  $a$ . For a tree  $t \in T_\Sigma$ , its nodes are referred to as follows: We denote by  $\epsilon$  the root of  $t$  while  $u.i$  denotes the  $i$ -th child of the node  $u$ . Denote by  $V(t)$  the set of nodes of  $t$ , e.g. for the tree  $t = f(a, f(a, b))$  we have  $V(t) = \{\epsilon, 1, 2, 2.1, 2.2\}$ . For  $v \in V(t)$ ,  $t[v]$  is the label of  $v$ ,  $t/v$  is the subtree of  $t$  rooted at  $v$ , and  $t[v \leftarrow t']$  is obtained from  $t$  by replacing  $t/v$  by  $t'$ . For a set  $\Lambda$  disjoint with  $\Sigma$ , we define  $T_\Sigma[\Lambda]$  as  $T_{\Sigma'}$  where  $\Sigma'_0 = \Sigma_0 \cup \Lambda$  and  $\Sigma'_k = \Sigma_k$  for  $k > 0$ .

A (*partial deterministic*) *attributed tree transducer* (or *att* for short) is a tuple  $A = (S, I, \Sigma, \Delta, a_0, R)$  where  $S$  and  $I$  are disjoint sets of *synthesized attributes* and *inherited attributes*, respectively. The sets  $\Sigma$  and  $\Delta$  are ranked alphabets of *input* and *output symbols*, respectively. We denote by  $a_0 \in S$  the *initial attribute* and define that  $R = (R_\sigma \mid \sigma \in \Sigma \cup \{\#\})$  is a collection of finite sets of rules. We implicitly assume *att*'s to include a unique symbol  $\# \notin \Sigma$  of rank 1, the so-called *root marker*, that only occurs at the root of a tree. Let  $\sigma \in \Sigma \cup \{\#\}$  be of rank  $k \geq 0$ . Let  $\pi$  be a variable for nodes for which we define  $\pi 0 = \pi$ . For every  $a \in S$  the set  $R_\sigma$  contains at most one rule of the form  $a(\pi) \rightarrow t$  and for every  $b \in I$  and  $i \in [k]$ ,  $R_\sigma$  contains at most one rule of the form  $b(\pi i) \rightarrow t'$  where  $t, t' \in T_\Delta[\{a'(\pi i) \mid a' \in S, i \in [k]\} \cup \{b(\pi) \mid b \in I\}]$ . The right-hand sides  $t, t'$  are denoted by  $\text{rhs}_A(\sigma, a(\pi))$  and  $\text{rhs}_A(\sigma, b(\pi i))$ , respectively, if they exist. If  $I = \emptyset$  then we call  $A$  a deterministic top-down tree transducer (or simply a *dt*). In this case, we call  $S$  a set of *states* instead of attributes.

To define the semantics of the *att*  $A$ , we first define the *dependency graph* of  $A$  for the tree  $s \in T_\Sigma$  as  $D_A(s) = (V, E)$  where  $V = \{(a_0, \epsilon)\} \cup ((S \cup I) \times (V(\#(s)) \setminus \{\epsilon\}))$  and  $E = \{((\gamma', uj), (\gamma, ui)) \mid u \in V(s), \gamma'(\pi j) \text{ occurs in } \text{rhs}_A(s[u], \gamma(\pi i)), \text{ with } 0 \leq i, j \text{ and } \gamma, \gamma' \in S \cup I\}$ . If  $D_A(s)$  contains a cycle for some  $s \in T_\Sigma$  then  $A$  is called *circular*. We define  $v0 = v$  for a node  $v$ . For a given tree  $s \in T_\Sigma \cup \{\#(s') \mid s' \in T_\Sigma\}$ , let  $N = \{a_0(\epsilon)\} \cup \{\alpha(v) \mid \alpha \in S \cup I, v \in V(s) \setminus \{\epsilon\}\}$ . For trees  $t, t' \in T_\Delta[N]$ ,  $t \Rightarrow_{A,s} t'$  holds if  $t'$  is obtained from  $t$  by replacing a node labeled by  $\gamma(vi)$ , where  $i = 0$  if  $\gamma \in S$  and  $i > 0$  if  $\gamma \in I$ , by  $\text{rhs}_A(s[v], \gamma(\pi i))[\gamma'(\pi j) \leftarrow \gamma'(vj) \mid \gamma' \in S \cup I, 0 \leq j]$ . If  $A$  is non-circular, then every  $t \in T_\Delta[N]$  has a unique normal form with respect to  $\Rightarrow_{A,s}$  denoted by  $\text{nf}(\Rightarrow_{A,s}, t)$ . The *translation realized by*  $A$ , denoted by  $\tau_A$ , is the set  $\{(s, \text{nf}(\Rightarrow_{A, \#(s)}, a_0(\epsilon))) \in T_\Sigma \times T_\Delta\}$ . As  $A$  is deterministic,  $\tau_A$  is a partial function. Thus we also write  $\tau_A(s) = t$  if  $(s, t) \in \tau_A$  and say that on input  $s$ ,  $A$  produces the tree  $t$ . Denote by  $\text{dom}(A)$  the *domain* of  $A$ , i.e., the set of all  $s \in T_\Sigma$  such that  $(s, t) \in \tau_A$  for some  $t \in T_\Delta$ . Similarly,  $\text{range}(A)$  denotes the *range* of  $A$ , i.e., the set of all  $t \in T_\Delta$  such that for some  $s \in T_\Sigma$ ,  $(s, t) \in \tau_A$ .

*Example 1.* Consider the *att*  $A_1 = (S, I, \Sigma, \Delta, a, R)$  where  $\Sigma = \{f^2, e^0\}$  and  $\Delta = \{g^1, e^0\}$ . Let the set of attributes of  $A$  be given by  $S = \{a\}$  and  $I = \{b\}$ . We define  $R_f = \{a(\pi) \rightarrow a(\pi 1), b(\pi 1) \rightarrow a(\pi 2), b(\pi 2) \rightarrow b(\pi)\}$ . Furthermore we define  $R_e = \{a(\pi) \rightarrow g(b(\pi))\}$  and  $R_\# = \{a(\pi) \rightarrow a(\pi 1), b(\pi 1) \rightarrow e\}$ . The tree transformation realized by  $A_1$  contains all pairs  $(s, t)$  such that if  $s$  has  $n$  leaves, then  $t$  is the tree over  $\Delta$  that contains  $n$  occurrences of the symbol  $g$ . The domain of  $A$  is  $T_\Sigma$  and its range is  $T_\Delta \setminus \{e\}$ . The dependency graph of  $A_1$  for the tree  $f(f(e, e), f(e, e))$  is depicted in Figure 1. As usual, occurrences of inherited and synthesized attributes are placed to the left and right of nodes, respectively. If clear from context, names of attribute occurrences are omitted.  $\square$

We emphasize that we always consider input trees to be trees over  $\Sigma$ . The root marker is a technical necessity. For instance, the translation of  $A_1$  in Example 1 is not possible without it. It is well known that whether or not a given *att*  $A$  is circular can be tested by computing the *is-dependencies* of  $A$  [11]. Informally, the is-dependency of a tree  $s$  depicts the dependencies between inherited



**Fig. 1.** Dependency Graph of the *att* in Example 1 for  $f(f(e, e), f(e, e))$ .

and synthesized attributes at the root of  $s$ . Formally, the is-dependency of  $s$  is the set  $IS_A(s) = \{(b, a) \in I \times S \mid a(1) \text{ is reachable from } b(1) \text{ in } D_A(s)\}$ . As a dependency graph is a directed graph, we say for  $v, v_1, v_2 \in V$ , that  $v_1$  is *reachable* from  $v_2$  if (a)  $v_1 = v_2$  or (b)  $v$  is reachable from  $v_2$  and  $(v, v_1) \in E$  as usual. If  $s = \sigma(s_1, \dots, s_k)$  and the is-dependency of  $s_1, \dots, s_k$  is known, then the is-dependency of  $s$  can be easily computed in a bottom-up fashion using the rules of  $R_\sigma$ . In the the rest of the paper, we only consider non-circular *att*'s.

We define an *attributed tree transducer with look-ahead (or att<sup>R</sup>)* as a pair  $\hat{A} = (B, A)$  where  $B$  is a deterministic bottom-up relabeling which preprocesses input trees for the *att*  $A$ . A (*deterministic*) *bottom-up relabeling*  $B$  is a tuple  $(P, \Sigma, \Sigma', F, R)$  where  $P$  is the set of states,  $\Sigma, \Sigma'$  are ranked alphabets and  $F \subseteq P$  is the set of final states. For  $\sigma \in \Sigma$  and  $p_1, \dots, p_k \in P$ , the set  $R$  contains at most one rule of the form  $\sigma(p_1(x_1), \dots, p_k(x_k)) \rightarrow p(\sigma'(x_1, \dots, x_k))$  where  $p \in P$  and  $\sigma' \in \Sigma'$ . These rules induce a derivation relation  $\Rightarrow_B$  in the obvious way. The translation realized by  $B$  is given by  $\tau_B = \{(s, t) \in T_\Sigma \times T_\Delta \mid s \Rightarrow_B^* p(t), p \in F\}$ . As  $\tau_B$  is a partial function, we also write  $\tau_B(s) = t$  if  $(s, t) \in \tau_B$ . The translation realized by  $\hat{A}$  is given by  $\tau_{\hat{A}} = \{(s, t) \in T_\Sigma \times T_\Delta \mid t = \tau_A(\tau_B(s))\}$ . We write  $\tau_{\hat{A}}(s) = t$  if  $(s, t) \in \tau_{\hat{A}}$  as usual. If  $A$  is a *dt* then  $\hat{A}$  is called a deterministic top-down transducer with look-ahead (or *dt<sup>R</sup>*).

### 3 The Single Path Property

In this section, we show that given an *att* with monadic output, it is decidable whether or not an equivalent *dt<sup>R</sup>* exists. Monadic output means that output symbols are at most of rank 1. First consider the following definition. For an *att*  $A$  with initial attribute  $a_0$ , an input tree  $s$  and  $v \in V(s)$ , we say that on input  $s$ , an attribute  $\alpha$  of  $A$  *processes* the node  $v$  if  $(a_0, \epsilon)$  is reachable from  $(\alpha, 1.v)$  in  $D_A(s)$ . Recall that the dependency graph for  $s$  is defined on the tree  $\#(s)$ . Now, consider an arbitrary *dt<sup>R</sup>*  $\hat{T} = (B', T')$  with monadic output. Then the behavior of  $T'$  is limited in a particular way: Let  $s$  be an input tree and  $s'$  be obtained from  $s$  via the relabeling  $B'$ . On input  $s'$ , the states of  $T'$  only process the nodes

on a single *path* of  $s'$ . A path is a sequence of nodes  $v_1, \dots, v_n$  such that  $v_i$  is the parent node of  $v_{i+1}$ . This property holds obviously as the output is monadic and hence at most one state occurs on the right-hand side of any rule of  $T'$ .

Using this property, we prove our claim. In the following, we fix an *att*  $A = (S, I, \Sigma, \Delta, a_0, R)$  with monadic output and show that if a  $dt^R$   $T = (B', T')$  equivalent to  $A$  exists then we can equip  $A$  with look-ahead such that attributes of  $A$  become limited in the same way as states of  $T'$ : They only process nodes of a single path of the input tree. Our proof makes use of the result of [1]. This result states that for a *functional two-way transducer* it is decidable whether or not an equivalent *one way transducer* exists. Such transducers are essentially attributed transducers and top-down transducers with monadic input and output, respectively. Functional means that the realized translation is a function. We show that  $A$  equipped with look-ahead so that attributes of  $A$  are limited as described earlier can be converted into a two-way transducer  $T_W$ . It can be shown that the procedure of [1] yields a one-way transducer  $T_O$  equivalent to  $T_W$  if and only if  $T$  exists. We then show that we can construct  $T$  from  $T_O$ .

Subsequently, we define the look-ahead with which we equip  $A$ . W.l.o.g. we assume that only right-hand sides of rules in  $R_\#$  are ground (i.e., in  $T_\Delta$ ). Clearly any *att* can be converted so that it conforms to this requirement. Let  $\alpha(\pi) \rightarrow \tau \in R_\sigma$  such that  $\tau$  is ground. First, we remove this rule from  $R_\sigma$ . Then we introduce a new inherited attribute  $\langle \tau \rangle$  and the rules  $\alpha(\pi) \rightarrow \langle \tau \rangle(\pi) \in R_\sigma$ , and  $\langle \tau \rangle(\pi 1) \rightarrow \tau \in R_\#$ . For all  $\sigma' \in \Sigma_k$  and  $j \in [k]$  we define  $\langle \tau \rangle(\pi j) \rightarrow \langle \tau \rangle(\pi) \in R_{\sigma'}$ .

Let  $s \in \text{dom}(A)$  and let  $v \in V(s)$ . We call  $\psi \subseteq I \times S$  the *visiting pair set at  $v$  on input  $s$*  if  $(b, a) \in \psi$  if and only if

- on input  $s$ , the attribute  $a$  processes  $v$  and
- $(b, a) \in \text{IS}_A(s/v)$

Let  $\psi$  be the visiting pair set at  $v$  on input  $s$ . In the following, we denote by  $\Omega_\psi$  the set consisting of all trees  $s' \in T_\Sigma$  such that  $\psi \subseteq \text{IS}_A(s')$ . Thus the set  $\Omega_\psi$  contains all trees  $s'$  such that the visiting pair set at  $v$  on input  $s[v \leftarrow s']$  is also  $\psi$ . If an arbitrary  $a \in S$  exists such that  $(b, a) \in \psi$  for some  $b \in I$  and the range of  $a$  when translating trees in  $\Omega_\psi$  is unbounded, i.e., if the cardinality of  $\{\text{nf}(\Rightarrow_{A, s'}, a(\epsilon)) \mid s' \in \Omega_\psi\}$  is unbounded, then we say that the *variation of  $\Omega_\psi$*  is unbounded. If  $\psi$  is the visiting pair set at  $v$  on input  $s$  and the variation of  $\Omega_\psi$  is unbounded then we also say that the *variation at  $v$  on input  $s$*  is unbounded. The variation plays a key role for proving our claim. In particular, the following property is derived from it: We say that  $A$  has the *single path property* if for all trees  $s \in \text{dom}(A)$  a path  $\rho$  exists such that the variation at  $v \in V(s)$  is bounded whenever  $v$  does not occur in  $\rho$ . The following lemma states that the single path property is a necessary condition for the *att*  $A$  to have an equivalent  $dt^R$ .

**Lemma 1.** *Let  $s \in \text{dom}(A)$  and  $v_1, v_2 \in V(s)$  such that  $v_1$  and  $v_2$  have the same parent node. If a  $dt^R$   $T$  equivalent to  $A$  exists then the variation at either  $v_1$  or  $v_2$  on input  $s$  is bounded.*

*Proof.* For simplicity assume that  $T$  is a *dt*. The proof for  $dt^R$  is obtained similarly. Assume that the variations at both  $v_1$  and  $v_2$  on input  $s$  is unbounded. As

$T$  produces monadic output trees, on input  $s$ , only one of the nodes  $v_1$  and  $v_2$  is processed by a state of  $T$ . W.l.o.g. let  $\psi$  be the visiting pair set at  $v_1$  on input  $s$  and assume that only  $v_2$  is processed by a state of  $T$ . Then for all  $s' \in \Omega_\psi$ ,  $T$  produces the same output on input  $s[v_1 \leftarrow s']$ . However,  $A$  does not produce the same output as the visiting pair set at  $v_1$  on input  $s[v_1 \leftarrow s']$  is  $\psi$  and the variation of  $\Omega_\psi$  is unbounded contradicting the equivalence of  $T$  and  $A$ .  $\square$

*Example 2.* Consider the *att*  $A_2 = (S, I, \Sigma, \Delta, a, R)$  where  $\Sigma = \{f^2, e^0, d^0\}$  and  $\Delta = \{f^1, g^1, e^0, d^0\}$ . The set of attributes are given by  $S = \{a, a_e, a_d\}$  and  $I = \{b_e, b_d, \langle e \rangle, \langle d \rangle\}$ . In addition to  $\langle e \rangle(\pi 1) \rightarrow \langle e \rangle(\pi)$  and  $\langle d \rangle(\pi 1) \rightarrow \langle d \rangle(\pi)$ , the set  $R_f$  contains the rules

$$\begin{array}{llll} a_d(\pi) \rightarrow f(a(\pi 1)) & b_d(\pi 1) \rightarrow a_d(\pi 1) & b_d(\pi 2) \rightarrow b_d(\pi) & a(\pi) \rightarrow a(\pi 2) \\ a_e(\pi) \rightarrow g(a(\pi 1)) & b_e(\pi 1) \rightarrow a_e(\pi 1) & b_e(\pi 2) \rightarrow b_e(\pi) & \end{array}$$

while  $R_\#$  contains in addition to  $\langle e \rangle(\pi 1) \rightarrow e$  and  $\langle d \rangle(\pi 1) \rightarrow d$  the rules

$$a(\pi) \rightarrow a(\pi 1) \quad b_e(\pi 1) \rightarrow a_e(\pi 1) \quad b_d(\pi 1) \rightarrow a_d(\pi 1).$$

Furthermore, we define  $R_e = \{a(\pi) \rightarrow b_e(\pi), a_e(\pi) \rightarrow \langle e \rangle(\pi)\}$  and  $R_d = \{a(\pi) \rightarrow b_d(\pi), a_d(\pi) \rightarrow \langle d \rangle(\pi)\}$ . Let  $s \in T_\Sigma$  and denote by  $n$  the length of the leftmost path of  $s$ . On input  $s$ ,  $A$  outputs the tree  $t$  of height  $n$  such that if  $v \in V(t)$  is not a leaf and the rightmost leaf of the subtree  $s/v$  is labeled by  $e$  then  $t[v] = g$ , otherwise  $t[v] = f$ . If  $v$  is a leaf then  $t[v] = s[v]$ .  $\square$

Clearly, the *att*  $A_2$  in Example 2 is equivalent to a  $dt^R$  and  $A_2$  has the single path property. In particular, it can be verified that the variations of all nodes that do not occur on the left-most path of the input tree are bounded. More precisely, if  $v$  does not occur on the leftmost path of the input tree then its visiting pair set is either  $\psi_e = \{(b_e, a)\}$  or  $\psi_d = \{(b_d, a)\}$ . Thus,  $\Omega_{\psi_e}$  consists of all trees in  $T_\Sigma$  whose rightmost leaf is labeled by  $e$ . For all such trees the attribute  $a$  yields the output  $b_e(e)$ . The case  $\psi_d$  is analogous.

In contrast, consider the *att*  $A_1$  in Example 1. Recall that it translates an input tree  $s$  into a monadic tree  $t$  of height  $n + 1$  if  $s$  has  $n$  leaves. This translation is not realizable by any  $dt^R$ . This is reflected in the fact that the *att* of Example 1 does *not* have the single path property. In particular, consider  $s = f(f(e, e), f(e, e))$ . The visiting pair set at all nodes of  $s$  is  $\psi = \{(a, b)\}$  (cf. Figure 1) and  $\Omega_\psi$  is  $T_\Sigma$ . It can be verified that the variation of  $\Omega_\psi$  is unbounded.

Recall that we aim to equip  $A$  with look-ahead for obtaining the  $att^R \hat{A}$  that has the following property: On input  $s \in \text{dom}(A)$ , attributes of  $A$  only process nodes of a single path of  $s$ . Before we show how to test whether or not  $A$  has the single path property, we describe how to construct  $\hat{A}$ . Denote by  $B$  the bottom-up relabeling of  $\hat{A}$  and let  $A'$  be  $A$  modified to process output trees produced by  $B$ . Let  $s \in \text{dom}(A)$  and let  $s'$  be obtained from  $s$  via the relabeling  $B$ . The idea is that on input  $s'$ , if attributes of  $A'$  process  $v \in V(s')$  then the variation of  $v$  on input  $s$  with respect to  $A$  is unbounded. Note that obviously  $V(s) = V(s')$ . Clearly, if  $A$  has the single path property then attributes of  $A'$  only process nodes of a single path of  $s'$ .

Now the question is how precisely do we construct  $\hat{A}$ ? It can be shown that all  $\psi \subseteq I \times S$  that are visiting pair sets of  $A$  can be computed. Let  $\psi$  be a visiting pair set. If the variation of  $\Omega_\psi$  is bounded, then a minimal integer  $\kappa_\psi$  can be computed such that for all  $a \in S$  such that  $(b, a) \in \psi$  for some  $b \in I$  and for all  $s' \in \Omega_\psi$ ,  $\text{height}(\text{nf}(\Rightarrow_{A, s'}, a(\epsilon))) \leq \kappa_\psi$ . Whether or not the variation of  $\Omega_\psi$  is bounded can be decided as finiteness of ranges of  $\text{att}$ 's is decidable [3]. Thus,  $\kappa = \max\{\kappa_\psi \mid \psi \text{ is a visiting pair set of } A \text{ and the variation of } \Omega_\psi \text{ is bounded}\}$  is computable.

Denote by  $T_\Delta^\kappa[I(\{\epsilon\})]$  the set of all trees in  $T_\Delta[I(\{\epsilon\})]$  of height at most  $\kappa$ . Informally, the idea is that the bottom-up relabeling of the  $\text{att}^R \hat{A}$  precomputes output subtrees of height at most  $\kappa$  that contain the inherited attributes of the root of the current input subtree. Hence, the  $\text{att}$  of  $\hat{A}$  does not need to compute those output subtrees itself; the translation is continued immediately with those output subtrees. Formally, the bottom-up relabeling  $B$  is constructed as follows. The states of  $B$  are sets  $\varrho \subseteq \{(a, \xi) \mid a \in S \text{ and } \xi \in T_\Delta^\kappa[I(\{\epsilon\})]\}$ . The idea is that if  $s \in \text{dom}_B(\varrho)$  and  $(a, \xi) \in \varrho$  then  $\xi = \text{nf}(\Rightarrow_{A, s}, a(\epsilon))$ . Given  $\sigma(s_1, \dots, s_k)$ ,  $B$  relabels  $\sigma$  by  $\sigma_{\varrho_1, \dots, \varrho_k}$  if for  $i \in [k]$ ,  $s_i \in \text{dom}_B(\varrho_i)$ . Note that knowing  $\varrho_1, \dots, \varrho_k$  and the rules in the set  $R_\sigma$  of  $A$ , we can easily compute  $\varrho$  such that  $\sigma(s_1, \dots, s_k) \in \text{dom}_B(\varrho)$  and hence the rules of  $B$ . In particular,  $\varrho$  contains all pairs  $(a, \xi)$  such that  $\xi = \text{nf}(\Rightarrow_{A, \sigma(s_1, \dots, s_k)}, a(\epsilon)) \in T_\Delta^\kappa[I(\{\epsilon\})]$ . Therefore,  $B$  contains the rule  $\sigma(\varrho_1(x_1), \dots, \varrho_k(x_k)) \rightarrow \varrho(\sigma_{\varrho_1, \dots, \varrho_k}(x_1, \dots, x_k))$ .

*Example 3.* Consider the  $\text{att}$   $A_2$  in Example 2. Recall that all nodes that do not occur on the leftmost path of the input tree  $s$  of  $A_2$  have bounded variation. Let  $v$  be such a node. Then the visiting pair set at  $v$  is either  $\psi_e = \{(a, b_e)\}$  or  $\psi_d = \{(a, b_d)\}$ . Assume the former. Then  $\text{nf}(\Rightarrow_{A_2, s/v}, a(\epsilon)) = b_e(\epsilon)$ . If we know beforehand that  $a$  produces  $b_e(\epsilon)$  when translating  $s/v$ , then there is no need to process  $s/v$  with  $a$  anymore. This can be achieved via a bottom-up relabeling  $B_2$  that precomputes all output trees of height at most  $\kappa = \kappa_{\psi_e} = \kappa_{\psi_d} = 1$ . In particular the idea is that if for instance  $v \in V(s)$  is relabeled by  $f_{\{(a, b_d(\epsilon))\}, \{(a, b_e(\epsilon))\}}$  then this means when translating  $s/v.1$  and  $s/v.2$ ,  $a$  produces  $b_d(\epsilon)$  and  $b_e(\epsilon)$ , respectively. The full definition of  $B_2$  is as follows: The states of  $B_2$  are  $\varrho_1 = \{(a_e, \langle e \rangle(\epsilon)), (a, b_e(\epsilon))\}$ ,  $\varrho_2 = \{(a_d, \langle d \rangle(\epsilon)), (a, b_d(\epsilon))\}$ ,  $\varrho_3 = \{(a, b_d(\epsilon))\}$ , and  $\varrho_4 = \{(a, b_e(\epsilon))\}$ . All states are final states. In addition to  $e \rightarrow \varrho_1(e)$  and  $d \rightarrow \varrho_2(d)$ ,  $B_2$  also contains the rules

$$\begin{aligned} f(\varrho(x_1), \varrho_1(x_2)) &\rightarrow \varrho_4(f_{\varrho, \varrho_1}(x_1, x_2)) & f(\varrho(x_1), \varrho_2(x_2)) &\rightarrow \varrho_3(f_{\varrho, \varrho_2}(x_1, x_2)) \\ f(\varrho(x_1), \varrho_3(x_2)) &\rightarrow \varrho_3(f_{\varrho, \varrho_3}(x_1, x_2)) & f(\varrho(x_1), \varrho_4(x_2)) &\rightarrow \varrho_4(f_{\varrho, \varrho_4}(x_1, x_2)), \end{aligned}$$

where  $\varrho \in \{\varrho_1, \dots, \varrho_4\}$ . It is easy to see that using  $B_2$ , attributes of  $A'_2$ , that is,  $A_2$  modified to make use of  $B_2$ , only process nodes of the leftmost path of  $s_2$ .  $\square$

With  $B$ , we obtain  $\hat{A} = (B, A')$  equivalent to  $A$ . Note that  $A$  is modified into  $A'$  such that its input alphabet is the output alphabet of  $B$  and its rules make use of  $B$ . In particular, the rules of  $A'$  for a symbol  $\sigma_{\varrho_1, \dots, \varrho_k}$  are defined as follows. First, we introduce for each state  $\varrho$  of  $B$  an auxiliary symbol  $\langle \varrho \rangle$  of rank 0 to  $A$ . We define that  $a(\pi) \rightarrow t \in R_{\langle \varrho \rangle}$  if  $(a, t[\pi \leftarrow \epsilon]) \in \varrho$ . Denote by  $[\pi \leftarrow v]$

the substitution that substitutes all occurrences of  $\pi$  by the node  $v$ , e.g. for  $t_1 = f(b(\pi))$  and  $t_2 = f(a(\pi 2))$  where  $f$  is a symbol of rank 1,  $a \in S$  and  $b \in I$ , we have  $t_1[\pi \leftarrow v] = f(b(v))$  and  $t_2[\pi \leftarrow v] = f(a(v 2))$ . Let  $t = \text{nf}(\Rightarrow_{A, \sigma(\langle \varrho_1 \rangle, \dots, \langle \varrho_k \rangle)})$ ,  $a(\epsilon) \in T_\Delta[I(\{\epsilon\}) \cup S([k])]$ . Then we define the rule  $a(\pi) \rightarrow t' \in R_{\sigma_{\varrho_1, \dots, \varrho_k}}$  for  $A'$  where  $t'$  is the tree such that  $t'[\pi \leftarrow \epsilon] = t$ . It should be clear that since all output subtrees of height at most  $\kappa$  are precomputed, attributes of  $A'$  only process nodes whose variation with respect to  $A$  are unbounded (cf. Example 3).

In parallel with the above construction of  $\hat{A}$ , we can decide whether or not a given  $\text{att } A$  has the single path property as follows. Let  $s \in \text{dom}(A)$  and let  $s'$  be the tree obtained from  $s$  via the relabeling  $B$ . If nodes  $v_1, v_2 \in V(s')$  with the same parent node exist such that on input  $s'$ , attributes of  $A'$  process both  $v_1$  and  $v_2$  then  $A$  does not have the single path property. Thus, to test whether  $A$  has the single path property, we construct the following  $\text{att}^R \check{A} = (\check{B}, \check{A}')$  from  $\hat{A} = (B, A')$ . Input trees of  $\check{A}$  are trees  $s \in \text{dom}(\hat{A})$  where two nodes  $v_1, v_2$  with the same parent node are annotated by flags  $f_1$  and  $f_2$  respectively. The relabeling  $\check{B}$  essentially behaves like  $B$  ignoring the flags. Likewise, the  $\text{att } \check{A}'$  behaves like  $A'$  with the restriction that output symbols are only produced if an annotated symbol is processed by a synthesized attribute or if a rule in  $R_\#$  is applied. For  $i = 1, 2$  we introduce a special symbol  $g_i$  which is only outputted if the node with the flag  $f_i$  is processed. Hence, we simply need to check whether there is a tree with occurrences of both  $g_1$  and  $g_2$  in the range of  $\check{A}$ . Obviously, the range of  $\check{A}$  is finite. Thus it can be computed.

**Lemma 2.** *It is decidable whether or not  $A$  has the single path property.*

## 4 From Tree to String Transducers and Back

Assume that  $A$  has the single path property. We now convert  $\hat{A} = (B, A')$  into a two-way transducer  $T_W$ . Recall that two-way transducers are essentially attributed tree transducers with monadic input and output<sup>1</sup>. Informally the idea is as follows: Consider a tree  $s \in \text{dom}(A)$  and let  $s'$  be obtained from  $s$  via  $B$ . As on input  $s'$ , attributes of  $A'$  only process nodes occurring on a single path  $\rho$  of  $s$ , the basic idea is to ‘cut off’ all nodes from  $s'$  not occurring in  $\rho$ . This way, we effectively make input trees of  $A'$  monadic. More formally,  $T_W$  is constructed by converting the input alphabet of  $A'$  to symbols of rank 1. Denote by  $\Sigma'$  the set of all output symbols of  $B$ . Accordingly, let  $\Sigma'_k$  with  $k \geq 0$  be the set of all symbols in  $\Sigma'$  of rank  $k$ . Let  $\sigma' \in \Sigma'_k$  with  $k > 0$ . Then the input alphabet  $\Sigma^W$  of  $T_W$  contains the symbols  $\langle \sigma', 1 \rangle, \dots, \langle \sigma', k \rangle$  of rank 1. Furthermore,  $\Sigma^W$  contains all

<sup>1</sup> Note that the two-way transducers in [1] are defined with a *left end marker*  $\vdash$  and a *right end marker*  $\dashv$ . While  $\vdash$  corresponds to the root marker of our tree transducers,  $\dashv$  has no counterpart. Monadic trees can be considered as strings with specific end symbols, i.e. symbols in  $\Sigma_0$ , that only occur at the end of strings. Thus,  $\dashv$  is not required. Two-way transducers can test if exactly one end symbol occurs in the input string and if it is the rightmost symbol. Thus they can simulate tree transducers with monadic input and output.



symbols in  $\Sigma'_0$ . Informally, a symbol  $\langle \sigma', i \rangle$  indicates that the next node is to be interpreted as the  $i$ -th child. Thus trees over  $\Sigma^W$  can be considered prefixes of trees over  $\Sigma'$ , e.g., let  $f \in \Sigma'_2$ ,  $g \in \Sigma'_1$  and  $e \in \Sigma'_0$  then  $\langle f, 2 \rangle \langle f, 1 \rangle \langle f, 1 \rangle e$  encodes the prefix  $f(x_1, f(f(e, x_1), x_1))$  while  $\langle f, 1 \rangle \langle g, 1 \rangle e$  encodes  $f(g(e), x_1)$ . Note that for monadic trees we omit parentheses for better readability. We call  $t_1 \in T_{\Sigma'}[\{x_1\}]$  a prefix of  $t_2 \in T_{\Sigma'}$  if  $t_2$  can be obtained from  $t_1$  by substituting nodes labeled by  $x_1$  by ground trees. The idea is that as attributes of  $A'$  only process nodes occurring on a single path of the input tree, such prefixes are sufficient to simulate  $A'$ .

The attributes of  $T_W$  are the same ones as those of  $A'$ . The rules of  $T_W$  are defined as follows: Firstly the rules for  $\#$  are taken over from  $A'$ . As before, assume that only rules for  $\#$  have ground right-hand sides. Let  $A'$  contains the rule  $a(\pi) \rightarrow t \in R_{\sigma'}$  where  $\sigma' \in \Sigma'_k$  and  $a, \alpha \in S$  and  $\alpha(\pi i)$  occurs in  $t$ . Then  $T_W$  contains the rule  $a(\pi) \rightarrow t[\pi i \leftarrow \pi 1] \in R_{\langle \sigma', i \rangle}$ , where  $[\pi i \leftarrow \pi 1]$  denotes the substitution that substitutes occurrences of  $\pi i$  by  $\pi 1$ . Analogously, if  $A'$  contains the rule  $b(\pi i) \rightarrow t' \in R_{\sigma'}$  where  $b \in I$  and for  $\alpha \in S$ ,  $\alpha(\pi i)$  occurs in  $t'$ , then  $T_W$  contains the rule  $b(\pi 1) \rightarrow t'[\pi i \leftarrow \pi 1] \in R_{\langle \sigma', i \rangle}$ . We remark that as  $A$  has the single path property,  $A'$  will never apply a rule  $b(\pi i) \rightarrow t'$  where  $\alpha(\pi j)$  with  $j \neq i$  occurs in  $t'$ . Thus, we do not need to consider such rules.

For the correctness of subsequent arguments, we require a technical detail: Let  $\tilde{s}$  be an input tree of  $T_W$ . Then we require that an output tree  $s$  of  $B$  exists such that  $\tilde{s}$  encodes a prefix of  $s$ . If such a tree  $s$  exists we say  $\tilde{s}$  *corresponds to*  $s$ . To check whether for a given input tree  $\tilde{s}$  of  $T_W$  an output tree  $s$  of  $B$  exists such that  $\tilde{s}$  corresponds to  $s$ , we proceed as follows. As  $B$  is a relabeling, its range is effectively recognizable. Denote by  $\bar{B}$  the bottom-up automaton recognizing it. Given  $\bar{B}$ , we can construct a bottom-up automaton  $\bar{B}'$  that accepts exactly those trees  $\tilde{s}$  for which an output tree  $s$  of  $B$  exists such that  $\tilde{s}$  corresponds to  $s$ . W.l.o.g. assume that for all states  $l$  of  $\bar{B}$ ,  $\text{dom}_B(l) \neq \emptyset$ . If for  $\sigma' \in \Sigma'_k$ ,  $\sigma'(l_1(x_1), \dots, l_k(x_k)) \rightarrow l(\sigma'(x_1, \dots, x_k))$  is a rule of  $\bar{B}$  then  $\langle \sigma', i \rangle(l_i(x_1)) \rightarrow l(\langle \sigma', i \rangle(x_1))$  is a rule of  $\bar{B}'$ . We define that  $\bar{B}'$  has the same accepting states as  $\bar{B}$ . Using  $\bar{B}'$ , we check whether for a given input tree  $\tilde{s}$  of  $T_W$  an output tree  $s$  of  $B$  exists such that  $\tilde{s}$  corresponds to  $s$  as follows. Before producing any output symbols, on input  $\tilde{s}$ ,  $T_W$  goes to the leaf of  $\tilde{s}$  and simulates  $\bar{B}'$  in a bottom-up fashion while going back to the root. If  $\tilde{s}$  is accepted by  $\bar{B}'$  then  $T_W$  begins to simulate  $A'$ , otherwise no output is produced. Thus the domain of  $T_W$  only consists of trees  $\tilde{s}$  for which an output tree  $s$  of  $B$  exists such that  $\tilde{s}$  corresponds to  $s$ . We remark that  $\bar{B}'$  may be nondeterministic which in turn means that  $T_W$  may be nondeterministic as well, however the translation it realizes is a function. In fact the following holds.

**Lemma 3.** *Consider the  $\text{att}^R \hat{A} = (B, A')$  and the two-way transducer  $T_W$  constructed from  $\hat{A}$ . Let  $\tilde{s}$  be a tree over  $\Sigma^W$ . If on input  $\tilde{s}$ ,  $T_W$  outputs  $t$  then for all  $s \in \text{range}(B)$  such that  $\tilde{s}$  corresponds to  $s$ ,  $A'$  also produces  $t$  on input  $s$ .*

In the following, consider the two-way transducer  $T_W$ . Assume that the procedure of [1] yields a one-way transducer  $T_O$  that is equivalent to  $T_W$ . Given  $T_O$ , we

construct a top-down transducer  $T'$  that produces output trees on the range of  $B$ . In particular,  $T'$  has the same states as  $T_O$ . Furthermore, a rule  $q(\langle \sigma', i \rangle(x_1)) \rightarrow t$  of  $T_O$  where  $\sigma' \in \Sigma'_k$  and  $i \in [k]$  induces the rule  $q(\sigma'(x_1, \dots, x_k)) \rightarrow \hat{t}$  for  $T'$  where  $\hat{t}$  is obtained from  $t$  by substituting occurrences of  $x_1$  by  $x_i$ , e.g., if  $t = f(g(q'(x_1)))$  then  $\hat{t} = f(g(q'(x_i)))$ . Recall that the domain of  $T_W$  only consists of trees  $\tilde{s}$  for which an output tree  $s$  of  $B$  exists such that  $\tilde{s}$  corresponds to  $s$ . As  $T_W$  and  $T_O$  are equivalent, the domain of  $T_O$  also consists of such trees. Hence, by construction, the following holds.

**Lemma 4.** *Consider the top-down transducer  $T'$  constructed from the one-way transducer  $T_O$ . Let  $\tilde{s}$  be a tree over  $\Sigma^W$ . If on input  $\tilde{s}$ ,  $T_O$  outputs  $t$  then for all  $s \in \text{range}(B)$  such that  $\tilde{s}$  corresponds to  $s$ ,  $T'$  also produces  $t$  on input  $s$ .*

With Lemmas 3 and 4, it can be shown that the following holds.

**Lemma 5.** *The top-down transducer  $T'$  and the att  $A'$  are equivalent on the range of  $B$ .*

Therefore it follows that  $\hat{A} = (B, A')$  and  $N = (B, T')$  are equivalent. Consequently,  $A$  and  $N$  are equivalent. We remark that there is still a technical detail left. Recall that our aim is to construct a  $dt^R T$  equivalent to  $A$ . However, the procedure of [1] may yield a nondeterministic  $T_O$ . Thus  $T'$  and hence  $N$  may be nondeterministic (but functional). However, as shown in [5], we can easily compute a  $dt^R$  equivalent to  $N$ .

The arguments above are based on the assumption that the procedure of [1] yields a one-way transducer equivalent to  $T_W$ . Now the question is, does such a one-way transducer always exist if a  $dt^R$  equivalent to  $A$  exists? The answer to this question is indeed affirmative. In particular the following holds.

**Lemma 6.** *Consider the att<sup>R</sup>  $\hat{A} = (B, A')$  equivalent to  $A$ . If a  $dt^R T$  equivalent to  $A$  exists, then a (nondeterministic) top-down transducer with look-ahead  $N = (B, N')$  exists such that  $\hat{A}$  and  $N$  are equivalent.*

*Proof.* (sketch) Recall that given  $\sigma(s_1, \dots, s_k)$ ,  $B$  relabels  $\sigma$  by  $\sigma_{\varrho_1, \dots, \varrho_k}$  if for  $i \in [k]$ ,  $s_i \in \text{dom}_B(\varrho_i)$ . In the following, denote by  $s_\varrho$  a fixed tree in  $\text{dom}_B(\varrho)$ .

Let  $T = (B', T')$ . We sketch the main idea of the proof, i.e., how to simulate  $B'$  using  $B$ . First consider the following property which we call the *substitute-property*. Let  $s \in T_\Sigma$  and  $\hat{s}$  be obtained from  $s$  via the relabeling  $B$ . Let  $v_1$  and  $v_2$  be nodes of  $s$  with the same parent. As  $T$  exists, the single path property holds for  $A$ . Thus on input  $\hat{s}$ ,  $v_1$  or  $v_2$  is not processed by attributes of  $A'$ . Assume that  $v_1$  is not processed and that  $s/v_1 \in \text{dom}_B(\varrho)$ . Then  $\tau_{\hat{A}}(s) = \tau_{\hat{A}}(s[v_1 \leftarrow s_\varrho])$  follows. Informally, this means that  $s/v_1$  can be substituted by  $s_\varrho$  without affecting the output of the translation. By definition,  $\hat{A}$  and  $A$  are equivalent while  $T$  and  $A$  are equivalent by our premise. Thus,  $\tau_T(s) = \tau_T(s[v_1 \leftarrow s_\varrho])$ .

Now we show how  $B'$  is simulated using  $B$ . Let  $\hat{q}$  be a state of  $N'$ . Each such state  $\hat{q}$  is associated with a state  $q$  of  $T'$  and a state  $l$  of  $B'$ . Consider the tree  $\hat{s}$  obtained from  $s$  via  $B$ . Assume that the node  $v$  labeled by  $\sigma_{\varrho_1, \dots, \varrho_k}$  is processed by  $\hat{q}$  in the translation of  $N'$  on input  $\hat{s}$ , i.e.,  $v$  has  $k$  child nodes.

It can be shown that  $\hat{q}$  can compute which of  $v$ 's child nodes is processed by attributes in the translation of  $A'$  on input  $\hat{s}$ . W.l.o.g. let  $v.1$  be that node and let  $s/v.i \in \text{dom}_B(\varrho_i)$  for  $i \in [k]$ . Due to the substitute-property,  $N$  can basically assume that  $s/v.i = s_{\varrho_i}$  for  $i \neq 1$ . For  $i \neq 1$ , let  $s_{\varrho_i} \in \text{dom}_{B'}(l_i)$ . Now  $N'$  guesses a state  $l_1$  for  $s/v.1$  such that  $\sigma(l_1(x_1), \dots, l_k(x_k)) \rightarrow l(\hat{\sigma}(x_1, \dots, x_k))$  is a rule of  $B'$ . The state  $\hat{q}$  then ‘behaves’ as  $q$  would when processing a node labeled by  $\hat{\sigma}$ . It can be guaranteed that  $N'$  verifies its guess.  $\square$

The  $dt^R N = (B, N')$  can be constructed such that on input  $s \in \text{range}(B)$  an attribute of  $A'$  processes the node  $v$  if and only if a state of  $N'$  processes  $v$ . The existence of such a transducer with look-ahead  $N$  implies the existence of a one-way transducer  $T_O$  equivalent to  $T_W$ . In fact,  $T_O$  is obtainable from  $N$  similarly to how  $T_W$  is obtainable from  $A_p$ . Therefore, the procedure of [1] yields a top-down transducer  $T_O$  equivalent to  $T_W$  if and only if  $T$  exists.

## 5 Final Results

The considerations in Sections 3 and 4 yield the following theorem.

**Theorem 1.** *For an att with monadic output, it is decidable whether or not an equivalent  $dt^R$  exists and if so then it can be constructed.*

We show that the result of Theorem 1 is also obtainable for nondeterministic functional attributed tree transducers with look-around. Look-around is a formalism similar to look-ahead. An attributed tree transducers with look-around (or  $att^U$ ) consists of a *top-down relabeling with look-ahead*  $R$  and an *att*  $A$  where output trees are computed as  $\tau_A(\tau_R(s))$  for input trees  $s$ . Before we prove our claim, we prove the following result. Denote by  $nATT^U$  and  $dATT^U$  the classes of translations realizable by nondeterministic and deterministic  $att^U$ 's, respectively. Denote by  $func$  the class all functions.

**Lemma 7.**  $nATT^U \cap func = dATT^U$ .

*Proof.* (sketch) Informally, the basic idea is the same as for functional top-down transducers in [5]. For simplicity, let  $A$  be a functional *att* without look-around. Consider a rule set  $R_\sigma$  of  $A$  where  $\sigma \in \Sigma$ . Let all rules in  $R_\sigma$  with the same left-hand side be ordered. Whenever several such rules are applicable, i.e., utilizing these particular rule leads to the production of a ground tree, the first applicable rule in the given order will be executed. Using look-around, we can test whether or not a rule is applicable.  $\square$

We remark that look-around is required; just look-ahead for instance is not sufficient as  $ATT \cap func \not\subseteq dATT^R$  can be shown.

Due to Lemma 7, it is sufficient to show that for a deterministic  $att^U$  it is decidable whether or not an equivalent  $dt^R$  exists. Roughly speaking, this result is obtained as follows. Let a deterministic  $att^U$   $\hat{A} = (R, A')$ , where  $R$  is a top-down relabeling with look-ahead and  $A'$  is an *att*, be given. To show that a  $dt^R$

equivalent to  $\hat{A}$  exists it is sufficient to show that a  $dt^R$   $T$  exists such that  $A'$  and  $T$  are equivalent on the range of  $R$ . This is because  $R$  is also a  $dt^R$  and  $dt^R$ 's are closed under composition [4]. The  $dt^R$   $T$  can be obtained by slightly modifying the procedure in Section 3.

**Theorem 2.** *For a functional  $att^U$  with monadic output, it is decidable whether or not an equivalent  $dt^R$  exists and if so then it can be constructed.*

Note that by definition  $dt^R$ 's with monadic output are by default *linear*. For a linear  $dt^R$  it is decidable whether or not an equivalent linear  $dt$  exists [12]. If such a  $dt$  exists it can be constructed. Hence, we obtain the following corollary.

**Corollary 1.** *For a functional  $att^U$  with monadic output, it is decidable whether or not an equivalent  $dt$  exists and if so then it can be constructed.*

## 6 Conclusion

We have shown how to decide for a given attributed transducer with look-around but restricted to monadic output, whether or not an equivalent deterministic top-down tree transducers (with or without look-ahead) exists. Clearly we would like to extend this result to non-monadic output trees. The latter seems quite challenging, as it is not clear whether or not the result [1] can be applied in this case. Other questions that remain are the exact complexities of our constructions.

## References

1. Baschenis, F., Gauwin, O., Muscholl, A., Puppis, G.: One-way definability of two-way word transducers. *Log. Methods Comput. Sci.* **14**(4) (2018)
2. Bloem, R., Engelfriet, J.: A comparison of tree transductions defined by monadic second order logic and by attribute grammars. *JCSS* **61**(1), 1–50 (2000)
3. Drewes, F., Engelfriet, J.: Decidability of the finiteness of ranges of tree transductions. *Inf. Comput.* **145**(1), 1–50 (1998)
4. Engelfriet, J.: Top-down tree transducers with regular look-ahead. *Math. Syst. Theory* **10**, 289–303 (1977)
5. Engelfriet, J.: On tree transducers for partial functions. *Inf. Process. Lett.* **7**(4), 170–172 (1978)
6. Engelfriet, J., Maneth, S.: Macro tree transducers, attribute grammars, and MSO definable tree translations. *Inf. Comput.* **154**(1), 34–91 (1999)
7. Engelfriet, J., Maneth, S.: Macro tree translations of linear size increase are MSO definable. *SIAM J. Comput.* **32**(4), 950–1006 (2003)
8. Filiot, E., Gauwin, O., Reynier, P., Servais, F.: From two-way to one-way finite state transducers. In: *LICS*. pp. 468–477 (2013)
9. Filiot, E., Maneth, S., Reynier, P., Talbot, J.: Decision problems of tree transducers with origin. *Inf. Comput.* **261**, 311–335 (2018)
10. Fülöp, Z.: On attributed tree transducers. *Acta Cybern.* **5**(3), 261–279 (1981)
11. Knuth, D.E.: Semantics of context-free languages. *Math. Syst. Theory* **2**(2), 127–145 (1968)
12. Maneth, S., Seidl, H.: When is a bottom-up deterministic tree translation top-down deterministic? In: *ICALP. LIPIcs*, vol. 168, pp. 134:1–134:18 (2020)