How to decide Functionality of Compositions of Top-Down Tree Transducers

Sebastian Maneth¹, Helmut Seidl², and Martin Vu¹

¹ Universität Bremen, Germany
² TU München, Germany

Abstract. We prove that functionality of compositions of top-down tree transducers is decidable by reducing the problem to the functionality of one top-down tree transducer with look-ahead.

1 Introduction

Tree transducers are fundamental devices that were invented in the 1970’s in the context of compilers and mathematical linguistics. Since then they have been applied in a huge variety of contexts such as, e.g., programming languages [13], security [10], or XML databases [9].

The perhaps most basic type of tree transducer is the top-down tree transducer [15,14] (for short transducer). One important decision problem for transducers concerns functionality: given a (nondeterministic) transducer, does it realize a function? This problem was shown to be decidable by Ésik [8] (even in the presence of look-ahead); note that this result also implies the decidability of equivalence of deterministic transducers [8], see also [11].

A natural and fundamental question is to ask whether functionality can also be decided for compositions of transducers. It is well known that compositions of transducers form a proper hierarchy, more precisely: compositions of \( n + 1 \) transducers are strictly more expressive than compositions of \( n \) transducers [6]. Even though transducers are well studied, the question of deciding functionality for compositions of transducers has remained open. In this paper we fill this gap and show that the question can be answered affirmatively.

Deciding functionality for compositions of transducers has several applications. For instance, if an arbitrary composition of (top-down and bottom-up) tree transducers is functional, then an equivalent deterministic transducer with look-ahead can be constructed [5]. Together with our result this implies that it is decidable for such a composition whether or not it is definable by a deterministic transducer with look-ahead; note that the construction of such a single deterministic transducer improves efficiency, because it removes the need of computing intermediate results of the composition. Also other recent definability results can now be generalized to compositions: for instance, given such a composition we can now decide whether or not an equivalent linear transducer or an equivalent homomorphism exists [12] (and if so, construct it).
Let us now discuss the idea of our proof in detail. Initially, we consider a composition \( \tau \) of two transducers \( T_1 \) and \( T_2 \). Given \( \tau \), we construct a ‘candidate’ transducer with look-ahead \( M \) with the property that \( M \) is functional if and only if \( \tau \) is functional. Our construction of \( M \) is an extension of the product construction in [2, p. 195]. The latter constructs a transducer \( N \) (without look-ahead) that is obtained by translating the right-hand sides of the rules of \( T_1 \) by the transducer \( T_2 \). It is well-known that in general, the transducer \( N \) is not equivalent to \( \tau \) [2] and thus \( N \) may not be functional even though \( \tau \) is. This is due to the fact that the transducer \( T_2 \) may

- copy or
- delete input subtrees.

Copying of an input tree means that the tree is translated several times and in general by different states. Deletion means that in a translation rule a particular input subtrees is not translated at all.

Imagine that \( T_2 \) copies and translates an input subtree in two different states \( q_1 \) and \( q_2 \), so that the domains \( D_1 \) and \( D_2 \) of these states differ and moreover, \( T_1 \) nondeterministically produces outputs in the union of \( D_1 \) and \( D_2 \). Now the problem that arises in the product construction of \( N \) is that \( N \) needs to guess the output of \( T_1 \), however, the two states corresponding to \( q_1 \) and \( q_2 \) cannot guarantee that the same guess is used. However, the same guess may be used. This means that \( N \) (seen as a binary relation) is a superset of \( \tau \). To address this problem we show that it suffices to change \( T_1 \) so that it only outputs trees in the intersection of \( D_1 \) and \( D_2 \). Roughly speaking this can be achieved by changing \( T_1 \) so that it runs several tree automata in parallel, in order to carry out the necessary domain checks.

Imagine now a transducer \( T_1 \) that translates two input subtrees in states \( q_1 \) and \( q_2 \), respectively, but has no rules for state \( q_2 \). This means that the translation of \( T_1 \) (and of \( \tau \)) is empty. However, the transducer \( T_2 \) deletes the position of \( q_2 \). This causes the translation of \( N \) to be non-empty. To address this problem we equip \( N \) with look-ahead. The look-ahead checks if the input tree is in the domains of all states of \( T_1 \) translating the current input subtree.

Finally, we are able to generalize the result to arbitrary compositions of transducers \( T_1, \ldots, T_n \). For this, we apply the extended composition described above to the transducers \( T_{n-1} \) and \( T_n \), giving us the transducer with look-ahead \( M \). The look-ahead of \( M \) can be removed and incorporated into the transducer \( T_{n-2} \) using a composition result of [2]. The resulting composition of \( n-1 \) transducers is functional if and only if the original composition is.

The details of all our proofs can be found in the Appendix.

2 Top-Down Tree Transducers

For \( k \in \mathbb{N} \), we denote by \([k]\) the set \( \{1, \ldots, k\} \). Let \( \Sigma = \{e_1^{k_1}, \ldots, e_n^{k_n}\} \) 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 $a(t_1, \ldots, t_k)$, where $a \in \Sigma_k$, $k \geq 0$, and $t_1, \ldots, t_k \in T_\Sigma$. Instead of $a()$ we simply write $a$. We fix the set $X$ of variables as $X = \{x_1, x_2, x_3, \ldots\}$.

Let $B$ be an arbitrary set. We define $T_\Sigma[B] = T_{\Sigma'}$ where $\Sigma'$ is obtained from $\Sigma$ by $\Sigma'_0 = \Sigma_0 \cup B$ while for all $k \geq 0$, $\Sigma'_k = \Sigma_k$. In the following, let $A, B$ be arbitrary sets. We let $A(B) = \{a(b) \mid a \in A, b \in B\}$.

**Definition 1.** A top-down tree transducer $T$ (or transducer for short) is a tuple of the form $T = (Q, \Sigma, \Delta, R, q_0)$ where $Q$ is a finite set of states, $\Sigma$ and $\Delta$ are the input and output ranked alphabets, respectively, disjoint with $Q$, $R$ is a finite set of rules, and $q_0 \in Q$ is the initial state. The rules contained in $R$ are of the form $q(a(x_1, \ldots, x_k)) \rightarrow t$, where $q \in Q$, $a \in \Sigma_k$, $k \geq 0$ and $t$ is a tree in $T_\Delta[Q(X)]$.

If $q(a(x_1, \ldots, x_k)) \rightarrow t \in R$ then we call $t$ a right-hand side of $q$ and $a$. The rules of $R$ are used as rewrite rules in the natural way, as illustrated by the following example.

**Example 1.** Consider the transducer $T = (\{q_0, q\}, \Sigma, \Delta, R, q_0)$ where $\Sigma_0 = \{e\}$, $\Sigma_1 = \{a\}$, $\Delta_0 = \{e\}$, $\Delta_1 = \{a\}$ and $\Delta_2 = \{f\}$ and $R$ consists the following rules (numbered 1 to 4):

1. $q_0(a(x_1)) \rightarrow f(q(x_1), q_0(x_1))$
2. $q_0(e) \rightarrow e$
3. $q(a(x_1)) \rightarrow a(f(q_1))$
4. $q(e) \rightarrow e$

On input $a(a(e))$, the transducer $T$ produces the output tree $f(a(e), f(e, e))$ as follows

Informally, when processing a tree $s \in T_\Sigma$, the transducer $T$ produces a tree $t$ in which all proper subtrees of $s$ occur as disjoint subtrees of $t$, ‘ordered’ by size. As the reader may realize, given an input tree $s$ of size $n$, the transducer $T$ produces an output tree that is of size $(n^2 + n)/2$. Hence, this translation has quadratic size increase, i.e., the size of the output tree is a most quadratic in size of the input tree. Note that transducers can have polynomial or exponential size increase [1].
Let $s \in T_{\Sigma}$. Then $T(s)$ contains all trees in $T_{\Delta}$ obtainable from $q_0(s)$ by applying rules of $T$.

Clearly, $T$ defines a binary relation over $T_{\Sigma}$ and $T_{\Delta}$. In the following, we denote by $\mathcal{R}(T)$ the binary relation that the transducer $T$ defines. We say that the transducer $T$ is functional if the relation $\mathcal{R}(T)$ is a function. Let $q$ be a state of $T$. We denote by $\text{dom}(q)$ the domain of $q$, i.e., the set of all trees $s \in T_{\Sigma}$ for which some tree $t \in T_{\Delta}$ is obtainable from $q(s)$ by applying rules of $T$. We define the domain of $T$ by $\text{dom}(T) = \text{dom}(q_0)$. For instance in Example 1, $\text{dom}(T) = T_{\Sigma}$. However, if we remove the rule 1 for instance then the domain of $T$ shrinks to the set $\{e\}$. We define $\text{dom}(q)$, the domain of a state $q$ of $T$, analogously.

A transducer $T = (Q, \Sigma, \Delta, R, q)$ is a top-down tree automaton (for short automaton) if $\Sigma = \Delta$ and all rules of $T$ are of the form $q(a(x_1, \ldots, x_k)) \rightarrow a(q_1(x_1), \ldots, q_k(x_k))$ where $a \in \Sigma_k$, $k \geq 0$.

Let $T_1$ and $T_2$ be transducers. As $\mathcal{R}(T_1)$ and $\mathcal{R}(T_2)$ are relations, they can be composed. Hence,

$$\mathcal{R}(T_1) \circ \mathcal{R}(T_2) = \{(s, u) \mid \text{for some } t, (s, t) \in \mathcal{R}(T_1) \text{ and } (t, u) \in \mathcal{R}(T_2)\}.$$  

If the output alphabet of $T_1$ and the input alphabet of $T_2$ coincide then the transducers $T_1$ and $T_2$ can be composed as well. The composition $T_1 \circ T_2$ of the transducers $T_1$ and $T_2$ defines a tree translation as follows. On input $s$, the tree $s$ is first translated by $T_1$. Afterwards, the tree produced by $T_1$ is translated by $T_2$ which yields the output tree. Clearly, $T_1 \circ T_2$ computes the relation $\mathcal{R}(T_1) \circ \mathcal{R}(T_2)$. We say that the composition $T_1 \circ T_2$ is functional if the relation $\mathcal{R}(T_1) \circ \mathcal{R}(T_2)$ is a function.

## 3 Functionality of Two-Fold Compositions

In this section we show that for a composition $\tau$ of two transducers, a transducer $M$ with look-ahead can be constructed such that $M$ is functional if and only if $\tau$ is functional. Before formally introducing the construction for $M$ and proving its correctness, we explain how to solve the challenges described in Section 1, i.e., we show how to handle copying and deleting rules. In the following, we call the product construction in [2, p. 195] simply the p-construction.

To see how precisely we handle copying rules, consider the transducers $T_1$ and $T_2$. Let the transducer $T_1$ consist of the rules

$$q_1(a(x_i)) \rightarrow b(q_1(x_i)) \quad q_1(e) \rightarrow e_i \mid i = 1, 2, 3$$

while transducer $T_2$ consist of the rules

$$q_2(b(x_i)) \rightarrow f(q_2'(x_1), q_2'(x_1)) \quad q_2'(e_j) \rightarrow e \mid j = 1, 2$$
$$q_2'(e_3) \rightarrow e' \quad q_2'(e_j) \rightarrow e \mid j = 1, 2.$$  

The composition $\tau = T_1 \circ T_2$ defines a relation that only contains a single pair: $\tau$ only translates the tree $a(e)$ into $f(e, e)$. Therefore, $\tau$ is functional. For $T_1$ and $T_2$, the p-construction yields the transducer $N$ with the rules
\[(q_1, q_2)(a(x_1)) \rightarrow f((q_1, q_2')(x_1), (q_1, q_2'')(x_1))\]
\[(q_1, q_2')(e) \rightarrow e'\]
\[(q_1, q_2'')(e) \rightarrow e.\]

On input \(a(e)\), the transducer \(N\) can produce either \(f(e, e)\) or \(f(e, e')\). Therefore, \(N\) and \(\tau\) are clearly not equivalent. Furthermore, the transducer \(N\) is obviously not functional even though the composition \(\tau\) is.

In order to obtain a better understanding of why this phenomenon occurs, we analyze the behavior of \(N\) and \(\tau\) on input \(a(e)\) in the following.

In the translation of \(\tau\), the states \(q_2'\) and \(q_2''\) process the same tree produced by \(q_1\) on input \(e\) due to the copying rule \(q_2(b(x_1)) \rightarrow f(q_2'(x_1), q_2''(x_1))\). Furthermore, \(q_2'\) and \(q_2''\) process a tree in \(\text{dom}(q_2') \cap \text{dom}(q_2'')\). More precisely, \(q_2'\) and \(q_2''\) both process either \(e_1\) or \(e_2\).

In the translation of \(N\) on the other hand, due to the rule \((q_1, q_2)(a(x_1)) \rightarrow f((q_1, q_2')(x_1), (q_1, q_2'')(x_1))\), the states \((q_1, q_2')\) and \((q_1, q_2'')\) process \(e\) by ‘guessing independently’ from each other what \(q_1\) might have produced on input \(e\). In particular, the problem is that \((q_1, q_2')\) can apply the rule \((q_1, q_2')(e) \rightarrow e'\) which eventually leads to the production of \(f(e, e')\). Applying this rule means that \((q_1, q_2')\) guesses that \(e_3\) is produced by \(q_1\). While this guess is valid, i.e., \(e_3\) is producible by \(q_1\) on input \(e\), quite clearly \(e_3 \notin \text{dom}(q_2')\).

In general, guesses performed by states of \(N\) cannot be ‘synchronized’, i.e., we cannot guarantee that states guess the same tree. Our solution to fix this issue is to restrict \((q_1, q_2')\) and \((q_1, q_2'')\) such that either state is only allowed to guess trees in \(\text{dom}(q_2') \cap \text{dom}(q_2'')\). To understand why this approach works in general consider the following example.

**Example 2.** Let \(T_1\) and \(T_2\) be arbitrary transducers. Let \(\tau = T_1 \circ T_2\) be functional. Let \(T_1\) on input \(s\) produce either \(b(t_1)\) or \(b(t_2)\). Let \(T_2\) contain the rule
\[q_2(b(x_1)) \rightarrow f(q_2^1(x_1), q_2^2(x_1))\]
where \(q_2\) is the initial state of \(T_2\). The application of this rule effectively means that the states \(q_2^1\) and \(q_2^2\) process the same subtree produced by \(T_1\). Let \(t_1, t_2 \in \text{dom}(q_2^1) \cap \text{dom}(q_2^2)\). Informally speaking, it does not matter whether the state \(q_2^1\) processes \(t_1\) or \(t_2\); for either input \(q_2^1\) produces the same output tree \(r\) and nothing else, otherwise, the functionality of \(\tau\) is contradicted. The same holds for \(q_2^2\).

Informally, Example 2 suggests that if \((q_1, q_2')\) and \((q_1, q_2'')\) only guess trees in \(\text{dom}(q_2') \cap \text{dom}(q_2'')\), then it does not matter which tree exactly those states guess if the composition is functional. The final result in either case is the same. Quite clearly this is the case in our example. (In effect, \(q_2''\) is forbidden to guess \(e_3\).) Thus, restricting \((q_1, q_2')\) and \((q_1, q_2'')\) basically achieves the same result as synchronizing their guesses if the composition is functional.

Now the question is how exactly do we restrict the states of \(N\)? Consider the states \((q_1, q_2')\) and \((q_1, q_2'')\) of \(N\) in our example. The trick is to restrict \(q_1\) such that \(q_1\) can only produce trees in \(\text{dom}(q_2') \cap \text{dom}(q_2'')\). Thus any guess is guaranteed to be in \(\text{dom}(q_2') \cap \text{dom}(q_2'')\). In order to restrict which output trees \(T_1\) can produce, we compose \(T_1\) with the domain automaton of \(T_2\).
For an arbitrary transducer $T = (Q, \Sigma, \Delta, R, q)$, the domain automaton $A$ of $T$ is constructed analogous to the automaton in [4, Theorem 3.1]. The set of states of $A$ is the power set of $Q$ where $\{q\}$ is the initial state of $A$. The idea is that if in a translation of $T$ on input $s$, the states $q_1, \ldots, q_n$ process the node $v$ of $s$ then $\{q_1, \ldots, q_n\}$ processes the node $v$ of $s$ in a computation of $A$. The rules of $A$ are thus defined as follows.

Let $S = \{q_1, \ldots, q_n\}$, $n > 0$, and $a \in \Sigma_k$. In the following, we denote by $\text{rhs}_T(q_j, a)$, where $j \in [n]$, the set of all right-hand sides of $q_j$ and $a$. For all non-empty subsets $I_1 \subseteq \text{rhs}_T(q_1, a), \ldots, I_n \subseteq \text{rhs}_T(q_n, a)$, we define a rule

$$S(a(x_1, \ldots, x_k)) \rightarrow a(S_1(x_1), \ldots, S_k(x_k))$$

where for $i \in [k]$, $S_i$ is defined as the set $\bigcup_{j=1}^{n} I_j(x_i)$. We denote by $I_j(x_i)$ the set of all states $q'$ such that $q'(x_i)$ occurs in some tree $\gamma$ in $I_j$; e.g., for

$$I_j = \{a(q(x_1), q'(x_2)), a(q_1(x_1), q_2(x_2)), q_3(x_1))\},$$

we have $I_j(x_1) = \{q, q_1, q_3\}$ and $I_j(x_2) = \{q', q_2\}$. We define that the state $\emptyset$ of $A$ realizes the identity. Hence, the rules for the state $\emptyset$ are defined in the obvious way.

We now explain why subsets $I_j$ of right-hand sides are used for the construction of rules of $A$. Recall that the idea is that if in a translation of $T$ on input $s$, the states $q_1, \ldots, q_n$ process the node $v$ of $s$ then $\{q_1, \ldots, q_n\}$ processes the node $v$ of $s$ in a computation of $A$. Due to copying rules, multiple instances of a state $q_1$ may access $v$. Two instance of $q_1$ may process $v$ in different manners. This necessitates the use of subsets $I_j$ of right-hand sides. For a better understanding, consider the following example.

**Example 3.** Let $T = (\{q_0, q\}, \Sigma, \Delta, R, q_0)$ where $\Sigma_0 = \Delta_0 = \{e\}$, $\Sigma_1 = \Delta_1 = \{a\}$ and $\Sigma_2 = \Delta_2 = \{f\}$. The set $R$ contains the following rules:

- $q_0(a(x_1)) \rightarrow f(q_0(x_1), q_0(x_1))$
- $q_0(f(x_1, x_2)) \rightarrow q_0(x_1)$
- $q_0(f(x_1, x_2)) \rightarrow f(q(x_1), q(x_2))$
- $q_0(e) \rightarrow e$.

Consider the input tree $s = a(f(e, e))$. Clearly, on input $s$, the tree $f(e, f(e', e'))$ is producible by $T$. In this translation, two instances of the state $q_0$ process the subtree $f(e, e)$ of $s$, however the instances of $q_0$ do not process $f(e, e)$ in the same way. The first instance of $q_0$ produces $e$ on input $f(e, e)$ while the second instance produces $f(e', e')$. These translations mean that the states $q_0$ and $q$ process the leastmost $e$ of $s$.

Consider the domain automaton $A$ of $T$. By definition, $A$ contains the rule

$$\{q_0\}(a(x_1)) \rightarrow a(\{q_0\}(x_1))$$

which is obtained from the right-hand side of the rule

$$q_0(a(x_1)) \rightarrow f(q_0(x_1), q_0(x_1))$$

of $T$. To simulate that the states $q_0$ and $q$ process the leastmost $e$ of $s$ in the translation from $s$ to $f(e, f(e', e'))$, we clearly require the rule

$$\{q_0\}(f(x_1, x_2)) \rightarrow f(\{q_0\}(x_1), \{q\}(x_2))$$

obtained from the right-hand sides of the rules $q_0(f(x_1, x_2) \rightarrow q_0(x_1)$ and $q_0(f(x_1, x_2) \rightarrow f(q(x_1), q(x_2))$ of $T$. 


For completeness, we list the remaining rules of $A$. The automaton $A$ also contains the rules

\[
\begin{align*}
\{q_0\} \{f(x_1, x_2)\} &\rightarrow f(\{q\}(x_1), \{q\}(x_2)) & \{q\} (a(x_1)) &\rightarrow a(\emptyset(x_1)) \\
\{q_0\} \{f(x_1, x_2)\} &\rightarrow f(\{q_0\}(x_1), \emptyset(x_2)) & \{q\} (f(x_1, x_2)) &\rightarrow f(\emptyset(x_1), \emptyset(x_2)) \\
\{q_0\} (e) &\rightarrow e & \{q\} (e) &\rightarrow e. \\
\emptyset (f(x_1, x_2)) &\rightarrow f(\emptyset(x_1), \emptyset(x_2)) & \emptyset (a(x_1)) &\rightarrow a(\emptyset(x_1)) \\
\emptyset (e) &\rightarrow e.
\end{align*}
\]

For the rules of the state $\{q_0, q\}$ consider the following. The right-hand sides of rules of $\{q_0, q\}$ are identical to the right-hand sides of rules of $\{q_0\}$, i.e., the rules for $\{q_0, q\}$ are obtained by substituting $\{q_0\}$ on the left-hand-side of rules of $A$ by $\{q_0, q\}$.

The automaton $A$ has the following property.

**Lemma 1.** Let $S \neq \emptyset$ be a state of $A$. Then $s \in \text{dom}(S)$ if and only if $s \in \bigcap_{q \in S} \text{dom}(q)$.

Obviously, Lemma 1 implies that $A$ recognizes the domain of $T$.

Using the domain automaton $A$ of $T_2$, we transform $T_1$ into the transducer $\hat{T}_1$.

Formally, the transducer $T_1$ is obtained from $T_1$ and $A$ using the p-construction.

In our example, the transducer $\hat{T}_1$ obtained from $T_1$ and $T_2$ includes the following rules

\[
\begin{align*}
(q_1, \{q_2\}) (a(x_1)) &\rightarrow b((q_1, \{q_2\}))(x_1) \\
(q_1, \{q_2, q''\}) (e) &\rightarrow e_j
\end{align*}
\]

where $j = 1, 2$. The state $(q_1, \{q_2\})$ is the initial state of $\hat{T}_1$. Informally, the idea is that in a translation of $\hat{\tau} = T_1 \circ T_2$, a tree produced by a state $(q, S)$ of $T_1$ is only processed by states in $S$. The following result complements this idea.

**Lemma 2.** If the state $(q, S)$ of $\hat{T}_1$ produces the tree $t$ and $S \neq \emptyset$ then $t \in \bigcap_{q_2 \in S} \text{dom}(q_2)$.

We remark that if a state of the form $(q, \emptyset)$ occurs then it means that in a translation of $\hat{\tau}$, no state of $T_2$ will process a tree produced by $(q, \emptyset)$. Note that as $A$ is nondeleting and linear, $T_1 \circ A$ defines the same relation as $T_1 \circ A$ [2] Th. 1].

Informally, the transducer $\hat{T}_1$ is a restriction of the transducer $\hat{T}_1$ such that $\text{range}(\hat{T}_1) = \text{range}(T_1) \cap \text{dom}(T_2)$. Therefore, the following holds.

**Lemma 3.** $\mathcal{R}(T_1) \circ \mathcal{R}(T_2) = \mathcal{R}(\hat{T}_1) \circ \mathcal{R}(T_2)$.

Due to Lemma 3 we focus on $\hat{T}_1$ instead of $T_1$ in the following.

Consider the transducer $\hat{N}$ obtained from $\hat{T}_1$ and $T_2$ using the p-construction.

By construction, the states of $\hat{N}$ are of the form $((q, S), q')$ where $(q, S)$ is a state of $\hat{T}_1$ and $q'$ is a state of $T_2$. In the following, we write $(q, S, q')$ instead for better readability. Informally, the state $(q, S, q')$ implies that in a translation of $\hat{\tau}$ the state $q'$ is supposed to process a tree produced by $(q, S)$. Because trees produced by $(q, S)$ are only supposed to be processed by states in $S$, we only consider states $(q, S, q')$ where $q' \in S$. For $\hat{T}_1$ and $T_2$, we obtain the transducer $\hat{N}$ with the following rules
In the folowing, we briefly explain our idea. In a translation of $\hat{N}$ on input $a(e)$, the subtree $e$ is processed by $(q_1, S, q'_2)$ and $(q_1, S, q''_2)$. Note that in a translation of $\hat{\tau}$ the states $q'_2$ and $q''_2$ would process the same tree produced by $(q_1, S)$ on input $e$. Consider the state $(q_1, S, q'_2)$. If $(q_1, S, q'_2)$, when reading $e$, makes a valid guess, i.e., $(q_1, S, q''_2)$ guesses a tree $t$ that is producible by $(q_1, S)$ on input $e$, then $t \in \text{dom}(q'_2)$ by construction of $\hat{T}_1$. Due to previous considerations (cf. Example 2), it is thus sufficient to ensure that all guesses of states of $\hat{N}$ are valid. While obviously in the case of $\hat{N}$, all guesses are indeed valid, guesses of transducers obtained from the p-construction are in general not always valid; in particular if deleting rules are involved.

To be more specific, consider the following transducers $T'_1$ and $T'_2$. Let $T'_1$ contain the rules

$$q_1(a(x_1, x_2)) \rightarrow b(q'_1(x_1), q''_1(x_2), q'''_1(x_2)) \quad q'_1(e) \rightarrow e$$

where $\text{dom}(q''_1)$ consists of all trees whose left-most leaf is labeled by $e$ while $\text{dom}(q'''_1)$ consists of all trees whose left-most leaf is labeled by $c$. Let $T'_2$ contain the rules

$$q_2(b(x_1, x_2, x_3)) \rightarrow q_2(x_1) \quad q_2(e) \rightarrow e_j \mid j = 1, 2.$$ 

As the translation of $T'_1$ is empty, obviously the translation of $\tau' = T'_1 \circ T'_2$ is empty as well. Thus, $\tau'$ is functional. However, the p-construction yields the transducer $N'$ with the rules

$$(q_1, q_2)(a(x_1, x_2)) \rightarrow (q'_1, q_2)(x_1) \quad (q'_1, q_2)(e) \rightarrow e_j \mid j = 1, 2$$

Even though $\tau' = T'_1 \circ T'_2$ is functional, the transducer $N'$ is not. More precisely, on input $a(e, s)$, where $s$ is an arbitrary tree, $N'$ can produce either $e_1$ or $e_2$ while $\tau'$ would produce nothing. The reason is that in the translation of $N'$, the tree $a(e, s)$ is processed by the state $(q_1, q_2)$ by applying the deleting rule $\eta = (q_1, q_2)(a(x_1, x_2)) \rightarrow (q'_1, q_2)(x_1)$. Applying $\eta$ means that $(q_1, q_2)$ guesses that on input $a(e, s)$, the state $q_1$ produces a tree of the form $b(t_1, t_2, t_3)$ by applying the rule $q_1(a(x_1, x_2)) \rightarrow b(q'_1(x_1), q''_1(x_2), q'''_1(x_2))$ of $T_1$. However, this guess is not valid, i.e., $q_1$ does not produce such a tree on input $a(e, s)$, as by definition $s \notin \text{dom}(q''_1)$ or $s \notin \text{dom}(q'''_1)$. The issue is that $N'$ itself cannot verify the validity of this guess because, due to the deleting rule $\eta$, $N'$ does not read $s$.

As the reader might have guessed our idea is that the validity of each guess is verified using look-ahead. First, we need to define look-ahead.

A transducer with look-ahead (or la-transducer) $M'$ is a transducer that is equipped with an automaton called the la-automaton. Formally, $M'$ is a tuple $M' = (Q, \Sigma, \Delta, R, q, B)$ where $Q$, $\Sigma$, $\Delta$ and $q$ are defined as for transducers and
$B$ is the la-automaton. The rules of $R$ are of the form $q(a(x_1 : l_1, \ldots, x_k : l_k)) \to t$ where $i \in [k]$, $l_i$ is a state of $B$. Consider the input $s$. The la-transducer $M'$ processes $s$ in two phases: First each input node of $s$ is annotated by the states of $B$ at its children, i.e., an input node $v$ labeled by $a \in \Sigma_k$ is relabeled by $(a, l_1, \ldots, l_k)$ if $B$ arrives in the state $l_i$ when processing the $i$-th subtree of $v$. Relabeling the nodes $s$ provides $M'$ with additional information about the subtrees of $s$, e.g., if the node $v$ is relabeled by $(a, l_1, \ldots, l_k)$ then the $i$-th subtree of $v$ is a tree in $\text{dom}(l_i)$. The relabeled tree is then processed by $M'$. To this end a rule $q(a(x_1 : l_1, \ldots, x_k : l_k)) \to t$ is interpreted as $q((a, l_1, \ldots, l_k)(x_1, \ldots, x_k)) \to t$.

In our example, the idea is to equip $N'$ with an la-automaton to verify the validity of guesses. In particular, the la-automaton is the domain automaton $A'$ of $T_1'$. Recall that a state of $A'$ is a set consisting of states of $T_1'$. To process relabeled trees the rules of $N'$ are as follows

$$(q_1, q_2)(a(x_1 : \{q_1'\}, x_2 : \{q_1'' : q_1'''\})) \to (q_1', q_2')(x_1) \quad (q_1', q_2')(e) \to e_j \mid j = 1, 2$$

Consider the tree $a(e, s)$, where $s$ is an arbitrary tree. The idea is that if the root of $a(e, s)$ is relabeled by $(a, \{q_1'\}, \{q_1'', q_1'''\})$, then due to Lemma \[ e \in \text{dom}(q_1') \]

and $s \in \text{dom}(q_1''') \cap \text{dom}(q_1''')$ and thus on input $a(e, s)$ a tree of the form $b(t_1, t_2, t_3)$ is producible by $q_1$ using the rule $q_1(a(x_1, x_2)) \to b(q_1'(x_1), q_1''(x_2), q_1'''(x_2))$. Quite clearly, the root of $a(e, s)$ is not relabeled. Thus, the translation of $N'$ equipped with the la-automaton $A'$ is empty as the translation of $\tau'$ is.

### 3.1 Construction of the LA-Transducer $M$

Recall that for a a composition $\tau$ of two transducers $T_1$ and $T_2$, we aim to construct an la-transducer $M$ such that $M$ is functional if and only if $\tau$ is functional.

In the following we show that combining the ideas presented above yields the la-transducer $M$. For $T_1$ and $T_2$, we obtain $M$ by first completing the following steps.

1. Construct the domain automaton $A$ of $T_2$
2. Construct the transducer $T_1$ from $T_1$ and $A$ using the p-construction
3. Construct the transducer $N$ from $T_1$ and $T_2$ using the p-construction

We then obtain $M$ by extending $N$ into a transducer with look-ahead. Note that the states of $N$ are written as $(q, S, q')$ instead of $((q, S), q')$ for better readability, where $(q, S)$ is a state of $T_1$ and $q'$ is a state of $T_2$. Recall that $(q, S, q')$ means that $q'$ is supposed to process a tree generated by $(q, S)$. Furthermore, recall that $S$ is a set of states of $T_2$ and that the idea is that trees produced by $(q, S)$ are only supposed to be processed by states in $S$. Thus, we only consider states $(q, S, q')$ of $N$ where $q' \in S$.

The transducer $M$ with look-ahead is constructed as follows. The set of states of $M$ and the initial state of $M$ are the states of $N$ and the initial state of $N$, respectively. The la-automaton of $M$ is the domain automaton $A$ of $T_1$.

We now define the rules of $M$. First, recall that a state of $A$ is a set consisting of states of $T_1$. Furthermore, recall that for a set of right-hand sides $\Gamma$ and a
variable $x$, we denote by $\Gamma(x)$ the set of all states $q$ such that $q(x)$ occurs in some $\gamma \in \Gamma$. For a right-hand side $\gamma$, the set $\gamma(x)$ is defined analogously. For all rules

$$\eta = (q, S, q')(a(x_1, \ldots, x_k)) \rightarrow \gamma$$

of $N$ we proceed as follows: If $\eta$ is obtained from the rule $(q, S)(a(x_1, \ldots, x_k)) \rightarrow \xi$ of $T_1$ and subsequently translating $\xi$ by the state $q'$ of $T_2$ then we define the rule

$$(q, S, q')(a(x_1 : l_1, \ldots, x_k : l_k)) \rightarrow \gamma$$

for $M$ where for $i \in [k]$, $l_i$ is a state of $A$ such that $\xi(x_i) \subseteq l_i$. Recall that relabeling a node $v$, that was previously labeled by $a$, by $\langle a, l_1, \ldots, l_k \rangle$ means that the $i$-th subtree of $v$ is a tree in $\text{dom}(l_i)$. By Lemma 1, $s \in \text{dom}(l_i)$ if and only if $s \in \bigcap_{l \in l_i} \text{dom}(\hat{q})$. Thus, if the node $v$ of a tree $s$ is relabeled by $\langle a, l_1, \ldots, l_k \rangle$ then it means that $(q, S)$ can process subtree of $s$ rooted at $v$ using the rule $(q, S)(a(x_1, \ldots, x_k)) \rightarrow \xi$.

In the following, we present a detailed example for the construction of $M$ for two transducers $T_1$ and $T_2$.

**Example 4.** Let the transducer $T_1$ contain the rules

$$q_0(f(x_1, x_2)) \rightarrow f(q_1(x_1), q_2(x_2)) \quad q_0(f(x_1, x_2)) \rightarrow q_3(x_2)$$
$$q_2(f(x_1, x_2)) \rightarrow f(q_2(x_1), q_1(x_2)) \quad q_1(f(x_1, x_2)) \rightarrow f(q_1(x_1), q_2(x_2))$$
$$q_2(c) \rightarrow c \quad q_1(c) \rightarrow c$$
$$q_3(d) \rightarrow d \quad q_1(d) \rightarrow d$$

and let the initial state of $T_1$ be $q_0$. Informally, when reading the symbol $f$, the states $q_1$ and $q_2$ nondeterministically decide whether or not to relabel $f$ by $f'$. However, the domain of $q_2$ only consists of trees whose leftmost leaf is labeled by $c$. The state $q_3$ only produces the tree $d$ on input $d$. Thus, the domain of $T_1$ only consists of trees of the form $f(s_1, s_2)$ where $s_1$ and $s_2$ are trees and either the leftmost leaf of $s_2$ is $c$ or $s_2 = d$.

The initial state of the transducer $T_2$ is $\hat{q}_0$ and $T_2$ contains the rules

$$\hat{q}_0(f(x_1, x_2)) \rightarrow f(\hat{q}_1(x_1), \hat{q}_2(x_1))$$
$$\hat{q}_0(d) \rightarrow d$$
$$\hat{q}_2(f(x_1, x_2)) \rightarrow f(\hat{q}_2(x_1), \hat{q}_2(x_2))$$
$$\hat{q}_1(c) \rightarrow c$$
$$\hat{q}_2(d) \rightarrow d$$

Informally, on input $s$, the state $\hat{q}_2$ produces $s$ if the symbol $f'$ does not occur in $s$; otherwise $\hat{q}_2$ produces no output. The state $\hat{q}_1$ realizes the identity. Hence, the domain of $T_2$ only consists of the tree $d$ and trees $f(s_1, s_2)$ with no occurrences of $f'$ in $s_1$.

Consider the composition $\tau = T_1 \circ T_2$. On input $s$, the composition $\tau$ yields $f(s_1, s_1)$ if $s$ is of the form $f(s_1, s_2)$ and the leftmost leaf of $s_2$ is labeled by $c$. If the input tree is of the form $f(s_1, d)$, the output tree $d$ is produced. Clearly, $\tau$ is functional. We remark that both phenomena described in Section 3 occur in the
composition $\tau$. More precisely, simply applying the $p$-construction to $T_1$ and $T_2$ yields a nondeterministic transducer due to ‘independent guessing’. Furthermore, not checking the validity of guesses causes nondeterminism on input $f(s_1, d)$.

In the following, we show how to construct the la-automaton $M$ from the transducers $T_1$ and $T_2$.

**Construction of the domain automaton $A$.** We begin by constructing the domain automaton $A$ of $T_2$. The set of states of $A$ is the power set of the set of states of $T_2$ and the initial state of $A$ is $\{q_0\}$. The rules of $A$ are

\[
\{q_0\} \rightarrow f(S(x_1), \emptyset(x_2))
\]

\[
\{q_0\} \rightarrow d
\]

\[
S \rightarrow f(S(x_1), S(x_2))
\]

\[
S \rightarrow e
\]

\[
S \rightarrow d
\]

where $S = \{q_1, q_2\}$. The state $\emptyset$ realizes the identity. The rules for the state $\emptyset$ are straightforward and hence omitted here. All remaining states, such as for instance $\{q_0, q_1\}$, are unreachable and hence the corresponding rules are irrelevant. Thus, we omit these rules as well. In the following, we only consider rules of states that are reachable.

**Construction of the transducer $\hat{T}_1$.** For $T_1$ and $A$, the $p$-construction yields the transducer $\hat{T}_1$. The transducer $\hat{T}_1$ contains the rules

\[
(q_0, \{q_0\}) \rightarrow f((q_1, S)(x_1), q_2(x_2))
\]

\[
(q_0, \{q_0\}) \rightarrow (q_3, \{q_0\})
\]

\[
q_1 \rightarrow f(q_1(x_1), q_1(x_2))
\]

\[
q_1 \rightarrow f'(q_1(x_1), q_1(x_2))
\]

\[
q_1 \rightarrow e
\]

\[
q_1 \rightarrow d
\]

\[
(q_1, S) \rightarrow f((q_1, S)(x_1), (q_1, S)(x_2))
\]

\[
(q_1, S) \rightarrow e
\]

\[
(q_1, S) \rightarrow d
\]

\[
q_2 \rightarrow f(q_2(x_1), q_2(x_2))
\]

\[
q_2 \rightarrow f'(q_2(x_1), q_2(x_2))
\]

\[
q_2 \rightarrow e
\]

\[
(q_3, \{q_0\}) \rightarrow d
\]

and the initial state of $\hat{T}_1$ is $(q_0, \{q_0\})$. For better readability, we just write $q_1$ and $q_2$ instead of $(q_1, \emptyset)$ and $(q_2, \emptyset)$, respectively.

**Construction of the transducer $N$.** For $\hat{T}_1$ and $T_2$, we construct the transducer $N$ containing the rules
however, since no rules are defined for the state \( q F \) or better readability, we again just write respectively. We remark that, by construction of the domain automaton, the initial state of \( N \) is \((q_0, \{\hat{q}_0\}, \hat{q}_0)\). Note that the states such as \((q_1, S, \hat{q}_0)\) are not considered as \( \hat{q}_0 \) is not contained in \( S \). We remark that though no nondeterminism is caused by ‘independent guessing’, \( N \) is still nondeterministic on input \( f(s_1, d) \) as the validity of guesses cannot be checked. To perform validity checks for guesses, we extend \( N \) with look-ahead.

**Construction of the look-ahead automaton \( \hat{A} \).** Recall that the look-ahead automaton of \( M \) is the domain automaton \( \hat{A} \) of \( \hat{T}_1 \). The set of states of \( \hat{A} \) is the power set of the set of states of \( \hat{T}_1 \). The initial state of \( \hat{A} \) is \( \{(q_0, \{\hat{q}_0\})\} \) and \( \hat{A} \) contains the following rules.

\[
\begin{align*}
\{(q_0, \{\hat{q}_0\})\} (f(x_1, x_2)) &\rightarrow f((q_1, S, \hat{q}_1)(x_1), (q_1, S, \hat{q}_2)(x_1)) \\
(q_0, \{\hat{q}_0\}) (f(x_1, x_2)) &\rightarrow (q_3, \{\hat{q}_0\}, \hat{q}_0)(x_2) \\
(q_1, S, \hat{q}_1) (f(x_1, x_2)) &\rightarrow f((q_1, S, \hat{q}_1)(x_1), (q_1, S, \hat{q}_1)(x_2)) \\
(q_1, S, \hat{q}_1)(e) &\rightarrow e \\
(q_1, S, \hat{q}_1)(d) &\rightarrow d \\
(q_1, S, \hat{q}_2)(f(x_1, x_2)) &\rightarrow f((q_1, S, \hat{q}_2)(x_1), (q_1, S, \hat{q}_2)(x_2)) \\
(q_1, S, \hat{q}_2)(e) &\rightarrow e \\
(q_1, S, \hat{q}_2)(d) &\rightarrow d \\
(q_3, \{\hat{q}_0\})(d) &\rightarrow d
\end{align*}
\]

For better readability, we again just write \( q_1 \) and \( q_2 \) instead of \((q_1, \emptyset)\) and \((q_2, \emptyset)\), respectively. We remark that, by construction of the domain automaton, \( \hat{A} \) also contains the rule

\[
\{(q_0, \{\hat{q}_0\})\}(f(x_1, x_2)) \rightarrow f((\{(q_1, S)\}(x_1), \{q_2, (q_3, \{\hat{q}_0\})\}(x_2)),
\]

however, since no rules are defined for the state \( \{q_2, (q_3, \{\hat{q}_0\})\} \), this rule can be omitted.

**Construction of the la-transducer \( M \).** Finally, we construct the la-transducer \( M \). The initial state of \( M \) is \((q_0, \{\hat{q}_0\}, \hat{q}_0)\) and the rules of \( M \) are
More precisely, we show that $N$ implies our claim. First of all, consider the transducers $M$ prove that $N$ the $p$-construction in our examples in Section 3. Notice that the relations defined $M$ to show that $N$ are proper supersets. However, as none such states $l$ but $l_1$ and $l_2$ are states of $M$ such that $\{q_1, S\} \subseteq l_1$ and $\{q_2\} \subseteq l_2$ and $l_1$ or $l_2$ is a proper superset. However, as none such states $l_1$ and $l_2$ are reachable by $M$, we have omitted rules of this form. Other rules are omitted for the same reason. □

3.2 Correctness of the LA-Transducer $M$

In the following we prove the correctness of our construction. More precisely, we prove that $M$ is functional if and only if $T_1 \circ T_2$ is. By Lemma [3] it is sufficient to show that $M$ is functional if and only if $T_1 \circ T_2$ is.

First, we prove that the following claim: If $M$ is functional then $T_1 \circ T_2$ is functional. More precisely, we show that $\mathcal{R}(T_1) \circ \mathcal{R}(T_2) \subseteq \mathcal{R}(M)$. Obviously, this implies our claim. First of all, consider the transducers $N$ and $N'$ obtained from the $p$-construction in our examples in Section [3]. Notice that the relations defined by $N$ and $N'$ are supersets of $\mathcal{R}(T_1) \circ \mathcal{R}(T_2)$ and $\mathcal{R}(T'_1) \circ \mathcal{R}(T'_2)$, respectively.
In the following, we show that this observation can be generalized. Consider arbitrary transducers $T$ and $T'$. We claim that the transducer $\hat{N}$ obtained from the $p$-construction for $T$ and $T'$ always defines a superset of the composition $R(T) \circ R(T')$. To see that our claim holds, consider a translation of $T \circ T'$ in which the state $q'$ of $T'$ processes a tree $t$ produced by the state $q$ of $T$ on input $s$. If the corresponding state $(q, q')$ of $\hat{N}$ processes $s$ then $(q, q')$ can guess that $q$ has produced $t$ and proceed accordingly. Thus $\hat{N}$ can effectively simulate the composition $T \circ T'$.

As $M$ is in essence obtained from the $p$-construction extended with look-ahead, $M$ ‘inherits’ this property. Note that the addition of look-ahead does not affect this property. Therefore our claim follows.

**Lemma 4.** $R(\hat{T}_1) \circ R(T_2) \subseteq R(M)$.

In fact an even stronger result holds.

**Lemma 5.** Let $(q_1, S)$ be a state of $\hat{T}_1$ and $q_2$ be a state of $T_2$. If on input $s$, $(q_1, S)$ can produce the tree $t$ and on input $t$, $q_2$ can produce the tree $r$ then $(q_1, S, q_2)$ can produce $r$ on input $s$.

Consider a translation of $\hat{T}_1 \circ T_2$ in which $T_2$ processes the tree $t$ produced by $\hat{T}_1$ on input $s$. We call a translation of $M$ synchronized if the translation simulates a translation of $\hat{T}_1 \circ T_2$, i.e., if a state $(q, S, q')$ of $M$ processes the subtree $s'$ of $s$ and the corresponding state of $q'$ of $T_2$ processes the subtree $t'$ of $t$ and $t'$ is produced by $(q, S)$ on input $s'$, then $(q, S, q')$ guesses $t'$.

We now show that if $\hat{T}_1 \circ T_2$ is functional, then so is $M$. Before we prove our claim consider the following auxiliary results.

**Lemma 6.** Consider an arbitrary input tree $s$. Let $\hat{s}$ be a subtree of $s$. Assume that in an arbitrary translation of $M$ on input $s$, the state $(q_1, S, q_2)$ processes $\hat{s}$. Then, a synchronized translation of $M$ on input $s$ exists in which the state $(q_1, S, q_2)$ processes the subtree $\hat{s}$.

It is easy to see that the following result holds for arbitrary transducers.

**Proposition 1.** Let $\tau = T_1 \circ T_2$ where $T_1$ and $T_2$ are arbitrary transducers. Let $s$ be a tree such that $\tau(s) = \{r\}$ is a singleton. Let $t_1$ and $t_2$ be distinct trees produced by $T_1$ on input $s$. If $t_1$ and $t_2$ are in the domain of $T_2$ then $T_2(t_1) = T_2(t_2) = \{r\}$.

Using Lemma 6 and Proposition 1, we now show that the following holds. Note that in the following $t/v$, where $t$ is some tree and $v$ is a node, denotes the subtree of $t$ rooted at the node $v$.

**Lemma 7.** Consider an arbitrary input tree $s$. Let $\hat{s}$ be a subtree of $s$. Let the state $(q_1, S, q_2)$ process $\hat{s}$ in a translation $M$ on input $s$. If $\hat{T}_1 \circ T_2$ is functional then $(q_1, S, q_2)$ can only produce a single output tree on input $\hat{s}$.

**Proof.** Assume to the contrary that $(q_1, S, q_2)$ can produce distinct trees $r_1$ and $r_2$ on input $\hat{s}$. For $r_1$, it can be shown that a tree $t_1$ exists such that
1. on input $\hat{s}$, the state $(q_1, S)$ of $\hat{T}_1$ produces $t_1$ and
2. on input $t_1$, the state $q_2$ of $T_2$ produces $r_1$.

It can be shown that a tree $t_2$ with the same properties exists for $r_2$. Informally, this means that $r_1$ and $r_2$ are producible by $(q_1, S, q_2)$ by simulating the ‘composition of $(q_1, S)$ and $q_2$’.

Due to Lemma 7, a synchronized translation of $M$ on input $s$ exists in which the state $(q_1, S, q_2)$ processes the subtree $\hat{s}$ of $s$. Let $g$ be the node at which $(q_1, S, q_2)$ processes $\hat{s}$. Let $\hat{q}_1, \ldots, \hat{q}_n$ be all states of $M$ of the form $(q_1, S, q_2')$, where $q_2'$ is some state of $T_2$, that occur in the synchronized translation of $M$ and that process $\hat{s}$. Note that by definition $q_2' \in S$. Due to Lemmas 2 and 5, we can assume that in the synchronized translation, the states $\hat{q}_1, \ldots, \hat{q}_n$ all guess that the tree $t_1$ has been produced by the state $(q_1, S)$ of $\hat{T}_1$ on input $\hat{s}$. Hence, we can assume that at the node $g$, the output subtree $r_1$ is produced. Therefore, a synchronized translation of $M$ on input $s$ exists, that yields an output tree $\hat{r}_1$ such that $\hat{r}_1/g = r_1$, where $\hat{r}_1/g$ denotes the subtree of $\hat{r}_1$ rooted at the node $g$. Analogously, it follows that a synchronized translation of $M$ on input $s$ exists, that yields an output tree $\hat{r}_2$ such that $\hat{r}_2/g = r_2$.

As both translation are synchronized, i.e., ‘simulations’ of translations of $\hat{T}_1 \circ T_2$ on input $s$, it follows that the trees $\hat{r}_1$ and $\hat{r}_2$ are producible by $\hat{T}_1 \circ T_2$ on input $s$. Due to Proposition 1, $\hat{r}_1 = \hat{r}_2$ and therefore $r_1 = \hat{r}_1/g = \hat{r}_2/g = r_2$.

Lemma 3 implies that if $M$ is functional then $\hat{T}_1 \circ T_2$ is functional as well. Lemma 4 implies that if $\hat{T}_1 \circ T_2$ is functional then so is $M$. Therefore, we deduce that due Lemmas 3 and 4, the following holds.

**Corollary 1.** $\hat{T}_1 \circ T_2$ is functional if and only if $M$ is functional.

In fact, Corollary 1 together with Lemma 3 imply that $\hat{T}_1 \circ T_2$ and $M$ are equivalent if $T_1 \circ T_2$ is functional, since it can be shown that $\text{dom}(\hat{T}_1 \circ T_2) = \text{dom}(M)$.

Since functionality for transducers with look-ahead is decidable [8], Corollary 1 implies that it is decidable whether or not $\hat{T}_1 \circ T_2$ is functional. Together with Lemma 3, we obtain:

**Theorem 1.** Let $T_1$ and $T_2$ be top-down tree transducers. It is decidable whether or not $T_1 \circ T_2$ is functional.

### 3.3 Functionality of Arbitrary Compositions

In this section, we show that the question whether or not an arbitrary composition is functional can be reduced to the question of whether or not a two-fold composition is functional.

**Lemma 8.** Let $\tau$ be a composition of transducers. Then two transducers $T_1, T_2$ can be constructed such that $T_1 \circ T_2$ is functional if and only if $\tau$ is functional.
Proof. Consider a composition of $n$ transducers $T'_1, \ldots, T'_n$. W.l.o.g. assume that $n > 2$. For $n \leq 2$, our claim follows trivially. Let $\tau$ be the composition of $T'_1, \ldots, T'_n$. We show that transducer $\hat{T}_1, \ldots, \hat{T}_{n-1}$ exist such that $\hat{T}_1 \circ \cdots \circ \hat{T}_{n-1}$ is functional if and only if $\tau$ is.

Consider an arbitrary input tree $s$. Let $t$ be a tree produced by the composition $T'_1 \circ \cdots \circ T'_{n-2}$ on input $s$. Analogously as in Proposition 1, the composition $T'_{n-1} \circ T'_n$, on input $t$, can only produce a single output tree if $\tau$ is functional. For the transducers $T'_{n-1}$ and $T'_{n}$, we construct the la-transducer $M$ according to our construction in Section 3.1. It can be shown that, the la-transducer $M$ our construction yields has the following properties regardless of whether or not $T'_{n-1} \circ T'_n$ is functional

(a) $\text{dom}(M) = \text{dom}(T'_{n-1} \circ T'_n)$ and
(b) on input $t$, $M$ only produces a single output tree if and only if $T'_{n-1} \circ T'_n$ does

Therefore, $\tau(s)$ is a singleton if and only if $T'_1 \circ \cdots \circ T'_{n-2} \circ M(s)$ is a singleton. Engelfriet has shown that every transducer with look-ahead can be decomposed to a composition of a deterministic bottom-up relabeling and a transducer (Theorem 2.6 of [4]). It is well known that (nondeterministic) relabelings are independent of whether they are defined by bottom-up transducers or by top-down transducers (Lemma 3.2 of [3]). Thus, any transducer with look-ahead can be decomposed into a composition of a nondeterministic top-down relabeling and a transducer. Let $R$ and $T$ be the relabeling and the transducer such that $M$ and $R \circ T$ are equivalent. Then obviously, $\tau(s)$ is a singleton if and only if $T'_1 \circ \cdots \circ T'_{n-2} \circ R \circ T(s)$ is a singleton.

Consider arbitrary transducers $\hat{T}_1$ and $\hat{T}_2$. Baker has shown that if $\hat{T}_2$ is non-deleting and linear then a transducer $T$ can be constructed such that $T$ and $\hat{T}_1 \circ \hat{T}_2$ are equivalent (Theorem 1 of [2]). By definition, any relabeling is non-deleting and linear. Thus, we can construct a transducer $\hat{T}$ such that $\hat{T}$ and $T'_{n-2} \circ R$ are equivalent. Therefore, it follows that $\tau(s)$ is a singleton if and only if $T'_1 \circ \cdots \circ T'_{n-3} \circ \hat{T}(s)$ is a singleton. This yields our claim. \hfill \Box

Lemma 8 and Theorem 1 yield that functionality of compositions of transducers is decidable.

Engelfriet has shown that any la-transducer can be decomposed into a composition of a nondeterministic top-down relabeling and a transducer [4]. Recall that while la-transducers generalize transducers, bottom-up transducers and la-transducers are incomparable [4]. Baker, however, has shown that the composition of $n$ bottom-up transducers can be realized by the composition of $n+1$ top-down transducers [2]. For any functional composition of transducers an equivalent deterministic la-transducer can be constructed [5]. Therefore we obtain our following main result.

**Theorem 2.** Functionality for arbitrary compositions of top-down and bottom-up tree transducers is decidable. In the affirmative case, an equivalent deterministic top-down tree transducer with look-ahead can be constructed.
4 Conclusion

We have presented a construction of an la-transducer for a composition of transducers which is functional if and only if the composition of the transducers is functional — in which case it is equivalent to the composition. This construction is remarkable since transducers are not closed under composition in general, neither does functionality of the composition imply that each transducer occurring therein, is functional. By Engelfriet’s construction in [5], our construction provides the key step to an efficient implementation (i.e., a deterministic transducer, possibly with look-ahead) for a composition of transducers – whenever possible (i.e., when their translation is functional). As an open question, it remains to see how large the resulting functional transducer necessarily must be, and whether the construction can be simplified if for instance only compositions of linear transducers are considered.

References

1. Aho, A.V., Ullman, J.D.: Translations on a context-free grammar. Inf. Control. 19(5), 439–475 (1971)
2. Baker, B.S.: Composition of top-down and bottom-up tree transductions. Inf. Control. 41(2), 186–213 (1979)
3. Engelfriet, J.: Bottom-up and top-down tree transformations - A comparison. Math. Syst. Theory 9(3), 198–231 (1975)
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.: Three hierarchies of transducers. Math. Syst. Theory 15(2), 95–125 (1982)
7. Engelfriet, J., Maneth, S., Seidl, H.: Deciding equivalence of top-down XML transformations in polynomial time. J. Comput. Syst. Sci. 75(5), 271–286 (2009)
8. Ésik, Z.: Decidability results concerning tree transducers I. Acta Cybern. 5(1), 1–20 (1980)
9. Hakuta, S., Maneth, S., Nakano, K., Iwasaki, H.: Xquery streaming by forest transducers. In: ICDE 2014, Chicago, USA, March 31 - April 4, 2014. pp. 952–963 (2014)
10. Küsters, R., Wilke, T.: Transducer-based analysis of cryptographic protocols. Inf. Comput. 205(12), 1741–1776 (2007)
11. Maneth, S.: A survey on decidable equivalence problems for tree transducers. Int. J. Found. Comput. Sci. 26(8), 1069–1100 (2015)
12. Maneth, S., Seidl, H., Vu, M.: Definability results for top-down tree transducers. In: DLT 2021, Porto, Portugal, August 16-20, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12811, pp. 291–303. Springer (2021)
13. Matsuda, K., Inaba, K., Nakano, K.: Polynomial-time inverse computation for accumulative functions with multiple data traversals. High. Order Symb. Comput. 25(1), 3–38 (2012)
14. Rounds, W.C.: Mappings and grammars on trees. Math. Syst. Theory 4(3), 257–287 (1970)
15. Thatcher, J.W.: Generalized sequential machine maps. J. Comput. Syst. Sci. 4(4), 339–367 (1970)
A Appendix

In the following, we first introduce additional notation and definitions used in the proofs in the Appendix.

A.1 Definitions

Set of Nodes Let $t$ be a tree. For $t$, its set $V(t)$ of nodes is a subset of $V = N^*$. More formally, $V(t) = \{ \epsilon \} \cup \{ iu \mid i \in [k], u \in V(t_i) \}$ where $t = a(t_1, \ldots, t_k)$, $a \in \Sigma_k$, $k \geq 0$ and $t_1, \ldots, t_k \in T_\Sigma$. For better readability we add dots between numbers. 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 $v$ and $t/v$ is the subtree of $t$ rooted at $v$.

Substitutions Let $t_1, \ldots, t_n$ be trees over $\Sigma$ and $v_1, \ldots, v_n$ be distinct nodes none of which is a prefix of the other, then we denote by $[v_i \leftarrow t_i \mid i \in [n]]$ the substitution that for each $i \in [n]$, replaces the subtree rooted at $v_i$ with $t_i$.

Let $t$ be a tree, $a \in \Sigma_0$ and $T$ be a set of trees. We denote by $t[a \leftarrow T]$ the set of all trees obtained by substituting leaves labeled by $a$ with some tree in $T$, i.e., the set of all trees of the form $t[v \leftarrow t_v \mid v \in V(t), t[v] = a]$ where for all $a$-leaves $v$, $t_v \in T$. Note that two distinct leaves labeled by $a$ may be replaced by distinct trees in $T$. If $T = \emptyset$ then we define $t[a \leftarrow T] = \emptyset$. For simplicity, we write $t[a \leftarrow t']$ if $T = \{ t' \}$.

Partial Trees and Semantic of a Transducer Recall that $T_\Sigma[B] = T_{\Sigma'}$ where $\Sigma'$ is obtained from $\Sigma$ by $\Sigma'_0 = \Sigma_0 \cup B$ while for all $k > 0$, $\Sigma'_k = \Sigma_k$. In the following, we call a tree in $T_{\Sigma'}[B]$ a partial tree.

The semantic of a transducer $T$, defined as in Section 2 is formally defined as follows. Let $q \in Q$ and $v$ be an arbitrary node. We denote by $[q]_v^T$ the partial function from $T_{\Sigma'}[B]$ to the power set of $T_\Delta[Q(V)]$ defined as follows

- for $s = a(s_1, \ldots, s_k)$, $a \in \Sigma_k$, and $s_1, \ldots, s_k \in T_{\Sigma'}[B]$,
  
  $$[q]_v^T(s) = \bigcup_{\xi \in \text{rhs}(q, a)} \xi[q(x_i) \leftarrow [q]_{v,i}(s_i) \mid q \in Q, i \in [k]]$$

- for $b \in B$, $[q]_v^T(b) = \{ q(v) \}$,

where $\text{rhs}(q, a)$ denotes the set of all right-hand sides of $q$ and $a$. The reason why input nodes of $s$ are added to the semantic of $T$ is that for some of our proofs we require that for states $q$ of $T$ it is traceable which input node $q$ currently processes.

If clear from context which transducer is meant, we omit the superscript $T$ and write $[q]_v$ instead of $[q]_v^T$. In the following, we write $[q]$ instead of $[q]_v$ for simplicity. We write $[q]_v^T(s) \Rightarrow t$ if $t \in [q]_v^T(s)$.

In the following, for trees in $T_\Delta[Q(X)]$ and $T_\Delta[Q(V)]$, we write $t(x \leftarrow v)$ to denote the substitution $t[q(x) \leftarrow q(v) \mid q \in Q]$ for better readability where $x \in X$ and $v \in V$. 
Recall that for a set $\Gamma$ of right-hand sides of a transducer $T$, $\Gamma^{(x_i)}$ denotes the set of all states $q$ of $T$ such that $q(x_i)$ occurs in some tree $\gamma$ in $\Gamma$. For a set $A$ of trees in $T^\Delta [Q(V)]$, we define $A(v)$ where $v$ is some node analogously; e.g., for $A = \{ f(q(v_1), f(q(v_2), q'(v_2))), f(q_1(v_1), q_2(v_2)) \}$, we have $A(v_1) = \{ q, q_1 \}$ and $A(v_2) = \{ q, q', q_2 \}$.

**B Properties of the Domain Automaton $A$**

In the following we consider the *domain automaton* $A$ introduced in Section 3 for a transducer $T$. In particular, we consider the properties of $A$. Recall that a state of $A$ is a set consisting of states of $T$. In Section 3 we have claimed that if in a translation of $T$ on input $s$, the states $q_1, \ldots, q_n$ process the node $v$ of $s$ then $\{ q_1, \ldots, q_n \}$ processes the node $v$ of $s$ in a computation of $A$. We now formally prove this statement. First we prove the following auxiliary result.

**Lemma 9.** Let $s \in T^\Sigma_2[X]$. Let $v_1, \ldots, v_n$ be the nodes of $s$ that are labeled by a symbol in $X$. Let $S_1$ and $S_2$ be states of $A$ and let for $j = 1, 2$,

$$\left[ S_j \right](s) \Rightarrow s[v_i \leftarrow S^j_i(v_i) \mid i \in [n]]$$

where for $i \in [n], S^j_i$ is a state of $A$. Then

$$\left[ S_1 \cup S_2 \right](s) \Rightarrow s[v_i \leftarrow S^1_i \cup S^2_i(v_i) \mid i \in [n]]$$

**Proof.** We prove our claim by structural induction over $s$. Let $s = a(s_1, \ldots, s_k)$ where $a \in \Sigma_k$, $k \geq 0$, and for $t \in [k]$, $s_t \in T^\Sigma_2[X]$. As

$$\left[ S_1 \right](s) \Rightarrow s[v_i \leftarrow S^1_i(v_i) \mid i \in [n]]$$

a rule $S_1(a(x_1, \ldots, x_k)) \rightarrow a(\hat{S}_1(x_1), \ldots, \hat{S}_k(x_k))$ exists such that for all $i \in [k]$, on input $s_i$, the function $[\hat{S}_i]_{s_i}$ yields the subtree of $s[v_i \leftarrow S^1_i(v_i) \mid i \in [n]]$ that is rooted at the node $i$. More formally,

$$[\hat{S}_i]_{s_i}(s_i) \Rightarrow s_i[v' \leftarrow S^1_i(i.v') \mid v' \in V(s_i) \text{ and } i.v' = v_i \text{ where } i \in [n]]$$

which in turn implies

$$[\hat{S}_i]_{s_i}(s_i) \Rightarrow s_i[v' \leftarrow S^1_i(v') \mid v' \in V(s_i) \text{ and } i.v' = v_i \text{ where } i \in [n]]$$

Analogously, it follows that a rule $S_2(a(x_1, \ldots, x_k)) \rightarrow a(\hat{S}'_1(x_1), \ldots, \hat{S}'_k(x_k))$ exists such that for all $i \in [k],

$$[\hat{S}'_i]_{s_i}(s_i) \Rightarrow s_i[v' \leftarrow S^2_i(v') \mid v' \in V(s_i) \text{ and } i.v' = v_i \text{ where } i \in [n]]$$

We now show that the automaton $A$ contains the rule

$$S_1 \cup S_2(a(x_1, \ldots, x_k)) \rightarrow a(\hat{S}_1 \cup \hat{S}'_1(x_1), \ldots, \hat{S}_k \cup \hat{S}'_k(x_k)). \quad (a)$$
Lemma 10. We now prove our statement. Thus, due to (a) and (b) our claim follows. ⊓⊔

By construction, the rule $S_1(a(x_1, \ldots, x_k)) \rightarrow a(\hat{S}_1(x_1), \ldots, \hat{S}_k(x_k))$ is defined only if for all $q \in S_1$ a non-empty set of right-hand sides $\Gamma_q \subseteq \text{rhs}_T(q, a)$ exists such that for $i \in [k]$, $\hat{S}_i = \bigcup_{q \in S_1} \Gamma_q(x_i)$.

Likewise, the rule $S_2(a(x_1, \ldots, x_k)) \rightarrow a(\hat{S}_1'(x_1), \ldots, \hat{S}_k'(x_k))$ is defined only if for all $q' \in S_2$ a non-empty set of right-hand sides $\Gamma_{q'} \subseteq \text{rhs}_T(q', a)$ exists such that for $i \in [k]$, $\hat{S}_i' = \bigcup_{q' \in S_2} \Gamma_{q'}(x_i)$. For all states $q \in S_1 \cup S_2$, we define

$$\tilde{\Gamma}_q = \begin{cases} 
\Gamma_q \cup \Gamma'_q & \text{if } q \in S_1 \cap S_2 \\
\Gamma_q & \text{if } q \in S_1 \setminus S_2 \\
\Gamma'_q & \text{if } q \in S_2 \setminus S_1 
\end{cases}$$

Clearly, the sets $\tilde{\Gamma}_q$ yield that the rule defined in (a) exists.

Now, consider the following. As for $i \in [k]$,

$$[\hat{S}_i](s_i) \Rightarrow s_i[v' \leftarrow S'_i(v') \mid v' \in V(s_i) \text{ and } i.v' = v_i \text{ where } i \in [n]]$$

and

$$[\hat{S}_i'](s_i) \Rightarrow s_i[v' \leftarrow S'_2(v') \mid v' \in V(s_i) \text{ and } i.v' = v_i \text{ where } i \in [n]]$$

the induction hypothesis yields

$$[\hat{S}_i \cup \hat{S}_i'](s_i) \Rightarrow s_i[v' \leftarrow S_1' \cup S_2'(v') \mid v' \in V(s_i) \text{ and } i.v' = v_i \text{ where } i \in [n]] \quad (b)$$

Thus, due to (a) and (b) our claim follows. □

We now prove our statement.

Lemma 10. Let $s \in T_{\Sigma}[X]$. Let $v_1, \ldots, v_n$ be the nodes of $s$ that are labeled by a symbol in $X$. Let $S$ be a state of $A$. For $q \in S$, let $[q]^T(s) \Rightarrow t_q$. Then,

$$[S]^A(s) \Rightarrow s[v_i \leftarrow S_i(v_i) \mid i \in [n]]$$

where $S_i = \bigcup_{q \in S} t_q(v_i)$.

Proof. Due to Lemma 9 it is sufficient to show that if $[q]^T(s) \Rightarrow t_q$, then

$$[[q]]^A(s) \Rightarrow s[v_i \leftarrow S'_i(v_i) \mid i \in [n]]$$

where $S'_i = t_q(v_i)$. We prove this claim by structural induction over $s$. Let $s = a(s_1, \ldots, s_k)$ where $a \in \Sigma_k$, $k \geq 0$, and for $i \in [k]$, $s_i \in T_{\Sigma}[X]$. Due to our premise, $\gamma \in \text{rhs}(q, a)$ exists such that

$$t_q \in \gamma[q'(x_i) \leftarrow [q'],(s_i) \mid q' \in Q, i \in [k]]. \quad (1)$$

By definition of $A$, $\gamma \in \text{rhs}(q, a)$ implies that the automaton $A$ contains the rule

$$\{q\}(a) \rightarrow a(\hat{S}_1(x_1), \ldots, \hat{S}_k(x_k)) \quad (a)$$
where $\hat{S}_i = \gamma(x_i)$ for $i \in [k]$.

Now consider $\gamma \in \text{rhs}(q, a)$ in conjunction with the variable $x_i$. Let $\gamma(x_i) = \{q_1, \ldots, q_m\}$. For $j \in [m]$, we denote by $U_j$ the set of all nodes of $\gamma$ that are labeled by $q_j(x_i)$. For all nodes $u \in U_j$, Equation 1 clearly implies $[q_j]^T(s_i) \Rightarrow t_{q/u}$. Recall that by definition, if $\hat{q}(v)$ occurs in $t_{q/u}$, where $\hat{q}$ is a state of $T$ and $v$ is some node then $v$ is of the form $i.v$. Clearly, $[q_j]^T(s_i) \Rightarrow t_{q/u}$ implies $[q_j]^T(s_i) \Rightarrow \eta_u$ where $\eta_u$ denotes the tree obtained from $t_{q/u}$ by substituting occurrences of $\hat{q}(i.v)$ by $\hat{q}(v)$.

Recall that $\eta_u(i.v')$ denotes the set of all states $q'$ in $Q$ such that $q'(v')$ occurs in $\eta_u$. In the following, let $S_{u,v} = \eta_u(i.v')$. Due to the induction hypothesis, it follows that for all $u \in U_j$,

$$\{[q_j]^A(s_i) \Rightarrow s_i[v' \leftarrow S_{u,v}(v') \mid v' \in V(s_i), s_i[v'] \in X]\}.$$  \hfill (2)

Due to Lemma 9 and Equation 2 it follows for all $j \in [m]$ that

$$\{[q_j]^A(s_i) \Rightarrow s_i[v' \leftarrow \bigcup_{u \in U_j} S_{u,v}(v') \mid v' \in V(s_i), s_i[v'] \in X]\}.$$  \hfill (3)

Recall that $\hat{S}_i = \gamma(x_i)$. Thus, Lemma 9 and Equation 3 yield

$$[\hat{S}_i]^A(s_i) \Rightarrow s_i[v' \leftarrow \bigcup_{j \in [m]} \bigcup_{u \in U_j} S_{u,v}(v') \mid v' \in V(s_i), s_i[v'] \in X]\text{.}$$ \hfill (b)

Note that for $v = i.v'$,

$$\bigcup_{j \in [m]} \bigcup_{u \in U_j} \eta_u(v') = \bigcup_{j \in [m]} \bigcup_{u \in U_j} S_{u,v} = t(v).$$

Therefore it follows that (a) and (b) yield our claim. \hfill \Box

Additionally, the domain automaton $A$ has the following property. If in a translation of $A$ on input $s$, the state $S$ processes the node $v$ of $s$ then a translation of $T$ on input $s$ exists such that $v$ is only processed by states in $S$. More formally, we prove thee following result.

**Lemma 11.** Let $s \in T_X[X]$. Let $v_1, \ldots, v_n$ be the nodes of $s$ that are labeled by a symbol in $X$. Let $S, S_1, \ldots, S_n$ be states of $A$. Let $[S]^A(s) \Rightarrow \hat{s}$ where

$$\hat{s} = s[v_i \leftarrow S_i(v_i) \mid i \in [n]].$$

For all $q \in S$, a tree $t$ exists such that $[q]^T(s) \Rightarrow t$ and for $i \in [n]$, $t(v_i) \subseteq S_i$.

**Proof.** We prove our claim by structural induction. Obviously, our claim holds if $s \in X$.

In the following, let $s \notin X$. Then, a node $v$ exist such that the subtree of $s$ rooted at $v$ is of the form $a(s_1, \ldots, s_k)$ where $a \in \Sigma_k$, $k \geq 0$, and $s_1, \ldots, s_k \in X$. Note that by definition $v$ can be a leaf. Consider the tree $s' = s[v \leftarrow x_1]$. Let
\(v'_1, \ldots, v'_m\) be the nodes of \(s'\) that are labeled by a symbol in \(X\). W.l.o.g. let \(v'_1 = v\).

Recall that the node \(v\) is labeled by the node \(a\) in \(s\). As \([S]^A(s) \Rightarrow \bar{s}\), it follows that a tree \(\bar{s}\) exists such that \([S]^A(s') \Rightarrow \bar{s}\) and the tree \(s\) can be obtained from \(\bar{s}\) by substituting \(S'_1(v)\) by \(\xi(x_i \leftarrow v.j \mid j \in [k])\) where \(\xi\) is a right-hand side of \(S'_1\) and \(a\). Note that by definition, states \(S'_1, \ldots, S'_m\) of \(A\) exists such that the tree \(\bar{s}\) is obtained from \(s'\) by relabeling the node \(v'_1\) of \(s'\) by \(S'_1(v_i)\). More formally, it holds that

\[
\bar{s} = s'[v'_1 \leftarrow S'_1(v'_1) \mid i \in [m]],
\]

and that

\[
\bar{s} = \bar{s}[S'_1(v) \leftarrow \xi(x_i \leftarrow v.j \mid j \in [k])].
\]

By induction hypothesis, as \([S]^A(s') \Rightarrow \bar{s}\), for all states \(q \in S\), a tree \(t'\) exists such that

1. \([q]^T(s') \Rightarrow t'\) and
2. for \(i \in [m]\), \(t'(v'_i) \subseteq S'_i\).

In particular, it holds that \(t'(v'_i) = t'(v) \subseteq S'_i\).

Let \(S'_1 = \{q_1, \ldots, q_n\}\). Recall that \(\xi\) is a right-hand side of \(S'_m\) and \(a\). W.l.o.g. let \(\xi = a(S_1(x_1), \ldots, S_k(x_k))\). Then, by definition of the rules of \(A\), it follows that for each \(j \in [n]\), a tree \(\gamma_j\) exists such that

(a) \(\gamma_j \in \text{rhs}_T(q_j, a)\)
(b) for \(i \in [k]\) it holds that \(\bigcup_{j \in [n]} \gamma_j(x_i) \subseteq S_i\).

As \(t'(v) \subseteq S'_1\), it follows that \([q]^T(s) \Rightarrow t\) where

\[
t = t'[q_j(v) \leftarrow \gamma_j(x_i \leftarrow v.i \mid i \in [k]) \mid j \in [n]]).
\]

Consider the node \(v.i\) where \(i \in [k]\). If \(\bar{S}(v.i)\) occurs in \(\bar{s}\) then it follows due to Equation 1 that \(\bar{S}(x_i)\) occurs in \(\xi\). Due to Equation 2 and Statement (b), it follows that \(t(v.i) \subseteq S_i\). Thus, our claim follows.

\(\square\)

Lemmas 10 and 11 yield Lemma 11.

\section*{C Properties of \(\hat{T}_1\)}

In the following we consider the properties of the transducer \(\hat{T}_1\). Recall that \(\hat{T}_1\) is obtained via the p-construction from the transducer \(T_1\) and the domain automaton \(A\) of \(T_2\). In particular, we formally prove the statements we made about \(\hat{T}_1\) in Section 3. First we formally prove Lemma 2 that is, we prove the following.

\textbf{Lemma 12.} Let \((q, S)\) be a state of \(\hat{T}_1\) and \(S \neq \emptyset\). If the tree \(t\) over \(\Delta\) is producible by \((q, S)\) then \(t \in_{q_2} \bigcap \text{dom}(q_2)\).
Proof. Let \( t \) be produced by \((q, S)\) on input \( s \) where \( s \in T_\Sigma \). Clearly, it is sufficient to show that \([S]^A(t) \Rightarrow t\) due to Lemma 1. We prove our claim by structural induction. Let \( s = a(s_1, \ldots, s_k) \) where \( a \in \Sigma_k \), \( k \geq 0 \), and \( s_1, \ldots, s_k \in T_\Sigma \). As \( t \) is produced by \((q, S)\) on input \( s \), a right-hand side \( \xi \) for \((q, S)\) and \( a \) exists such that

\[
t \in \xi[(q', S')(x_i) \leftarrow [(q', S')^{\tilde{T}_1}(s_i) \mid (q', S') \in \hat{Q}_1, i \in [k]].
\]

(1)

This means that

\[
t \in \xi[u \leftarrow t/u \mid u \in V(\xi), \xi[u] \text{ is of the form } (q', S')(x_i)].
\]

The following statements hold:

1. By definition of \( \tilde{T}_1 \), it follows that \([S]^A(\xi) \Rightarrow \xi'\) where

\[
\xi' = \xi[u \leftarrow S'(u) \mid u \in V(\xi), \xi[u] \text{ is of the form } (q', S')(x_i)].
\]

2. Consider the node \( u \). Let \( u \) be labeled by \((q', S')(x_i)\) in \( \xi \). This means that \( u \) is labeled by \( S'(u) \) in \( \xi' \). By Equation 1 \( t/u \) can be produced by \((q', S')\) on input \( s_i \).

By induction hypothesis, \([S']^A(t') \Rightarrow t'\) for all trees \( t' \) producible by \((q', S')\).

Thus, \([S']^A(t/u) \Rightarrow t/u\). By definition \([S']^A(t/u) \Rightarrow t/u\) implies

\[
[S']^A(t/u) \Rightarrow t/u
\]

because \( t/u \) is ground.

Statements (1) and (2) yield that \([S]^A(t) \Rightarrow t\).

\( \square \)

We now show that the converse holds as well.

Lemma 13. Let \( s \in T_\Sigma \) and \( t \) be producible by the state \( q_1 \) of \( T_1 \) on input \( s \).
Let \( S \subseteq Q_2 \) such that \( t \in \bigcap_{q \in S} \text{dom}(q) \). Then \( t \) be producible by the state \((q_1, S)\) of \( \tilde{T}_1 \) on input \( s \).

Proof. We prove our claim by structural induction. Let \( s = a(s_1, \ldots, s_k), a \in \Sigma_k \), \( k \geq 0 \), and \( s_1, \ldots, s_k \in T_\Sigma \). As \( t \) be producible by the state \( q_1 \) of \( T_1 \) on input \( s \), \( \xi \in \text{rhs}_{T_1}(q_1, a) \) exists such that

\[
t \in \xi[q'(x_i) \leftarrow [q']^{T_i}_1(s_i) \mid q' \in Q_1, i \in [k]].
\]

In essence, this means that

\[
t \in \xi[u \leftarrow t/u \mid u \in V(\xi), \xi[u] \text{ is of the form } q'(x_i)].
\]

Hence, it follows that if a node \( u \) is labeled by \( q'(x_i) \) in \( \xi \) then

\[
[q']^{T_i}_1(s_i) \Rightarrow t/u.
\]

(*)
By our premise \( t \in \bigcap_{q \in S} \text{dom}(q) \). Due to Lemma 1, it follows that \([S]_A(t) \Rightarrow t\). Therefore, for all leafs \( u \) of \( \xi \) that are labeled by a symbol in of the form \( q_i(x_i) \), a state \( S_u \) of \( A \) exists such that

\[
[S]_A(\xi) \Rightarrow \xi[u \leftarrow S_u(u) \mid u \in V(\xi), \xi[u] \in Q_1(X)]
\]

and \([S_u]_A(t/u) \Rightarrow t/u\). By definition of \( \hat{T}_1 \), the former implies that

\[
(q_1, S)(a(x_1, \ldots, x_k)) \Rightarrow \xi[u \leftarrow (q_1', S_u)(x_i) \mid u \in V(\xi), \xi[u] = q'(x_i)] \quad (\dagger)
\]

is a rule of \( \hat{T}_1 \). The later implies \([S_u]_A(t/u) \Rightarrow t/u\) as \( t_u \) is ground. Therefore, \( t/u \in \bigcap_{q \in S_u} \text{dom}(q) \) due to Lemma 1.

Consider an arbitrary node \( \overset{\dagger}{u} \) of \( \xi \). Let \( \overset{\dagger}{u} \) labeled by \( q_i(x_i) \) in \( \xi \). Then, \( t/\overset{\dagger}{u} \in \bigcap_{q \in S_u} \text{dom}(q) \). Furthermore, due to (\ast\), it follows that \([q_1^{\overline{\hat{T}_1}}(s_i) \Rightarrow t/\overset{\dagger}{u}\).

Then, the induction hypothesis yields that \([\langle q, S_u \rangle]^{\overline{\hat{T}_1}}(s_i) \rightarrow t/\overset{\dagger}{u}\). Note that \([\langle q, S_u \rangle]^{\overline{\hat{T}_1}}(s_i) \rightarrow t/\overset{\dagger}{u}\) implies \([\langle q, S_u \rangle]^{\hat{T}_1}(s_i) \rightarrow t/\overset{\dagger}{u}\) because \( s_i \) is ground. Along with (\dagger), this yields our claim. \( \square \)

Lemmas 12 and 13 allow us to prove the following statement, which implies Lemma 3.

**Lemma 14.** \( \text{dom}(\hat{T}_1) = \text{dom}(T_1 \circ T_2) \) and for \( s \in T_\Sigma \), \( \hat{T}_1(s) = T_1(s) \cap \text{dom}(T_2) \).

**Proof.** First we show that \( \text{dom}(\hat{T}_1) = \text{dom}(T_1 \circ T_2) \). Let \( s \in \text{dom}(\hat{T}_1) \), i.e., a tree \( t \) over \( \Delta \) exists such that \([q_1^0, q_2^0)]^{\overline{\hat{T}_1}}(s) \Rightarrow t\), where \((q_1^0, q_2^0)\) is the initial state of \( \hat{T}_1 \). By construction of \( T_1 \), it follows that \([q_1^0]^{\overline{T_1}}(s) \Rightarrow t\) and by Lemma 12 \( t \in \text{dom}(q_2^0) \). Hence \( s \in \text{dom}(T_1 \circ T_2) \). For the converse, let \( s \in \text{dom}(T_1 \circ T_2) \). Then, a tree \( t \) over \( \Delta \) exists such that \([q_1^0, q_2^0)]^{\overline{T_1}}(s) \Rightarrow t\), where \( q_1^0 \) is the initial state of \( T_1 \), and \( t \in \text{dom}(q_2^0) \). Hence, due to Lemma 13 it follows that \([q_1^0, q_2^0)]^{\overline{T_1}}(s) \Rightarrow t\) and thus, \( s \in \text{dom}(\hat{T}_1) \).

Now we show that \( \hat{T}_1(s) = T_1(s) \cap \text{dom}(T_2) \). Let \([q_1^0, q_2^0)]^{\overline{T_1}}(s) \Rightarrow t\). By construction of \( \hat{T}_1 \), \([q_1^0]^{\overline{T_1}}(s) \Rightarrow t\) holds. By Lemma 12 \( t \in \text{dom}(q_2^0) \). Therefore, our claim follows. Conversely, let \( t \in T_1(s) \cap \text{dom}(T_2) \). Then, clearly \( t \in \text{dom}(q_2^0) \) and \([q_1^0)]^{\overline{T_1}}(s) \Rightarrow t\). By Lemma 13 \([q_1^0, q_2^0)]^{\overline{T_1}}(s) \Rightarrow t\) which yields our claim. \( \square \)

## D Correctness of the LA-Transducer \( M \)

In this section, we present the formal proof of correctness for the la-transducer \( M \), i.e., we show that \( M \) is functional if and only if \( T_1 \circ T_2 \) is functional. Recall that due to Lemma 3 it is sufficient to consider \( T_1 \circ T_2 \).

In the following, denote by \( L \) the set of states of the la-automaton of \( M \). W.l.o.g. we assume that for all states \( l \) in \( L \), \( \text{dom}(l) \neq \emptyset \). In the remainder of this section, our proofs employ partial trees in \( T_\Sigma[L] \). Consider such a tree \( s \). Recall that in a translation of \( M \) input trees are first preprocessed by a relabeling induced by the la-automaton of \( M \). We demand that in a translation
of $M$ the tree $s$ is relabeled as follows: If the $i$-th child of the node $v$ of $s$ is labeled by $l \in L$ then we require that $v$ be relabeled by a symbol of the form $(a_{i-1}...a_i)$. Furthermore, consider the la-automaton $B = \langle \{(p, p')\}, \Sigma, \Sigma, \mathcal{R}, \{p\} \rangle$ where $\Sigma = \{ f^2, a^0, b^0 \}$ and $\mathcal{R}$ contains the rules

$$
\begin{align*}
& p(f(x_1, x_2)) \rightarrow f(p(x_1), p(x_2)) \quad p'(f(x_1, x_2)) \rightarrow f(p(x_1), p'(x_2)) \\
& p'(f(x_1, x_2)) \rightarrow f(p'(x_1), p'(x_2)) \quad p'(f(x_1, x_2)) \rightarrow f(p'(x_1), p(x_2)) \\
& p(a) \rightarrow a \quad p'(b) \rightarrow b.
\end{align*}
$$

Informally, the state $p$ checks whether or not the leftmost leaf of its input tree is $a$. The state $p'$ does the same for $b$. Consider the tree $s = f(a, f(p, b))$. For $s$ the tree $(f, p, p')(a, (f, p', p')(p, b))$ is a valid relabeling. The tree $(f, p, p')(a, (f, p', p')(p, b))$ on the other hand is not.

### D.1 If $M$ is functional then $\hat{T}_1 \circ T_2$ is functional

In this section we formally prove the only-if statement of Corollary \[\texttt{I}\] i.e., we show that if $M$ is functional then $\hat{T}_1 \circ T_2$ is functional. More precisely we formally show that $\mathcal{R}(\hat{T}_1) \circ \mathcal{R}(T_2) \subseteq \mathcal{R}(M)$. Obviously this implies our result.

In the following we formally prove Lemma \[\texttt{I}\] More precisely we prove the following lemma which is a more detailed version of Lemma \[\texttt{I}\].

**Lemma 15.** Let $(q_1, S)$ be a state of $\hat{T}_1$ and $q_2$ be a state of $T_2$. Let $s \in T_\Sigma$. Consider the state $(q_1, S, q_2)$ of $M$. If

$$
[(q_1, S)]^{T_1}(s) \Rightarrow t \quad \text{and} \quad [q_2]^{T_2}(t) \Rightarrow r
$$

then $[(q_1, S, q_2)]^M(s) \Rightarrow r$.

**Proof.** We prove our claim by induction on the structure of $s$. Let $s = \alpha(s_1, \ldots, s_k)$ where $\alpha \in \Sigma_k$, $k \geq 0$, and for $i \in [k]$, $s_1, \ldots, s_k \in T_\Sigma$. First, we prove the following claim.

**Claim 16.** If $[(q_1, S)]^{\hat{T}_1}(s) \Rightarrow t$ and $[q_2]^{T_2}(t) \Rightarrow r$ and $[(q_1, S, q_2)]^M(s) \Rightarrow r$, then trees $\xi$ and $\psi$ exist such that

$$
\xi \in \text{rhs}_{\hat{T}_1}(q_1, S, a) \quad \text{and} \quad [q_2]^{T_2}(\xi) \Rightarrow \psi.
$$

Furthermore, $\xi$ and $\psi$ have the following properties.

1. Let $u$ be a node of $\xi$. If a node of $\psi$ is labeled by $q'_2(u)$ then a state $(q'_1, S')$ of $\hat{T}_1$ exists such that $u$ is labeled by $(q'_1, S')(x_i)$ in $\xi$, where $i \in [k]$, and $q'_1 \in S'$.
2. It holds that

$$
\begin{align*}
& t \in \xi[q(x_i) \leftarrow [q]^{\hat{T}_1}(s_i) \mid q \in Q'_1, i \in [k]] \\
& \text{and} \\
& r \in \psi[q(u) \leftarrow [q]^{T_2}(t/u) \mid q \in Q_2, u \in V(\xi)].
\end{align*}
$$


Proof of Claim. By definition, \( [(q_1,S)]^{\hat{T}_1}(s) \Rightarrow t \) implies \( [q_1]^{T_1}(s) \Rightarrow t \) and thus

\[
t \in \xi'[q(x_i) \leftarrow [q]^{T_1}(s_i) \mid q \in Q_1, i \in [k]]
\]

for some \( \xi' \in \text{rhs}(q_1,a) \). Furthermore, by Lemma 12 \( [(q_1,S)]^{\hat{T}_1}(s) \Rightarrow t \) implies that \( t \in \bigcap_{q' \in S} \text{dom}(q') \). In the following, let \( S = \{q_2^1, \ldots, q_2^n\} \). As \( t \in \bigcap_{q' \in S} \text{dom}(q') \) and due to Equation 1 for all \( j \in [n] \), trees \( \psi_j \) and \( r_j \) exists such that

(a) \( [q_2^j]^{T_2}(\xi') \Rightarrow \psi_j \)
(b) \( [q_2^j]^{T_2}(t) \Rightarrow r_j \) such that

\[
r_j \in \psi_j[q_2^j(u) \leftarrow [q_2^j]^{T_2}(t/u) \mid q_2^j \in Q_2, u \in V(\xi')].
\]

By our premise, the state \((q_1,S,q_2)\) of \( M \) is defined which implies that \( q_2 \in S \). W.l.o.g. let \( q_2^1 = q_2 \). Furthermore, as \( [q_2]^{T_2}(t) \Rightarrow r \), we can also assume that \( r_1 = r \).

Recall that by definition, if a node of \( \psi_j \) is labeled by \( q_2^j(u) \), where \( q_2^j \in Q_2 \) and \( u \) is a node, then the node \( u \) is labeled by some symbol in \( Q_1(X) \) in \( \xi' \). Let \( u_1, \ldots, u_m \) be the nodes of \( \xi' \) that are labeled by a symbol in \( Q_1(X) \).

In the following we first prove Statement (1). Due to Lemma 10 (a) implies

\[
[S]^A(\xi') \Rightarrow \xi'[u_i \leftarrow S_i(u_i) \mid i \in [m]],
\]

where \( A \) is the domain automaton of \( T_2 \) and \( S_i = \bigcup_{j \in [n]} \psi_j(u_i) \), which in turn implies that \( \hat{T}_1 \) contains the rule \((q_1,S)(a(x_1, \ldots, x_k)) \Rightarrow \xi \) where

\[
\xi = \xi'[u_i \leftarrow (q', S_i)(x_i) \mid \xi[u_i] = q'(x_i), i \in [k]]
\]

as \( \hat{T}_1 \) is obtained from the p-construction of \( T_1 \) and \( A \). In the following, consider the node \( u_i \). Assume that in \( \psi_j \) a node labeled by \( q_2^j(u_i) \) occurs. Recall that this means that the node \( u_i \) is labeled by a symbol of the form \( q'(x_i) \) in \( \xi' \). By construction, \( u_i \) is labeled by \( q'(x_i) \) in \( \xi' \) if and only if \( u_i \) is labeled by \( q'(x_i) \) in \( \xi' \). As \( S_i = \bigcup_{j \in [n]} \psi_j(u_i) \), obviously \( q_2^j \in S_i \).

As \( q_2^2 = q_2 \), Statement (1) follows with \( \psi = \psi_1 \). Note that clearly, for all \( j \in [n] \), it holds that

\[
[q_2^j]^{T_2}(\xi') \Rightarrow \psi_j \quad \text{if and only if} \quad [q_2^j]^{T_2}(\xi) \Rightarrow \psi_j.
\]

We now prove Statement (2). In particular, we prove the first part of Statement (2), i.e., that

\[
t \in \xi'[q(x_i) \leftarrow [q]^{T_1}(s_i) \mid q \in Q_1', i \in [k]].
\]

Let the node \( u_i \) be labeled by \((q_1', S_i)(x_i)\) in \( \xi \). Consider an arbitrary state \( q_2^j \in S_i \) where \( i \in [m] \). In particular, this means that \( q_2^j \in \psi_j(u_i) \) for some \( j \in [n] \). In other words, a node \( g \) exists such that \( g \) is labeled by \( q_2^j(u_i) \) in \( \psi_j \). Clearly, Statement (b) implies that \( q_2^j \) can produce the tree \( r_j/g \) on input \( t/u_i \). This
Proof. Let \( \eta \) be obtained from a rule \((q_1, S)(a(x_1, \ldots, x_k)) \rightarrow \xi \) of \( \hat{T}_1 \) such that for \( i \in [k] \), \( \xi(x_i) \subseteq l_i \). The application of \( \eta \) implies \( s_i \in l_i \) for \( i \in [k] \). This implies \( s \in \text{dom}(\{(q_1, S)\}) \).

The second part of Statement (2) follows due to Statement (b) and Equation \( \text{ Equation 2 } \). 

Let \( \xi \) and \( \psi \) be as in Claim \( \text{ Claim 16 } \). Due to Statement (1) of Claim \( \text{ Claim 16 } \), it follows that \( M \) contains the rule

\[
(q_1, S, q_2)(a(x_1: l_1, \ldots, x_k: l_k)) \rightarrow \gamma
\]

where \( \gamma \) is obtained from \( \psi \) by substituting occurrences of \( q_2'(u) \) in \( \psi \), where \( q_2' \in Q_2 \) and \( u \) is a leaf of \( \xi \) labeled by a symbol of the form \((q_1', S')(x_i)\) by \((q_1', S', q_2')(x_i)\). Furthermore, for \( i \in [k] \), \( \xi(x_i) = l_i \).

We now show that \( [[(q_1, S, q_2)]^M(s) \Rightarrow r] \). Recall that \( s = a(s_1, \ldots, s_k) \) where \( a \in \Sigma_k, k \geq 0 \), and for \( i \in [k] \), \( s_1, \ldots, s_k \in T_\Sigma \). Note that as \( \xi(x_i) = l_i \) and due to Statement (2) of Claim \( \text{ Claim 16 } \) and Lemma \( \text{ Lemma 4 } \), \( s_i \in \text{dom}(l_i) \) for \( i \in [k] \).

Consider a node \( g \). By definition of \( \gamma \), \( g \) is labeled by \((q_1', S', q_2')(x_i)\) in \( \gamma \) if and only if \( g \) is labeled by \( q_2'(u) \) in \( \psi \) and the node \( u \) is labeled by \((q_1', S')(x_i)\) in \( \xi \). Statement (2) of Claim \( \text{ Claim 16 } \) implies that \( [[(q_1', S')]^T_{\hat{T}_1}(s_i) \Rightarrow t/u \) and \( [[q_2']^T_{\hat{T}_1}(t/u) \Rightarrow r/g] \). Therefore, by induction hypothesis, \( [[(q_1', S', q_2')]^M(s_i) \Rightarrow r/g \). Clearly, our claim follows.

Clearly, Lemma \( \text{ Lemma 15 } \) implies Lemma \( \text{Lemma 4} \). Lemma \( \text{Lemma 15} \) also yields the following two auxiliary results.

**Lemma 17.** Let \((q_1, S)\) be a state of \( \hat{T}_1 \) and \( q_2 \) be a state of \( T_2 \) such that \( q_2 \in S \). Then, for the state \((q_1, S, q_2)\) of \( M \), \( \text{dom}((q_1, S, q_2)) = \text{dom}((q_1, S)) \) holds.

**Proof.** Let \( s = a(s_1, \ldots, s_k) \), \( a \in \Sigma_k, k \geq 0 \) and \( s_1, \ldots, s_k \in T_\Sigma \). Let \( s \in \text{dom}((q_1, S, q_2)) \), i.e., \( [[(q_1, S, q_2)]^M(s) \Rightarrow r \) for some tree \( r \). Consider the first rule of \( M \) applied in this translation. Let

\[
\eta = (q_1, S, q_2)(a(x_1: l_1, \ldots, x_k: l_k)) \rightarrow \gamma
\]

be this rule. By construction \( \eta \) is obtained from a rule \((q_1, S)(a(x_1, \ldots, x_k)) \rightarrow \xi \) of \( \hat{T}_1 \) such that for \( i \in [k] \), \( \xi(x_i) \subseteq l_i \). The application of \( \eta \) implies \( s_i \in l_i \) for \( i \in [k] \). This implies \( s \in \text{dom}((q_1, S)) \).
Conversely, let \( s \in \text{dom}((q_1, S)) \), i.e., \( [(q_1, S)](s) \rightarrow t \) for some tree \( t \). Note that the state \((q_1, S, q_2)\) of \( M \) implies \( q_2 \in S \). Due to Lemma 12 it follows that \( \left[ q_2 \right]^{T_2}(t) \neq \emptyset \). Therefore, we deduce that due to Lemma 15 \( s \in \text{dom}((q_1, S, q_2)) \).

\[
\text{Lemma 18. Let } s \in T_\Sigma[L]. \text{ Let } M(s) \Rightarrow r_M \text{ and let } (q_1, S, q_2)(v) \text{ occurs in } r_M, \text{ where } (q_1, S, q_2) \text{ is a state of } M \text{ and } v \text{ is a node of } s \text{ labeled by a symbol } l \in L. \text{ Then } \text{dom}(l) \subseteq \text{dom}((q_1, S, q_2)).
\]

**Proof.** Let the parent node of \( v \) be labeled by \( a \in \Sigma_k \) where \( k > 0 \). W.l.o.g., let \( v \) be the first child of its parent node. Then, clearly the occurrence of \((q_1, S, q_2)(v)\) in \( r_M \) originates from the application of a rule \((q'_1, S', q'_2)(a(x_1 : l_1, \ldots, x_k : l_k)) \rightarrow \gamma \) of \( M \) such that \((q_1, S, q_2)(x_1) \) occurs in \( \gamma \). Recall that by definition, \( l_1, \ldots, l_k \) are sets of states of \( T_1 \). By the definition of relabelings of trees in \( T_\Sigma[L] \), the parent node of \( v \) is relabeled by a symbol of the form \((a, l, l'_2, \ldots, l'_k)\) which implies \( l = l_1 \).

Consider the rule \((q'_1, S', q'_2)(a(x_1 : l_1, \ldots, x_k : l_k)) \rightarrow \gamma \). Recall that by construction of \( M \), this rule is obtained from a rule \((q'_1, S')(a(x_1, \ldots, x_k)) \rightarrow \xi \) of \( T_1 \) such that for \( i \in [k] \), \( \xi(x_i) \subseteq l_i \). Note that the occurrence of \((q_1, S, q_2)(x_1)\) in \( \gamma \) implies that \((q_1, S)(x_1)\) occurs in \( \xi \). Therefore, the state \((q_1, S)\) of \( T_1 \) is included in \( l \). As \( s \in \text{dom}(l) \) if and only if \( s \in \bigcap_{q \in l} \text{dom}(q) \), our claim follows due to Lemma 17. 

### D.2 If \( \hat{T}_1 \circ T_2 \) is functional then \( M \) is functional.

In this section we formally prove the only-if statement of Corollary 1, i.e., we show that if \( \hat{T}_1 \circ T_2 \) is functional then \( M \) is functional.

First we introduce the following definition. Recall that we have introduced synchronized translations of \( M \) in Section 3.4. In the following, we extend this definition. Let \( s \in T_\Sigma[L] \). We call the trees \( s, t, r \) and \( r_M \) **synchronized** if

1. \( \hat{T}_1(s) \Rightarrow t \) and \( T_2(t) \Rightarrow r \) and \( M(s) \Rightarrow r_M \) and
2. the tree \( r_M \) is obtained from \( r \) by substituting all occurrences of \( q'_2(u) \) in \( r \) by \((q'_1, S', q'_2)(v)\), where \((q'_1, S')\) and \( q'_2 \) are states of \( T_1 \) and \( T_2 \), respectively, and \( u \) is a leaf of \( t \) labeled by \((q'_1, S')(v)\).

Informally, \( s, t, r \) and \( r_M \) are synchronized if on input \( s \), \( M \) produces the tree \( r_M \) by accurately simulating \( \hat{T}_1 \circ T_2 \). More precisely: Recall that when a state \((q_1, S, q_2)\) of \( M \) processes a subtree \( s' \) of \( s \) then \((q_1, S, q_2)\) guesses what the state \((q_1, S)\) of \( T_1 \) might have produced before producing output according to this guess. Informally, if all such guesses of \( M \) are correct, i.e., the states of \( \hat{T}_1 \) have indeed produced the trees \( M \) has guessed, then \( s, t, r \) and \( r_M \) are synchronized.

Before we prove a more detailed version of Lemma 6 recall that by definition, a state \( l \) in \( L \) is a set of states of \( T_1 \). Consider a tree \( s \in T_\Sigma[L] \). Informally, if a symbol \( l \in L \) occurs at some leaf of \( s \) then \( l \) can be considered a placeholder for some tree \( s' \) such that \( s' \in \bigcap_{(q_1, S) \in l} \text{dom}(q_1, S) \). We now show that the following holds.

Lemma 19. Let $s \in T_L$. Let $M(s) \Rightarrow r_M$ and let $(q_1, S, q_2)(v)$ occur in $r_M$. Then trees $t$, $r$ and $r'_M$ exist such that $s$, $t$, $r$ and $r'_M$ are synchronized and $(q_1, S, q_2)(v)$ occurs in $r'_M$. Furthermore, let $v'$ be a leaf of $s$ that is labeled by $l \in L$. Then, it holds that if $(q'_1, S')(v')$ occurs in $t$ then $(q'_1, S') \in L$.

Proof. We prove our claim by structural induction. First, let $\bar{v}$ be a node of $s$ such that the subtree of $s$ rooted at $\bar{v}$ is of the form $a(l_1, \ldots, l_k)$ where $a \in \Sigma_k$, $k \geq 0$, and $l_1, \ldots, l_k \in L$. Note that by definition $\bar{v}$ can be a leaf. Then, a state $l \in L$ exists such that

$$l(a(x_1, \ldots, x_k)) \rightarrow a(l_1(x_1), \ldots, l_k(x_k))$$

is a rule of the la-automaton of $M$. Furthermore, as $M(s) \Rightarrow r_M$, on input $\bar{s} = s[\bar{v} \mapsto l]$, the $M$ produces the tree $\bar{r}_M$ such that

$$r_M \in \bar{r}_M[\langle q \rangle(\bar{v}) \leftarrow [q_1^M(a(l_1, \ldots, l_k)) \mid q \text{ is a state of } M].$$

We remark that all trees in $[q_1^M(a(l_1, \ldots, l_k))]$ are of the form $\gamma(x_i \leftarrow v.i \mid i \in [k])$ where $\gamma \in \text{rhs}(q, a, l_1, \ldots, l_k)$. Recall that by our premise, $(q_1, S, q_2)(v)$ occurs in $r_M$. Then one of the following cases arises:

(a) $(q_1, S, q_2)(v)$ does not already occur in $\bar{r}_M$.
(b) $(q_1, S, q_2)(v)$ already occurs in $\bar{r}_M$.

First, we consider case (b). By induction hypothesis, as $M(\bar{s}) \Rightarrow \bar{r}_M$ and a node labeled by $(q_1, S, q_2)(v)$ occurs in $\bar{r}_M$, trees $\bar{t}$, $\bar{r}$ and $\bar{r}'_M$ exist such that $\bar{s}$, $\bar{t}$, $\bar{r}$ and $\bar{r}'_M$ are synchronized and $(q_1, S, q_2)(v)$ occurs in $\bar{r}'_M$. Furthermore, by induction hypothesis, it holds that if $(q'_1, S')(v)$ occurs in $\bar{t}$ then $(q'_1, S') \in L$.

First, we construct the tree $t$. Recall that the la-automaton of $M$ is the domain automaton of $\bar{T}_1$. Therefore the existence of rule

$$l(a(x_1, \ldots, x_k)) \rightarrow a(l_1(x_1), \ldots, l_k(x_k))$$

of the la-automaton implies that for all states $(q'_1, S') \in L$, a right-hand side $\xi \in \text{rhs}_{\bar{r}}$, $((q'_1, S'), a)$ exists such that $\xi(x_i) \subseteq l_i$ for $i \in [k]$ (*).

In the following, we define $t_{(q'_1, S')} = \xi(x_i \leftarrow \bar{v}.i \mid i \in [k])$ if $(q'_1, S') \in L$. Then clearly $\bar{T}_1(s) \Rightarrow t$ where

$$t = \bar{t}[(q'_1, S')(\bar{v}) \leftarrow t_{(q'_1, S')} \mid (q'_1, S') \in Q_1].$$

We now show that for arbitrary nodes $v'$ of $s$ it holds that if $v'$ is labeled by $l$ in $s$ and $(q'_1, S')(v')$ occurs in $t$ then $(q'_1, S') \in L$. Due to (*), this holds for all nodes $v'$ that are descendants of $\bar{v}$. Now assume that $v'$ is not a descendant of $\bar{v}$. Let $v'$ be labeled by the symbol $l \in L$ in $s$. Then obviously, the node $v'$ is also labeled by $l$ in $\bar{s}$. Thus, by definition of $\bar{t}$, if $(q'_1, S')(v')$ occurs in $\bar{t}$ then $(q'_1, S') \in L$. By construction of $t$, $(q'_1, S')(v')$ occurs in $\bar{t}$ if and only if $(q'_1, S')(v')$ occurs in $t$. This yields our claim.

We now construct $r$ and $r'_M$. First recall that, by induction hypothesis, the trees $\bar{s}$, $\bar{t}$, $\bar{r}$ and $\bar{r}'_M$ are synchronized. Therefore, for an arbitrary node $g$ the
following holds: $g$ is labeled by $(q_1', S', q_2')(v)$ in $\bar{r}_M$ if and only if $g$ is labeled by $q_2'(u)$ in $\bar{r}$ and $u$ is a node of $\bar{l}$ labeled by $(q_1', S')(v)$ $(\dagger)$.

Now let the node $g$ be labeled by $q_2'(u)$ in $\bar{r}$ and let the node $u$ be labeled by $(q_1', S')(\bar{v})$ in $\bar{l}$. Consider the right-hand side $\xi'$ assigned to the state $(q_1', S')$ in $(\ast)$. Due to how rules of $\bar{T}_1$ are defined, it holds that

$$\left[S'\right]^{A}(\xi') \Rightarrow \xi'[u \leftarrow S(u) \mid u \in V(\xi'), \xi'[u] = (\bar{q}, \bar{S})(x_i)].$$

Note that $(\dagger)$ implies $q_2' \in S$. This follows as the state $(q_1', S', q_2')$ is defined. Therefore by Lemma $\Pi$ a tree $\psi'$ exists such that

1. $\left[q_2'\right]^{T_2}(\xi') \Rightarrow \psi'$ and
2. if the node $u'$ is labeled by $(\bar{q}, \bar{S})(x_i)$ in $\xi'$ then $\psi'(u') \subseteq \bar{S}$.

The later implies that if $\bar{q}_2(u')$ occurs in $\psi'$ then $\bar{q}_2 \in \bar{S}$. Due to $(\ast)$, for $i \in [k]$, it holds that $\xi'(x_i) \subseteq l_i$. Therefore, by construction of $M$ the rule

$$(q_1', S', q_2')(a(x_1 : l_1, \ldots, x_k : l_k)) \Rightarrow \gamma'$$

is defined where $\gamma'$ is obtained from $\psi'$ by substituting occurrences of $\bar{q}_2(u')$ in $\psi$ by $(\bar{q}_1, \bar{S}, \bar{q}_2)(x_i)$, where $(\bar{q}_1, \bar{S})$ and $\bar{q}_2$ are states of $\bar{T}_1$ and $T_2$, respectively, and $u'$ is a leaf of $\xi'$ labeled by a symbol of the form $(\bar{q}_1, \bar{S})(x_i)$.

For the node $g$ we define $r_{T_2,g} = \psi'(u' \leftarrow u.u' \mid u' \in V)$. Additionally, we define $r_{M,g} = \gamma'(x_i \leftarrow \bar{v}, i \mid i \in [k])$.

Recall that $(\dagger)$ holds. Then, $T_2(t) \Rightarrow r$ where

$$r = \bar{r}[g \leftarrow r_{T_2,g} \mid \bar{r}[g] = q_2'(u) \text{ and } \bar{r}[u] = (q_1', S')(\bar{v})]$$

and $M(s) \Rightarrow r_M'$ where

$$r_M' = \bar{r}_M[g \leftarrow r_{M,g} \mid \bar{r}_M[g] = (q_1', S', q_2')(\bar{v})].$$

Note that the node $\bar{v}$ of $s$ is relabeled by $(a, l_1, \ldots, l_k)$ via the relabeling induced by the rule $l(a(x_1, \ldots, x_k)) \rightarrow a(l_1(x_1), \ldots, l_k(x_k))$ of the $a$-automaton of $M$. Thus, $r_M'$ is well defined. Clearly, $\bar{T}_1(s) \Rightarrow t$ and $T_2(t) \Rightarrow r$ and $M(s) \Rightarrow r_M'$. Due to $(\dagger)$ and the construction of $r$ and $r_M'$, it follows that the second part of the synchronized-property holds as well.

We now consider case (a). As $(q_1, S, q_2)(v)$ occurs in $r_M$ but not in $\bar{r}_M$, it follows that $v = \bar{v}, i$ for some $i \in [k]$. W.l.o.g. let $v = \bar{v}, 1$, i.e., $v$ is the first child of the node $\bar{v}$. Furthermore, it follows that a rule

$$(\bar{q}_1, \bar{S}, \bar{q}_2)(a(x_1 : l_1, \ldots, x_k : l_k)) \rightarrow \bar{\gamma}$$

exists such that $(\bar{q}_1, \bar{S}, \bar{q}_2)(\bar{v})$ occurs in $\bar{r}_M$ and $(q_1, S, q_2)(x_1)$ occurs in $\bar{\gamma}$. Let the rule of $M$ above be obtained from the rule $(\bar{q}_1, \bar{S})(a(x_1, \ldots, x_k)) \rightarrow \xi$ of $\bar{T}_1$ and subsequently translating $\xi$ by the state $\bar{q}_2$ of $T_2$. In particular, this means that a tree $\psi$ exists such that
(a) $\bar{q}_2I^2(\xi) \Rightarrow \bar{\psi}$ and

(b) $\tilde{\gamma}$ is obtained from $\psi$ by substituting occurrences of $q'_2(u)$ in $\bar{\psi}$ by $(q'_1, S', q'_2)(x_1)$, where $(q'_1, S')$ and $q'_2$ are states of $T_1$ and $T_2$, respectively, and $u$ is a leaf of $\xi$ labeled by $(q'_1, S')(x_1)$.

By induction hypothesis, as $M(\bar{s}) \Rightarrow r_M$ and a node labeled by $(\bar{q}_1, \bar{S}, \bar{q}_2)(\bar{v})$ occurs in $\bar{r}_M$, trees $\bar{t}$, $\bar{r}$ and $\bar{r}'_M$ exist such that $\bar{s}$, $\bar{t}$, $\bar{r}$ and $\bar{r}'_M$ are synchronized.

Let the node $\tilde{g}$ be labeled by $(\bar{q}_1, \bar{S}, \bar{q}_2)(\bar{v})$ in $\bar{r}'_M$. Due to the synchronized property, the node $\tilde{g}$ is labeled by $q_2(\bar{u})$ in $\bar{r}$, where $\bar{u}$ is a node that is labeled by $(q_1, S)(\bar{v})$ in $\bar{t}$.

To construct the trees $t$, $r$ and $r'_M$, we then proceed as in case (b) but set

\[ t_{(\bar{q}_1, \bar{S})} = \tilde{\xi}(x_i \leftarrow \bar{v}.i \mid i \in [k]), \]
\[ r_{T_2, \tilde{g}} = \tilde{\psi}(u' \leftarrow \bar{u}.u' \mid u' \in V) \text{ and} \]
\[ r_{M, \tilde{g}} = \tilde{\gamma}(x_i \leftarrow \bar{v}.i \mid i \in [k]). \]

This yields our claim. \hfill \Box

Lemma 19 and and Proposition 1 allow us to formally prove the following version of Lemma 17.

**Lemma 20.** Let $s \in T_2[L]$ such that only a single node $v$ of $s$ is labeled by a symbol in $L$. Let $v$ be labeled by $l \in L$. Let $M(s) \Rightarrow r_M$ such that $(q_1, S, q_2)(v)$ occurs in $r_M$.

Consider the tree $\bar{s} = s[v \leftarrow s']$ where $s' \in \text{dom}(l)$. If $T_1 \circ T_2(\bar{s})$ is a singleton then $[(q_1, S, q_2)](s')$ is a singleton.

**Proof.** Note that by Lemma 18 $s' \in \text{dom}((q_1, S, q_2))$. Hence, $[(q_1, S, q_2)](s') \neq \emptyset$. Assume that $[(q_1, S, q_2)](s')$ is not a singleton, i.e., assume that distinct trees $r_1, r_2$ exist such that $r_1, r_2 \in [(q_1, S, q_2)](s')$.

We claim that for $r_1$, a tree $t_1$ exists such that

1. on input $s'$, the state $(q_1, S)$ of $T_1$ produces $t_1$ and
2. on input $t_1$, the state $q_2$ of $T_2$ produces $r_2$.

We will later prove this claim in detail. It can be shown that a tree $t_2$ with the same properties exists for $r_2$.

Using this claim and Proposition 1 we now prove that contrary to the assumption $r_1 = r_2$.

Due to Lemma 19 as $M(s) \Rightarrow r_M$ and $(q_1, S, q_2)(v)$ occurs in $r_M$, it follows that trees $t$, $r$ and $r'_M$ exist such that $s$, $t$, $r$ and $r'_M$ are synchronized and $(q_1, S, q_2)(v)$ occurs in $r'_M$. Moreover, if $(q', S')(v)$ occurs in $t$, where $(q', S')$ is some state of $T_1$, then $(q', S') \in l$. Recall that by our premise, $v$ is labeled by $l$ in $s$. Therefore, $\text{dom}(l) \subseteq \text{dom}((q', S'))$ due to Lemma 1. Consequently, $s' \in \text{dom}((q', S'))$. Therefore, for all states $(q', S')$ of $T_1$ such that $(q', S')(v)$ occurs in $t$, a tree $t'$ exists such that $[(q', S')]T_1(s') \Rightarrow t'$. In the following, let
Before, we prove our claim consider the following. Let
Claim 21. statement of Lemma 20.

Let \( q_2' \) be a state of \( T_2 \) and \( u \) be a node. By definition of

\( t \) and \( r \), if \( q_2'(u) \) occurs in \( r \), then the node \( u \) is labeled by a symbol of the form

\( (q', S', q_2') \) in \( t \). Furthermore, the synchronized property implies that \( q_2' \in S' \).

This follows as a state \( (q', S', q_2') \) of \( M \) has the property that \( q_2' \in S' \). The
subtree of \( \hat{t} \) rooted at \( u \) is a tree \( t' \) such that \( [(q', S')]^{\hat{T}_1} \Rightarrow t' \). By Lemma \[20\]
it follows that \( t' \in \text{dom}(q_2) \). Therefore, it follows easily that \( T_2(t) \Rightarrow r \) where

\[
\hat{r} = r[q_2'(u) \leftarrow r_u \mid q_2' \in Q_2, \hat{t}[u] = t'_j \text{ and } [q_2']^{T_2}(t'_j) \Rightarrow r_u].
\]

By our premise a node \( g \) exists such that \( g \) is labeled by \( (q_1, S, q_2)(v) \) in

\( r'_M \). As the trees \( s, t, r \) and \( r'_M \) are synchronized, \( g \) is labeled by \( q_2(u) \) in \( r \)
where \( u \) is a node of \( t \) such that \( t[u] = (q_1, S)(v) \). Due to our claim, a tree \( t_1 \)
exists such that \( [(q_1, S)]^{\hat{T}_1}(s') \Rightarrow t_1 \) and and \([q_2]^{T_2}(t_1) \Rightarrow r_1 \). W.l.o.g. we assume that \( (q_1, S') = (q_1, S) \) and \( t_1 = t_2 \). Then, it follows easily that on input \( \hat{s} \), the
composition \( \hat{T}_1 \circ T_2 \) can produce a tree \( \hat{r}_1 \) such that \( \hat{r}_1/g = r_1 \). Analogously, it
follows easily that on input \( \hat{s} \), the composition \( \hat{T}_1 \circ T_2 \) can produce a tree \( \hat{r}_2 \) such
that \( \hat{r}_2/g = r_2 \). Due to Proposition \[16\] \( \hat{r}_1 = \hat{r}_2 \) and therefore

\[
r_1 = \hat{r}_1/g = \hat{r}_2/g = r_2.
\]

Now all that is left is to prove our previous claim that for \( r_1 \) and \( r_2 \), trees \( t_1 \) and \( t_2 \) exist such that

\[
\begin{align*}
- \quad [(q_1, S)]^{\hat{T}_1}(s') \Rightarrow t_1 & \text{ and and } [q_2]^{T_2}(t_1) \Rightarrow r_1 \\
- \quad [(q_1, S)]^{\hat{T}_1}(s') \Rightarrow t_2 & \text{ and and } [q_2]^{T_2}(t_2) \Rightarrow r_2.
\end{align*}
\]

We prove our claim for \( r_1 \). The proof for \( r_2 \) is analogous. Let \( s' = a(s_1, \ldots, s_k) \)
where \( a \in \Sigma_k, k \geq 0, \) and \( s_1, \ldots, s_k \in T_2 \). As \( r_1 \) is producible by \( (q_1, S, q_2) \) on
input \( s' \), it follows that

\[
r_1 \in \gamma[q_M(x_i) \leftarrow [q_M](s_i) \mid i \in [k] \text{ and } q_M \text{ is a state of } M]\]

(2)

where \( (q_1, S, q_2)(a(x_1: l_1, \ldots, x_k : l_k)) \rightarrow \gamma \) is a rule of \( M \), \( l_1, \ldots, l_k \) are states of the la-automaton of \( M \) and for \( i \in [k], s_i \in \text{dom}(l_i) \).

Before we prove our claim, we prove the following result by induction on the
statement of Lemma \[20\]

Claim 21. Let \( q_M \) be a state of \( M \) and let \( q_M(x_i) \) occur \( \gamma \) where \( i \in [k] \). Then
the set \([q_M](s_i)\) is a singleton.

Proof of Claim. Before, we prove our claim consider the following. Let \( q'_M \) be a
state of \( M \). Then, by Lemma \[18\] \( s' \in \text{dom}(q'_M) \), if \( q'_M(v) \) occurs in \( r_M(\hat{t}) \).

Now, we prove our claim. W.l.o.g., we consider the case \( i = 1 \). Consider the
tree \( \bar{s} = a(l_1, s_2, \ldots, s_k) \). Due to (\( \hat{t} \)), it follows that on input \( \bar{s} \), any state \( q'_M \)
such that \(q_M'(v)\) occurs in \(r_M\) can produce some partial tree, i.e., a tree with leaves with label of the form \(\hat{q}_M(v.1)\) where \(\hat{q}_M\) is a state of \(M\). In particular, by our premise, \((q_1, S, q_2)(v)\) occurs in \(r_M\). As the tree \(r_1\) is producible by \((q_1, S, q_2)\) on input \(s'\) by applying the rule \((q_1, S, q_2)(a(x_1: l_1, \ldots, x_k: l_k)) \to \gamma\), it follows easily that on input \(\bar{s}\), the state \((q_1, S, q_2)\) generates a tree \(\bar{r}\) such that \(q_M(v.1)\) occurs in \(\bar{r}\) if \(q_M(x_1)\) occurs \(\gamma\).

Thus, it follows that \(M\) on input \(s[v \leftarrow \bar{s}]\) produces a tree in which \(q_M(v.1)\) occurs. Clearly, the node \(v.1\) is labeled by \(l_1\) in \(\bar{s}\). Note that \(s_1 \in \text{dom}(l_1)\).

By induction hypotheses, \([q_M](s_i)\) is a singleton.

We now prove our main claim.

**Claim 22.** A tree \(t_1\) exists such that \([[(q_1, S)]^{T_1}(s')] \Rightarrow t_1\) and \([q_2]^{T_2}(t_1) \Rightarrow r_1\).

**Proof of Claim.** Let \(i \in [k]\) and \(q_M(x_i)\) occur in \(\gamma\). By Claim 21, the set \([q_M](s_i)\) is a singleton. Let \([q_M](s_i) = \{r'\}\). Let \(q_M = \langle q_1', S', q_2' \rangle\) where \(q_1', S'\) and \(q_2'\) are states of \(T_1\) and \(T_2\), respectively. Due to Lemma 17, \(s_i \in \text{dom}(q_M)\) implies \(s_i \in \text{dom}((q_1', S'))\), i.e., \([[(q_1', S')]^{T_i}(s_i)\) is not empty. In the following, we show that if \([q_M](s_i) = \{r'\}\) then for all trees \(t'\) contained in \([[(q_1', S')]^{T_i}(s_i)\), it holds that \([q_2]^{T_2}(t') = \{r'\}\) (*).

In the following, consider such a tree \(t'\). By Lemma 12

\[
t' \in \bigcap_{q_2 \in S'} \text{dom}(q_2)
\]

and thus \(t' \in \text{dom}(q_2')\). Recall that by definition the state \(q_M = \langle q_1', S', q_2' \rangle\) implies \(q_2' \in S'\). Therefore, the set \([q_2']^{T_2}(t')\) is not empty. As \([q_M](s_i) = \{r'\}\), we deduce that \([q_2']^{T_2}(t') = \{r'\}\) due to Lemma 15. Therefore, (*) follows.

By definition, the rule \((q_1, S, q_2)(a(x_1: l_1, \ldots, x_k: l_k)) \to \gamma\) of \(M\) is defined only if a rule \((q_1, S)(a(x_1, \ldots, x_k)) \to (q_1', S', l_i)\) of \(T_1\) and a tree \(\psi\) exist such that

1. \([q_2']^{T_2}(\xi) \Rightarrow \psi\)
2. the tree \(\gamma\) is obtained from \(\psi\) by substituting all occurrences of \(q_2'(u)\) in \(\psi\) by \((q_1', S', q_2')(x_i)\), where \((q_1', S')\) and \(q_2'\) are states of \(T_1\) and \(T_2\), respectively, and \(u\) is a leaf of \(\xi\) labeled by \((q_1', S')(x_i)\)
3. for \(i \in [k]\), it holds that \((q_1', S') \in l_i\).

By definition of \(r_1\) (see Equation 2), for \(i \in [k]\), it holds that \(s_i \in l_i\). Therefore, it follows due to Statement 3 that if \((q_1', S')(x_i)\) occurs in \(\xi\) then \(s_i \in \text{dom}((q_1', S'))\). Thus, \([[(q_1, S)]^{T_1}(s_i)] \Rightarrow t_1\) where

\[
t_1 \in \xi_1[(q_1', S')(x_i)] \Rightarrow [(q_1', S')]^{T_1}(s_i) \in Q_1', i \in [k]
\]

and \([q_2']^{T_2}(t_1) \Rightarrow r_1'\) where

\[
r_1' \in \psi[q_2'(u) \leftarrow [q_2']^{T_2}(t_1/u) \mid q_2' \in Q_2, u \in V(\xi)].
\]

Note that the tree \(t_1/u\) is produced by the state \((q_1', S')\) on input \(s_i\) if \(\xi[u] = (q_1', S')(x_i)\). We remark that the node \(g\) is labeled by \(q_2(u)\) in \(\psi\) where \(u\) is a
node of $\xi$ such that $u$ is labeled by a symbol of the form $(q'_1, S')(x_i)$ if and only if $g$ is labeled by $(q'_1, S', q'_2)(x_i)$ in $\gamma$ due to Statement 2. Due to ($\ast$), it follows that

$$\left[(q'_1, S', q'_2)\right]^M(s_i) = \{r'\} = \left[q'_2\right]^{T_2}(t_1/u).$$

Therefore, ($\ast$) yields $r'_1/g = r_1/g$. Due to the definition of $\gamma$ and $\psi$, i.e. Statement 2, our claim follows.

\[ \square \]