Deciding Equivalence of Linear Tree-to-Word Transducers in Polynomial Time

Adrien Boiret, Raphaela Palenta

Deciding Equivalence of Linear Tree-to-Word Transducers in Polynomial Time

Adrien Boiret[1,⋆] and Raphaela Palenta[2]

[1] CRIStAL, University Lille 1, Avenue Carl Gauss, 59655 Villeneuve d'Ascq Cedex, France, adrien.boiret@inria.fr

[2] Department of Informatics, Technical University of Munich, Boltzmannstr. 3, 85748 Garching, Germany, palenta@in.tum.de

Abstract. We show that the equivalence of linear top-down tree-to-word transducers is decidable in polynomial time. Linear tree-to-word transducers are non-copying but not necessarily order-preserving and can be used to express XML and other document transformations. The result is based on a partial normal form that provides a basic characterization of the languages produced by linear tree-to-word transducers.

Keywords: Tree Transducer, Deciding Equivalence, Partial Normal Form

1 Introduction

Tree transformations are widely used in functional programming and document processing. Tree transducers are a general model for transforming structured data like a database in a structured or even unstructured way. Consider the following internal representation of a client database that should be transformed to a table in HTML.

```
{ {
   name: "Alexander"
   surname: "Walker"
   nickname: "Alex"
   title: "Prof."
   salutation: "Mr."
  }
  ...
}
```

→

```
<table>
  <tr>  <th> name <\th>
        <th> surname <\th>
        <th> nickname <\th>
        <th> title <\th>
        <th> salutation <\th> <\tr>
  <tr>  <td> Alexander <\td>
        <td> Walker <\td>
        <td> Alex <\td>
        <td> Prof. <\td>
        <td> Mr. <\td> <\tr>
  ...
<\table>
```

Deterministic top-down tree transducers can be seen as functional programs that transform trees from the root to the leaves with finite memory. Transformations where the output is not produced in a structured way or where, for example, the output is a string, can be modeled by tree-to-word transducers.

⋆ This work was partially supported by a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020

In this paper, we study deterministic linear tree-to-word transducers (LTWs), a subset of deterministic tree-to-word transducers that are non-copying, but not necessarily order-preserving. Processing the subtrees in an arbitrary order is important to avoid reordering of the internal data for different use cases. In the example of the client database the names may be needed in different formats, e.g.

```
<salutation> <name> <surname>
<surname>, <name>
<title> <surname>
<title> <surname>, <name>
```

The equivalence of unrestricted tree-to-word transducers was a long standing open problem that was recently shown to be decidable [12]. The algorithm by [12] provides an co-randomized polynomial algorithm for linear transducers. We show that the equivalence of LTWs is decidable in polynomial time and provide a partial normal form.

To decide equivalence of LTWs, we start in Section 3 by extending the methods used for sequential (linear and order-preserving) tree-to-word transducers (STWs), discussed in [13]. The equivalence for these transducers is decidable in polynomial time [13]. Moreover a normal form for sequential and linear tree-to-word transducers, computable in exponential time, is known [7, 1]. Two equivalent LTWs do not necessarily transform their trees in the same order. However, the differences that can occur are quite specific and characterized in [1]. We show how they can be identified. We use the notion of *earliest* states, inspired by the existing notion of earliest sequential transducers [7]. In this earliest form, two equivalent STWs can transform subtrees in different orders only if they fulfill specific properties pertaining to the periodicity of the words they create. Computing this normal form is exponential in complexity as the number of states may increase exponentially. To avoid this size increase we do not compute these earliest transducers fully, but rather locally. This means we transform two LTWs with different orders to a *partial normal form* in polynomial time (see Section 4) where the order of their transformation of the different subtrees are the same. LTWs that transform the subtrees of the input in the same order can be reduced to sequential tree-to-word transducers as the input trees can be reordered according to the order in the transformation.

Due to space constraints some proofs are omitted. The full version of the paper can be found at `http://arxiv.org/abs/1606.03758`.

Related Work. Different other classes of transducers, such as tree-to-tree transducers [5], macro tree transducers [6] or nested-word-to-word transducers [13] have been studied. Many results for tree-to-tree transducers are known, e.g. deciding equivalence [10], minimization algorithms [10] and Gold-style learning algorithms [8]. In contrast, transformations where the output is not generated in a structured way like a tree are not that well understood. In macro-tree transducers, the decidability of equivalence is a well-known and long-standing question [2]. However, the equivalence of linear size increase macro-tree transducers that are equivalent to MSO definable transducers is decidable [3, 4].

2 Preliminaries

Let Σ be a ranked alphabet with $\Sigma^{(n)}$ the symbols of rank n. Trees on Σ ($\mathcal{T}_\Sigma$) are defined inductively: if $f \in \Sigma^{(n)}$, and $t_1, ..., t_n \in \mathcal{T}_\Sigma$, then $f(t_1, ..., t_n) \in \mathcal{T}_\Sigma$ is a tree. Let Δ be an alphabet. An element $w \in \Delta^*$ is a word. For two words u, v we denote the concatenation of these two words by uv. The length of a word w is denoted by $|w|$. We call ε the empty word. We denote a^{-1} the inverse of a symbol a where $aa^{-1} = a^{-1}a = \varepsilon$. The inverse of a word $w = u_1 \dots u_n$ is $w^{-1} = u_n^{-1} \dots u_1^{-1}$.

A *context-free grammar* (CFG) is defined as a tuple (Δ, N, S, P), where Δ is the alphabet of G, N is a finite set of *non-terminal symbols*, $S \in N$ is the initial non-terminal of G, P is a finite set of rules of form $A \to w$, where $A \in N$ and $w \in (\Delta \cup N)^*$. A CFG is deterministic if each non-terminal has at most one rule.

We define the language $L_G(A)$ of a non-terminal A recursively: if $A \to u_0 A_1 u_1 ... A_n u_n$ is a rule of P, with u_i words of Δ^* and A_i non-terminals of N, and w_i a word of $L_G(A_i)$, then $u_0 w_1 u_1 ... w_n u_n$ is a word of $L_G(A)$. We define the context-free language L_G of a context-free grammar G as $L_G(S)$.

A *straight-line program* (SLP) is a deterministic CFG that produces exactly one word. The word produced by an SLP (Δ, N, S, P) is called w_S.

We denote the *longest common prefix of all words* of a language L by $\mathsf{lcp}(L)$. Its *longest common suffix* is $\mathsf{lcs}(L)$.

A word u is said to be *periodic* of period w if w is the smallest word such that $u \in w^*$. A language L is said to be *periodic* of period w if w is the smallest word such that $L \subseteq w^*$.

A language L is *quasi-periodic* on the left (resp. on the right) of handle u and period w if w is the smallest word such that $L \subseteq uw^*$ (resp. if $L \subseteq w^*u$). A language is quasi-periodic if it is quasi-periodic on the right or left. If L is a singleton or empty, it is periodic of period ε. Iff L is periodic, it is quasi-periodic on the left and the right of handle ε. If L is quasi-periodic on the left (resp. right) then $\mathsf{lcp}(L)$ (resp. $\mathsf{lcs}(L)$) is the shortest word of L.

3 Linear Tree-to-Word Transducers

A *linear tree-to-word transducer* (LTW) is a tuple $M = (\Sigma, \Delta, Q, \mathsf{ax}, \delta)$ where

- Σ is a ranked alphabet,
- Δ is an alphabet of output symbols,
- Q is a finite set of states,
- the axiom ax is of the form $u_0 q(x) u_1$, where $q \in Q$ and $u_0, u_1 \in \Delta^*$,
- δ is a set of rules of the form $q, f \to u_0 q_1(x_{\sigma(1)}) \dots q_n(x_{\sigma(n)}) u_n$ where $q, q_1, \dots, q_n \in Q$, $f \in \Sigma$ of rank n, $u_0, \dots, u_n \in \Delta^*$ and σ is a permutation from $\{1, \dots, n\}$ to $\{1, \dots, n\}$. There is at most one rule per pair q, f.

The partial function $[\![M]\!]_q$ of a state q on an input tree $f(t_1, \dots, t_n)$ is defined inductively as

- $u_0 [\![M]\!]_{q_1}(t_{\sigma(1)}) \dots [\![M]\!]_{q_n}(t_{\sigma(n)}) u_n$, if $q, f \to u_0 q_1(x_{\sigma(1)}) \dots q_n(x_{\sigma(n)}) u_n \in \delta$
- undefined, if q, f is not defined in δ.

The partial function $[\![M]\!]$ of an LTW M with axiom $u_0q(x)u_1$ on an input tree t is defined as $[\![M]\!](t) = u_0[\![M]\!]_q(t)u_1$.

Two LTWs M and M' are equivalent if $[\![M]\!] = [\![M']\!]$.

A *sequential tree-to-word transducer* (STW) is an LTW where for each rule of the form $q, f \to u_0q_1(x_{\sigma(1)})u_1 \dots q_n(x_{\sigma(n)})u_n$, σ is the identity on $1 \dots n$.

We define *accessibility* of states as the transitive and reflexive closure of appearance in a rule. This means state q is accessible from itself, and if $q, f \to u_0q_1(x_{\sigma(1)}) \dots q_n(x_{\sigma(n)})u_n$, and q is accessible from q', then all states q_i, $1 \leq i \leq n$, are accessible from q'.

We denote by $\mathsf{dom}(M)$ (resp. $\mathsf{dom}(q)$) the domain of an LTW M (resp. a state q), i.e. all trees $t \in \mathcal{T}_\Sigma$ such that $[\![M]\!](t)$ is defined (resp. $[\![M]\!]_q(t)$). We only consider LTWs with non-empty domains and assume w.l.o.g. that no state q in an LTW has an empty domain by eliminating transitions using states with empty domain.

We denote by L_M (resp. L_q) the range of $[\![M]\!]$ (resp. $[\![M]\!]_q$), i.e. the set of all images $[\![M]\!](t)$ (resp. $[\![M]\!]_q(t)$). The languages L_M and L_q for each $q \in Q$ are all context-free languages. We call a state q *(quasi-)periodic* if L_q is (quasi-)periodic.

Note that a word u in a rule of an LTW can be represented by an SLP without changing the semantics of the LTW. Therefore a set of SLPs is added to the transducer and a word on the right-hand side of a rule is represented by an SLPs. The decidability of equivalence of STWs in polynomial time still holds true with the use of SLPs.

The results of this paper require SLP compression to avoid exponential blow-up. SLPs are used to prevent exponential blow-up in [11], where morphism equivalence on context-free languages is decided in polynomial time.

The equivalence problem for sequential tree-to-word transducer can be reduced to the morphism equivalence problem for context-free languages [13]. This reduction relies on the fact that STWs transform their subtrees in the same order. As LTWs do not necessarily transform their subtrees in the same order the result cannot be applied on LTWs in general. However, if two LTWs transform their subtrees in the same order, then the same reduction can be applied. To formalize that two LTWs transform their subtrees in the same order we introduce the notion of state co-reachability. Two states q_1 and q_2 of LTWs M_1, M_2, respectively, are co-reachable if there is an input tree such that the two states are assigned to the same node of the input tree in the translations of M_1, M_2, respectively.

Two LTWs are *same-ordered* if for each pair of co-reachable states q_1, q_2 and for each symbol $f \in \Sigma$, neither q_1 nor q_2 have a rule for f, or if $q_1, f \to u_0q_1'(x_{\sigma_1(1)}) \dots q_n'(x_{\sigma_1(n)})u_n$ and $q_2, f \to v_0q_1''(x_{\sigma_2(1)}) \dots q_n''(x_{\sigma_2(n)})v_n$ are rules of q_1 and q_2, then $\sigma_1 = \sigma_2$.

If two LTWs are same-ordered the input trees can be reordered according to the order in the transformations. Therefore for each LTW a tree-to-tree transducer is constructed that transforms the input tree according to the transformation in the LTW. Then all permutations σ in the LTWs are replaced by the identity. Thus

the LTWs can be handled as STWs and therefore the equivalence is decidable in polynomial time [13].

Theorem 1. *The equivalence of same-ordered* LTW*s is decidable in polynomial time.*

3.1 Linear Earliest Normal Form

In this section we introduce the two key properties that are used to build a normal form for linear tree-to-word transducers, namely the *earliest* and *erase-ordered* properties. The earliest property means that the output is produced as early as possible, i.e. the longest common prefix (resp. suffix) of L_q is produced in the rule in which q occurs, and as left as possible. The erase-ordered property means that all states that produce no output are ordered according to the input tree and pushed to the right in the rules.

An LTW is in *earliest form* if

- each state q is *earliest*, i.e. $\mathsf{lcp}(L_q) = \mathsf{lcs}(L_q) = \varepsilon$,
- and for each rule $q, f \to u_0 q_1(x_{\sigma(1)}) \dots q_n(x_{\sigma(n)}) u_n$, for each $i, 1 \leq i \leq n$, $\mathsf{lcp}(L_{q_i} u_i) = \varepsilon$.

In [1, Lemma 9] it is shown that for each LTW M an equivalent earliest LTW M' can be constructed in exponential time. Intuitively, if $\mathsf{lcp}(L_q) = v \neq \varepsilon$ (resp. $\mathsf{lcs}(L_q) = v \neq \varepsilon$) then q' is constructed with $L_{q'} = v^{-1} L_q$ (resp. $L_{q'} = L_q v^{-1}$) and $q(x)$ is replaced by $vq'(x)$ (resp. $q'(x)v$). If $\mathsf{lcp}(L_q u) = v \neq \varepsilon$ and v is a prefix of $u = vv'$ then we push v through L_q by constructing q' with $L_{q'} = v^{-1} L_q v$ and replace $q(x)u$ by $vq'(x)v'$.

Note that the construction to build the earliest form M' of an LTW M creates a same-ordered M'. Furthermore, if a state q of M and a state q' of M' are co-reachable, then q' is an "earliest" version of q, where some word u was pushed out of the production of q to make it earliest, and some word v was pushed through the production of q to ensure that the rules have the right property: there exists $u, v \in \Delta^*$ such that for all $t \in \mathsf{dom}(q)$, $[\![M']\!]_{q'}(t) = v^{-1} u^{-1} [\![M]\!]_q(t) v$.

Theorem 2. *For each* LTW *an equivalent same-ordered and earliest* LTW *can be constructed in exponential time.*

The exponential time complexity is caused by a potential exponential size increase in the number of states as it is shown in [7, Example 5].

We call a state q that produces only the empty word, i.e. $L_q = \{\varepsilon\}$, an *erasing state*. As erasing states do not change the transformation and can occur at any position in a rule we need to fix their position for a normal form.

An LTW M is *erase-ordered* if for each rule $q, f \to u_0 q_1(x_{\sigma(1)}) \dots q_n(x_{\sigma(n)}) u_n$ in M, if q_i is erasing then for all $j \geq i$, q_j is erasing, $\sigma(i) \leq \sigma(j)$ and $u_j = \varepsilon$.

We test whether $L_q = \{\varepsilon\}$ in polynomial time and then reorder a rule according to the erase-ordered property. If an LTW is earliest it is still earliest after the reordering.

Lemma 3 (extended from [1, Lemma 18]). *For each (earliest)* LTW *an equivalent (earliest) erase-ordered* LTW *can be constructed in polynomial time.*

Example 4. Consider the rule $q_0, f \to q_1(x_4)q_2(x_3)q_1(x_2)q_4(x_1)$ where q_2 translates trees of the form $f^n(g), n \geq 0$ to $(abc)^n$, q_4 translates trees of the form $f^n(g), n \geq 0$ to $(abc)^{2n}$, q_1 translates trees of the form $f^n(g), n \geq 0$ to ε. Thus the rule is not erase-ordered. We reorder the rule to the equivalent and erase-ordered rule $q_0, f \to q_2(x_3)q_4(x_1)q_1(x_2)q_1(x_4)$.

If two equivalent LTWs are earliest and erase-ordered, then they are not necessarily same-ordered. For example, the rule $q, f \to q_4(x_1)q_2(x_3)q_1(x_2)q_1(x_4)$ is equivalent to the rule in the above example but the two rules are not same-ordered. However, in earliest and erase-ordered LTWs, we can characterize the differences in the orders of equivalent rules: Just as two words u, v satisfy the equation $uv = vu$ if and only if there is a word w such that $u \in w^*$ and $v \in w^*$, the only way for equivalent earliest and erase-ordered LTWs to not be same-ordered is to switch periodic states.

Theorem 5 ([1]). *Let M and M' be two equivalent erase-ordered and earliest* LTW*s and q, q' be two co-reachable states in M, M', respectively. Let*
$q, f \to u_0 q_1(x_{\sigma_1(1)}) \dots q_n(x_{\sigma_1(n)})u_n$ *and* $q', f \to v_0 q'_1(x_{\sigma_2(1)}) \dots q'_n(x_{\sigma_2(n)})v_n$
be two rules for q, q'. Then
- *for $k < l$ such that $\sigma_1(k) = \sigma_2(l)$, all q_i, $k \leq i \leq l$, are periodic of the same period and all $u_j = \varepsilon$, $k \leq j < l$,*
- *for k, l such that $\sigma_1(k) = \sigma_2(l)$, $[\![M]\!]_{q_k} = [\![M']\!]_{q'_l}$.*

As the subtrees that are not same-ordered in two equivalent earliest and erase-ordered states are periodic of the same period the order of these can be changed without changing the semantics. Therefore the order of these subtrees can be fixed such that equivalent earliest and erase-ordered LTWs are same-ordered. Then the equivalence is decidable in polynomial time, see Theorem 1. However, building the earliest form of an LTW is in exponential time.

To circumvent this difficulty, we will show that the first part of Theorem 5 still holds even on a *partial normal form*, where only quasi-periodic states are earliest and the longest common prefix of parts of rules $q(x)u$ with $L_q u$ being quasi-periodic is the empty word.

Theorem 6. *Let M and M' be two equivalent erase-ordered* LTW*s such that*
- *all quasi-periodic states q are earliest, i.e.* $\mathsf{lcp}(q) = \mathsf{lcs}(q) = \varepsilon$
- *for each part $q(x)u$ of a rule where $L_q u$ is quasi-periodic,* $\mathsf{lcp}(L_q u) = \varepsilon$

Let q, q' be two co-reachable states in M, M', respectively and
$q, f \to u_0 q_1(x_{\sigma_1(1)}) \dots q_n(x_{\sigma_1(n)})u_n$ *and* $q', f \to v_0 q'_1(x_{\sigma_2(1)}) \dots q'_n(x_{\sigma_2(n)})v_n$
be two rules for q, q'. Then for $k < l$ such that $\sigma_1(k) = \sigma_2(l)$, all q_i, $k \leq i \leq l$, are periodic of the same period and all $u_j = \varepsilon$, $k \leq j < l$.

4 Partial Normal Form

In this section we introduce a partial normal form for LTWs that does not suffer from the exponential blow-up of the earliest form. Inspired by Theorem 6, we

wish to solve order differences by switching adjacent periodic states of the same period. Remember that the earliest form of a state q is constructed by removing the longest common prefix (suffix) of L_q to produce this prefix (suffix) earlier. It follows that all non-earliest states from which q can be constructed following the earliest form are quasi-periodic.

We show that building the earliest form of a quasi-periodic state or a part of a rule $q(x)u$ with $L_q u$ being quasi-periodic is in polynomial time. Therefore building the following partial normal form is in polynomial time.

Definition 7. *A linear tree-to-word transducer is in* partial normal form *if*

1. *all quasi-periodic states are earliest,*
2. *it is erase-ordered and*
3. *for each rule* $q, f \to u_0 q_1(x_{\sigma(1)}) \dots q_n(x_{\sigma(n)}) u_n$ *if* $L_{q_i} u_i L_{q_{i+1}}$ *is quasi-periodic then* $q_i(x_{\sigma(i)}) u_i q_{i+1}(x_{\sigma(i+1)})$ *is earliest and* $\sigma(i) < \sigma(i+1)$.

4.1 Eliminating Non-Earliest Quasi-Periodic States

In this part, we show a polynomial time algorithm to build an earliest form of a quasi-periodic state. From which an equivalent LTW can be constructed in polynomial time such that any quasi-periodic state is earliest, i.e. $\mathsf{lcp}(L_q) = \mathsf{lcs}(L_q) = \varepsilon$. Additionally, we show that the presented algorithm can be adjusted to test if a state is quasi-periodic in polynomial time.

As quasi-periodicity on the left and on the right are symmetric properties we only consider quasi-periodic states of the form uw^* (quasi-periodic on the left). The proofs in the case w^*u are symmetric and therefore omitted here. In the end of this section we shortly discuss the introduced algorithms for the symmetric case w^*u.

To build the earliest form of a quasi-periodic state we use the property that each state accessible from a quasi-periodic state is as well quasi-periodic. However, the periods can be shifted as the following example shows.

Example 8. Consider states q, q_1 and q_2 with rules $q, f \to aq_1(x_1)c$, $q_1, f \to aaq_2(x_1)ab$, $q_2, f \to q_2(x_1)abc$, $q_2, g \to abc$. State q accepts trees of the form $f^n(g)$, $n \geq 2$, and produces the language $aaa(abc)^n$, i.e. q is quasi-periodic of period abc. State q_1 accepts trees of the form $f^n(g)$, $n \geq 1$, and produces the language $aa(abc)^n ab$, i.e. q_1 is quasi-periodic of period cab. State q_2 accepts trees of the form $f^n(g)$, $n \geq 0$ and produces the language $(abc)^{n+1}$, i.e. q_2 is (quasi-)periodic of period abc.

We introduce two definitions to measure the shift of periods. We denote by $\rho_n[u]$ the *from right-to-left shifted word of* u *of shift* n, $n \leq |u|$, i.e. $\rho_n[u] = u'^{-1}uu'$ where u' is the prefix of u of size n. If $n \geq |u|$ then $\rho_n[u] = \rho_m[u]$ with $m = n \mod |u|$.

For two quasi-periodic states q_1, q_2 of period $u = u_1u_2$ and $u' = u_2u_1$, respectively, we denote the *shift in their period* by $s(q_1, q_2) = |u_1|$.

The size of the periods of a quasi-periodic state and the states accessible from this state can be computed from the size of the shortest words of the languages produced by these states.

Lemma 9. *If q is quasi-periodic on the left with period w, and q' accessible from q, then q' is quasi-periodic with period ε or a shift of w. Moreover we can calculate the shift $s(q, q')$ in polynomial time.*

We now use these shifts to build, for a state q in M that is quasi-periodic on the left, a transducer M^q equivalent to M where each occurrence of q is replaced by its equivalent earliest form, i.e. a periodic state and the corresponding prefix.

Algorithm 1. *Let q be a state in M that is quasi-periodic on the left. M^q starts with the same states, axiom, and rules as M.*

- *For each state p accessible from q, we add a copy p^e to M^q.*
- *For each rule $p, f \to u_0 q_1(x_{\sigma(1)}) \dots q_n(x_{\sigma(n)}) u_n$ in M with p accessible from q, we add a rule $p^e, f \to u_p q_1^e(x_{\sigma(1)}) q_2^e(x_{\sigma(2)}) \dots q_n^e(x_{\sigma(n)})$ with $u_p = \rho_{s(q,p)} \left[\mathsf{lcp}(p)^{-1} u_0 \mathsf{lcp}(q_1) \dots \mathsf{lcp}(q_n) u_n\right]$ in M^q.*
- *We delete state q in M^q and replace any occurrence of $q(x)$ in a rule or the axiom of M^q by $\mathsf{lcp}(q) q^e(x)$.*

Note that $\mathsf{lcp}(p)^{-1} u_0 \mathsf{lcp}(q_1) \dots \mathsf{lcp}(q_n) u_n$ is equivalent to deleting the prefix of size $|\mathsf{lcp}(p)|$ from the word $u_0 \mathsf{lcp}(q_1) \dots \mathsf{lcp}(q_n) u_n$.

Intuitively, to build the earliest form of a state q that is quasi-periodic on the left we need to push all words and all longest common prefixes of states on the right-hand side of a rule of q to the left. Pushing a word to the left through a state needs to shift the language produced by this state. We explain the algorithm in detail on state q from Example 8.

Example 10. Remember that q produces the language $aaa(abc)^n, n \geq 2$ and q_1, q_2 accessible from q produce languages $aa(abc)^n ab, n \geq 1$ and $(abc)^{n+1}, n \geq 0$, respectively. Therefore $\mathsf{lcp}(q) = aaaabcabc$, $\mathsf{lcp}(q_1) = aaabcab$ and $\mathsf{lcp}(q_2) = abc$. We start with state q. As there is only one rule for q the longest common prefix of q and the longest common prefix of this rule are the same and therefore eliminated.

$$\begin{aligned} q^e, f &\to \rho_{s(q,q)}[\mathsf{lcp}(q)^{-1} a \mathsf{lcp}(q_1) c] q_1^e(x_1) \\ &\to \rho_{s(q,q)}[(aaaabcabc)^{-1} aaaabcabc] q_1^e(x_1) \\ &\to q_1^e(x_1) \end{aligned}$$

As there is only one rule for q_1 the argumentation is the same and we get $q_1^e, f \to q_2^e$. For the rule q_2, f we calculate the longest common prefix of the right-hand side $\mathsf{lcp}(q_2) abc = abcabc$ that is larger than the longest common prefix of q_2. Therefore we need to calculate the shift $s(q, q_2) = s(q, q_1) + s(q_1, q_2) = |c| + |ab| = 3$ as q_1 is accessible from q in rule q, f and q_2 is accessible from q_1 in rule q_1, f. This leads to the following rule.

$$\begin{aligned} q_2^e, f &\to \rho_{s(q,q_2)}[\mathsf{lcp}(q_2)^{-1} \mathsf{lcp}(q_2) abc] q_2^e(x_1) \\ &\to \rho_3[(abc)^{-1} abcabc] q_2^e(x_1) \\ &\to abc q_2^e(x_1) \end{aligned}$$

As the longest common prefix of q_2 is the same as the longest common prefix of the right-hand side of rule q_2, g we get $q_2^e, g \to \varepsilon$. The axiom of M^q is $\mathsf{lcp}(q) q^e(x_1) = aaaabcabc q^e(x_1)$.

Lemma 11. *Let M be an* LTW *and q be a state in M that is quasi-periodic on the left. Let M^q be constructed by Algorithm 1 and p^e be a state in M^q accessible from q^e. Then M and M^q are equivalent and p^e is earliest.*

To replace all quasi-periodic states by their equivalent earliest form we need to know which states are quasi-periodic. Algorithm 1 can be modified to test an arbitrary state for quasi-periodicity on the left in polynomial time. The only difference to Algorithm 1 is that we do not know how to compute $\mathsf{lcp}(p)$ in polynomial time and $s(q,p)$ does not exist. We therefore substitute $\mathsf{lcp}(p)$ by some smallest word of L_p and we define a mock-shift $s'(q,p)$ as follows

- $s'(q,q) = 0$ for all q,
- if $q, f \to u_0 q_1(x_{\sigma(1)}) \dots q_n(x_{\sigma(n)}) u_n$, we say $s'(q,q_i) = |u_i w_{q_{i+1}} \dots w_{q_n} u_n|$, where w_q is a shortest word of L_q,
- if $s'(q_1,q_2) = n$ and $s'(q_2,q_3) = m$ then $s'(q_1,q_3) = n+m$.

If several definitions of $s'(q,p)$ exist, we use the smallest. If p is accessible from a quasi-periodic q, then $s'(q,p) = s(q,p)$.

Algorithm 2. *Let $M = (\Sigma, \Delta, Q, \mathsf{ax}, \delta)$ be an* LTW *and q be a state in M. We build an* LTW *T^q as follows.*

- *For each state p accessible from q, we add a copy p^e to T^q.*
- *The axiom is $w_q q^e(x)$ where w_q is a shortest word of L_q.*
- *For each rule $p, f \to u_0 q_1(x_{\sigma(1)}) \dots q_n(x_{\sigma(n)}) u_n$ in M with p accessible from q, we add a rule*

$$p^e, f \to u_p q_1^e(x_{\sigma(1)}) q_2^e(x_{\sigma(2)}) \dots q_n^e(x_{\sigma(n)})$$

in T^q, where u_p is constructed as follows.

  - *We define $u = u_0 w_1 \dots w_n u_n$, where w_i is a shortest word of L_{q_i}.*
  - *Then we remove from u its prefix of size $|w'|$, where w' is a shortest word of L_p. We obtain a word u'.*
  - *Finally, we set $u_p = \rho_{s'(q,p)}[u']$.*

As the construction of Algorithms 1 and 2 are the same if the state q is quasi-periodic, $[\![M]\!]_q$ and $[\![T^q]\!]$ are equivalent if q is quasi-periodic. Moreover, q is quasi-periodic if $[\![M]\!]_q$ and $[\![T^q]\!]$ are equivalent.

Lemma 12. *Let q be a state of an* LTW *M and T^q be constructed by Algorithm 2. Then M and T^q are same-ordered and q is quasi-periodic on the left if and only if $[\![M]\!]_q = [\![T^q]\!]$ and q^e is periodic.*

As M and T^q are same-ordered we can test the equivalence in polynomial time, cf. Theorem 1. Moreover testing a CFG for periodicity is in polynomial time and therefore testing a state for quasi-periodicity is in polynomial time.

Algorithm 2 can be applied to a part $q(x)u$ of a rule to test $L_q u$ for quasi-periodicity on the left. In this case for each rule $q, f \to u_0 q_1(x_{\sigma(1)}) \dots q_n(x_{\sigma(n)}) u_n$ a rule $\hat{q}, f \to u_0 q_1(x_{\sigma(1)}) \dots q_n(x_{\sigma(n)}) u_n u$ is added to M and each occurrence of the part $q(x)u$ in a rule of M is replaced by $\hat{q}(x)$. We then apply the above algorithm to $\hat{q}$ and test $[\![M]\!]_{\hat{q}}$ and $[\![T^{\hat{q}}]\!]$ for equivalence and $\hat{q}^e$ for periodicity.

We introduced algorithms to test states for quasi-periodicity on the left and to build the earliest form for such states. These two algorithms can be adapted for states that are quasi-periodic on the right. There are two main differences. First, as the handle is on the right the shortest word of a language L that is quasi-periodic on the right is $\mathsf{lcs}(L)$. Second, instead of pushing words through a periodic language to the left we need to push words through a periodic language to the right.

Hence, we can test each state q of an LTW M for quasi-periodicity on the left and right. If the state is quasi-periodic we replace q by its earliest form. Algorithm 1 and 2 run in polynomial time if SLPs are used. This is crucial as the shortest word of a CFG can be of exponential size. However, the operations that are needed in the algorithms, namely constructing the shortest word of a CFG and removing the prefix or suffix of a word, are in polynomial time using SLPs, cf. [9].

Theorem 13. *Let M be an* LTW*. Then an equivalent* LTW *M' where all quasi-periodic states are earliest can be constructed in polynomial time.*

4.2 Switching Periodic States

In this part we obtain the partial normal form by ordering periodic states of an erase-ordered transducer where all quasi-periodic states are earliest. Ordering means that if the order of the subtrees in the translation can differ, we choose the one similar to the input, i.e. if $q(x_3)q'(x_1)$ and $q'(x_1)q(x_3)$ are equivalent, we choose the second order. We already showed how we can build a transducer where each quasi-periodic state is earliest and therefore periodic. However, we need to make parts of rules earliest such that periodic states can be switched as the following example shows.

Example 14. Consider the rule $q, h \to q_1(x_2)bq_2(x_1)$ where q_1, q_2 have the rules $q_1, f \to bcabcaq_1(x)$, $q_1, g \to \varepsilon$, $q_2, f \to cabq_2(x)$, $q_2, g \to \varepsilon$. States q_1 and q_2 are earliest and periodic but not of the same period as a subword is produced in between. We replace the non-earliest and quasi-periodic part $q_1(x_2)b$ by their earliest form. This leads to $q, h \to bq_1^e(x_2)q_2(x_1)$ with $q_1^e, f \to cabcabq_1^e(x)$, $q_1^e, g \to \varepsilon$. Hence, q_1^e and q_2 are earliest and periodic of the same period and can be switched in the rule.

To build the earliest form of a quasi-periodic part of a rule $q(x)u$ each occurrence of this part is replaced by a state $\hat{q}(x)$ and for each rule $q, f \to u_0q_1(x_{\sigma(1)})\dots q_n(x_{\sigma(n)})u_n$ a rule $\hat{q}, f \to u_0q_1(x_{\sigma(1)})\dots q_n(x_{\sigma(n)})u_nu$ is added. Then we apply Algorithm 1 on $\hat{q}$ to replace $\hat{q}$ and therefore $q(x)u$ by their earliest form. Iteratively this leads to the following theorem.

Theorem 15. *For each* LTW *M where all quasi-periodic states are earliest we can build in polynomial time an equivalent* LTW *M' such that each part $q(x)u$ of a rule in M where L_qu is quasi-periodic is earliest.*

In Theorem 6 we showed that order differences in equivalent erase-ordered LTWs where all quasi-periodic states are earliest and all parts of rules $q(x)u$ are earliest are caused by adjacent periodic states. As these states are periodic of the same period and no words are produced in between these states can be reordered without changing the semantics of the LTWs.

Lemma 16. *Let M be an* LTW *such that*
- *M is erase-ordered,*
- *all quasi-periodic states in M are earliest and*
- *each $q_i(x_{\sigma(i)})u_i$ in a rule of M that is quasi-periodic is earliest.*

Then we can reorder adjacent periodic states $q_i(x_{\sigma(i)})q_{i+1}(x_{\sigma(i+1)})$ of the same period in the rules of M such that $\sigma(i) < \sigma(j)$ in polynomial time. The reordering does not change the transformation of M.

We showed before how to construct a transducer with the preconditions needed in Lemma 16 in polynomial time. Note that replacing a quasi-periodic state by its earliest form can break the erase-ordered property. Thus we need to replace all quasi-periodic states by its earliest form *before* building the erase-ordered form of a transducer. Then Lemma 16 is the last step to obtain the partial normal form for an LTW.

Theorem 17. *For each* LTW *we can construct an equivalent* LTW *that is in partial normal form in polynomial time.*

4.3 Testing Equivalence in Polynomial Time

It remains to show that the equivalence problem of LTWs in partial normal form is decidable in polynomial time. The key idea is that two equivalent LTWs in partial normal form are same-ordered.

Consider two equivalent LTWs M_1, M_2 where all quasi-periodic states and all parts of rules $q(x)u$ with $L_q u$ is quasi-periodic are earliest. In Theorem 6 we showed if the orders σ_1, σ_2 of two co-reachable states q_1, q_2 of M_1, M_2, respectively, for the same input differ then the states causing this order differences are periodic with the same period. The partial normal form solves this order differences such that the transducers are same-ordered.

Lemma 18. *If M and M' are equivalent and in partial normal form then they are same-ordered.*

As the equivalence of same-ordered LTWs is decidable in polynomial time (cf. Theorem 1) we conclude the following.

Corollary 19. *The equivalence problem for* LTW*s in partial normal form is decidable in polynomial time.*

To summarize, the following steps run in polynomial time and transform a LTW M into its partial normal form.

1. Test each state for quasi-periodicity. If it is quasi-periodic replace the state by its earliest form.
2. Build the equivalent erase-ordered transducer.
3. Test each part $q(x)u$ in each rule from right to left for quasi-periodicity on the left. If it is quasi-periodic replace the part by its earliest form.
4. Order adjacent periodic states of the same period according to the input order.

This leads to our main theorem.

Theorem 20. *The equivalence of* LTW*s is decidable in polynomial time.*

Acknowledgement

We would like to thank the reviewers for their very helpful comments and suggestions.

References

1. Boiret, A.: Normal form on linear tree-to-word transducers. In: Language and Automata Theory and Applications. pp. 439–451. Springer (2016)
2. Engelfriet, J.: Some open question and recent results on tree transducers and tree languages. In: Formal Language Theory, Perspectives and Open Problems. pp. 241–286. Academic Press (1980)
3. Engelfriet, J., Maneth, S.: Macro tree translations of linear size increase are MSO definable. SIAM Journal on Computing 32(4), 950–1006 (2003)
4. Engelfriet, J., Maneth, S.: The equivalence problem for deterministic MSO tree transducers is decidable. Information Processing Letters 100(5), 206–212 (2006)
5. Engelfriet, J., Rozenberg, G., Slutzki, G.: Tree transducers, L systems and two-way machines. In: Proceedings of the tenth annual ACM symposium on Theory of computing. pp. 66–74. ACM (1978)
6. Engelfriet, J., Vogler, H.: Macro tree transducers. Journal of Computer and System Sciences 31(1), 71–146 (1985)
7. Laurence, G., Lemay, A., Niehren, J., Staworko, S., Tommasi, M.: Normalization of sequential top-down tree-to-word transducers. In: Language and Automata Theory and Applications, pp. 354–365. Springer (2011)
8. Lemay, A., Maneth, S., Niehren, J.: A learning algorithm for top-down XML transformations. In: Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. pp. 285–296 (2010)
9. Lohrey, M.: The Compressed Word Problem for Groups. Springer (2014)
10. Maneth, S., Seidl, H.: Deciding equivalence of top-down XML transformations in polynomial time. In: Programming Language Technologies for XML. pp. 73–79 (2007)
11. Plandowski, W.: The complexity of the morphism equivalence problem for context-free languages. Ph.D. thesis, Warsaw University (1995)
12. Seidl, H., Maneth, S., Kemper, G.: Equivalence of deterministic top-down tree-to-string transducers is decidable. In: IEEE 56th Annual Symposium on Foundations of Computer Science. pp. 943–962 (2015)

13. Staworko, S., Laurence, G., Lemay, A., Niehren, J.: Equivalence of deterministic nested word to word transducers. In: Fundamentals of Computation Theory. pp. 310–322. Springer (2009)