

Equivalence of Deterministic Top-Down Tree-to-String Transducers is Decidable

Helmut Seidl
Fakultät für Informatik
TU München
Garching, Germany
Email: seidl@in.tum.de

Sebastian Maneth
School of Informatics
University of Edinburgh
Edinburgh, UK
Email: smaneth@inf.ed.ac.uk

Gregor Kemper
Fakultät für Mathematik
TU München
Garching, Germany
Email: kemper@ma.tum.de

Abstract

We show that equivalence of deterministic top-down tree-to-string transducers is decidable, thus solving a long standing open problem in formal language theory. We also present efficient algorithms for subclasses: polynomial time for total transducers with unary output alphabet (over a given top-down regular domain language), and co-randomized polynomial time for linear transducers; these results are obtained using techniques from multi-linear algebra. For our main result, we prove that equivalence can be certified by means of inductive invariants using polynomial ideals. This allows us to construct two semi-algorithms, one searching for a proof of equivalence, one for a witness of non-equivalence. Furthermore, we extend our result to deterministic top-down tree-to-string transducers which produce output not in a free monoid but in a free group.

I. INTRODUCTION

Transformations of structured data are at the heart of functional programming [1], [2], [3], [4], [5] and also application areas such as compiling [6], document processing [7], [8], [9], [10], [11], [12], [13], automatic translation of natural languages [14], [15], [16], [17] or even cryptographic protocols [18]. The most fundamental model of such transformations is given by (finite-state tree) *transducers* [19], [6]. Transducers traverse the input by means of finitely many mutually recursive functions — corresponding to finitely many *states*. Accordingly, these transducers are also known as *top-down* tree transducers [20], [21] or, if additional parameters are used for accumulating output values, *macro* tree transducers [22]. Here we only deal with deterministic transducers and denote them DT and DMT transducers, respectively (equivalence is undecidable already for very restricted classes of non-deterministic transducers [23]).

When the output is produced in a structured way, i.e., in the case of tree-to-tree transducers, many properties, at least of transducers without parameters, are fairly well understood. An algorithm for deciding equivalence of DT transducers already dates back to the 80s [24]. More recently, canonical forms have been provided allowing for effective minimization [25] as well as Gold-style learning of transformations from examples [26]. In various applications, though, the output is *not* generated in a structured way. This may be the case when general scripting languages are employed [27], non-tree operations are required [28] or simply, because the result is a string.

Assume, e.g., that we want to generate a well-formed XML document from an internal tree-like representation where the elements of the document do not only have tags and contents but also attributes. The output for the input tree

```
frame(
  defs(height(20), defs(width(50), end)),
  content(button("Do_not_press!"), ...)
)
```

then should look like:

```
<frame_height = "20" width = "50">
  <button>Do_not_press!</button>
  ...
</frame>
```

This translation could be accomplished by a tree-to-string transducer with, among others, the following rules:

$$\begin{aligned}
q(\text{frame}(x_1, x_2)) &\rightarrow \langle \text{frame } q_1(x_1)q(x_2) \rangle / \text{frame} \rangle \\
q_1(\text{end}) &\rightarrow \rangle \\
q_1(\text{defs}(x_1, x_2)) &\rightarrow q_2(x_1)q_1(x_2) \\
q_2(\text{height}(x_1)) &\rightarrow \text{height} = "q_3(x_1)" \\
q_2(\text{width}(x_1)) &\rightarrow \text{width} = "q_3(x_1)" \\
q(\text{button}(x_1)) &\rightarrow \langle \text{button} \rangle q_3(x_1) \langle / \text{button} \rangle \\
&\dots
\end{aligned}$$

According to the peculiarities of XML, arbitrary many attribute value pairs may be positioned inside the start tag of an element. The given rules generate the closing bracket of the start tag from the node end which terminates the list of attribute definitions. At the expense of an empty right-hand side, the closing bracket could also be generated by the rule for the tag `frame` directly. In this case, the two first rules should be replaced with:

$$\begin{aligned}
q(\text{frame}(x_1, x_2)) &\rightarrow \langle \text{frame } q_1(x_1) \rangle q(x_2) \langle / \text{frame} \rangle \\
q_1(\text{end}) &\rightarrow \epsilon
\end{aligned}$$

while all remaining rules stay the same. This example indicates that already simple tasks for structured data may be solved by transducers processing their inputs in different ways.

Following the tradition since [29], we denote *tree-to-string transducers* by prefixing the letter y (which stands for “yield”, i.e., the function that turns a tree into the string of its leaf labels, read from left to right). In [9], [28] dedicated transducers for XML have been introduced. Beyond the usual operations on trees, these transducers also support concatenation of outputs. Decidability of equivalence for these transducers has been open. Since they can be simulated by tree-to-string transducers, our main result implies that equivalence is decidable for both types of transducers (where for [28] we mean the parameter-less version of their transducers).

Amazingly little has been known so far for tree-to-string transducers, possibly due to the multitude of ways in which they can produce the same output string. As a second example, consider the transducer M with initial state q , on input trees with a ternary symbol f and a leaf symbol a , defined by:

$$\begin{aligned}
q(f(x_1, x_2, x_3)) &\rightarrow q(x_3)aq_1(x_2)bq(x_2) \\
q_1(f(x_1, x_2, x_3)) &\rightarrow q_1(x_3)q_1(x_2)q_1(x_2)ba \\
q_1(e) &\rightarrow ba \\
q(e) &\rightarrow ab.
\end{aligned}$$

Furthermore, let M' be the transducer with single state q' and the rules:

$$\begin{aligned}
q'(f(x_1, x_2, x_3)) &\rightarrow abq'(x_2)q'(x_2)q'(x_3) \\
q'(e) &\rightarrow ab.
\end{aligned}$$

Some thought reveals that the transducers M and M' are equivalent, although the output is generated in a quite “un-aligned” way with respect to x_2, x_3 . Note that these two transducers do not fall into any class of tree-to-string transducers for which equivalence has been known to be decidable so far. One class where equivalence is already known to be decidable, are the *linear and order-preserving* deterministic tree-to-string transducers as studied in [30]. A transducer is linear, if each input variable x_i occurs at most once in every right-hand side. A transducer is order-preserving if the variables x_i appear in ascending order (from left to right) in each right-hand side. Equivalence for these can be decided by a reduction to Plandowski’s result [31] even in polynomial time [30]. This class of transducers is sufficiently well-behaved so that periodicity and commutation arguments over the output strings can be applied to derive canonical normal forms [32] and enable Gold-style learning [33]. Apart from these stronger normal form results, equivalence itself can indeed be solved for a much larger class of tree-to-string translations, namely for those definable in MSO logic [34], or equivalently, by macro tree-to-string translations of *linear size increase* [35]. This proof gracefully uses Parikh’s theorem and the theory of semi-linear

sets. More precisely, for a Parikh language L (this means L , if the order of symbols is ignored, is equivalent to a regular language) it is decidable whether there exists an output string with equal number of a 's and b 's (for given letters $a \neq b$). The idea of the proof is to construct L which contains $a^n b^m$ if and only if, on input t , transducer M_1 outputs a at position n and transducer M_2 outputs b at position m .

Our main result generalizes the result of [34] by proving that equivalence of *unrestricted* deterministic top-down tree-to-string transducers is decidable. By that, it solves an intriguing problem which has been open for at least thirty-five years [29]. The difficulty of the problem may perhaps become apparent as it encompasses not only the equivalence problem for MSO definable transductions, but also the famous HDTOL sequence equivalence problem [36], [37], [38], the latter is the sub-case when the input is restricted to monadic trees [39]. Opposed to the attempts, e.g., in [30], we refrain from any arguments based on the combinatorial structure of finite state devices or output strings. We also do not follow the line of arguments in [34] based on semi-linear sets. Instead, we proceed in two stages. **In the first stage, we consider transducers with unary output alphabets only** (Sections III–V). In this case, a produced output string can be represented by its length, thus turning the transducers effectively into tree-to-integer transducers. For a given input tree, the output behavior of the states of such a unary y DT transducer is collected into a vector. Interestingly, the output vector for an input tree turns out to be a *multi-affine* transformation of the corresponding output vectors of the input subtrees. As the property we are interested in can be expressed as an affine equality to be satisfied by output vectors, we succeed in replacing the *sets* of reachable output vectors of the transducer by their *affine closures*. This observation allows us to apply exact fixpoint techniques as known from abstract interpretation of programs [40], to effectively compute these affine closures and thus to decide equivalence. In the next step, we generalize these techniques to a larger class of transducers, namely, unary *non-self-nested* transducers. These are transducers which additionally have parameters, but may use these only in a restricted way. Although they are more expressive than unary y DT transducers, the effect of the transducer for each input symbol still is multi-affine and therefore allows a similar (yet more expensive) construction as for unary y DT transducers for deciding equivalence. In the final step, we ultimately show that the restriction of *non-self-nestedness* can be lifted. Then however, multi-affinity is no longer available. In order to attack this problem, we turn it upside down: instead of maintaining affine spaces generated by sets of input trees, we maintain their *dual*, namely suitable properties satisfied by input trees. Indeed, we show that *inductive invariants* based on conjunctions of polynomial equalities are sufficient for proving equivalence. This result is obtained by representing conjunctions of equalities by polynomial ideals [41] and applying Hilbert's basis theorem to prove that finite conjunctions suffice. We also require compatibility of polynomial ideals with Cartesian products, a property which may be of independent interest. For the specific case of monadic input alphabets, we obtain a decision procedure which resembles techniques for effectively proving polynomial program invariants by means of weakest pre-conditions [42], [43]. For non-monadic input symbols, we obtain two semi-decision procedures, one enumerating all potential proofs of equivalence, while the other searches for counter-examples.

Having established decidability for all y DMT transducers with unary output alphabet, we indicate in the second stage how these transducers are able to simulate y DT transducers with multiple output symbols when these are viewed as *digits* in a suitable number system (Section VII). The corresponding construction maps *linear* y DT transducers into unary non-self-nested y DMT transducers, and general y DT transducers into general unary y DMT transducers. In this way, our algorithms for deciding equivalence of unary transducers give rise to algorithms for deciding equivalence of linear and general y DT transducers, respectively. While decidability of (in-)equivalence for linear y DT transducers has been known (via the result of [34]), the resulting randomized polynomial complexity bounds has only recently been improved to a truly polynomial upper bound by Boiret and Palenta [44]. No other method, on the other hand, allows to decide equivalence of unrestricted y DT transducers.

Related Work

Decision procedures for equivalence of deterministic tree transducers have been provided for various sub-classes of transducers (see, e.g., [39] for a recent survey). The equivalence problem for y DT transducers has already been mentioned as a difficult open question in [29]. Still, little progress on the question has been made for

tree transducers where the outputs are unstructured strings. The strongest result known so far is the decidability of equivalence for MSO definable tree-to-string transductions [34]; this class is equal to y DMT transducers of linear size increase [35]. For the specific sub-class of linear y DT transducers, we obtain an algorithm with by far better complexity bounds as those provided by the construction in [34]. General MSO definable tree-to-string transductions on the other hand, can be simulated by y DT transducers with regular look-ahead (see [45], [46]). Since equivalence of y DT transducers with regular look-ahead can be reduced to equivalence of y DT transducers without look-ahead but relative to a DTTA automaton, our decidability result encompasses the decidability result for MSO definable tree-to-string transductions. It is more general, since y DT transducers may have more than linear size increase and thus may not be MSO definable. The same holds true for arbitrary y DMT transducers with unary output alphabet. It is unclear how or whether the suggested methods can be generalized to the equivalence problem of unrestricted y DT transducers.

The methods employed to obtain our novel results have the following predecessors. The algorithm for deciding equivalence in the case of non-self-nested y DMT transducers is related to the algorithm in [47] for deciding *ambiguity* equivalence of non-deterministic finite tree automata. Two automata are ambiguity equivalent if they agree for each input tree, in the numbers of accepting runs. While vector spaces and multi-linear mappings were sufficient in case of finite automata, we required *affine* spaces and *multi-affine* mappings in case of linear y DT transducers.

The known algorithm for deciding equivalence of y DT transducers with *monadic* alphabet is based on a reduction to the HDTOL sequence equivalence problem. The latter can be solved [38] via establishing an increasing chain of finite sets of word equations which are guaranteed to eventually agree in their sets of solutions. Instead, our elegant *direct* algorithm for monadic unary y DMT transducers is related to an algorithm effective program verification. In [42], [43] a decision procedure is presented which allows to check whether a given polynomial equality is invariably true at a given program point of a polynomial program, i.e., a program with non-deterministic branching and polynomial assignments of numerical variables. Similar to the new algorithm for every state p of the DTTA automaton, the verification algorithm characterizes the required conjunction of polynomial equalities at each program point by polynomial ideals. These ideals then are characterized as the least solution of a set of constraints which closely resembles those in equation (10). To the best of our knowledge, the algorithm for solving the equivalence problem in the unrestricted case, is completely new.

Organization of the remaining paper

After providing basic notations and concepts in the next section, we first concentrate on unary transducers. In Section III, we provide an algorithm for deciding equivalence of y DT transducers with unary output alphabet. In Section IV, we generalize this algorithm to non-self-nested y DMT transducers with unary output alphabet. These methods are based on least fixpoint iterations over affine spaces. The case of y DMT transducers with unary output alphabet and arbitrary nesting is considered in Section V. This section makes use of polynomial ideals in a non-trivial way. As kind of warm-up, a dedicated algorithm for y DMT transducers with monadic input alphabet is provided which is based on *least* fixpoint iteration over polynomial ideals. The general method for arbitrary input alphabets goes beyond that. It is based on the notion of *inductive invariants* which provide proofs of equivalence. The strongest inductive invariant, however, can only be characterized non-effectively via a *greatest fixpoint* over polynomial ideals. Section V provides us with a decision procedure of equivalence based on two semi-algorithms. In-equivalence can easily be verified by providing a witness inputs for which the outputs differ. Searching for a proof of equivalence, on the other hand, may be rather difficult. Therefore, Section VI provides a systematic means to identify candidate inductive invariants. Section VII indicates how the algorithms for unary y DT transducers can be used to decide equivalence for y DT transducers with arbitrary output alphabets by providing an appropriate simulation. Section VIII then shows that equivalence of y DT transducers remains decidable if the output is not considered as a string, i.e., an element of the *free monoid*, but as an element in the *free group*. This means that we now allow symbols to have positive or negative polarities and assume that symbols with opposite polarities may cancel each other out. In order to deal with this situation, we consider a

setting where the output symbols of a *y*DT transducer are interpreted as square matrices with rational entries and show that in this setting, equivalence is still decidable. By recalling that the free group with two generators is a sub-group of the special linear group of (2×2) matrices with entries in \mathbb{Z} , we thus arrive at the desired result. Finally, Section IX discusses applications of the obtained results to various models of transducers as proposed in the literature.

A preliminary version of this paper was presented at FOCS'2015 [48]. That version has been extended with a practical algorithm for enumerating inductive invariants (Section VI) and by allowing output not just in the free monoid of strings, but instead in the free group (Section VIII).

II. PRELIMINARIES

For a finite set S , we denote by $|S|$ its cardinality. For $m \in \mathbb{N}$ let $[m]$ denote the set $\{1, \dots, m\}$. A ranked alphabet Σ is a finite set of symbols each with an associated natural number called rank. By $a^{(m)}$ we denote that a is of rank m and by $\Sigma^{(m)}$ the set of symbols in Σ of rank m . For an m -tuple \mathbf{t} and $i \in [m]$ we denote by \mathbf{t}_i the i th component of \mathbf{t} . The set \mathcal{T}_Σ of trees over Σ is the smallest set T such that if $\mathbf{t} \in T^m$ for $m \geq 0$ then also $f \mathbf{t} \in T$ for all $f \in \Sigma^{(m)}$. For the tree $f()$ we also write f . Thus a tree consists of a symbol of rank m together with an m -tuple of trees. We fix the sets of input variables $X = \{x_1, x_2, \dots\}$ and formal parameters $Y = \{y_1, y_2, \dots\}$ where for $m \in \mathbb{N}$, $X_m = \{x_1, \dots, x_m\}$ and $Y_m = \{y_1, \dots, y_m\}$.

A (*deterministic top-down*) *tree automaton* (DTTA automaton for short) is a tuple $A = (P, \Sigma, p_0, \rho)$ where P is a finite set of states, Σ a ranked alphabet, $p_0 \in P$ the initial state, and ρ the transition function. For every $f \in \Sigma^{(m)}$ and $p \in P$, $\rho(p, f)$ is undefined or is in P^m . The transition function allows to define for each $p \in P$ the set $\text{dom}(p) \subseteq \mathcal{T}_\Sigma$ by letting $f \mathbf{t} \in \text{dom}(p)$ for $f \in \Sigma^{(m)}$, $m \geq 0$, and $\mathbf{t} \in \mathcal{T}_\Sigma^m$ iff $\rho(p, f) = \mathbf{p}$ and $\mathbf{t}_i \in \text{dom}(\mathbf{p}_i)$ for $i \in [m]$. The language $\mathcal{L}(A)$ of A is given by $\mathcal{L}(A) = \text{dom}(p_0)$. The *size* $|A|$ of A is defined as $|P| + |\Sigma| + |\rho|$ where $|\rho| = \sum_{m \geq 0} |\rho^{-1}(P^m)| \cdot (m + 1)$.

Let $l \geq 0$. A *deterministic macro tree-to-string transducer* (with l parameters) (*y*DMT transducer for short) is a tuple $M = (Q, \Sigma, \Delta, q_0, \delta)$, where Q is a ranked alphabet of states all of rank $l + 1$, Σ is a ranked alphabet of input symbols, Δ is an alphabet of output symbols, $q_0 \in Q$ is the initial state, and δ is the transition function. For every $q \in Q$, $m \geq 0$, and $f \in \Sigma^{(m)}$, $\delta(q, f)$ is either undefined or is in R , where R is the smallest set such that $\epsilon \in R$ and if $T, T_1, \dots, T_l \in R$, then also

- (1) $aT \in R$ for $a \in \Delta$,
- (2) $y_j T \in R$ for $j \in [l]$, and
- (3) $q'(x_i, T_1, \dots, T_l) T \in R$ for $q' \in Q$ and $i \in [m]$.

Again, we represent the fact that $\delta(q, f) = T$ also by the rule:

$$q(f(x_1, \dots, x_m), y_1, \dots, y_l) \rightarrow T.$$

A state $q \in Q$ induces a partial function $\llbracket q \rrbracket_M$ from \mathcal{T}_Σ to total functions $(\Delta^*)^l \rightarrow \Delta^*$ defined recursively as follows. Let $t = f \mathbf{t}$ with $f \in \Sigma^{(m)}$, $m \geq 0$, and $\mathbf{t} \in \mathcal{T}_\Sigma^m$. Here, we consider a *call-by-value* (or *inside-out*) mode of evaluation for the arguments of states. Thus for $\mathbf{w} \in (\Delta^*)^l$, $\llbracket q \rrbracket_M(t)(\mathbf{w})$ is defined whenever $\delta(q, f) = T$ for some T and for each occurrence of a subtree $q'(x_i, T_1, \dots, T_l)$ in T , $\llbracket q' \rrbracket_M(\mathbf{t}_i)$ is also defined. In this case, the output is obtained by evaluating T in call-by-value order with \mathbf{t}_i taken for x_i and \mathbf{w}_j for y_j . In function applications (especially for higher-order) we often leave out parenthesis; e.g. we write $\llbracket T \rrbracket_M \mathbf{t} \mathbf{w}$ for $\llbracket T \rrbracket_M(\mathbf{t})(\mathbf{w})$. We obtain,

$$\llbracket q \rrbracket_M(f \mathbf{t}) \mathbf{w} = \llbracket T \rrbracket_M \mathbf{t} \mathbf{w}$$

where the evaluation function $\llbracket T \rrbracket_M$ is defined as follows:

$$\begin{aligned} \llbracket \epsilon \rrbracket_M \mathbf{t} \mathbf{w} &= \epsilon \\ \llbracket aT' \rrbracket_M \mathbf{t} \mathbf{w} &= a \llbracket T' \rrbracket_M \mathbf{t} \mathbf{w} \\ \llbracket y_j T' \rrbracket_M \mathbf{t} \mathbf{w} &= \mathbf{w}_j \llbracket T' \rrbracket_M \mathbf{t} \mathbf{w} \\ \llbracket q'(x_i, T_1, \dots, T_l) T' \rrbracket_M \mathbf{t} \mathbf{w} &= \llbracket q' \rrbracket_M \mathbf{t}_i (\llbracket T_1 \rrbracket_M \mathbf{t} \mathbf{w}, \dots, \llbracket T_l \rrbracket_M \mathbf{t} \mathbf{w}) \llbracket T' \rrbracket_M \mathbf{t} \mathbf{w}. \end{aligned}$$

The transducer M realizes the (partial) translation $M : \mathcal{T}_\Sigma \rightarrow \Delta^*$ which, for $t \in \mathcal{T}_\Sigma$, is defined as $M(t) = \llbracket q_0 \rrbracket_M(t)(\epsilon, \dots, \epsilon)$ if $\llbracket q_0 \rrbracket_M(t)$ is defined and is undefined otherwise; the domain of this translation is denoted $\text{dom}(M)$. The y DMT transducer M is a *deterministic top-down tree-to-string transducer* (y DT transducer for short) if $l = 0$. The y DMT transducer M is *total* if $\delta(q, f)$ is defined for all $q \in Q$ and $f \in \Sigma$, and M is *unary* if $|\Delta| = 1$. In the latter case, the output can also be represented by a number, namely, the length of the output. Finally, a y DMT transducer is *self-nested*, if there is a right-hand side T in δ so that T contains an occurrence of a tree $q'(x_i, T_1, \dots, T_l)$ where one of the trees T_j contains another tree $q''(x_i, T'_1, \dots, T'_l)$ for the same x_i . A y DMT transducer is called *non-self-nested*, if it is *not* self-nested.

As for DTTA automata, we define the size $|M|$ of a y DT or y DMT transducer M as the sum of the sizes of the involved alphabets, here Q, Σ and Δ , together with the size of the corresponding transition function where the size of a transition $\delta(q, f) = T$ is one plus the sum of numbers of occurrences of output symbols, parameters, and states in T .

In the following three sections, we consider transducers with unary output alphabet $\Delta = \{d\}$ only. In this case, we prefer to let the transducer produce the *lengths* of the output directly. Then, the right-hand sides T may no longer contain symbols d , but constant numbers c (representing d^c). Likewise, concatenation is replaced with addition. For convenience, we also allow multiplication with constants to compactly represent repeated addition of the same subterm.

Example 1: Consider the y DMT transducer with set $Q = \{q_0, q\}$ of states and initial state q_0 and the following transition rules:

$$\begin{aligned} q_0(f(x_1, x_2), y_1) &\rightarrow q(x_1, q(x_2, d)) \\ q(a(x_1), y_1) &\rightarrow y_1 q(x_1, y_1) \\ q(e, y_1) &\rightarrow \epsilon \end{aligned}$$

where the output is considered as a string. This y DMT transducer is non-self-nested and realizes a translation τ which maps each input tree $f(a^n(e), a^m(e))$ to the string in $d^{n \cdot m}$. As the output alphabet is unary, we prefer to represent the output length by these rules:

$$\begin{aligned} q_0(f(x_1, x_2), y_1) &\rightarrow q(x_1, q(x_2, 1)) \\ q(a(x_1), y_1) &\rightarrow y_1 + q(x_1, y_1) \\ q(e, y_1) &\rightarrow 0. \end{aligned}$$

A DTTA automaton accepting the domain of the given y DMT transducer may use states from $\{p_0, p\}$ with initial state p_0 where

$$\rho(p_0, f) = (p, p) \quad \rho(p, a) = p \quad \rho(p, e) = ().$$

■

Thus, right-hand sides T now are constructed according to the grammar:

$$T ::= c \mid y_j \mid q(x_i, T_1, \dots, T_l) \mid T_1 + T_2 \mid c \cdot T'$$

where the non-negative numbers c may be taken from some fixed range $\{0, 1, \dots, h\}$. The size $|T|$ then is defined as the size of T as an expression, i.e.,

$$\begin{aligned} |c| &= |y_j| = 1 & |T_1 + T_2| &= |T_1| + |T_2| \\ |q(x_i, T_1, \dots, T_l)| &= 2 + |T_1| + \dots + |T_l| & |c \cdot T'| &= 2 + |T'|. \end{aligned}$$

Thus, e.g., for $T = 2 + 3 \cdot q(x_1, 1, 0)$, $|T| = 4 + |q(x_1, 1, 0)| = 4 + 4 = 8$.

From Arbitrary to Binary Input Alphabets

Here we state a technical lemma that allows to restrict the rank of output symbols of our transducers to two. Let Σ be a ranked alphabet and \perp a special symbol not in Σ . By $\text{bin}(\Sigma)$ we denote the ranked alphabet $\{\perp^{(0)}\} \cup \{\sigma^{(2)} \mid \sigma \in \Sigma\}$. For sequences s of trees over Σ we define their *binary encoding* $\text{bin}(s)$ as: $\text{bin}(s) = \perp$ if s is the empty sequence, and $\text{bin}(s) = \sigma(\text{bin}(t_1 t_2 \dots t_m), \text{bin}(s'))$ if $s = \sigma(t_1, \dots, t_m) s'$ with $\sigma \in \Sigma^{(m)}$, $m \geq 0$, $t_1, \dots, t_m \in \mathcal{T}_\Sigma$, and $s' \in \mathcal{T}_\Sigma^*$. Likewise for $S \subseteq \mathcal{T}_\Sigma$, $\text{bin}(S) = \{\text{bin}(s) \mid s \in S\}$. Note that this encoding corresponds to the *first-child-next-sibling* encoding of unranked trees, here applied to ranked trees.

As an example, consider the tree $t = f(b, g(c), h(b, c))$. Then the encoding $\text{bin}(t)$ is given by:

$$\text{bin}(t) = f(b(\perp, g(c(\perp, \perp), h(b(\perp, c(\perp, \perp)), \perp))), \perp)$$

Lemma 1: Let $M = (Q, \Sigma, \Delta, q_0, \delta)$ be a y DMT transducer and let m be the maximal rank of symbols in Σ . Then a y DMT transducer $M' = (Q', \text{bin}(\Sigma), \Delta, q'_0, \delta')$ together with a DTTA automaton B can be constructed in time polynomial in $|M|$ such that

- (1) $\text{bin}(\text{dom}(M)) = \mathcal{L}(B) \cap \text{dom}(M')$,
- (2) $M'(\text{bin}(t)) = M(t)$ for all $t \in \text{dom}(M)$,
- (3) $|Q'| = m|Q|$,
- (4) M' is a total y DT if M is.

Proof: The DTTA automaton B is meant to check whether a tree in $\text{bin}(\Sigma)$ is an encoding of a tree in $\text{bin}(\mathcal{T}_\Sigma)$, i.e., $\mathcal{L}(B) = \text{bin}(\mathcal{T}_\Sigma)$. Such an automaton can be constructed as $B = (P, \text{bin}(\Sigma), 1, \rho)$ where $P = \{0\} \cup [m]$ and ρ is given by:

$$\begin{aligned} \rho(j+1, f) &= (k, j) \quad \text{if } f \in \Sigma^{(k)} \\ \rho(0, \perp) &= () \end{aligned}$$

For this DTTA automaton B , $\mathcal{L}(B) = \text{bin}(\mathcal{T}_\Sigma)$ holds.

The y DMT transducer M' is defined as follows. $Q' = \{\langle q, i \rangle \mid q \in Q, m \geq 1, i \in [m]\}$ and $q'_0 = \langle q_0, 1 \rangle$. Let $q \in Q$ and $f \in \Sigma$ of arity $k \geq 0$. If $\delta(q, f)$ is defined and equals T , then we let $\delta'(\langle q, 1 \rangle, f) = T'$ where T' is obtained from T by replacing every occurrence of $q'(x_i, T_1, \dots, T_l)$ with $\langle q', i \rangle(x_1, T'_1, \dots, T'_l)$ where each T'_i is obtained from T_i in the same way. Furthermore for every $2 \leq i \leq k$, we define $\delta'(\langle q, i \rangle, f) = \langle q, i-1 \rangle(x_2, y_1, \dots, y_l)$ and finally for all $q' \in Q'$, $\delta'(q', \perp) = \epsilon$.

By construction, M' is total whenever M is. Also, the bounds to the number of states is obvious. For the correctness of the construction, we observe that whenever for $i \geq 1$, the state $\langle q, i \rangle$ is called with the encoding of a list $t_1 \dots t_k$ with $k \geq i$, then the same output is produced as by M when applied to the input tree t_i , i.e.,

$$\llbracket \langle q, i \rangle \rrbracket_{M'}(\text{bin}(t_1 \dots t_k))(y_1, \dots, y_l) = \llbracket q \rrbracket_M(t_i)(y_1, \dots, y_l)$$

The proof is by induction on the structure of the sequence $t_1 \dots t_k$, where for subterms T of right-hand sides of M and corresponding subterms T' of right-hand sides of M' ,

$$\llbracket T' \rrbracket_{M'}(\text{bin}(t_1 \dots t_k), s') = \llbracket T \rrbracket_M(t_1, \dots, t_k)$$

holds for all s' . From this, the first two assertions follow. ■

Example 2: Consider the following rule:

$$q(f(x_1, x_2, x_3), y_1) \rightarrow q(x_1, q(x_2, q(x_3, y_1)))$$

for the ternary symbol $f \in \Sigma$. By the construction provided for Lemma 1, this rule is simulated by means of the rules:

$$\begin{aligned} q_1(f(x_1, x_2), y_1) &\rightarrow q_1(x_1, q_2(x_1, q_3(x_1, y_1))) \\ q_2(g(x_1, x_2), y_1) &\rightarrow q_1(x_2, y_1) \quad (g \in \Sigma) \\ q_3(g(x_1, x_2), y_1) &\rightarrow q_2(x_2, y_1) \quad (g \in \Sigma) \end{aligned}$$

where the state q_1 corresponds to the state q and the states q_2, q_3 traverse the encoding of a list $t_1 t_2 t_3$ where q is called for t_2 and t_3 , respectively. \blacksquare

We remark that non-self-nestedness may not be preserved by our construction.

III. UNARY TRANSDUCERS WITHOUT PARAMETERS

We first consider a single unary total y DT transducer and show that equivalence of two states relative to a DTTA automaton can be decided in polynomial time. This result then is extended to decide equivalence of two not necessarily total unary y DT transducers. Let $M = (Q, \Sigma, \{d\}, q_0, \delta)$ be a total unary y DT transducer, and $A = (P, \Sigma, p_0, \rho)$ a DTTA automaton. Assume that $Q = [n]$ for some natural number n . Our goal is to decide for given $q, q' \in Q$ whether or not $\llbracket q \rrbracket_M(t) = \llbracket q' \rrbracket_M(t)$ for all $t \in \mathcal{L}(A)$. For every $t \in \mathcal{T}_\Sigma$ and $q \in Q$, $\llbracket q \rrbracket_M(t) = d^r$ with $r \in \mathbb{N}$, i.e., $\llbracket q \rrbracket_M$ can be seen as a tree-to-integer translation mapping t to r ; we denote r by $\llbracket t \rrbracket_q$ and write $\llbracket t \rrbracket$ for the vector $(\llbracket t \rrbracket_1, \dots, \llbracket t \rrbracket_n) \in \mathbb{N}^n$, or, more generally, in \mathbb{Q}^n . For a vector $\mathbf{v} \in \mathbb{Q}^n$ we again denote its i th component by \mathbf{v}_i . Then for $q \in Q$, $m \geq 0$, $f \in \Sigma^{(m)}$, and $\mathbf{t}_1, \dots, \mathbf{t}_m \in \mathcal{T}_\Sigma$, the output $\llbracket f(\mathbf{t}_1, \dots, \mathbf{t}_m) \rrbracket_q \in \mathbb{Q}$ can be computed arithmetically by

$$\llbracket f(\mathbf{t}_1, \dots, \mathbf{t}_m) \rrbracket_q = \llbracket \delta(q, f) \rrbracket_M(\llbracket \mathbf{t}_1 \rrbracket, \dots, \llbracket \mathbf{t}_m \rrbracket) \quad (1)$$

where for $T \in (\Delta \cup Q(X_m))^*$ and a vector $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_m)$ of vectors $\mathbf{x}_i \in \mathbb{Q}^n$ the number $\llbracket T \rrbracket_M \mathbf{x}$ is given by:

$$\begin{aligned} \llbracket c \rrbracket_M \mathbf{x} &= c \\ \llbracket j(x_i) \rrbracket_M \mathbf{x} &= \mathbf{x}_{ij} \\ \llbracket T'_1 + T'_2 \rrbracket_M \mathbf{x} &= \llbracket T'_1 \rrbracket_M \mathbf{x} + \llbracket T'_2 \rrbracket_M \mathbf{x} \\ \llbracket c \cdot T' \rrbracket_M \mathbf{x} &= c \cdot \llbracket T' \rrbracket_M \mathbf{x}. \end{aligned}$$

By structural induction on T , we conclude that

$$\llbracket T \rrbracket_M \mathbf{x} = b_0 + \sum_{i=1}^m \sum_{j=1}^n b_{ij} \cdot \mathbf{x}_{ij}$$

for suitable numbers $b_0, b_{ij} \in \mathbb{Q}$. Thus, $\llbracket T \rrbracket_M$ and hence also $\llbracket \delta(q, f) \rrbracket_M$ constitutes a *multi-affine* mapping from $(\mathbb{Q}^n)^m$ to \mathbb{Q} . Technically, a multi-affine mapping H is affine in each argument. This means that the transformation H' corresponding to the k th argument and fixed $\mathbf{x}_1, \dots, \mathbf{x}_{k-1}, \mathbf{x}_{k+1}, \dots, \mathbf{x}_m$, which is defined by:

$$H'(\mathbf{x}') = H(\mathbf{x}_1, \dots, \mathbf{x}_{k-1}, \mathbf{x}', \mathbf{x}_{k+1}, \dots, \mathbf{x}_m)$$

is affine, i.e.,

$$H'(\mathbf{y}_0 + \sum_{r=1}^n \lambda_r (\mathbf{y}_r - \mathbf{y}_0)) = H'(\mathbf{y}_0) + \sum_{r=1}^n \lambda_r (H'(\mathbf{y}_r) - H'(\mathbf{y}_0))$$

holds for vectors $\mathbf{y}_0, \dots, \mathbf{y}_n \in \mathbb{Q}^n$ and $\lambda_1, \dots, \lambda_n \in \mathbb{Q}$. Accordingly, we define the (output) semantics of $f \in \Sigma$ of arity m as the function $\llbracket f \rrbracket : (\mathbb{Q}^n)^m \rightarrow \mathbb{Q}^n$ by:

$$\llbracket f \rrbracket \mathbf{x} = (\llbracket \delta(1, f) \rrbracket_M \mathbf{x}, \dots, \llbracket \delta(n, f) \rrbracket_M \mathbf{x}) \quad (2)$$

which again is multi-affine.

Theorem 2: Let Σ be a fixed ranked alphabet, and A a DTTA automaton over Σ . Let M a total unary y DT transducer with input alphabet Σ , and q, q' states of M . It is decidable in polynomial time whether or not $\llbracket q \rrbracket_M(t) = \llbracket q' \rrbracket_M(t)$ for all $t \in \mathcal{L}(A)$.

Proof: By repeated application of the transformations $\llbracket f \rrbracket$, $f \in \Sigma$, every tree $t \in \mathcal{T}_\Sigma$ gives rise to a vector $\llbracket t \rrbracket \in \mathbb{Q}^n$. For a set $S \subseteq \mathcal{T}_\Sigma$, let $\llbracket S \rrbracket = \{\llbracket t \rrbracket \mid t \in S\}$. Then two states q, q' are equivalent relative to $S \subseteq \mathcal{T}_\Sigma$ iff $H_{qq'}(\mathbf{v}) = 0$ for all $\mathbf{v} = (\mathbf{v}_1, \dots, \mathbf{v}_n) \in \llbracket S \rrbracket$, where the function $H_{qq'}$ is given by $H_{qq'}(\mathbf{v}) = \mathbf{v}_q - \mathbf{v}_{q'}$. The


```

forall ( $p \in P$ )  $B_p := \emptyset$ ;
repeat
  done := true;
  forall ( $p, p_1, \dots, p_m \in P, f \in \Sigma$  with  $\rho(p, f) = (p_1, \dots, p_m)$ )
    forall ( $(\mathbf{v}_1, \dots, \mathbf{v}_m) \in B_{p_1} \times \dots \times B_{p_m}$ )
       $\mathbf{v} := \llbracket f \rrbracket(\mathbf{v}_1, \dots, \mathbf{v}_m)$ ;
      if  $\mathbf{v} \notin \text{aff}(B_p)$ 
         $B_p := B_p \cup \{\mathbf{v}\}$ ;
      done := false;
until (done = true);

```

Figure 1. Computing bases for the closures $\text{aff}(\{\llbracket t \rrbracket \mid t \in \text{dom}(p)\}), p \in P$.

set of vectors in $\llbracket \mathcal{L}(A) \rrbracket$ can be characterized by means of a constraint system. Consider the collection of sets $V_p, p \in P$, which are the least sets with

$$V_p \supseteq \llbracket f \rrbracket(V_{p_1}, \dots, V_{p_m}) \quad (3)$$

whenever $\rho(p, f) = (p_1, \dots, p_m)$ holds. Then $\{\llbracket t \rrbracket \mid t \in \mathcal{L}(A)\}$ is precisely given by the set V_{p_0} .

For a set $V \subseteq \mathbb{Q}^n$ of n -dimensional vectors, let $\text{aff}(V)$ denote the *affine closure* of V . This set is obtained from V by adding all affine combinations of elements in V :

$$\text{aff}(V) = \{\mathbf{s}_0 + \sum_{j=1}^r \lambda_j \cdot (\mathbf{s}_j - \mathbf{s}_0) \mid r \geq 0, \mathbf{s}_0, \dots, \mathbf{s}_r \in V, \lambda_1, \dots, \lambda_r \in \mathbb{Q}\}.$$

Every set $V \subseteq \mathbb{Q}^n$ has a subset $B \subseteq V$ of cardinality at most $n + 1$ such that the affine closures of B and V coincide. A set B with this property of minimal cardinality is also called *affine basis* of $\text{aff}(V)$. For an affine function $H : \mathbb{Q}^n \rightarrow \mathbb{Q}$ such as $H_{qq'}$ and every subset $V \subseteq \mathbb{Q}^n$, the following three statements are equivalent:

- 1) $H(\mathbf{v}) = 0$ for all $\mathbf{v} \in V$;
- 2) $H(\mathbf{v}) = 0$ for all $\mathbf{v} \in \text{aff}(V)$;
- 3) $H(\mathbf{v}) = 0$ for all $\mathbf{v} \in B$ if B is any subset of V with $\text{aff}(B) = \text{aff}(V)$.

Instead of verifying that $H(\mathbf{v}) = 0$ holds for all elements \mathbf{v} of V_{p_0} , it suffices to test $H(\mathbf{v}) = 0$ for all elements \mathbf{v} of an affine basis $B \subseteq V_{p_0}$. Accordingly, we are done if we succeed in computing an affine basis of the set $\text{aff}(V_{p_0})$. It is unclear, though, how the least solution $V_p, p \in P$, of the constraint system (3) can be computed. Instead of computing this least solution, we propose to consider the least solution of the constraint system (3), not over *arbitrary* subsets but over *affine* subsets of \mathbb{Q}^n only. Like the set $\mathcal{P}(\mathbb{Q}^n)$ of all subsets of \mathbb{Q} (ordered by subset inclusion), the set $\mathcal{A}(\mathbb{Q}^n)$ of all affine subsets of \mathbb{Q} (still ordered by subset inclusion) forms a complete lattice, but where the least upper bound operation is not given by set union. Instead, for a family \mathcal{B} of affine sets, the least affine set containing all $B \in \mathcal{B}$ is given by:

$$\bigsqcup \mathcal{B} = \text{aff}(\bigcup \mathcal{B}).$$

We remark that affine mappings commute with least upper bounds, i.e., for every affine mapping $F : \mathbb{Q}^n \rightarrow \mathbb{Q}^n$,

$$\begin{aligned}
F(\bigsqcup \mathcal{B}) &= F(\text{aff}(\bigcup \mathcal{B})) = \text{aff}(F(\bigcup \mathcal{B})) \\
&= \text{aff}(\{F(\mathbf{v}) \mid \mathbf{v} \in \bigcup \mathcal{B}\}) \\
&= \bigsqcup \{F(B) \mid B \in \mathcal{B}\}.
\end{aligned}$$

Let $V_p, p \in P$, and $V_p^\sharp, p \in P$, denote the least solutions of (3) w.r.t. the complete lattices $\mathcal{P}(\mathbb{Q}^n)$ and $\mathcal{A}(\mathbb{Q}^n)$, respectively. Since for each $f \in \Sigma$, $\llbracket f \rrbracket$ is affine in each of its arguments, it follows by the transfer lemma of [49] (see also [50]), that

$$\text{aff}(V_p) = V_p^\sharp \quad (p \in P).$$

The complete lattice $\mathcal{A}(\mathbb{Q}^n)$, on the other hand, satisfies the *ascending chain* condition. This means that every increasing sequence of affine subsets is ultimately stable. Therefore, the least solution $V_p^\sharp, p \in P$, of the constraint system (3) over $\mathcal{A}(\mathbb{Q}^n)$ can be effectively computed by fixpoint iteration. One such fixpoint iteration algorithm is presented in Figure 1. Each occurring affine subset of \mathbb{Q}^n is represented by a basis. For the resulting basis B_{p_0} of the affine subset $V_{p_0}^\sharp = \text{aff}(V_{p_0})$ we finally may check whether or not $H_{qq'}(\mathbf{v}) = 0$ for all $\mathbf{v} \in B_{p_0}$, which completes the procedure.

The algorithm of Figure 1 starts with empty sets B_p for all $p \in P$. Then it repeatedly performs one round through all transitions $\rho(p, f) = (p_1, \dots, p_m)$ of A while the flag *done* is **false**. For each transition $\rho(p, f) = (p_1, \dots, p_m)$, the transformation $\llbracket f \rrbracket$ is applied to every m -tuple $\mathbf{v} = (\mathbf{v}_1, \dots, \mathbf{v}_m)$ with $\mathbf{v}_i \in B_{p_i}$. The resulting vectors then are added to B_p — whenever they are not yet contained in the affine closure $\text{aff}(B_p)$ of B_p . The iteration terminates when during a full round of the *repeat-until* loop, no further element has been added to any of the B_p .

In the following, we assume a uniform cost measure where arithmetic operations are counted as 1. Thus, evaluating a right-hand side $\delta(q, f)$ takes time at most proportional to the number of symbols occurring in $\delta(q, f)$. Concerning the complexity of the algorithm, we note:

- The algorithm performs at most $h \cdot (n + 1)$ rounds on the *repeat-until* loop (h and n are the number of states of A and M , respectively);
- In each round of the *repeat-until* loop, for each transition of A , at most $(n + 1)^m$ tuples are considered (m is the maximal arity of an input symbol);
- for each encountered vector, time $O(n^3)$ is sufficient to check whether the vector is contained in the affine closure of the current set B_p (see, e.g., chapter 28.1 of [51]).

Accordingly, a full round of the *repeat-until* loop can be executed in time $O(|A| \cdot |M| \cdot n^m)$ — giving us an upper complexity bound $O(|A| \cdot |M| \cdot hn^{m+4})$ for the algorithm where m can be chosen as 2, according to Lemma 1. ■

The base algorithm as presented in the proof of Theorem 2, can be further improved as follows:

- We replace the Round-robin iteration by a worklist iteration which re-schedules the evaluation of a transition of A for a state $p' \in P$ and an input symbol f only if B_{p_i} for one of the states p_i in $\rho(p', f)$ has been updated.
- We keep track of the set of tuples which have already been processed for a given pair (p', f) , f an input symbol and p' state of A . This implies that throughout the whole fixpoint iteration, for each such pair (p', f) , inclusion in the affine closure must only be checked for $(n + 1)^m$ tuples.
- For a non-empty affine basis B , we can maintain a single element $v' \in B$, together with a basis of the linear space L_B corresponding to B , spanned by the vectors $(v - v'), v \in B \setminus \{v'\}$. By maintaining a basis of L_B in *Echelon* form, membership in $\text{aff}(B)$ can be tested in time $O(n^2)$.

Applying these three optimizations, the overall complexity comes down to $O(|A| \cdot |M| \cdot n^{m+2})$.

So far, we have compared the output behavior of two states of a unary total y DT transducer M relative to some DTTA automaton only. Our decision procedure for equivalence, however, readily extends to arbitrary unary y DT transducers. Note that the exponential upper bound of Theorem 3 is sharp, since non-emptiness for unary y DT transducers is already EXPTIME-complete (see Theorem 9 of [39]).

Theorem 3: Equivalence for (possibly partial) unary y DT transducers can be decided in deterministic exponential time. If the transducers are linear, then the algorithm runs in polynomial time.

Proof: First, we, w.l.o.g., may assume that we are given two states q_0, q'_0 of a single y DT transducer, and the task is to decide whether the partial mappings $\llbracket q_0 \rrbracket_M$ and $\llbracket q'_0 \rrbracket_M$ coincide, i.e., whether (1) $\llbracket q_0 \rrbracket_M(t)$ is defined iff $\llbracket q'_0 \rrbracket_M(t)$ is defined, and (2) $\llbracket q_0 \rrbracket_M(t) = \llbracket q'_0 \rrbracket_M(t)$ whenever both are defined. In order to decide the former task, we construct DTTA automata A, A' where the languages of A and A' are precisely given by the domains of the translations $\llbracket q_0 \rrbracket_M$ and $\llbracket q'_0 \rrbracket_M$, respectively.

The set of states and transitions of A can be determined as the smallest subset P of sets $Q' \subseteq [n]$ together with the partial function ρ as follows. First, $\{q_0\} \in P$ which also serves as the initial state of A . Then for every element $Q' \in P$ and every input symbol $f \in \Sigma$ of some arity m , where $\delta(q, f)$ is defined for each $q \in Q'$, every set Q'_i is contained in P for $i = 1, \dots, m$, where Q'_i is the set of all states $q' \in [n]$ such that $q'(x_i)$ occurs in the right-hand side $\delta(q, f)$ for some $q \in Q'$. In this case then ρ has the transition $\rho(Q', f) = Q'_1 \dots Q'_m$. The DTTA automaton A' is obtained by starting with the initial state $\{q'_0\}$ instead of $\{q_0\}$, and subsequently proceeding analogously to the construction of A . Assume that the number of states of A and A' are q and q' , respectively. Then property (1) is satisfied iff $\mathcal{L}(A) = \mathcal{L}(A')$. This can be verified in time polynomial in the sizes of A and A' . Now assume that $\mathcal{L}(A) = \mathcal{L}(A')$. Then we construct a total y DT transducer M' from M by adding to the transition function of M a transition $q(f(x_1, \dots, x_m)) \rightarrow \epsilon$ for every state q and input symbol f — where M does not yet provide a transition. By construction, $\llbracket q \rrbracket_{M'}(t) = \llbracket q \rrbracket_M(t)$ whenever $\llbracket q \rrbracket_M(t)$ is defined. Therefore for every $t \in \mathcal{L}(A)$, $\llbracket q_0 \rrbracket_{M'}(t) = \llbracket q'_0 \rrbracket_{M'}(t)$ iff $\llbracket q_0 \rrbracket_M(t) = \llbracket q'_0 \rrbracket_M(t)$. Using the algorithm of Theorem 2, the latter can be decided in time polynomial in the sizes of A and M' .

The size of the DTTA automaton A characterizing the domain of the y DT transducer M is at most exponential in the size of M . In case, however, that M is linear, the size of the corresponding automaton A is at most linear in the size of M . From that, the complexity bounds of the theorem follow. ■

Theorems 2 and 3 can be applied to decide *Abelian* equivalence of y DT transducers with arbitrary output alphabet. *Abelian* equivalence of two deterministic tree-to-string transducers means that the outputs for every input tree coincide up to the ordering of output symbols.

IV. NON-SELF-NESTED UNARY TRANSDUCERS WITH PARAMETERS

We consider unary *non-self-nested* y DMT transducers and show that their equivalence problem can be solved in co-randomized polynomial time. This implies equivalence with the same complexity for (arbitrary) linear y DT transducers.

Deterministic macro tree-to-string transducers (y DMT transducers) combine y DT transducers with the nesting present in macro grammars. Each state of a y DMT transducer takes a fixed number of *parameters* (of type output tree). Recall that a y DMT transducer is non-self-nested if whenever $q'(x_j, \dots)$ occurs nested in $q(x_i, \dots)$ implies that $i \neq j$. Note that non-self-nested y DMT transducers are strictly more powerful than y DT transducers as shown in the following lemma.

Lemma 4: The translation of Example 1, which is realized by a non-self-nested unary y DMT transducer, cannot be realized by any y DT transducer.

Proof: For convenience, we prove a slightly stronger result, namely, that this translation also cannot be realized by any y DT transducer even if it is equipped with *regular look-ahead* (a y DT^R transducer). Assume for a contradiction, that a given y DT^R transducer N realizes the translation of M where N has a finite set Q of states and uses the finite bottom-up automaton B for providing look-ahead information about the input. Let $n_1 \neq n_2$ so that $a^{n_1}(e)$ and $a^{n_2}(e)$ correspond to the same look-ahead state of B . Then for $i = 1, 2$ and $j = 1, 2$,

$$N(f(a^{n_i}(e), a^{n_j}(e))) = c + \sum_{q \in Q} c_q \llbracket q \rrbracket_N(a^{n_i}(e)) + \sum_{q \in Q} c'_q \llbracket q \rrbracket_N(a^{n_j}(e))$$

for suitable numbers c, c_q, c'_q (independent of i, j). For $j = 1, 2$, consider the *difference* in the outputs:

$$\begin{aligned} \Delta_j &= N(f(a^{n_1}(e), a^{n_j}(e))) - N(f(a^{n_2}(e), a^{n_j}(e))) \\ &= \sum_{q \in Q} c_q (\llbracket q \rrbracket_N(a^{n_1}(e)) - \llbracket q \rrbracket_N(a^{n_2}(e))) \end{aligned}$$

and observe that it is independent of j . According to our assumption, N realizes the translation of M . Therefore,

$$\begin{aligned} 0 &= \Delta_1 - \Delta_2 \\ &= (n_1 - n_2) \cdot n_1 - (n_1 - n_2) \cdot n_2 \\ &= (n_1 - n_2) \cdot (n_1 - n_2) \\ &\neq 0 \end{aligned}$$

— a contradiction. Hence the translation of M cannot be realized by any $y\text{DT}^R$ transducer. \blacksquare

As in the case for unary $y\text{DT}$ transducers, we first consider *total* unary $y\text{DMT}$ transducers only, but relative to a DTTA automaton A . Assume that a unary $y\text{DMT}$ transducer M is given by $M = ([n], \Sigma, \{d\}, i_0, \delta)$. Recall that we assume that all states have exactly $l + 1$ parameters where the first one is the input tree and the remaining l parameters accumulate output strings, i.e., numbers. The output for a state q and an m -ary input symbol $f \in \Sigma$, then is given by:

$$\llbracket f(\mathbf{t}_1, \dots, \mathbf{t}_m) \rrbracket_{q\mathbf{y}} = \llbracket T \rrbracket_M(\llbracket \mathbf{t}_1 \rrbracket_{\mathbf{y}}, \dots, \llbracket \mathbf{t}_m \rrbracket_{\mathbf{y}}) \quad (4)$$

when $q(f(x_1, \dots, x_m), y_1, \dots, y_l) \rightarrow T$ is a rule of M , and \mathbf{y} is a vector of parameters in \mathbb{Q}^l . Here, $\llbracket T' \rrbracket_{\mathbf{x}\mathbf{y}}$ for a vector $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_m)$ of vectors $\mathbf{x}_i \in (\mathbb{Q}^l \rightarrow \mathbb{Q})^n$ is defined by:

$$\begin{aligned} \llbracket c \rrbracket_M \mathbf{x}\mathbf{y} &= c \\ \llbracket y_k \rrbracket_M \mathbf{x}\mathbf{y} &= \mathbf{y}_k \\ \llbracket c \cdot T' \rrbracket_M \mathbf{x}\mathbf{y} &= c \cdot \llbracket T' \rrbracket_{\mathbf{x}\mathbf{y}} \\ \llbracket T'_1 + T'_2 \rrbracket_M \mathbf{x}\mathbf{y} &= \llbracket T'_1 \rrbracket_M \mathbf{x}\mathbf{y} + \llbracket T'_2 \rrbracket_M \mathbf{x}\mathbf{y} \\ \llbracket j(x_i, T'_1, \dots, T'_l) \rrbracket_M \mathbf{x}\mathbf{y} &= \mathbf{x}_{ij}(\llbracket T'_1 \rrbracket_M \mathbf{x}\mathbf{y}, \dots, \llbracket T'_l \rrbracket_M \mathbf{x}\mathbf{y}). \end{aligned}$$

By structural induction, we verify that for all input trees $t \in \mathcal{T}_\Sigma$ and all states q of the $y\text{DMT}$ transducer M , $\llbracket t \rrbracket_q$ is an *affine function* $\mathbb{Q}^l \rightarrow \mathbb{Q}$, i.e., $\llbracket t \rrbracket_q \mathbf{y} = \mathbf{v}_{q0} + \mathbf{v}_{q1}\mathbf{y}_1 + \dots + \mathbf{v}_{ql}\mathbf{y}_l$ for suitable $\mathbf{v}_{qj} \in \mathbb{Q}$. Accordingly, $\llbracket t \rrbracket$ can be represented as the two-dimensional matrix $\mathbf{v} = (\mathbf{v}_{qj}) \in \mathbb{Q}^{n \times (l+1)}$.

Now assume that the arguments $\mathbf{x}_i, i = 1, \dots, m$, are all vectors of affine functions $\mathbb{Q}^l \rightarrow \mathbb{Q}$, and let \mathbf{x} denote the triply indexed set (\mathbf{x}_{ijk}) of coefficients in \mathbb{Q} ($i = 1, \dots, m, j = 1, \dots, n$ and $k = 0, \dots, l$) representing these functions. Then

$$\llbracket f \rrbracket_q \mathbf{x}\mathbf{y} = \mathbf{r}_{q0}^{(f)} + \mathbf{r}_{q1}^{(f)} \cdot \mathbf{y}_1 + \dots + \mathbf{r}_{ql}^{(f)} \cdot \mathbf{y}_l \quad (5)$$

where $\mathbf{r}_{qk}^{(f)}$ is a polynomial over the variables \mathbf{x} . Thus, $\llbracket f \rrbracket$ can be represented by the matrix $\mathbf{r}^{(f)} = (\mathbf{r}_{jk}^{(f)}) \in \mathbb{Q}[\mathbf{x}]^{n \times (l+1)}$.

Example 3: Consider the $y\text{DMT}$ transducer M from Example 1 which we extend to a total $y\text{DMT}$ transducer by adding the rules

$$q_0(a(e), y_1) \rightarrow 0 \quad q_0(e, y_1) \rightarrow 0 \quad q(f(x_1, x_2), y_1) \rightarrow 0$$

Then we obtain:

$$\begin{aligned} \llbracket f \rrbracket_{q_0}(\mathbf{x}_1, \mathbf{x}_2)(\mathbf{y}_1) &= \mathbf{x}_{1q0} + \mathbf{x}_{1q1} \cdot (\mathbf{x}_{2q0} + \mathbf{x}_{2q1} \cdot 1) \\ &= \mathbf{x}_{1q0} + \mathbf{x}_{1q1} \cdot \mathbf{x}_{2q0} + \mathbf{x}_{1q1} \cdot \mathbf{x}_{2q1} \\ \llbracket a \rrbracket_q(\mathbf{x}_1)(\mathbf{y}_1) &= \mathbf{y}_1 + \mathbf{x}_{1q0} + \mathbf{x}_{1q1} \cdot \mathbf{y}_1 \\ &= \mathbf{x}_{1q0} + (1 + \mathbf{x}_{1q1}) \cdot \mathbf{y}_1 \\ \llbracket e \rrbracket_q(\mathbf{y}_1) &= 0. \end{aligned}$$

In this section, we first examine the case that the $y\text{DMT}$ transducer M is non-self-nested (such as the $y\text{DMT}$ transducer from Example 1). Then each polynomial $\mathbf{r}_{jk}^{(f)}$ in (5) is a sum of products:

$$a \cdot \mathbf{x}_{i_1 j_1 k_1} \cdot \dots \cdot \mathbf{x}_{i_s j_s k_s}$$

where the i_1, \dots, i_s are pairwise distinct, i.e., each argument \mathbf{x}_i contributes at most one factor to each product. We conclude that the transformation $\llbracket f \rrbracket$ is *multi-affine*. This means that the mapping $\llbracket f \rrbracket$ when applied to an m -tuple of values in $\mathbb{Q}^{n \times (l+1)}$ (i.e., vectors of *affine functions*) is an *affine function* of each of the \mathbf{x}_i and \mathbf{y} , when the other arguments are kept constant. Thus, $\llbracket f \rrbracket$ commutes with affine combinations in any of the arguments \mathbf{x}_i and, for each sequence $\mathbf{x}_1, \dots, \mathbf{x}_m$ of matrices in $\mathbb{Q}^{n \times (l+1)}$, again results in an affine function of \mathbf{y} . \blacksquare

As in the case of y DT transducers, we can construct a constraint system analogously to the system of constraints (3) whose unknowns are indexed with the states from the automaton A — only that now each unknown V_p receives a set of values in $\mathbb{Q}^{n \times (l+1)}$ (vectors of affine transformations) instead of values in \mathbb{Q}^n (plain vectors). This constraint system has a least solution where the value for V_p is the set of all affine transformations $\llbracket t \rrbracket, t \in \text{dom}(p)$.

The two states q_0, q'_0 are equivalent with empty parameters relative to A iff $H(\llbracket t \rrbracket) = 0$ for all $t \in \mathcal{L}(A)$ where $H(\mathbf{v}) = \mathbf{v}_{q_0 0} - \mathbf{v}_{q'_0 0}$ for $\mathbf{v} = (\mathbf{v}_{qk}) \in \mathbb{Q}^{n \times (l+1)}$ (recall that for the affine function $\mathbf{v}_q = (\mathbf{v}_{q0}, \dots, \mathbf{v}_{ql})$, $\mathbf{v}_q(0, \dots, 0) = \mathbf{v}_{q0}$). As in the last section, the function H for testing equivalence of states, is affine.

For a set $V \subseteq \mathbb{Q}^{n \times (l+1)}$ of matrices, let $\text{aff}(V)$ denote the *affine closure* of V . This closure is defined analogously as for vectors. Only note that now an affine basis of the affine closure of V may have up to $n \cdot (l+1) + 1$ elements (compared to $n + 1$ in the last section). Now let H denote any affine function $H : \mathbb{Q}^{n \times (l+1)} \rightarrow \mathbb{Q}$. Analogously to the last section, for every set $V \subseteq \mathbb{Q}^{n \times (l+1)}$, $H(v) = 0$ holds for all $v \in V$ iff $H(v) = 0$ holds for all v in a basis of $\text{aff}(V)$. We conclude that it suffices to determine for each state p' of A , an *affine basis* $B_{p'}$ of the set $V_{p'}$ and then verify that $H(\mathbf{v}) = 0$ for all $\mathbf{v} \in B_{p_0}$ if p_0 is the initial state of A . With a similar algorithm as in the last section this is possible using a polynomial number of arithmetic operations only — given that the maximal arity of input symbols is bounded. Therefore, we obtain:

Theorem 5: Assume that M is a non-self-nested total unary y DMT transducer and A is a DTTA automaton. Then for every pair q, q' of states of M , it is decidable whether q and q' are equivalent relative to A . If the arity of input symbols is bounded by a constant, the algorithm requires only a polynomial number of arithmetic operations.

In case of non-self-nested y DMT transducers and multi-affine functions, the *lengths* of occurring numbers, however, can no longer be ignored. In order to calculate an upper bound to the occurring numbers, we first note that for each state p' of A , the basis of $\text{aff}(V_{p'})$ as calculated by our algorithm, is of the form $\llbracket t \rrbracket$ for a tree in $\mathcal{L}(A)$ of depth at most $((l+1) \cdot n + 1) \cdot h$ if n, l and h are the numbers of states and parameters of M , and the number of states of A , respectively. Concerning the lengths of occurring numbers, we prove:

Lemma 6: Assume that M is a non-self-nested unary y DMT transducer M where the ranks of input symbols are bounded by m , and the constants occurring in right-hand sides of rules are bounded by h . Then

$$\llbracket q \rrbracket_M(t)(\mathbf{y}_1, \dots, \mathbf{y}_l) \leq (h+1)^{(|M| \cdot (m+1))^N} \cdot b$$

if N is the depth of t and b is the maximum of the argument numbers $\mathbf{y}_1, \dots, \mathbf{y}_l$.

Proof: The proof is by induction on the depth of t . Thus, assume that $t = f \mathbf{t}$ with $\mathbf{t} = (\mathbf{t}_1, \dots, \mathbf{t}_m)$, $m \geq 0$, and assume that the induction hypothesis holds for the \mathbf{t}_i . Let $q(f(x_1, \dots, x_m), y_1, \dots, y_l) \rightarrow T$ be a rule of M . Then for $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_l)$,

$$\llbracket q \rrbracket_M(t) \mathbf{y} = \llbracket T \rrbracket_M \mathbf{t} \mathbf{y}$$

For T let $a(T)$ denote the nesting depth of calls $q(x_i, \dots)$. Note that $a(T) \leq m$ since M is assumed to be non-self-nested. Since $|T| \leq |M|$, and the depth of each \mathbf{t}_i is less than the depth of t , the assertion follows from the following claim:

$$(\llbracket T \rrbracket_M \mathbf{t} \mathbf{y}) \leq (h+1)^{|T| \cdot (a(T)+1) \cdot (|M| \cdot (m+1))^{N-1}} \cdot b$$

if N is the maximal depth of a tree \mathbf{t}_i . The proof of this claim is again by induction, but now on the structure of right-hand side T .

If T is a constant or equals \mathbf{y}_j for some j , the claim obviously holds. In case T equals a sum $T_1 + T_2$ or a scalar product $c \cdot T'$, the claim also follows easily from the inductive hypothesis. It remains to consider the case where $T = q'(x_i, T_1, \dots, T_l)$. By inductive hypothesis for the T_i , we find that for every i ,

$$\llbracket T_i \rrbracket_M \mathbf{t} \mathbf{y} \leq (h+1)^{|T_i| \cdot a(T) \cdot (|M| \cdot (m+1))^{N-1}} \cdot b$$

since the nesting depth of each T_i is at most $a(T) - 1$. Therefore,

$$\begin{aligned}
& \llbracket q'(x_i, T_1, \dots, T_l) \rrbracket_M \mathbf{t} \mathbf{y} \\
&= \llbracket q' \rrbracket_M (\mathbf{t}_i) (\llbracket T_1 \rrbracket_M \mathbf{t} \mathbf{y}, \dots, \llbracket T_l \rrbracket_M \mathbf{t} \mathbf{y}) \\
&\leq (h+1)^{(|M| \cdot (m+1))^{N-1}} \cdot (h+1)^{|T| \cdot a(M) \cdot (|M| \cdot (m+1))^{N-1}} \cdot b \\
&\leq (h+1)^{(|M| \cdot (m+1))^{N-1} + |T| \cdot a(M) \cdot (|M| \cdot (m+1))^{N-1}} \cdot b \\
&\leq (h+1)^{|T| \cdot (a(M)+1) \cdot (|M| \cdot (m+1))^{N-1}} \cdot b
\end{aligned}$$

since $|T_i| \leq |T|$ and the nesting-depths of any of the $|T_i|$ is bounded by $a(T) - 1$. This completes the proof. \blacksquare

Accordingly, the *bit length* $Z(N)$ of numbers occurring in $\llbracket t \rrbracket$ for trees of depth N is bounded by $O((|M| \cdot (m+1))^N)$ where m is the maximal rank of an input symbol. This means that, for $m > 1$, the *bit lengths* of occurring numbers can only be bounded by an exponential in the sizes of M and A . Still, in-equivalence can be decided in *randomized polynomial time*:

Theorem 7: In-equivalence of states of a non-self-nested total unary y DMT transducer relative to a DTTA automaton, is decidable in *randomized polynomial time*, i.e., there is a polynomial probabilistic algorithm which in case of equivalence, always returns **false**, while in case of non-equivalence returns **true** with probability at least 0.5.

Proof: Assume that M is a non-self-nested total unary y DMT transducer and A a DTTA automaton. Our goal is to decide whether or not $\llbracket q_0 \rrbracket_M(t) = \llbracket q'_0 \rrbracket_M(t)$ for all $t \in \mathcal{L}(A)$. Now let k denote any prime number. Then the set of integers modulo k , \mathbb{Z}_k , again forms a field. This means that we can realize the algorithm for determining affine closures of the sets V_p as well as the check whether an affine mapping H returns 0 for all elements of an affine basis now over \mathbb{Z}_k . The resulting algorithm allows us to decide whether the outputs for q_0, q'_0 coincide for all inputs from $\mathcal{L}(A)$ modulo the prime number k by using polynomially many operations on numbers of length $O(\log(k))$ only. In particular, if non-equivalence is found, then q_0, q'_0 cannot be equivalent relative to A over \mathbb{Q} either.

Let 2^D be an upper bound to $Z(n \cdot (l+1) + 1) \cdot h$ (n, l the number of states of M and the number of parameters of states of M , respectively, and h the number of states of A) where D is polynomial in the sizes of M and A . Then we have:

Lemma 8: q_0, q'_0 are equivalent relative to A iff q_0, q'_0 are equivalent relative to A modulo 2^D distinct primes.

Proof: Assume that the latter holds. Then the product already of the smallest 2^D primes vastly exceeds 2^{2^D} . Therefore by the Chinese remainder theorem, $H(\llbracket t \rrbracket) = 0$ holds also over \mathbb{Q} for all $t \in \mathcal{L}(A)$ of depth at most $(n \cdot (l+1) + 1) \cdot h$. Therefore, q_0, q'_0 must be equivalent. \blacksquare

Clearly, if q_0 and q'_0 are equivalent relative to $\mathcal{L}(A)$, then they are also equivalent relative to $\mathcal{L}(A)$ modulo every prime number k . Therefore now assume that q_0 and q'_0 are not equivalent relative to $\mathcal{L}(A)$. Let K denote the set of all primes k so that q_0 and q'_0 are still equivalent relative to $\mathcal{L}(A)$ modulo k . By Lemma 8, this set has less than 2^D elements. Now consider the interval $[0, D \cdot e^D]$. Note that each number in this range has polynomial length only. When D is suitably large, this interval contains at least $e^D \geq 4 \cdot 2^D$ prime numbers (see, e.g., [52]). Therefore, with probability at least 0.75, a prime randomly drawn from this range is not contained in K and therefore witnesses that q_0 and q'_0 are not equivalent relative to $\mathcal{L}(A)$. Since a random prime can be drawn in polynomial time with probability 0.75, and $0.75 \cdot 0.75 \geq 0.5$, the assertion of the theorem follows. \blacksquare

V. GENERAL UNARY TRANSDUCERS WITH PARAMETERS

In the following, we drop the restriction that the y DMT transducer M is necessarily non-self-nested. Then the polynomials $\mathbf{p}_{jk}^{(f)}$ are no longer necessarily multi-linear in the variables $\mathbf{x}_1, \dots, \mathbf{x}_m$. Accordingly, techniques based on affine closures of the sets $\llbracket \text{dom}(p) \rrbracket$ are no longer appropriate.

Instead, we propose to generally reason about *properties* satisfied by the elements of the sets $\llbracket \text{dom}(p) \rrbracket$. Let $\mathbf{z} = \{\mathbf{z}_{qk} \mid q = 1, \dots, n, k = 0, \dots, l\}$ denote a fresh set of variables. The key concept which we introduce here is the notion of an *inductive invariant* of M relative to the DTTA automaton A . As candidate invariants we only require conjunctions of equalities $r \doteq 0$ where $r \in \mathbb{Q}[\mathbf{z}]$ is a polynomial over the variables \mathbf{z} with rational

coefficients. Instead of referring to such a conjunction directly, it is mathematically more convenient to consider the *ideal* generated from the polynomials in the conjunction. Formally, an ideal of a ring R is a subset $J \subseteq R$ such that for all $a, a' \in J$, $a + a' \in J$ and for all $a \in J$ and $r \in R$, $r \cdot a \in J$. The smallest ideal containing a set S of elements, is the set $\langle S \rangle = \{\sum_{j=1}^k r_j \cdot s_j \mid k \geq 0, r_1, \dots, r_k \in R, s_1, \dots, s_k \in S\}$. The smallest ideal containing ideals J_1, J_2 is their *sum* $J_1 + J_2 = \{s_1 + s_2 \mid s_1 \in J_1, s_2 \in J_2\}$.

Using ideals instead of conjunctions of polynomial equalities is justified because for every $S \subseteq \mathbb{Q}[\mathbf{z}]$ and every $\mathbf{v} \in \mathbb{Q}^{n \times (l+1)}$, it holds that $s(\mathbf{v}) = 0$ for all $s \in \langle S \rangle$ iff $s(\mathbf{v}) = 0$ for all $s \in S$.

An inductive invariant I of the y DMT transducer M relative to A is a family of ideals $I_p \subseteq \mathbb{Q}[\mathbf{z}], p \in P$, such that for all transitions $\rho(p, f) = (p_1, \dots, p_m)$,

$$I_p \subseteq \{r' \in \mathbb{Q}[\mathbf{z}] \mid r'[\mathbf{r}^{(f)} / \mathbf{z}] \in \langle I_{p_1}(\mathbf{x}_1) \cup \dots \cup I_{p_m}(\mathbf{x}_m) \rangle\} \quad (6)$$

holds where we used $[\mathbf{v} / \mathbf{z}]$ to denote the substitution of the expressions \mathbf{v}_{jk} for the variables \mathbf{z}_{jk} . Likewise for an ideal $J \subseteq \mathbb{Q}[\mathbf{z}]$, $J(\mathbf{x}_i)$ denotes the ideal:

$$J(\mathbf{x}_i) = \{s[\mathbf{x}_i / \mathbf{z}] \mid s \in J\}.$$

The constraint (6) for the transition $\rho(p, f) = (p_1, \dots, p_m)$ formalizes the following intuition. For every polynomial r' , the polynomial $r'[\mathbf{r}^{(f)} / \mathbf{z}]$ can be understood as the *weakest precondition* of r' w.r.t. the semantics $\mathbf{r}^{(f)}$ of the input symbol f . It is a polynomial in the variables \mathbf{x} where the variables in \mathbf{x}_i refer to the i th argument of f . The constraint (6) therefore expresses that the weakest precondition of every polynomial in I_p can be generated from the polynomials provided by I for the states p_i — after the variables \mathbf{z} therein have been appropriately renamed with \mathbf{x}_i .

We verify for every inductive invariant I that each polynomial in the ideal I_p constitutes a valid property of all input trees in $\text{dom}(p)$. This means:

Theorem 9: Assume that I is an inductive invariant of the y DMT transducer M relative to A . Then for every state p of A and polynomial $r' \in I_p$, $r'(\llbracket t \rrbracket) = 0$ holds for all $t \in \text{dom}(p)$.

Proof: By structural induction on t , we prove that $r'(\llbracket t \rrbracket) = 0$ holds. Assume that $\rho(p, f) = (p_1, \dots, p_m)$ and $t = f(\mathbf{t}_1, \dots, \mathbf{t}_m)$ where (by induction hypothesis) for every $i = 1, \dots, m$ and every $r' \in I_{p_i}$, $r'(\llbracket \mathbf{t}_i \rrbracket) = 0$ holds. Since $\llbracket t \rrbracket_{qk} = \mathbf{r}_{qk}^{(f)}(\llbracket \mathbf{t}_1 \rrbracket, \dots, \llbracket \mathbf{t}_m \rrbracket)$, we have that

$$r'(\llbracket t \rrbracket) = r'[\mathbf{r}^{(f)} / \mathbf{z}](\llbracket \mathbf{t}_1 \rrbracket, \dots, \llbracket \mathbf{t}_m \rrbracket)$$

Since I is inductive, $r'[\mathbf{r}^{(f)} / \mathbf{z}]$ can be rewritten as a sum:

$$r'[\mathbf{r}^{(f)} / \mathbf{z}] = \sum_{i=1}^m \sum_{\mu_i=1}^{u_i} r'_{i\mu_i} s_{i\mu_i}[\mathbf{x}_i / \mathbf{z}]$$

for suitable polynomials $r'_{i\mu_i}$ where for $i = 1, \dots, m$, all $s_{i\mu_i} \in I_{p_i}$. Therefore for all μ_i , $s_{i\mu_i}(\llbracket \mathbf{t}_i \rrbracket) = 0$, and thus

$$r'(\llbracket t \rrbracket) = r'[\mathbf{r}^{(f)} / \mathbf{z}](\llbracket \mathbf{t}_1 \rrbracket, \dots, \llbracket \mathbf{t}_m \rrbracket) = 0.$$

■

We conclude that every inductive invariant I with $r' \in I_p$ provides us with a *certificate* that $r'(\llbracket t \rrbracket) = 0$ holds for all $t \in \text{dom}(p)$. In the next step, we convince ourselves that the reverse implication also holds, i.e., for all polynomials r' for which $r'(\llbracket t \rrbracket) = 0$ holds for all $t \in \text{dom}(p)$, an inductive invariant I exists with $r' \in I_p$. In order to prove this statement, we consider the family \bar{I} of ideals $\bar{I}_p, p \in P$, where

$$\bar{I}_p = \{r' \in \mathbb{Q}[\mathbf{z}] \mid \forall t \in \text{dom}(p). r'(\llbracket t \rrbracket) = 0\}.$$

Thus, \bar{I}_p is the set of *all* polynomials which represent a polynomial property of trees in $\text{dom}(p)$. We next prove that \bar{I} is indeed an inductive invariant.

Theorem 10: \bar{I} is an inductive invariant of the y DMT transducer M relative to A .

Proof: For any set $V \subseteq \mathbb{Q}^{n \times (l+1)}$ of vectors, let $\mathcal{I}(V)$ denote the set of polynomials r' over \mathbf{z} which vanish on V , i.e., with $r'(\mathbf{v}) = 0$ for all $\mathbf{v} \in V$. We remark that for disjoint sets of variables $\mathbf{x}_1, \dots, \mathbf{x}_m$ with $\mathbf{x}_i = \{\mathbf{x}_{ijk} \mid j = 1, \dots, n, k = 0, \dots, l\}$, and arbitrary sets $V_i \subseteq \mathbb{Q}^{n \times (l+1)}$,

$$\mathcal{I}(V_1 \times \dots \times V_m) = \langle \mathcal{I}(V_1)(\mathbf{x}_1) \cup \dots \cup \mathcal{I}(V_m)(\mathbf{x}_m) \rangle \quad (7)$$

holds when considered as ideals of $\mathbb{Q}[\mathbf{x}] = \mathbb{Q}[\mathbf{x}_1 \cup \dots \cup \mathbf{x}_m]$. This means that the set of polynomials which vanish on the Cartesian product $V_1 \times \dots \times V_m$ is exactly given by the ideal in $\mathbb{Q}[\mathbf{x}]$ which is generated by the polynomials in $\mathbb{Q}[\mathbf{x}_i]$ which vanish on the set V_i ($i = 1, \dots, m$).

We remark that the ideal of $\mathbb{Q}[\mathbf{x}]$ generated from $\mathcal{I}(V_i)(\mathbf{x}_i)$ is exactly given by $\mathcal{I}(\top^{i-1} \times V_i \times \top^{m-i})$ where $\top = \mathbb{Q}^{n \times (l+1)}$. Accordingly, equality (7) can be rewritten to:

$$\mathcal{I}(V_1 \times \dots \times V_m) = \sum_{i=1}^m \mathcal{I}(\top^{i-1} \times V_i \times \top^{m-i}).$$

Thus, equality (7) is a consequence of the following lemma, which we could not find in the literature. Although formulated for \mathbb{Q} , the lemma holds (with the same proof) for any field.

Lemma 11: Let $V_1 \subseteq \mathbb{Q}^{m_1}, V_2 \subseteq \mathbb{Q}^{m_2}$ be subsets of vectors, with m_1, m_2 positive integers. Then

$$\mathcal{I}(V_1 \times V_2) = \mathcal{I}(V_1 \times \mathbb{Q}^{m_2}) + \mathcal{I}(\mathbb{Q}^{m_1} \times V_2).$$

Proof: Since $V_1 \times V_2 \subseteq V_1 \times \mathbb{Q}^{m_2}$, it follows that $I_1 := \mathcal{I}(V_1 \times \mathbb{Q}^{m_2}) \subseteq \mathcal{I}(V_1 \times V_2)$. Likewise, $I_2 := \mathcal{I}(\mathbb{Q}^{m_1} \times V_2) \subseteq \mathcal{I}(V_1 \times V_2)$, and the inclusion “ \supseteq ” follows.

The proof of the reverse inclusion uses Gröbner bases (for basic notions and concepts on Gröbner bases see the textbook by Becker and Weisspfenning [41]). Let $\mathbf{x} = \{\mathbf{x}_1 \dots, \mathbf{x}_{m_1+m_2}\}$ a suitable finite set of variables. Fix a monomial ordering on the polynomial ring $\mathbb{Q}[\mathbf{x}]$. With respect to this monomial ordering, let G_1, G_2 be Gröbner bases of I_1 and I_2 , respectively. Clearly $G_1 \cup G_2$ generates the sum $I_1 + I_2$. Since I_1 is generated by polynomials in $\mathbb{Q}[\mathbf{x}_1, \dots, \mathbf{x}_{m_1}]$, we have $G_1 \subset \mathbb{Q}[\mathbf{x}_1, \dots, \mathbf{x}_{m_1}]$, and also $G_2 \subset \mathbb{Q}[\mathbf{x}_{m_1+1}, \dots, \mathbf{x}_{m_1+m_2}]$. It follows by Buchberger’s criterion (see [41, Section 5.5]) that $G_1 \cup G_2$ is a Gröbner basis of $I_1 + I_2$. This implies that each polynomial $f \in \mathbb{Q}[\mathbf{x}]$ has a unique normal form $g := \text{nf}(f)$, which (by definition) has no monomial that is divisible by the leading monomial of any polynomial in $G_1 \cup G_2$, and which satisfies $f - g \in I_1 + I_2$. Moreover, if $f \in I_1 + I_2$ then $g = 0$.

For the proof of the reverse inclusion, take $f \in \mathbb{Q}[\mathbf{x}]$ that does *not* lie in $I_1 + I_2$. So $g := \text{nf}(f) \neq 0$. We have to show $f \notin \mathcal{I}(V_1 \times V_2)$, so we have to find $\mathbf{v} \in V_1$ and $\mathbf{w} \in V_2$ such that $f(\mathbf{v}, \mathbf{w}) \neq 0$. Considered as a polynomial in the variables $\mathbf{x}_1, \dots, \mathbf{x}_{m_1}$, g has a nonzero term $c\mathbf{x}_1^{e_1} \dots \mathbf{x}_{m_1}^{e_{m_1}}$ with $c \in \mathbb{Q}[\mathbf{x}_{m_1+1}, \dots, \mathbf{x}_{m_1+m_2}]$. Since none of the monomials of c are divisible by any leading monomial of a polynomial from G_2 , the Gröbner basis property of G_2 implies $c \notin I_2$, so there exists $\mathbf{w} \in V_2$ such that $c(\mathbf{w}) \neq 0$.

Now consider the polynomial $g_{\mathbf{w}} := g(\mathbf{x}_1, \dots, \mathbf{x}_{m_1}, \mathbf{w}) \in \mathbb{Q}[\mathbf{x}_1, \dots, \mathbf{x}_{m_1}]$. This is nonzero since one of its coefficients, $c(\mathbf{w})$, is nonzero. Moreover, no monomial from $g_{\mathbf{w}}$ is divisible by any leading monomial of a polynomial from G_1 , so $g_{\mathbf{w}} \notin I_1$, implying that there is a vector $\mathbf{v} \in V_1$ with $g_{\mathbf{w}}(\mathbf{v}) \neq 0$. This means $g(\mathbf{v}, \mathbf{w}) \neq 0$. But $(f - g)(\mathbf{v}, \mathbf{w}) = 0$ since $f - g \in I_1 + I_2 \subseteq \mathcal{I}(V_1 \times V_2)$, so we obtain $f(\mathbf{v}, \mathbf{w}) \neq 0$, finishing the proof. ■
For each state p of A , let $V_p = \{\llbracket t \rrbracket \mid t \in \text{dom}(p)\}$. Then the ideal \bar{I}_p is exactly given by $\bar{I}_p = \mathcal{I}(V_p)$. Assume that $r' \in \bar{I}_p$ and $\rho(p, f) = (p_1, \dots, p_m)$ holds. Then for all tuples of trees $(\mathbf{t}_1, \dots, \mathbf{t}_m)$ with $\mathbf{t}_i \in \text{dom}(p_i)$ ($i = 1, \dots, m$), $r'(\llbracket f(\mathbf{t}_1, \dots, \mathbf{t}_m) \rrbracket) = 0$ holds. Therefore,

$$r'[\mathbf{r}^{(f)}/\mathbf{z}](\mathbf{v}_1, \dots, \mathbf{v}_m) = 0$$

holds for all $(\mathbf{v}_1, \dots, \mathbf{v}_m) \in V_{p_1} \times \dots \times V_{p_m}$. Therefore,

$$\begin{aligned} r'[\mathbf{r}^{(f)}/\mathbf{z}] &\in \mathcal{I}(V_{p_1} \times \dots \times V_{p_m}) \\ &= \langle \mathcal{I}(V_{p_1})(\mathbf{x}_1) \cup \dots \cup \mathcal{I}(V_{p_m})(\mathbf{x}_m) \rangle \quad \text{by (7)} \\ &= \langle \bar{I}(\mathbf{x}_1) \cup \dots \cup \bar{I}(\mathbf{x}_m) \rangle. \end{aligned}$$

As a consequence, \bar{I} satisfies the constraints (6) and therefore is an inductive invariant of M relative to A . ■

The inductive invariant \bar{I} is the *largest* invariant and, accordingly, the *greatest* fixpoint of the inclusions (6). Since the set of polynomial ideals in $\mathbb{Q}[\mathbf{x}]$ has unbounded *decreasing* chains, it is unclear whether \bar{I} can be effectively computed.

Let us first consider the case where the input alphabet of M (and thus also of A) is monadic. Then the variables from \mathbf{z} can be reused for the copy \mathbf{x}_1 of variables for the first (and only) argument of $f \in \Sigma^{(1)}$, implying that every polynomial $\mathbf{r}_{qk}^{(f)}$ can be considered as a constant or a polynomial again over the variables \mathbf{z} . Thus, the constraints in (6) to be satisfied by an inductive invariant I , can be simplified to:

$$I_p \subseteq \{r' \in \mathbb{Q}[\mathbf{z}] \mid r'(\mathbf{r}^{(b)}) = 0\} \quad \text{if } \rho(p, b) = () \quad (8)$$

$$I_p \subseteq \{r' \in \mathbb{Q}[\mathbf{z}] \mid r'[\mathbf{r}^{(f)}/\mathbf{z}] \in I_{p_1}\} \quad \text{if } \rho(p, f) = p_1 \quad (9)$$

According to the second constraint, the demand for an equality $r' \doteq 0$ to hold at p is transformed by the monadic input symbol f into the demand for the equality $r'[\mathbf{r}^{(f)}/\mathbf{z}] \doteq 0$ to hold at p_1 . The propagation of these demands generated for the equality $H \doteq 0$ to hold at p_0 can be expressed by the following system of constraints:

$$\begin{aligned} I_{p_0} &\supseteq \langle H \rangle \\ I_{p_1} &\supseteq \{r'[\mathbf{r}^{(f)}/\mathbf{z}] \mid r' \in I_p\} \quad \text{if } \rho(p, f) = p_1 \end{aligned} \quad (10)$$

Recall that Hilbert's basis theorem implies that each ideal $J \subseteq \mathbb{Q}[\mathbf{z}]$ can be represented by a finite set of polynomials s_1, \dots, s_u so that $J = \langle s_1, \dots, s_u \rangle$ and likewise, that each *increasing* chain $J_0 \subseteq J_1 \subseteq \dots$ of ideals is ultimately stable. Therefore, the system (10) has a *least solution*, which is attained after finitely many fixpoint iterations. We claim:

Lemma 12: Assume that I is the least solution of the system of constraints (10). Then I is an inductive invariant iff for each transition $\rho(p, b) = ()$ of A , $r'(\mathbf{r}^{(b)}) = 0$ for all $r' \in I_p$. In this case, it is the *least* inductive invariant I' with $H \in I'_{p_0}$. Otherwise, no inductive invariant with this property exists.

Proof: We have that I is a solution of (10) iff I satisfies the constraints (9). Moreover, $r'(\mathbf{r}^{(b)}) = 0$ for all $r' \in I_p$ holds for all $\rho(p, b) = ()$ of A iff I satisfies the constraints (8). Therefore, I is an inductive invariant with $H \in I_{p_0}$, iff these two assumptions are met.

Now assume that I is the least solution of (10). If it passes all tests on the transitions $\rho(p, b)$, it therefore must be the least inductive invariant I' with $H \in I'_{p_0}$. If it does not pass all tests, then there cannot be any inductive invariant I' with $H \in I'_{p_0}$. This can be seen as follows. Assume for a contradiction that there is an inductive invariant I' with $H \in I'_{p_0}$. Since I' satisfies the constraints in (9), I' is also a solution of (10). Therefore, $I_p \subseteq I'_p$ for all states p of A . Now since already I does not pass all tests on transitions $\rho(p, b) = ()$, then I' cannot pass all these tests either. But then I' does not satisfy the constraints (9) and therefore fails to be an inductive invariant — contradiction. ■

Since the least solution of system (10) can effectively be computed and the tests required by Lemma 12 can also be effectively performed, we obtain:

Theorem 13: For a total unary y DMT transducer M with a monadic input alphabet, it is decidable whether or not two states are equivalent relative to a DTTA automaton A .

Proof: Let H denote the polynomial $\mathbf{z}_{j_0 0} - \mathbf{z}_{j'_0 0}$. Then $H(\llbracket t' \rrbracket) = 0$ for all $t' \in \text{dom}(p_0)$ holds iff $H \in \bar{I}_{p_0}$. Now, $H \in I_{p_0}$ for some inductive invariant I iff $H \in I'_{p_0}$ for the least inductive invariant I' which, by Lemma 12 can be effectively computed. Since membership of a polynomial in an ideal can be effectively decided, the claim of the theorem follows. ■

In the following, we finally drop also the assumption that the y DMT transducer has a monadic input alphabet. What we keep is the assumption that the output alphabet is unary. For this case, we prove that equivalence is still decidable. An indicator for the extra complication due to non-monadic input symbols is that we are only able to provide two *semi*-algorithms, one which provides a proof of equivalence if equivalence holds, and another which provides an input tree for which the output differ — whenever non-equivalence holds.

In case of non-monadic input symbols, it is no longer clear whether computing the greatest solution of constraint system (6) can be replaced by computing the least solution over some suitably defined alternative constraint system over ideals. What we still know is that every ideal of $\mathbb{Q}[\mathbf{z}]$ can be represented by a finite set of polynomials with coefficients, which can be chosen from \mathbb{Z} . Since the validity of the inclusions (6) can be effectively decided for any given candidate invariant I , the set of *all* inductive invariants of M relative to A is recursively enumerable. Accordingly, if $\mathbf{z}_{j_0 0} - \mathbf{z}_{j'_0 0} = 0$ holds for all vectors $\llbracket t \rrbracket, t \in \text{dom}(p_0)$, an inductive invariant certifying this fact, will eventually be found in the enumeration. In this way, we obtain a *semi*-decision procedure for equivalence of the states j_0, j'_0 of M relative to A . The fact, on the other hand, that $\mathbf{z}_{j_0 0} - \mathbf{z}_{j'_0 0} = 0$ does not hold for all $\llbracket t \rrbracket, t \in \text{dom}(p_0)$, is witnessed by a specific tree $t \in \text{dom}(p_0)$ for which $\llbracket t \rrbracket_{j_0 0} - \llbracket t \rrbracket_{j'_0 0} \neq 0$. Since $\text{dom}(p_0)$ is recursively enumerable as well, we obtain another *semi*-decision procedure, now for non-equivalence of states j_0, j'_0 of M relative to A . Putting these two *semi*-decision procedures together, we obtain:

Theorem 14: Assume that M is a total y DMT transducer with unary output alphabet, A is a DTTA automaton. Then it is decidable whether or not two states j_0, j'_0 of M are equivalent relative to A .

In the same way as in the last section, Theorem 14 provides us with a decision procedure for possibly partial unary y DMT transducers. We obtain our main technical result:

Theorem 15: Equivalence for (possibly partial) unary y DMT transducers is decidable.

VI. A MORE PRACTICAL ALGORITHM

Clearly, checking all input trees is perhaps not the most systematic way of identifying a counter example to equivalence. Likewise, enumerating all mappings $p \mapsto \mathcal{I}_p$, in quest for a sufficiently strong inductive invariant seems completely impossible to be turned into a practical algorithm. Therefore, in this section we provide more realistic implementations of the two semi-algorithms to decide equivalence.

To accomplish the task of identifying counter examples, we take a closer look at the greatest fixpoint iteration to determine the greatest inductive invariant $p \mapsto \bar{\mathcal{I}}_p$. For $p \in P$, let $\bar{\mathcal{I}}_p^{(0)} = \langle 1 \rangle_{\mathbb{Q}[\mathbf{z}]} = \mathbb{Q}[\mathbf{z}]$, i.e., the full polynomial ring, and for $d > 0$, define $\bar{\mathcal{I}}_p^{(d)}$ as

$$\bar{\mathcal{I}}_p^{(d)} = \bigcap \{ \llbracket f \rrbracket^\#(\bar{\mathcal{I}}_{p_1}^{(d-1)}, \dots, \bar{\mathcal{I}}_{p_k}^{(d-1)}) \mid \rho(p, f) = p_1 \dots p_k \} \quad (11)$$

$$\begin{aligned} \llbracket f \rrbracket^\#(I_1, \dots, I_k) &= \{ r \in \mathbb{Q}[\mathbf{z}] \mid r[r^{(f)}/\mathbf{z}] \in \langle I_1(\mathbf{x}_1) \cup \dots \cup I_k(\mathbf{x}_k) \rangle_{\mathbb{Q}[\mathbf{z}]} \} \\ &= (\langle \mathbf{z}_{qi} - r_{qi}^{(f)} \mid q = 1, \dots, n, i = 0, \dots, l \rangle_{\mathbb{Q}[\mathbf{z}]} \oplus \\ &\quad \langle I_1(\mathbf{x}_1) \cup \dots \cup I_k(\mathbf{x}_k) \rangle_{\mathbb{Q}[\mathbf{z}]}) \cap \mathbb{Q}[\mathbf{z}] \end{aligned} \quad (12)$$

For every $p \in P$ and $d \geq 0$, let $\text{dom}_d(p)$ denote the set of all input trees $t \in \text{dom}(p)$ of depth at most d where we consider leaves to have depth 1. Then we have:

- Lemma 16:* 1) For every $d \geq 0$, $\bar{\mathcal{I}}_p^{(d)} = \{ r \in \mathbb{Q}[\mathbf{z}] \mid \forall t \in \text{dom}_d(p). r(\llbracket t \rrbracket) = 0 \}$;
2) For every $p \in P$, $\bar{\mathcal{I}}_p = \bigcap \{ \bar{\mathcal{I}}_p^{(d)} \mid d \geq 0 \}$.

Proof: Statement 1 follows by induction on d along the same lines as the proofs of Theorems 9 and 10. Statement 2 follows from statement 1 as the intersection to the right consists of all polynomials $r \in \mathbb{Q}[\mathbf{z}]$ so that $r(\llbracket t \rrbracket) = 0$ for all $t \in \text{dom}(p)$ — which precisely is the definition of $\bar{\mathcal{I}}_p$. ■

From statement 1 of Lemma 16, we conclude that there is a counter example to equivalence of q, q' of depth d iff $H_{q, q'} \notin \bar{\mathcal{I}}_{p_0}^{(d)}$ for the initial state p_0 of the DTTA automaton B . Thus, the semi-algorithm for falsifying equivalence, can be formulated as:

```

for ( $d \geq 0$ ) {
    determine  $p \mapsto \bar{\mathcal{I}}_p^{(d)}$ ;
    if ( $H_{q, q'} \notin \bar{\mathcal{I}}_{p_0}^{(d)}$ ) return “not equivalent”;
}

```

We turn to the efficient enumeration of candidate invariants. Let us again fix some bound d . This time, the bound d is used as the degree bound to the polynomials to be considered during the fixpoint iteration. Let $\mathbb{Q}_d[\mathbf{z}]$ denote the set of all polynomials in $\mathbb{Q}[\mathbf{z}]$ of total degree at most d . Here, the total degree of a polynomial r is the maximal sum of exponents of a monomial occurring in r . For an ideal $\mathcal{I} \subseteq \mathbb{Q}[\mathbf{z}]$, let us denote \mathcal{I}_d as the intersection $\mathcal{I} \cap \mathbb{Q}_d[\mathbf{z}]$. This set of polynomials can be considered as a vector space of finite dimension and can also be considered as a *pseudo ideal* in the sense of Colon [53].

The intersection \mathcal{I}_d can be effectively computed as follows. Given a Gröbner basis G for \mathcal{I} relative to some graded lexicographical ordering of monomials, it suffices to extract the subset G_d of polynomials in G of total degrees bounded by d . Then a set of generators of \mathcal{I}_d considered as a vector space is given by the set of polynomials $g \cdot m$ with $g \in G_d$ for monomials m with $\text{def}(m) + \deg(g) \leq d$. Moreover, the ideal $\langle G_d \rangle_{\mathbb{Q}[\mathbf{z}]}$ is the smallest ideal $\mathcal{I}' \subseteq \mathcal{I}$ so that $\mathcal{I}' \cap \mathbb{Q}_d[\mathbf{z}] = \mathcal{I} \cap \mathbb{Q}_d[\mathbf{z}]$.

Let α_d denote the function that maps each ideal \mathcal{I} to the corresponding ideal $\langle G_d \rangle_{\mathbb{Q}[\mathbf{z}]}$. Let \mathbb{D}_d denote the set of all ideals generated from Gröbner bases of total degree at most d . In this complete lattice, all decreasing sequences are ultimately stable. The idea is to compute increasingly precise abstractions of the mapping $p \mapsto \bar{\mathcal{I}}_p$ by means of the complete lattices \mathbb{D}_d . For $d \geq 1$, we put up the constraint system over \mathbb{D}_d , consisting of all constraints

$$I_p \subseteq \alpha_d(\llbracket f \rrbracket^\#(I_{p_1}, \dots, I_{p_k})) \quad (13)$$

for every transition $\rho(p, f) = p_1 \dots p_k$. Since all right-hand sides are monotonic, the greatest solution of this system exists. Since \mathbb{D}_d has finite descending chains only, the greatest solution is attained after finitely many fixpoint iterations. Let $p \mapsto \mathcal{I}_{p,d}$ denote the resulting greatest solution. We have:

- Lemma 17:* 1) For all $d \geq 1$, $p \mapsto \mathcal{I}_{p,d}$ is an inductive invariant;
 2) For all $d \geq 1$, $\mathcal{I}_{p,d} \subseteq \mathcal{I}_{p,d+1}$ holds for every $p \in P$;
 3) There exists some $d \geq 1$ such that $\mathcal{I}_{p,d} = \bar{\mathcal{I}}_p$ for all $p \in P$.

Proof: The first two statements are obvious. In order to prove the third statement, consider the greatest inductive invariant $p \mapsto \bar{\mathcal{I}}_p$. Thus in particular, $\mathcal{I}_{p,d} \subseteq \bar{\mathcal{I}}_p$ for all p and d . For every state p , let G_p denote the Gröbner base of $\bar{\mathcal{I}}_p$. Let \bar{d} denote the maximal global degree of any polynomial in the set $\bigcup \{G_p \mid p \in P\}$. Then $p \mapsto \bar{\mathcal{I}}_p$ is a solution of the constraint system (13). Accordingly, $\bar{\mathcal{I}}_p \subseteq \mathcal{I}_{p,\bar{d}}$. Hence $\bar{\mathcal{I}}_p = \mathcal{I}_{p,\bar{d}}$, and statement 3 follows. \blacksquare

In light of the argument for proving statement 3 of Lemma 17, we observe that for any d , $p \mapsto \mathcal{I}_{p,d}$ represents the largest inductive invariant which can be represented by a Gröbner basis with maximal total degree d . Given Lemma 17, the semi-algorithm for verifying equivalence therefore looks as follows:

```

for ( $d \geq 1$ ) {
    determine  $p \mapsto \mathcal{I}_{p,d}$ ;
    if ( $H_{q,q'} \in \mathcal{I}_{p_0,d}$ ) return “equivalent”;
}

```

This algorithm now provides a systematic way to generate inductive invariants of increasing precision thus complementing the systematic enumeration method of counter examples of increasing depths.

VII. FROM YDT TRANSDUCERS TO UNARY YDMTT TRANSDUCERS

In the following, we show that every total y DT transducer can be simulated by a total y DMT transducer with a unary output alphabet and polynomial size. This is the content of the next lemma. Assume that the output alphabet is given by $\Delta = [s]$. By considering the elements of Δ as non-zero digits of the number system with base $s + 1$, each element $w \in \Delta^*$ can be uniquely represented by a natural number. If $w = w_1 \dots w_k$, $w_j \in [s]$, this number is given by $[w]_{s+1} = \sum_{j=1}^k w_j \cdot (s+1)^j$. In particular, $[\epsilon]_{s+1} = 0$, i.e., the empty string is represented by 0. We have:

Lemma 18: Assume that M is a total y DT transducer with set $[n]$ of states and output alphabet $[s]$. Then a unary y DMT transducer N with the same set of states and a single parameter, can be constructed in polynomial time so that for every state $q \in [n]$ and input tree t , $\llbracket q \rrbracket_N(t)(\epsilon) = \llbracket q \rrbracket_M(t)_{s+1}$.

Moreover, if M is linear, then N is non-self-nested.

Proof: Let $M = (Q, \Sigma, \Delta, q_0, R)$ where $Q = [n]$. We define $N = (Q, \Sigma, \{d\}, q_0, R')$ as follows. For every rule $q(f(x_1, \dots, x_k)) \rightarrow T$ in R we let the rule $q(f(x_1, \dots, x_k), y_1) \rightarrow \mathcal{U}[T]$ be in R' . The parameter y_1 is meant to contain the right context (in unary). The mapping $\mathcal{U}[T]$ is defined as follows:

$$\begin{aligned} \mathcal{U}[aT] &= a + (s+1) \cdot \mathcal{U}[T] \\ \mathcal{U}[q'(x_i)T] &= q'(x_i, \mathcal{U}[T]) \\ \mathcal{U}[\epsilon] &= y_1. \end{aligned}$$

For the y DMT transducer N we prove the following invariant:

$$\llbracket q \rrbracket_N(t)([w]_{s+1}) = \llbracket q \rrbracket_M(t) w]_{s+1}.$$

From that, the statement follows by choosing $q = q_0$ and $w = \epsilon$. In order to prove the invariant, we proceed by induction on the structure of t . So assume that $t = f(\mathbf{t}_1, \dots, \mathbf{t}_m)$, $m \geq 0$, where $\delta(q, f) = T$. By induction, we may assume that the invariant already holds for $\mathbf{t}_1, \dots, \mathbf{t}_k$ and all output words w . Then we prove that for all subsequences T' of T and all words $w \in [s]^*$ the following invariant holds:

$$\llbracket \mathcal{U}[T'] \rrbracket_N \mathbf{t} ([w]_{s+1}) = \llbracket T' \rrbracket_M \mathbf{t} w]_{s+1}$$

where $\mathbf{t} = (\mathbf{t}_1, \dots, \mathbf{t}_m)$. The invariant for t follows because $\llbracket q \rrbracket_N(t)(y_1) = \llbracket \mathcal{U}[T] \rrbracket_N \mathbf{t} (y_1)$ and $\llbracket q \rrbracket_M(t) = \llbracket T \rrbracket_M \mathbf{t}$. If $T' = \epsilon$, we have that

$$\llbracket \epsilon \rrbracket_M \mathbf{t} w]_{s+1} = [w]_{s+1} = \llbracket \mathcal{U}[\epsilon] \rrbracket_N \mathbf{t} ([w]_{s+1})$$

and the invariant holds.

If $T' = aT''$ for some output symbol $a \in [s]$, we have that

$$\begin{aligned} \llbracket aT'' \rrbracket_M \mathbf{t} w]_{s+1} &= [a \llbracket T'' \rrbracket_M \mathbf{t} w]_{s+1} \\ &= a + (s+1) \cdot \llbracket T'' \rrbracket_M \mathbf{t} w]_{s+1} \\ &= a + (s+1) \cdot \llbracket \mathcal{U}[T''] \rrbracket_N \mathbf{t} ([w]_{s+1}) \quad \text{by induction for } T'' \\ &= \llbracket \mathcal{U}[aT''] \rrbracket_N \mathbf{t} ([w]_{s+1}) \end{aligned}$$

and the invariant holds, by induction, for T'' .

Therefore, it remains to consider the case where $T' = q'(x_i)T''$. Then:

$$\begin{aligned} \llbracket q'(x_i)T'' \rrbracket_M \mathbf{t} w]_{s+1} &= \llbracket q' \rrbracket_M(\mathbf{t}_i) \llbracket T'' \rrbracket_M \mathbf{t} w]_{s+1} \\ &= \llbracket q' \rrbracket_N(\mathbf{t}_i)(\llbracket T'' \rrbracket_M \mathbf{t} w]_{s+1}) \quad \text{by induction for } \mathbf{t}_i \text{ and } w' = \llbracket T'' \rrbracket_M \mathbf{t} w \\ &= \llbracket q' \rrbracket_N(\mathbf{t}_i)(\llbracket \mathcal{U}[T''] \rrbracket_N \mathbf{t} ([w]_{s+1})) \quad \text{by induction for } T'' \\ &= \llbracket \mathcal{U}[q'(x_i)T''] \rrbracket_N \mathbf{t} ([w]_{s+1}) \end{aligned}$$

and the assertion follows. — Obviously, if M is linear then N is non-self-nested. ■

VIII. OUTPUT IN THE FREE GROUP

In some applications, the output produced by a transducer cannot be considered as a sequence of uninterpreted letters, but consists in a sequence of operators. A first step in direction of dealing with such interpreted output is to consider the output to be an element of a *free group*. In that, we go beyond the free monoid as output domain and assume that for each output letter a , there is an *inverse* letter a^- together with the cancellation rules

$$aa^- = a^-a = \epsilon.$$

And we ask whether equivalence of transducers remains decidable if output strings are considered equivalent up to applications of these rewrite rules.

Let us first consider output generated in the free group \mathcal{F}_1 with a single generator. The free group \mathcal{F}_1 is isomorphic to the integral ring \mathbb{Z} . This means that our construction for y DMT transducers with unary output alphabet can be readily applied also to y DMT transducers with output in \mathcal{F}_1 . We obtain:

Theorem 19: Assume that M is a total y DMT transducer with output in the free group \mathcal{F}_1 , and A is a DTTA automaton. Then it is decidable whether or not two states j_0, j'_0 of M are equivalent relative to A . If M is non-self-nested, in-equivalence can even be decided in randomized polynomial time. \blacksquare

We now consider the case when we have two distinct output symbols a, b and together with their respective inverses a^- and b^- , i.e., we consider outputs in the free group \mathcal{F}_2 with two generators. We do not know how the result for \mathcal{F}_1 and y DMT transducers can be generalized to this more general situation. What we can do, however, is to consider y DT transducers where output symbols are interpreted as $l \times l$ matrices with entries in \mathbb{Q} . Let $\mathcal{M}_l(\mathbb{Q})$ denote the monoid of all such matrices where the monoid operation is matrix multiplication. Given an interpretation $\alpha : \Delta \rightarrow \mathcal{M}_l(\mathbb{Q})$ of output symbols, every output string w then represents a matrix $\alpha(w) \in \mathcal{M}_l(\mathbb{Q})$. Accordingly, the semantics $\llbracket t \rrbracket$ of an input tree t turns into a vector of matrices from $\mathcal{M}_l(\mathbb{Q})$, i.e., $\llbracket t \rrbracket \in \mathbb{Q}^{n \cdot (l \times l)}$. Moreover, every input symbol f of arity $k \geq 0$ corresponds to a mapping:

$$\llbracket f \rrbracket : \mathbb{Q}^{n \cdot (l \times l)} \times \dots \times \mathbb{Q}^{n \cdot (l \times l)} \rightarrow \mathbb{Q}^{n \cdot (l \times l)}$$

(n the number of states of the transducer) which transforms the vectors corresponding to the semantics of the argument trees into the vector corresponding to the result vector of matrices for the whole input tree. By induction on the structure of right-hand sides, we find that the entry $(\llbracket f \rrbracket(\mathbf{x}_1, \dots, \mathbf{x}_k))_{q\lambda\mu} =: r_{q\lambda\mu}^{(f)}$ is a polynomial in the variables $\mathbf{x}_{iq'\lambda'\mu'}$, $i \in \{1, \dots, k\}$, $q' \in \{1, \dots, n\}$, $\lambda', \mu' \in \{1, \dots, l\}$. In the particular case that the y DT transducer is linear, each $r_{q\lambda\mu}^{(f)}$ is multi-affine in the vectors $\mathbf{x}_1, \dots, \mathbf{x}_k$.

Example 4: Consider the two matrices

$$m_1 = \begin{bmatrix} 3 & 1 \\ 0 & 1 \end{bmatrix} \quad m_2 = \begin{bmatrix} 3 & 2 \\ 0 & 1 \end{bmatrix}$$

Let $\Delta = \{a_1, a_2\}$. The monoid homomorphism $\alpha : \Delta^* \rightarrow \mathcal{M}_2(\mathbb{Q})$ mapping a_i to m_i is injective and given by

$$\alpha(a_{j_1} \dots a_{j_s}) = \begin{bmatrix} 3^s & w \\ 0 & 1 \end{bmatrix} \quad \text{where} \quad w = \sum_{\lambda=0}^{s-1} 3^\lambda \cdot j_\lambda$$

In this way, the free monoid of strings over the alphabet Δ can be represented by the sub-monoid of $\mathcal{M}_2(\mathbb{Q})$ generated by m_1, m_2 . Consider, e.g., a transition

$$q(f(x_1, x_2)) \longrightarrow a_1 q'(x_2) a_2 q''(x_1)$$

of some deterministic y DT transducer with output in $\mathcal{M}_2(\mathbb{Q})$ according to the given α . Then according to our definition,

$$\begin{aligned} (\llbracket f \rrbracket(\mathbf{x}_1, \mathbf{x}_2))_q &= m_1 \cdot \mathbf{x}_{2q'} \cdot m_2 \cdot \mathbf{x}_{1q''} \\ &= \begin{bmatrix} 3 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x}_{2q'11} & \mathbf{x}_{2q'12} \\ \mathbf{x}_{2q'21} & \mathbf{x}_{2q'22} \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x}_{1q''11} & \mathbf{x}_{1q''12} \\ \mathbf{x}_{1q''21} & \mathbf{x}_{1q''22} \end{bmatrix} \end{aligned}$$

Let $\mathcal{H}_{j_0 j'_0}$ denote the set of polynomials $\mathbf{z}_{j_0 \lambda \mu} - \mathbf{z}_{j'_0 \lambda \mu}$, $\lambda, \mu \in \{1, \dots, l\}$. Then the states j_0, j'_0 are equivalent relative to the DTTA automaton A iff $r(\llbracket t \rrbracket) = 0$ for all $r \in \mathcal{H}_{j_0 j'_0}$ and $t \in \text{dom}(p_0)$. In general, this can be decided by the algorithm which we applied in the proof of Theorem 14, but adapted to the new construction of the polynomials $r_{q\lambda\mu}^{(f)}$ and where the single linear target equality $H_{j_0 j'_0}$ is replaced with the conjunction of the finitely many target equalities from $\mathcal{H}_{j_0 j'_0}$. In the case of linear y DT transducers, the algorithm from Section IV can be adapted accordingly. Therefore, we obtain:

Theorem 20: Assume that M is a total y DT transducer with output in the monoid $\mathcal{M}_l(\mathbb{Q})$, and A is a DTTA automaton. Then it is decidable whether or not two states j_0, j'_0 of M are equivalent relative to A . If the y DT transducer is linear and the ranks of the input symbols is bounded, in-equivalence is decidable in randomized polynomial time. ■

We remark that extending the matrix monoid considered in Example 4 with inverses, would not provide us with a free group. Instead, however, we can consider the subgroup of 2×2 matrices generated from the elements:

$$a = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$$

This group is also known as *Sanov* group \mathcal{S} (see, e.g., Example 4.5.1 of [54]). Since both matrices have determinant 1, all elements in \mathcal{S} have integer coefficients only. In particular, the inverses of the two generators are given by:

$$a^- = \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix} \quad b^- = \begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix}$$

The Sanov group is particularly useful for us, since \mathcal{S} with subgroup generators a and b is isomorphic to the free group \mathcal{F}_2 freely generated from a, b . Thus, our results on y DT transducers with output in matrix monoids in Theorem 20 gives us:

Theorem 21: Assume that M is a total y DT transducer with output in the free group \mathcal{F}_2 , and A is a DTTA automaton. Then it is decidable whether or not two states j_0, j'_0 of M are equivalent relative to A . If the y DT transducers are linear and the ranks of their input symbols are bounded, in-equivalence is decidable in randomized polynomial time. ■

Since the free group with two generators has a free group with $l \geq 2$ generators as a subgroup, Theorem 21 implies that equivalence of total y DT transducers relative to some DTTA automaton is also decidable when the outputs are in a free group with $l \geq 2$ generators.

Lemma 18 allows to apply our decision procedures for unary y DMT transducers to decide equivalence for y DT transducers with arbitrary output alphabets. Via Lemma 18, equivalence for linear (possibly partial) y DT transducers is reduced to equivalence of non-self-nested unary y DMT transducers, while equivalence for arbitrary (possibly partial and non-linear) y DT transducers is reduced to equivalence of general unary y DMT transducers. In summary, we obtain:

Theorem 22: Equivalence of arbitrary y DT transducers is decidable. If the y DT transducers are linear and the ranks of their input symbols are bounded, in-equivalence is decidable in randomized polynomial time.

The second part of Theorem 22 follows from Theorem 7 and Lemma 1. One particular subcase of Theorem 22 is when the input alphabet is monadic. This case is known to be equivalent to the sequence equivalence problem of HDT0L systems [39], [38]. By Lemma 18, this problem can be reduced to the equivalence problem for unary y DMT transducers with monadic output alphabet, for which a *direct* algorithm based on fixpoint iteration over polynomial ideals has been presented in the last section. The equivalence problem for y DT transducers with non-monadic input alphabets, as shown to be decidable in Theorem 22, seems to be significantly more difficult.

IX. APPLICATION TO OTHER TYPES OF TRANSDUCERS

Tree transducers can be equipped with regular look-ahead. For top-down transducers this increases the expressive power. A top-down or macro tree-to-string transducer *with regular look-ahead* (y DT^R and y DMT^R transducer) consists of an ordinary such transducer together with a complete deterministic bottom-up tree automaton B . For a y DT^R transducer, a rule is of the form

$$q(f(x_1 : p_1, \dots, x_m : p_m)) \rightarrow T$$

where T is as for ordinary y DT transducers, and the p_i are states of B . The rule is applicable to an input tree $f \mathbf{t}$ if B arrives in state p_i on input tree \mathbf{t}_i for every $i \in \{1, \dots, m\}$. Our result extends to look-ahead, using the

technique as in [25]: one changes the input ranked alphabet to contain state information of B , and changes the transducer to check the correctness of the information.

By a result of [46] the class of translations realized by yDT^R transducers is equal to the class realized by macro transducers which use each parameter in a rule precisely once (and have look-ahead). In fact, by the results of [45] we can state the result in terms of $yDMT^R$ transducers that are *finite-copying in the parameters* ($yDMT_{fcp}^R$ transducers). A $yDMT^R$ transducer is finite-copying in the parameters if there exists a k such that for every input tree s , state q (of rank $l + 1$), and $j \in [l]$, the number of occurrences of y_j in $\llbracket q \rrbracket(s)$ is $\leq k$.

Corollary 23: Equivalence of yDT^R transducers and $yDMT_{fcp}^R$ transducers is decidable.

A variation of transducers that has been considered in the context of XML, are transducers of *unranked trees*. In an unranked tree, the number of children of a node is not determined by the label of that node, but is independent. For instance the term $a(a(a, b), a, a, a)$ represents an unranked tree. XMLdocument trees are naturally modeled by unranked trees.

There are several models of top-down tree transducers for unranked trees. In [28] macro forest transducers, and their parameterless version, the *forest transducers*, are defined.

The rules of a forest transducer are very similar to the rules of a yDT transducer and are of the form $q(a(x_1, x_2)) \rightarrow T$ where T is a string as in a yDT transducer, with the only difference that T may contain special symbols “(” and “)” of opening and closing brackets, and if so, then T must be well-balanced with respect to these brackets. If such a rule is applied to an unranked a -labeled node u , then x_1 represents the first subtree of u , and x_2 represents the next sibling of u . The special bracket symbols in right-hand sides have the obvious interpretation of generating a tree structure. Obviously, when checking equivalence of two forest transducers, we may consider their output as *strings*. Thus, the equivalence problem for deterministic forest transducers is just a direct instance of the equivalence problem for yDT transducers.

Corollary 24: Equivalence is decidable for deterministic forest transducers.

Another, much earlier model of unranked top-down tree transducers is the *unranked top-down tree transducer* [9]. Instead of state calls $q(x_i)$ as in an ordinary ranked top-down tree transducer, they use calls of the form L where L is a regular language over the set of states Q of the transducers, plus the special symbol 0. If the current input node has k -many children, then a word of length k from L is chosen in order to determine which states translate the children nodes (where 0 means that no state translates the corresponding node). Such a transducer is deterministic, if for every k and every L in the right-hand side of a rule, L contains at most one string of length k . As an example, consider the unranked top-down tree transducer with the following rules:

$$\begin{aligned} q_0(a(\cdots)) &\rightarrow a(L) \\ q(a(\cdots)) &\rightarrow a(L)L \end{aligned}$$

where L is the regular language q^* consting of all strings of the form $qq \cdots q$. For the input tree $s = a(a(a(a)))$, this transducer first applies the first rule to obtain $a(q(s_1))$ where $s_1 = a(a(a))$. We now apply the second rule to obtain $a(a(q(s_2))q(s_2))$ where $s_2 = a(a)$. Two more applications of the second rule give $a(a(a(q(a))q(a))a(q(a))q(a))$, and finally we obtain the output tree $a(a(a(a)a)a(a)a)$.

As mentioned by Perst and Seidl, any unranked top-down tree transducer can be realized by a forest transducer. However, they only mention this for nondeterministic transducers. Thus, given a deterministic unranked top-down tree transducer M , we can construct an equivalent nondeterministic forest transducer N . It follows from the explanation above that we may consider N as a nondeterministic top-down tree-to-string transducer.

By an old result [55], for any functional DT transducer, an equivalent DT^R transducer can be constructed. Since “yield” which turns a tree into its string of leaf symbols is a function, it directly follows that also for any functional yDT transducer, one can construct an equivalent yDT^R transducer. Thus, we can construct for N an equivalent yDT^R transducer, for which equivalence is decidable by Corollary 23.

Corollary 25: Equivalence is decidable for deterministic unranked top-down tree transducers.

Moreover by the constructions from section VIII, the results stated in in corollaries 23,24 and 25 also hold when output is not considered in the free monoid, but in the free group.

X. CONCLUSION

We present algorithms for deciding equivalence of deterministic top-down tree-to-string transducers (y DT transducers). For y DT transducers with general output alphabets, we provide a construction which encodes outputs over arbitrary output alphabets into outputs over a unary alphabet. This construction requires to introduce an extra parameter. For arbitrary y DT transducers, it results in y DMT transducers with unary output alphabet, which are non-self-nested whenever the original y DT transducer is linear. For the case of *non-self-nested* unary y DMT transducers, we show that multi-affine mappings and affine spaces are sufficient to decide equivalence, whereas in the general case, we had to resort to polynomial ideals.

The key concept which helped us to arrive at a decision procedure in the general case, are inductive invariants certifying assertions. While such invariants can be automatically inferred for monadic input alphabets, we were less explicit for non-monadic input alphabets. Here, we only prove that an inductive invariant certifying a polynomial equality *exists*, whenever the equality holds. Since enumerating all inductive invariants is rather impractical, we presented a more explicit method which allows to systematically construct the best inductive invariant up to a given maximal degree. Together with an explicit enumeration of potential counter-examples, decidability of equivalence of arbitrary y DMT transducers with unary output alphabet follows. This result means that *Abelian* equivalence, i.e., equivalence up to the ordering of symbols in the output, is decidable for general y DMT transducers. The same holds true for *growth* equivalence where only the lengths of output strings matter.

By means of our simulation of arbitrary output alphabets with unary ones, we obtained a randomized polynomial algorithm for deciding in-equivalence of linear y DT transducers. The strongest result, however, is decidability of equivalence of general y DT transducers with arbitrary output alphabets. Both algorithms are then extended to the case when output is no longer considered to be in a free monoid, but in a free group.

Still, our decision procedures leave room for generalizations. The equivalence problem for y DMT transducers with unary output alphabet being solved, the equivalence problems (as stated in [29]) for y DMT transducers with *arbitrary* output alphabets, and even for DMT transducers with tree output, remain open.

REFERENCES

- [1] P. Wadler, Deforestation: Transforming programs to eliminate trees, Theor. Comput. Sci. 73 (2) (1990) 231–248.
- [2] S. Marlow, P. Wadler, Deforestation for higher-order functions, in: Functional Programming, 1993, pp. 154–165.
- [3] J. Voigtländer, A. Kühnemann, Composition of functions with accumulating parameters, J. Funct. Program. 14 (3) (2004) 317–363.
- [4] J. Voigtländer, Tree transducer composition as program transformation, Ph.D. thesis, TU Dresden (2005).
- [5] K. Matsuda, K. Inaba, K. Nakano, Polynomial-time inverse computation for accumulative functions with multiple data traversals, in: PEPM, 2012, pp. 5–14.
- [6] Z. Fülöp, H. Vogler, Syntax-Directed Semantics; Formal Models Based on Tree Transducers, Springer-Verlag, 1998.
- [7] W3C, XSL Transformations (XSLT), available online <http://www.w3.org/TR/xslt> (16 Nov. 1999).
- [8] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, J. Siméon, XQuery 1.0: An XML query language (second edition), Tech. rep., W3C Recommendation (2010). URL <http://www.w3.org/TR/xquery/>
- [9] S. Maneth, F. Neven, Structured Document Transformations Based on XSL, in: DBPL, 1999, pp. 80–98.
- [10] J. Engelfriet, S. Maneth, A comparison of pebble tree transducers with macro tree transducers, Acta Inf. 39 (9) (2003) 613–698.

- [11] S. Maneth, A. Berlea, T. Perst, H. Seidl, XML type checking with macro tree transducers, in: PODS, 2005, pp. 283–294.
- [12] S. Maneth, T. Perst, H. Seidl, Exact XML type checking in polynomial time, in: ICDT, 2007, pp. 254–268.
- [13] S. Hakuta, S. Maneth, K. Nakano, H. Iwasaki, Xquery streaming by forest transducers, in: ICDE, 2014, pp. 952–963.
- [14] Y. Liu, Q. Liu, S. Lin, Tree-to-string alignment template for statistical machine translation, in: ACL, 2006, pp. 609–616.
- [15] Y. Liu, Y. Huang, Q. Liu, S. Lin, Forest-to-string statistical translation rules, in: Annual Meeting-Association for Computational Linguistics, Vol. 45, 2007, p. 704.
- [16] A. Maletti, J. Graehl, M. Hopkins, K. Knight, The power of extended top-down tree transducers, SIAM Journal on Computing 39 (2) (2009) 410–430.
- [17] F. Braune, N. Seemann, D. Quernheim, A. Maletti, Shallow local multi-bottom-up tree transducers in statistical machine translation, in: ACL, 2013, pp. 811–821.
- [18] R. Küsters, T. Wilke, Transducer-based analysis of cryptographic protocols, Inf. Comput. 205 (12) (2007) 1741–1776.
- [19] S. Maneth, Models of tree translation, Ph.D. thesis, University of Leiden (2003).
- [20] J. W. Thatcher, Generalized sequential machine maps, J. Comput. Syst. Sci. 4 (4) (1970) 339–367.
- [21] W. C. Rounds, Mappings and grammars on trees, Mathematical Systems Theory 4 (3) (1970) 257–287.
- [22] J. Engelfriet, H. Vogler, Macro Tree Transducers, Journal of Computer and System Sciences (JCSS) 31 (1985) 71–146.
- [23] T. V. Griffiths, The unsolvability of the equivalence problem for lambda-free nondeterministic generalized machines, J. ACM 15 (3) (1968) 409–413.
- [24] Z. Ésik, Decidability results concerning tree transducers I, Acta Cybernetica 5 (1980) 1–20.
- [25] J. Engelfriet, S. Maneth, H. Seidl, Deciding equivalence of top-down XML transformations in polynomial time, J. Comput. Syst. Sci. 75 (5) (2009) 271–286.
- [26] A. Lemay, S. Maneth, J. Niehren, A learning algorithm for top-down XML transformations, in: PODS, 2010, pp. 285–296.
- [27] S. H. Jensen, M. Madsen, A. Møller, Modeling the HTML DOM and browser API in static analysis of javascript web applications, in: SIGSOFT FSE, 2011, pp. 59–69.
- [28] T. Perst, H. Seidl, Macro forest transducers, Inf. Process. Lett. 89 (2004) 141–149.
- [29] J. Engelfriet, Some Open Questions and Recent Results on Tree Transducers and Tree Languages, in: R. Book (Ed.), Formal Language Theory; Perspectives and Open Problems, Academic Press, New York, 1980, pp. 241–286.
- [30] S. Staworko, G. Laurence, A. Lemay, J. Niehren, Equivalence of deterministic nested word to word transducers, in: FCT, 2009, pp. 310–322.
- [31] W. Plandowski, Testing equivalence of morphisms on context-free languages, in: ESA, 1994, pp. 460–470.
- [32] G. Laurence, A. Lemay, J. Niehren, S. Staworko, M. Tommasi, Normalization of sequential top-down tree-to-word transducers, in: LATA, 2011, pp. 354–365.

- [33] G. Laurence, A. Lemay, J. Niehren, S. Staworko, M. Tommasi, Learning sequential tree-to-word transducers, in: LATA, 2014, pp. 490–502.
- [34] J. Engelfriet, S. Maneth, The equivalence problem for deterministic MSO tree transducers is decidable, *Inform. Proc. Letters* 100 (2006) 206–212.
- [35] J. Engelfriet, S. Maneth, Macro tree translations of linear size increase are MSO definable, *SIAM J. Comput.* 32 (4) (2003) 950–1006.
- [36] K. Culik II, J. Karhumäki, A new proof for the DOL sequence equivalence problem and its implications, in: G. Rozenberg, A. Salomaa (Eds.), *The book of L*, Springer, Berlin, 1986, pp. 63–74.
- [37] K. Ruohonen, Equivalence problems for regular sets of word morphisms, in: G. Rozenberg, A. Salomaa (Eds.), *The book of L*, Springer, Berlin, 1986, pp. 393–401.
- [38] J. Honkala, A short solution for the HDTOL sequence equivalence problem, *Theor. Comput. Sci.* 244 (1-2) (2000) 267–270.
- [39] S. Maneth, Equivalence problems for tree transducers: A brief survey, in: AFL, 2014, pp. 74–93.
- [40] M. Müller-Olm, H. Seidl, A note on Karrs algorithm, in: ICALP, 2004, pp. 1016–1028.
- [41] T. Becker, V. Weispfenning, *Gröbner Bases. A Computational Approach to Commutative Algebra*. 2nd Edition, Graduate Texts in Mathematics, Springer Verlag, 1998.
- [42] A. Letichevsky, M. Lvov, Discovery of invariant equalities in programs over data fields, *Applicable Algebra in Engineering, Communication and Computing* 4 (4) (1993) 21–29.
- [43] M. Müller-Olm, H. Seidl, Computing polynomial program invariants, *Inf. Process. Lett.* 91 (5) (2004) 233–244.
- [44] A. Boiret, R. Palenta, Deciding equivalence of linear tree-to-word transducers in polynomial time, in: S. Brlek, C. Reutenauer (Eds.), *Developments in Language Theory - 20th International Conference, DLT 2016, LNCS 9840*, Springer, 2016, pp. 355–367.
- [45] J. Engelfriet, S. Maneth, Macro tree transducers, attribute grammars, and MSO definable tree translations, *Inf. Comput.* 154 (1) (1999) 34–91.
- [46] J. Engelfriet, S. Maneth, Output string languages of compositions of deterministic macro tree transducers, *J. Comput. Syst. Sci.* 64 (2) (2002) 350–395.
- [47] H. Seidl, Deciding equivalence of finite tree automata, *SIAM J. Comput.* 19 (3) (1990) 424–437.
- [48] H. Seidl, S. Maneth, G. Kemper, Equivalence of deterministic top-down tree-to-string transducers is decidable, in: FOCS, 2015, pp. 943–962.
- [49] K. R. Apt, G. D. Plotkin, Countable nondeterminism and random assignment, *J. ACM* 33 (4) (1986) 724–767.
- [50] C. Mathematics of Program Construction Group, Fixed-point calculus, *Inf. Process. Lett.* 53 (3) (1995) 131–136.
- [51] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*. Third Edition, The MIT Press Cambridge, Massachusetts, 2009.
- [52] G. H. Hardy, E. M. Wright, *An introduction to the theory of numbers*, sixth Edition, Oxford University Press, Oxford, 2008.

- [53] M. Colón, Polynomial approximations of the relational semantics of imperative programs, *Sci. Comput. Program.* 64 (1) (2007) 76–96.
- [54] C. Löh, Geometric group theory, an introduction, Tech. rep., Fakultät für Mathematik, Universität Regensburg, 93040 Regensburg Germany (2015).
URL <http://www.mathematik.uni-regensburg.de/loeh/>
- [55] J. Engelfriet, On tree transducers for partial functions, *Inf. Process. Lett.* 7 (4) (1978) 170–172.