# Equivalence of Deterministic Top-Down Tree-to-String Transducers Is Decidable

HELMUT SEIDL, Technical University of Munich
SEBASTIAN MANETH, Universität of Bremen
GREGOR KEMPER, Technical University of Munich

We prove 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: for linear transducers or total transducers with unary output alphabet (over a given top-down regular domain language), as well as for transducers with the single-use restriction. These results are obtained using techniques from multi-linear algebra. For our main result, we introduce polynomial transducers and prove that for these, validity of a polynomial invariant can be certified by means of an inductive invariant of polynomial ideals. This allows us to construct two semi-algorithms, one searching for a certificate of the invariant and one searching for a witness of its violation. Via a translation into polynomial transducers, we thus obtain that equivalence of general $y$DT transducers is decidable. In fact, our translation also shows that equivalence is decidable when the output is not in a free monoid but in a free group.

CCS Concepts: • **Theory of computation → Tree languages**; **Transducers**;

Additional Key Words and Phrases: Tree-to-string transducer, macro tree transducer, equivalence problem, decidability, polynomial ideals

**ACM Reference format:**
Helmut Seidl, Sebastian Maneth, and Gregor Kemper. 2018. Equivalence of Deterministic Top-Down Tree-to-string Transducers Is Decidable. *J. ACM* 65, 4, Article 21 (April 2018), 30 pages.
https://doi.org/10.1145/3182653



## 1 INTRODUCTION

Transformations of structured data are at the heart of functional programming (Wadler 1990; Marlow and Wadler 1993; Voigtländer and Kühnemann 2004; Voigtländer 2005; Matsuda et al. 2012) and also application areas such as compiling (Fülöp and Vogler 1998), document processing (W3C 1999; Boag et al. 2010; Maneth and Neven 1999; Engelfriet and Maneth 2003a; Maneth et al. 2005, 2007; Hakuta et al. 2014), automatic translation of natural languages (Liu et al. 2006, 2007; Maletti et al. 2009; Braune et al. 2013), or even cryptographic protocols (Küsters and Wilke 2007). The most fundamental model of such transformations is given by (finite-state tree) *transducers* (Maneth 2003; Fülöp and Vogler 1998). Transducers traverse the input by means of finitely many

Authors' addresses: H. Seidl and G. Kemper, Fakultät für Informatik, U München, Boltzmannstr. 3, 84857 Garching, Germany; emails: seidl@in.tum.de, kemper@ma.tum.de; S. Maneth, Department of Mathematics and Informatics, niversität Bremen, P.O. Box 3300440, 28334 Bremen, Germany; email: maneth@uni-bremen.de.

https://doi.org/10.1145/3182653

mutually recursive functions—corresponding to finitely many *states*. Accordingly, these transducers are also known as *top-down* tree transducers (Thatcher 1970; Rounds 1970). Here we only deal with deterministic transducers (equivalence is undecidable already for very restricted classes of non-deterministic transducers (Griffiths 1968)).

When the output is produced in a structured way, i.e., in the case of top-down tree-to-tree transducers (DT *transducers*, for short), many properties are fairly well understood. An algorithm for deciding equivalence of DT transducers already dates back to the 1980s (Ésik 1980). More recently, canonical forms have been provided, allowing for effective minimization (Engelfriet et al. 2009) as well as Gold-style learning of transformations from examples (Lemay et al. 2010). In various applications, though, the output is *not* generated in a structured way. This may be the case when general scripting languages are employed (Jensen et al. 2011), non-tree operations are required (Perst and Seidl 2004), or simply because the result is a string.

Assume, e.g., that we want to generate a well-formed XML document from an internal treelike 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:

$$
\begin{aligned}
q(\texttt{frame}(x_1, x_2)) &\to \langle\texttt{frame}\ q_1(x_1) q(x_2)\langle\texttt{/frame}\rangle \\
q_1(\texttt{end}) &\to \rangle \\
q_1(\texttt{defs}(x_1, x_2)) &\to q_2(x_1) q_1(x_2) \\
q_2(\texttt{height}(x_1)) &\to \texttt{␣height = "}q_3(x_1)\texttt{"} \\
q_2(\texttt{width}(x_1)) &\to \texttt{␣width = "}q_3(x_1)\texttt{"} \\
q(\texttt{button}(x_1)) &\to \langle\texttt{button}\rangle q_3(x_1)\langle\texttt{/button}\rangle \\
&\ldots
\end{aligned}
$$

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 `end`-labeled node that 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:

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

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 quite different ways.

Following the tradition since Engelfriet (1980), 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 (Maneth and Neven 1999; Perst and Seidl 2004), 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 Perst and Seidl (2004) 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 $e$, defined by:

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

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

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

Some thought reveals that the transducers $M$ and $M'$ are equivalent, although the output is generated in a rather "non-aligned" way with respect to the input variables $x_2$, $x_3$. 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 (Staworko et al. 2009). 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 every right-hand side. Equivalence for these can be decided by a reduction to Plandowski (1994) even in polynomial time. This class of transducers is sufficiently well behaved such that periodicity and commutation arguments over the output strings can be applied to derive canonical normal forms (Laurence et al. 2011) and enable Gold-style learning (Laurence et al. 2014). Even for arbitrary linear $y$DT transducers, it has been shown that equivalence is decidable in polynomial time (Boiret and Palenta 2016).

Apart from these results, which are based on normal forms, equivalence itself can indeed be decided for a class that is much larger than linear $y$DT translations, namely for tree-to-string translations definable in MSO logic (Engelfriet and Maneth 2006) or, equivalently, by macro tree-to-string translations of *linear size increase* (Engelfriet and Maneth 2003b). 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 a language $L$ 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 $m$. Note that the complexity of this procedure has not been considered.

Our main result generalizes the result of Engelfriet and Maneth (2006) by proving that equivalence of *unrestricted* deterministic top-down tree-to-string transducers is decidable. By that, it solves an intriguing problem that has been open for more than 35 years (Engelfriet 1980). 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 HDT0L sequence equivalence problem (Culik II and Karhumäki 1986; Ruohonen 1986; Honkala 2000), the latter is the sub-case when the input is restricted to monadic trees (Maneth 2015). Opposed to the attempts, e.g., in Staworko et al. (2009), 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 Engelfriet and Maneth (2006) based on semi-linear sets. Instead, we translate the free monoid of strings into a monoid of square matrices with entries in $\mathbb{Z}$. Recall that the *Sanov* subgroup of $2 \times 2$ matrices is isomorphic to the *free group* with two group generators (Löh 2015). By choosing the interpretation of output symbols appropriately, we can thus simulate $y$DT transducers by transducers that output numbers. Using the previous obseration, the same can be done even for *yDT* transducers with outputs in the free group; such transducers can also handle *inverses* of output symbols and cancellation of symbols by means of their inverses.

To generally deal with transducers that output numbers, we introduce the concept of *polynomial* transducers. Such a transducer proceeds as a $y$DT transducer but produces single integers as outputs. In particular, the transducer is able to add and multiply output numbers.

While considering subclasses of $y$DT transducers, our translation (via the encoding as matrices) gives appropriate subclasses of polynomial transducers. For instance, the linear $y$DT transducers can be translated into *affine* polynomial transducers. Likewise, $y$DT transducers satisfying the *single-use* restriction (*sur* for short) give rise to polynomial transducers that are *multiplicatively single-use* (for precise definitions, see Section 2). Total sur $y$DT transducers whose domain is specified by means of a top-down deterministic tree automaton (for short, DTT *automaton*) can simulate *deterministic streaming ranked-tree-to-string transducers*, as introduced by Alur and D'Antoni (2017). Such a transducer consists of a bottom-up deterministic automaton that, while processing the input tree, may update the string values maintained in a finite set of variables. Thereby, the new values at a node are constructed from the variables at the child nodes by concatenation where each variable is used at most once (and after use is reset to $\epsilon$). Viewed as a tree-to-string transducer, this model coincides with $y$DT transducers extended with regular look-ahead as considered in Section 8—subject to the sur restriction.

We observe that for a given input tree, the output values of the states of a total polynomial transducer can be collected into a vector, where the output vector for an input tree is a *polynomial transformation* of the corresponding output vectors of the input subtrees. First consider the case where these transformations are affine. We are interested in invariants satisfied by all input trees accepted by a given DTT automaton. If the invariant happens to be an *affine equality*, then *sets* of output vectors attained at the states of the automaton can be replaced by their *affine closures*. This observation allows us to apply exact fixpoint techniques as known from abstract interpretation of programs (Müller-Olm and Seidl 2004b) to effectively compute these affine closures and thus to decide whether a given affine equality is an invariant. Via the translation from linear $y$DT transducers, this method provides us with a decision procedure for equivalence of linear $y$DT transducers that runs in randomized polynomial time. Applying the translation of *msur* transducers into *affine* transducers (essentially from Benedikt et al. (2017)), we furthermore arrive at a decision procedure for *sur* $y$DT transducers that runs in randomized exponential time.

In the case of general polynomial transducers, methods based on affine closures can no longer be applied. Instead, we maintain sets of invariants for the states of the domain automaton. Indeed, we show that to verify one particular polynomial equality, it suffices to consider *inductive invariants* based on finite conjunctions of polynomial equalities only. For that, we rely on Hilbert's basis theorem as well as the technical property that polynomial ideals are compatible with Cartesian products. For the specific case of monadic input alphabets, we obtain an explicit algorithm for constructing inductive invariants that resembles techniques for effectively proving polynomial program invariants by means of weakest pre-conditions (Letichevsky and Lvov 1993; Müller-Olm and Seidl 2004a). For non-monadic input symbols, we obtain two semi-decision procedures, one enumerating all candidates for inductive invariants and one searching for a counter-example. Via

the translation of yDT transducers into polynomial transducers, the resulting algorithm can be applied to decide whether or not two arbitrary yDT transducers are equivalent.

### Related Work

Decision procedures for equivalence of deterministic tree transducers have been provided for various sub-classes of transducers (see Maneth (2015) for a recent survey). The equivalence problem for yDT transducers is mentioned as a difficult open question as early as in Engelfriet (1980). 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 (Engelfriet and Maneth 2006); this class is equal to macro tree-to-string translations of linear size increase (Engelfriet and Maneth 2003b) and can be simulated by yDT transducers with regular look-ahead (see Engelfriet and Maneth (1999) and Engelfriet and Maneth (2002)). Since equivalence of yDT transducers with regular look-ahead can be reduced to equivalence of yDT transducers without look-ahead but relative to a DTT automaton, our decidability result encompasses the decidability result for MSO definable tree-to-string transductions. It is more general, since yDT transducers may have more than linear size increase (in fact, they can have exponential size increase) and thus may not be MSO definable. It is unclear how or whether the suggested methods can be generalized to the equivalence problem of unrestricted yDT transducers.

The methods employed to obtain our novel results have the following predecessors. The algorithm for deciding affine invariants of affine transducers is related to the algorithm in Seidl (1990) 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 affine transducers.

The known algorithm for deciding equivalence of yDT transducers with *monadic* alphabet is based on a reduction to the HDT0L sequence equivalence problem. The latter can be solved (Honkala 2000) via establishing an increasing chain of finite sets of word equations that are guaranteed to eventually agree in their sets of solutions. Instead, our *direct* algorithm for monadic yDT transducers is related to an algorithm for effective program verification. In Letichevsky and Lvov (1993) and, later, Müller-Olm and Seidl (2004a), a decision procedure is presented that allows us to check whether a given polynomial equality is invariably true at a given program point of a polynomial program, i.e., a sequential program with non-deterministic branching and polynomial assignments of numerical variables. 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. Cast in the formalism of *polynomial automata*, the complexity of this algorithm is investigated in Benedikt et al. (2017). Our algorithm for deciding polynomial invariants of general polynomial transducers, as well as for solving the equivalence problem of unrestricted yDT transducers, is new.

### Organization of the Paper

In the next section, we provide basic notations and concepts. In particular, we introduce the new notion of a *polynomial* transducer that produces integer outputs, and we indicate how total yDT transducers can be translated into polynomial transducers. This translation can also be applied to transducers that do not produce outputs in a free monoid but in a free *group*. Furthermore, we identify subclasses of polynomial transducers that correspond (via the given translation) to subclasses of yDT transducers. In Section 4, we provide a polynomial algorithm for deciding *affine* invariants of *affine* transducers for a given top-down regular subset of inputs. This algorithm is based on a least fixpoint iteration over affine spaces. It can be extended to an exponential

algorithm for deciding Π-*homogeneous* invariants of an *msur* transducer (with partition Π). Via the translation into polynomial transducers, a decision procedure is obtained for equivalence of *linear* and *sur* yDT transducers. The lengths of numbers used by the algorithm of Section 4 may, however, grow due to the required uses of multiplication. Therefore, we provide explicit upper bounds to that growth in Section 5. Section 6 then considers the case of polynomial invariants of arbitrary polynomial transducers for given top-down regular subsets of inputs. It makes use of polynomial ideals in a non-trivial way. As kind of warm-up, a dedicated algorithm for polynomial transducers with monadic input alphabet is provided that 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* that provide proofs of polynomial invariants. The strongest inductive invariant, however, can only be characterized non-effectively via a *greatest fixpoint* over polynomial ideals. Section 6 provides us therefore with a decision procedure for polynomial invariants that is based on two *semi*-algorithms. In-validity can be verified by providing a witness input for which the invariant is violated. Searching for a proof of the invariant, on the other hand, is possible but may be difficult. Therefore, Section 7 provides a systematic means to identify candidate inductive invariants. Again, the decision procedure for polynomial invariants gives rise to a decision procedure of equivalence of general yDT transducers. Finally, Section 8 discusses applications of the obtained results to various models of transducers as proposed in the literature, notably streaming ranked-tree-to-string transducers (Alur and D'Antoni 2017).

A preliminary version of this article was presented at the 56th Annual Symposium on Foundations of Computer Science (Seidl et al. 2015). That version has been carefully revised and simplified by introducing the new concept of *polynomial* transducers. This concept allows to elegantly deal with inverses of output symbols as well as the subclass of *sur* transducers. We also extended the article with a practical algorithm for enumerating inductive invariants (Section 7).

## 2 PRELIMINARIES

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

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

A *deterministic tree-to-string transducer* (yDT transducer for short) is a tuple $M = (Q, \Sigma, \Delta, q_0, \delta)$, where $Q$ is a ranked alphabet of states all of rank 1, $\Sigma$ is a ranked alphabet of input symbols, $\Delta$ is an alphabet of output symbols, $q_0 \in Q$ is the initial state, and $\delta$ is the transition function. For every $q \in Q$, $k \geq 0$, and $f \in \Sigma^{(k)}$, $\delta(q, f)$ is either undefined or is in $R$, where $R$ is the smallest set such that $\epsilon \in R$ and if $T \in R$, then also

(1) $aT \in R$ for $a \in \Delta$, and
(2) $q'(x_i)\,T \in R$ for $q' \in Q$ and $i \in [k]$.

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

$$q(f(x_1, \ldots, x_k)) \to T.$$

A state $q \in Q$ induces a partial function $[\![q]\!]_M$ from $\mathcal{T}_\Sigma$ to $\Delta^*$ defined recursively as follows. Let $t = f\,\mathbf{t}$ with $f \in \Sigma^{(k)}$, $k \geq 0$, and $\mathbf{t} \in \mathcal{T}_\Sigma^k$. Then $[\![q]\!]_M(t)$ is defined whenever $\delta(q, f) = T$ for some $T$ and for each occurrence of a subtree $q'(x_i)$ in $T$, $[\![q']\!]_M(\mathbf{t}_i)$ is also defined. In this case, the output is obtained by evaluating $T$ in call-by-value order with $\mathbf{t}_i$ taken for $x_i$. In function applications, we often leave out parenthesis; e.g., we write $[\![T]\!]_M\,\mathbf{t}$ for $[\![T]\!]_M(\mathbf{t})$. We obtain

$$[\![q]\!]_M(f\,\mathbf{t}) = [\![T]\!]_M\,\mathbf{t},$$

where the evaluation function $[\![T]\!]_M$ is defined as follows:

$$\begin{aligned} [\![\epsilon]\!]_M\,\mathbf{t} &= \epsilon \\ [\![aT']\!]_M\,\mathbf{t} &= a\,[\![T']\!]_M\,\mathbf{t} \\ [\![q'(x_i)T']\!]_M\,\mathbf{t} &= [\![q']\!]_M\,\mathbf{t}_i\,[\![T']\!]_M\,\mathbf{t}. \end{aligned}$$

The transducer $M$ realizes the (partial) translation $M : \mathcal{T}_\Sigma \to \Delta^*$ that, for $t \in \mathcal{T}_\Sigma$, is defined as $M(t) = [\![q_0]\!]_M(t)$ if $[\![q_0]\!]_M(t)$ is defined and is undefined otherwise; the domain of this translation is denoted $\mathrm{dom}(M)$. If the translations realized by two transducers $M, N$ are equal, then $M$ is *equivalent* to $N$.

We introduce the following subclasses of transducers. The yDT transducer $M$ is

- *total* if $\delta(q, f)$ is defined for all $q \in Q$ and $f \in \Sigma$;
- *linear* if in each right-hand side $\delta(q, f)$, each variable $x_i$ occurs at most once;
- *strongly single-use* (abbreviated: *ssur*), if for each $f \in \Sigma$ of rank $k \geq 0$, each term $q'(x_i)$, $q' \in Q$, $i \in [k]$, occurs at most once in once in the union of all right-hand sides $\delta(q, f)$, $q \in Q$.

We remark that a *linear* yDT transducer is not necessarily strongly single use and that a strongly single-use yDT transducer is not necessarily linear. In fact, there are translations realized by linear yDT transducers for which no equivalent strongly single-use yDT transducer exists, see Theorem 5.6 of Engelfriet and Maneth (1999). The converse is easy to see: Let $M$ be the strongly single-use yDT transducer with the rules $q_0(d(x_1)) \to q_a(x_1)q_b(x_1)q_c(x_1)$ and for $\gamma \in \{a, b, c\}$, $q_\gamma(d(x_1)) \to \gamma q_\gamma(x_1)$ and $q_\gamma(d(x_1)) \to \epsilon$. The range of $M$ is $\{a^n b^n c^n \mid n \geq 0\}$ which is not context-free. The translation of $M$ cannot be realized by any linear yDT transducer, because ranges of linear yDT transducers are context-free: They are the *yield* of ranges of linear deterministic top-down tree transducers, i.e., the yield of regular tree languages (see, e.g., Propositions 20.2 and 16.5 of Gécseg and Steinby (1997)). Hence, they are context-free (see, e.g., Proposition 14.3 of Gécseg and Steinby (1997)).

In Engelfriet and Maneth (1999), a generalization of the strongly single-use property is introduced that is a proper generalization also of linearity. It is called *single-use restricted (in the input)* for short *sur*. The idea of *sur* is to define a partition $\Pi$ of the set of states (called *sur partition*) such that whenever states $q_1, q_2$ translate the same input node, then $q_1, q_2 \in Q'$ for some $Q' \in \Pi$.

Formally, for a partition $\Pi$ of $Q$, let $C_\Pi$ denote the *conflict relation* defined by

$$C_\Pi = \{(q_1, q_2) \mid q_1 \neq q_2 \wedge \exists Q' \in \Pi \text{ such that } \{q_1, q_2\} \subseteq Q'\}.$$

If $(q_1, q_2) \in C_\Pi$, then the states $q_1, q_2$ are *conflicting*. We call a partition $\Pi$ of $Q$ a *sur partition* for $M$ if for every input symbol $f \in \Sigma$ with rank $k \geq 1$, every $i \in [k]$, every $Q' \in \Pi$ for all distinct occurrences $q_1'(x_i)$, $q_2'(x_i)$ in the concatenation of $\delta(q', f), q' \in Q'$, $q_1', q_2'$ must be different but belong to the same equivalence class of $\Pi$, i.e., $(q_1', q_2') \in C_\Pi$ holds. The transducer $M$ is *single-use*

*restricted* (*sur* for short) if there exists a *sur* partition for $M$. If $M$ is *ssur*, then $M$ is *sur* for the partition $\Pi = \{Q\}$. Likewise, a linear transducer is *sur* for $\Pi = \{\{q\} \mid q \in Q\}$.

In the following, we restrict ourselves to transducers that are *total* but assume that we are given a DTT automaton that characterizes the subset of *legal* inputs, i.e., those for which we are interested in the results produced by the transducers. As for DTT automata, we define the size $|M|$ of a $y$DT transducer $M$ as the sum of the sizes of the involved alphabets, here $Q, \Sigma$, and $\Delta$, 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 and states in $T$.

### Polynomial Transducers

In the following, we introduce another form of transducers that differ from $y$DT transducers in that the outputs are not considered as strings but as *numbers*. Assume that the output alphabet of a $y$DT is *unary*, i.e., $\Delta = \{d\}$. In this case, we may interpret the $y$DT transducer to produce the *lengths* of outputs, instead of the output directly. For that, each occurrence of the output letter $d$ is interpreted as 1 while concatenation of outputs is interpreted as addition. For convenience, we may also allow multiplication with constants to compactly represent repeated addition of the same subterm.

*Example 2.1.* Consider the following rule of a transducer with output alphabet $\{d\}$:

$$q(f(x_1, x_2)) \to d\, q_1(x_1)\, d\, q_2(x_2)\, d\, q_1(x_1)\, d.$$

Then the length of the output can be generated by

$$q(f(x_1, x_2)) \to 4 + 2 \cdot q_1(x_1) + q_2(x_2).$$

We remark that, according to this interpretation, each right-hand side $\delta(q, f)$ of the unary $y$DT transducer gives rise to an *affine* function in the terms $q'(x_i)$, i.e., can be equivalently re-organized into

$$a_0 + a_1 \cdot q_1(x_{i_1}) + \cdots + a_r \cdot q_r(x_{i_r})$$

for some $r \geq 0$, constants $a_0, \ldots, a_r \in \mathbb{Z}$, states $q_1, \ldots, q_r$, and $x_{i_1}, \ldots, x_{i_r} \in X$.

Subsequently, we generalize this idea by additionally allowing *multiplication* of the terms $q'(x_i)$ in right-hand sides. Right-hand sides then represent *polynomials* in the terms $q'(x_i)$. Technically, a *polynomial* transducer $M$ is a tuple $(Q, \Sigma, \delta, q_0)$, where $Q, \Sigma, q_0$ are as for a $y$DT, while $\delta$ is a function from pairs $(q, f) \in Q \times \Sigma$ to expressions of the form:

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

where $c \in \mathbb{Z}$ is a number. Thus, right-hand sides now are arbitrary polynomials in the terms $q'(x_i)$. Again, we denote the fact that $\delta(q, f) = T$ by the rule $q(f(x_1, \ldots, x_k)) \to T$. Analogously to the semantics of $y$DT transducers, we define the semantics $[\![q]\!]_M$ of a polynomial transducer $M$ by $[\![q]\!]_M(f\ \mathbf{t}) = [\![T]\!]_M\ \mathbf{t}$, where the evaluation function $[\![T]\!]_M$ is given by:

$$\begin{aligned}
[\![c]\!]_M\ \mathbf{t} &= c \\
[\![T_1 + T_2]\!]_M\ \mathbf{t} &= [\![T_1]\!]_M\ \mathbf{t} + [\![T_2]\!]_M\ \mathbf{t} \\
[\![T_1 \cdot T_2]\!]_M\ \mathbf{t} &= [\![T_1]\!]_M\ \mathbf{t} \cdot [\![T_2]\!]_M\ \mathbf{t} \\
[\![q'(x_i)]\!]_M\ \mathbf{t} &= [\![q']\!]_M\ \mathbf{t}_i.
\end{aligned}$$

*Example 2.2.* Consider the input alphabet $\Sigma = \{f, a, b\}$, where $f$ has rank 2 and $a, b$ both have ranks 0. Consider the polynomial transducer $M$ with the following rules:

$$\begin{array}{ll} q_1(f(x_1,x_2)) \to q_1(x_2) + q_2(x_2) \cdot q_1(x_1) & \\ q_2(f(x_1,x_2)) \to q_2(x_1) \cdot q_2(x_2) & \\ q_1(a) \to 1 & q_1(b) \to 2 \\ q_2(a) \to 3 & q_2(b) \to 3, \end{array}$$

where the initial state is $q_1$. Assume that $w \in \{a, b\}^*$ is the sequence of leaves of the input tree $t \in \mathcal{T}_\Sigma$, i.e., $w = \mathsf{yield}(t)$. Then the value produced by $q_2$ for $t$ is given by $3^{|w|}$. Accordingly, the output of $q_1$ for $t$ is the representation of $w$ in the number system with base 3, where $a$ and $b$ correspond to the digits 1 and 2, respectively.

The size $|T|$ is defined as the number of leaves of $T$, i.e.,

$$\begin{array}{lll} |c| & = 1 & |T_1 + T_2| = |T_1| + |T_2| \\ |q(x_i)| & = 1 & |T_1 \cdot T_2| \ = |T_1| + |T_2|. \end{array}$$

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

For a polynomial transducer $M$ and a state $p$ of a DTT automaton $A$, we are interested in *invariants*, i.e., properties that hold for all input trees in $\mathrm{dom}(p)$. W.l.o.g. assume that the set $Q$ of states of $M$ is given by $Q = [n]$ for some $n \geq 1$. Let $t \in \mathcal{T}_\Sigma$. The vector $[\![t]\!]_M \in \mathbb{Z}^n$ of outputs produced by $M$ for $t$ is defined by

$$[\![t]\!]_{M,q} = [\![q]\!]_M(t), \qquad q \in [n].$$

For $t = f\,\mathbf{t}$, the vector $[\![t]\!]_M$ can be constructed inductively from the output vectors $[\![\mathbf{t}_i]\!]_M$, $i = 1, \ldots, k$, by:

$$\begin{array}{lll} [\![t]\!]_{M,q} & = [\![\delta(q,f)]\!]([\![\mathbf{t}_1]\!]_M, \ldots, [\![\mathbf{t}_k]\!]_M) & \text{where} \\ [\![c]\!](\mathbf{z}_1, \ldots, \mathbf{z}_k) & = c & \\ [\![q'(x_i)]\!](\mathbf{z}_1, \ldots, \mathbf{z}_k) & = \mathbf{z}_{iq'} & \\ [\![T_1 + T_2]\!](\mathbf{z}_1, \ldots, \mathbf{z}_k) & = [\![T_1]\!](\mathbf{z}_1, \ldots, \mathbf{z}_k) + [\![T_2]\!](\mathbf{z}_1, \ldots, \mathbf{z}_k) & \\ [\![T_1 \cdot T_2]\!](\mathbf{z}_1, \ldots, \mathbf{z}_k) & = [\![T_1]\!](\mathbf{z}_1, \ldots, \mathbf{z}_k) \cdot [\![T_2]\!](\mathbf{z}_1, \ldots, \mathbf{z}_k). & \end{array}$$

Let $H$ denote a function $\mathbb{Z}^n \to \mathbb{Z}$. Then $H \doteq 0$ is an *invariant* of $M$ at state $p$, iff $H([\![t]\!]_M) = 0$ for all $t \in \mathrm{dom}(p)$. Subsequently, we consider *polynomial* invariants where the function $H$ can be expressed by a *polynomial* in its arguments. In this case, $H$ can be considered as a mapping $\mathbb{Q}^n \to \mathbb{Q}$.

As for yDT transducers, we consider specific classes of polynomial transducers. In the next section, we will see that these classes can be used to simulate interesting classes of tree-to-string transducers. Let $M = (Q, \Sigma, \delta, q_0)$ denote a polynomial transducer. Then $M$ is called *affine* if each right-hand side is a sum of products

$$z \cdot q_1(x_{i_1}) \cdot \ldots \cdot q_r(x_{i_r}) \tag{1}$$

with $z \in \mathbb{Z}$, where all $i_1, \ldots, i_r$ are pairwise distinct. The polynomial transducer from Example 2.2 is affine in this sense.

The notion of affinity can be relaxed by allowing products of calls $q(x_i) \cdot q'(x_i)$ for states $q \neq q'$ – if only $q$ and $q'$ are *conflicting*. Technically, conflicting states for polynomial transducers are defined analogously to conflicting states for yDT transducers. Let $\Pi$ denote a partition of $Q$ and $C_\Pi$ the corresponding conflict relation. The monomial (1) is called $\Pi$-*homogeneous* if $i_s = i_{s'}$ for $s \neq s'$ implies that $q_s, q_{s'}$ are distinct but belong to the same equivalence class of $\Pi$, i.e., $(q_s, q_{s'}) \in C_\Pi$. The partition $\Pi$ of the set $Q$ is called *multiplicative sur partition* for $M$ if for every $f \in \Sigma$ of rank $k \geq 1$ and every $1 \leq i \leq k$,

**M1** Each monomial occurring in a right-hand side of $M$, is $\Pi$-homogeneous;
**M2** $(q_1, q_2) \in C_\Pi$ implies $(q_1', q_2') \in C_\Pi$—whenever $q_1'(x_i)$ occurs in $\delta(q_1, f)$ and $q_2'(x_i)$ occurs in $\delta(q_2, f)$.

In the same spirit as in the definition of *sur*, this definition can be equivalently formulated by stating that for each $f \in \Sigma$ and each $Q' \in \Pi$, each monomial in the product $\delta(q', f), q' \in Q'$, once rewritten as a sum of products, must be $\Pi$-homogeneous. The polynomial transducer $M$ then is called *multiplicatively sur* (short form: *msur*) if there is a multiplicative *sur* partition for $M$. In this sense, each affine polynomial transducer $M$ is *msur* for $\Pi = \{\{q\} \mid q \in Q\}$. In fact, at the expense of a blow-up in the number of states of $M$, each polynomial transducer that is *msur* is equivalent to an *affine* polynomial transducer. For a multiplicative *sur* partition $\Pi$ for $M$, let $\mathcal{P}_\Pi(Q)$ denote the set of subsets $S \subseteq Q$ that consist of conflicting elements only, i.e., with $S \subseteq Q'$ for some $Q' \in \Pi$.

Lemma 2.3. *Assume that the total polynomial transducer $M$ is* msur *with set $Q$ of states and* msur *partition $\Pi$. Then an affine total polynomial transducer $M'$ can be constructed with $\mathcal{P}_\Pi(Q)$ as set of states where for each $S \in \mathcal{P}_\Pi(Q)$ and each input tree $t$,*

$$[\![S]\!]_{M'}(t) = \prod_{q \in Q'} [\![q]\!]_M(t), \tag{2}$$

*holds.*

Proof. We define the transition function $\delta'$ of $M'$ as follows. Consider an element $Q' \in \mathcal{P}_\Pi(Q)$ and some input symbol $f \in \Sigma$ with rank $k \geq 0$. If $k = 0$, then $\delta(S, f) = \prod_{q \in S} \delta(q, f)$. Now assume that $k > 0$. Consider the right-hand sides $\delta(q, f), q \in S$. As all pairs of distinct states $q, q'$ in $S$ are conflicting, each monomial $m$ in the product $u = \prod_{q \in S} \delta(q, f)$ is $\Pi$-homogeneous. Therefore, it can be re-arranged into a product

$$z \cdot \left( \prod_{q_1 \in S_1} q_1(x_{i_1}) \right) \cdot \ldots \cdot \left( \prod_{q_r \in S_r} q_r(x_{i_r}) \right)$$

for $z \in \mathbb{Z}$ and suitable subsets $S_1, \ldots, S_r \in \mathcal{P}_\Pi(Q)$ such that $i_s \neq i_{s'}$ for $s \neq s'$. The right-hand side $\delta'(S, f)$ then is obtained from the product $u$ be replacing each such monomial $m$ with

$$z \cdot S_1(x_{i_1}) \cdot \ldots \cdot S_r(x_{i_r}).$$

By construction, the resulting transducer is affine, and the validity of Equation (2) follows by induction on the structure of input trees. □

We remark that a related construction has recently been proposed for polynomial automata on words (Proposition 9 in Benedikt et al. (2017)).

## 3 TREE-TO-MATRIX TRANSDUCERS

Instead of considering transducers where the outputs are plain numbers, we may also consider transducers where the outputs are interpreted as $l \times l$ matrices with entries in $\mathbb{Z}$. The set $\mathcal{M}_l(\mathbb{Z})$ of all such matrices forms a monoid where the monoid operation is matrix multiplication. A tree-to-matrix transducer $M = (M_0, \alpha)$ then consists of a yDT transducer $M_0$ together with *homomorphism* $\alpha : \Delta^* \to \mathcal{M}_l(\mathbb{Z})$ to be applied to the output produced by $M_0$. W.r.t. the homomorphism $\alpha$, an output string $w$ represents the matrix $\alpha(w) \in \mathcal{M}_l(\mathbb{Z})$. Every total tree-to-matrix transducer, on the other hand, can be simulated by a total polynomial transducer. We have:

Lemma 3.1. *Assume that $M = (M_0, \alpha)$ is a total tree-to-matrix transducer with set of states $Q$ using $l \times l$ output matrices over $\mathbb{Z}$. Then a total polynomial transducer $M'$ with set of states $Q \times [l]^2$ can be*

*constructed in polynomial time such that for every input tree $t$,*

$$[\![\langle q, ij\rangle]\!]_{M'}(t) = (\alpha([\![q]\!]_{M_0}(t)))_{ij} \tag{3}$$

*for all $q \in Q$ and $1 \le i, j \le l$.*

*If $M_0$ is linear, then $M'$ is affine. If $M_0$ is* sur *with partition $\Pi$, then $M'$ is* msur *with partition $\Pi' = \{\{\langle q, ij\rangle \mid q \in Q'\} \mid Q' \in \Pi, 1 \le i, j \le l\}$.*

Proof. For every state $q$ of $M_0$, the state $\langle q, ij\rangle$, $1 \le i, j \le l$ is meant to produce the $(i, j)$-th entry of the matrix produced by $(M_0, \alpha)$ for a given input. Technically, this means that for $f \in \Sigma$ of rank $k \ge 0$ and a rule $q(f(x_1, \ldots, x_k)) \to T$ of $M_0$, $M'$ has the rules

$$\langle q, ij\rangle(f(x_1, \ldots, x_k)) \to \mathcal{E}_{ij}[T]$$

for $1 \le i, j \le l$, where

$$\begin{aligned}
\mathcal{E}_{ij}[\epsilon] &= \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \\
\mathcal{E}_{ij}[a\,T] &= \textstyle\sum_{r=1}^{l} (\alpha(a))_{ir} \cdot \mathcal{E}_{rj}[T] \qquad \text{for } a \in \Delta \\
\mathcal{E}_{ij}[q'(x_\kappa)\,T] &= \textstyle\sum_{r=1}^{l} \langle q', ir\rangle(x_\kappa) \cdot \mathcal{E}_{rj}[T].
\end{aligned}$$

By structural induction on input trees $t$, and for each tree then on the structure of right-hand sides, we verify that Equation (3) indeed holds.

Now assume that the transducer $M_0$ is linear, i.e., each variable $x_\kappa$ occurs in any right-hand side at most once. Then each right-hand side of $M'$ can be re-arranged into a sum of products of the form

$$a \cdot \langle q_1, i_1 j_1\rangle(x_{\kappa_1}) \cdot \ldots \cdot \langle q_r, i_r j_r\rangle(x_{\kappa_r}),$$

where $r \ge 0$ and $\kappa_1, \ldots, \kappa_r$ are all distinct. From that follows that the resulting polynomial transducer $M'$ is *affine*.

Finally, assume that the transducer $M_0$ is *sur* with *sur* partition $\Pi$ and that $\Pi' = \{\{\langle q, ij\rangle \mid q \in Q'\} \mid Q' \in \Pi\}$ holds.

We verify that $M'$ with partition $\Pi'$ satisfies the properties **M1** and **M2**. Each monomial in the right-hand side of $M'$ is of the form $a \cdot q_{1 i_1 j_1}(x_{i_1}) \cdot \ldots \cdot q_{r i_r j_r}(x_{i_r})$ with $a \in \mathbb{Z}$. As it arises from a suitable product of the matrices for the states $q_1, \ldots, q_r$ of $M_0$ occurring in the right-hand side, Property *sur* of $M_0$ implies property **M1** of $M'$. Now consider two distinct states $\langle q_1, i_1 j_1\rangle, \langle q_2, i_2 j_2\rangle$ of $M'$ so that $(\langle q_1, i_1 j_1\rangle, \langle q_2, i_2 j_2\rangle) \in C_{\Pi'}$. This implies that $q_1 \ne q_2$, and the pair $(q_1, q_2)$ is contained in $C_\Pi$. For every input symbol $f \in \Sigma$, the right-hand sides $\delta'(\langle q_\nu, i_\nu j_\nu\rangle, f)$ are given by $\mathcal{E}_{i_\nu j_\nu}(\delta(q_\nu, f))$ for $\nu = 1, 2$. Assume that $\langle q'_\nu, i'_\nu j'_\nu\rangle$ occurs in $\delta'(\langle q_\nu, i_\nu j_\nu\rangle, f)$. Then $q'_\nu$ occurs in $\delta(q_\nu, f)$. Since $\Pi$ is a *sur* partition, $(q'_1, q'_2) \in C_\Pi$. Therefore, $(\langle q'_1, i'_1 j'_1\rangle, \langle q'_2, i'_2 j'_2\rangle) \in C_{\Pi'}$ holds—proving Property **M2**.

*Example 3.2.* Consider the two matrices

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

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

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

In this way, the free monoid of strings over the alphabet $\Delta$ can be represented by the sub-monoid of $\mathcal{M}_2(\mathbb{Z})$ generated by $m_1, m_2$.

As a corollary, we obtain:

Corollary 3.3. *Assume that $M$ is a total yDT transducer with $n$ states and output alphabet $\Delta = \{a_1, a_2\}$. Then a total polynomial transducer $M'$ can be constructed within polynomial time such that for every input tree $t$ and $j = 1, 2$, the following holds:*

$$(\alpha(M(t))_{1j} = [\![q_j]\!]_{M'}(t)$$

*for suitable states $q_1, q_2$ of $M'$.*

*If $M$ is linear, then $M'$ is affine. If $M$ is* sur*, then $M'$ is* msur*.*

We remark that extending the matrix monoid considered in Example 3.2 with *inverses* would not provide us with a free group. Nonetheless, we can re-use the given construction to simulate yDT transducers with output in the free group $\mathcal{F}_2$ generated from $a_1, a_2$. The free group $\mathcal{F}_2$ additionally has elements $a_1^-, a_2^-$ such that $a_1 a_1^- = a_1^- a_1 = \epsilon$, and, likewise, $a_2 a_2^- = a_2^- a_2 = \epsilon$.

Instead of the free monoid generated by the matrices $m_1, m_2$ in Example 3.2, however, we consider the *subgroup* of $2 \times 2$ matrices that is generated by the elements:

$$b_1 = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \qquad b_2 = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}.$$

This group is also known as *Sanov* group $\mathcal{S}$ (see, e.g., Example 4.5.1 of Löh (2015)). 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:

$$b_1^- = \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix} \qquad b_2^- = \begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix}.$$

The Sanov group is useful for us, since $\mathcal{S}$ with subgroup generators $b_1$ and $b_2$ is isomorphic to the *free* group $\mathcal{F}_2$ that is freely generated from two elements $a_1, a_2$. Thus, Corollary 3.3 can be extended to yDT transducers with outputs in the free group.

Corollary 3.4. *Assume that $M$ is a total yDT transducer with $n$ states and output in the free group generated by $\{a_1, a_2\}$. Let $\alpha : \{a_1, a_2, a_1^-, a_2^-\}^* \to \mathcal{M}_2(\mathbb{Z})$ be the homomorphism such that $\alpha(a_i) = b_i$ and $\alpha(a_i^-) = b_i^-$ ($b_1, b_2$ are the two generators of the Sanov group). Then a total polynomial transducer $M'$ can be constructed within polynomial time such that for every input tree $t$ the following holds:*

$$\alpha(M(t)) = \begin{bmatrix} [\![q_{11}]\!]_{M'}(t) & [\![q_{12}]\!]_{M'}(t) \\ [\![q_{21}]\!]_{M'}(t) & [\![q_{22}]\!]_{M'}(t) \end{bmatrix}$$

*for suitable states $q_{11}, q_{12}, q_{21}, q_{22}$ of $M'$.*

*If $M$ is linear, then $M'$ is affine. If $M$ is* sur*, then $M'$ is* msur*.*

Via the translation of total yDT transducers into polynomial transducers, any decision procedure of affine invariants thus translates into a decision procedure for equality of yDT transducers.

Lemma 3.5. *Assume that $M_1, M_2$ are yDT transducers with outputs in a free monoid or in the free group. Then a DTT automaton $A$, a total polynomial transducer $M'$ can be constructed together with affine functions $H_1, \ldots, H_4$ and a state $p_0$ of $A$ such that $M_1$ and $M_2$ are equivalent iff the following properties are satisfied:*

*(1) Both the domains of $M_1$ and $M_2$ equal $\mathcal{L}(A)$;*
*(2) $H_1 \doteq 0, H_2 \doteq 0, H_3 \doteq 0$, and $H_4 \doteq 0$ all are invariants of $M'$ at $p_0$.*

*If $M_1$ and $M_2$ are linear, then $M'$ is affine. If $M_1$ and $M_2$ are* sur*, then $M'$ is* msur*.*

Proof. First, we, w.l.o.g., may assume that we are given two states $q_1, q_2$ of a single yDT transducer $M$, and the task is to decide whether the partial mappings $[\![q_1]\!]_M$ and $[\![q_2]\!]_M$ coincide, i.e., whether (1) $[\![q_1]\!]_M(t)$ is defined iff $[\![q_2]\!]_M(t)$ is defined, and (2) $[\![q_1]\!]_M(t) = [\![q_2]\!]_M(t)$ whenever both are defined. To decide the former task, we construct DTT automata $A_1, A_2$, where the languages of $A_1$ and $A_2$ are precisely given by the domains of the translations $[\![q_1]\!]_M$ and $[\![q_2]\!]_M$, respectively.

For $\nu = 1, 2$, the set of states and transitions of $A_\nu$ can be determined as the smallest set $P_\nu$ of sets $Q' \subseteq [n]$ together with the partial function $\rho_\nu$ as follows. First, $\{q_\nu\} \in P_\nu$ that also serves as the initial state of $A_\nu$. Then for every element $Q' \in P_\nu$ and every input symbol $f \in \Sigma$ of some arity $k$, where $\delta(q, f)$ is defined for each $q \in Q'$, every set $Q'_i$ is contained in $P_\nu$ for $i = 1, \ldots, k$, 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 $\rho_\nu$ has the transition $\rho_\nu(Q', f) = Q'_1 \ldots Q'_k$. Property (1) is satisfied iff $\mathcal{L}(A_1) = \mathcal{L}(A_2)$. This can be verified in time polynomial in the sizes of $A_1$ and $A_2$. We remark that the sizes of the DTT automata $A_1, A_2$ can be at most exponential in the size of $M$. In case, however, that $M$ is linear, the sizes of the corresponding automata $A_1, A_2$ are at most linear in the size of $M$.

Now assume that $\mathcal{L}(A_1) = \mathcal{L}(A_2)$. We choose the state $p_0$ as the initial state of $A_1$, i.e., $p_0 = \{q_1\}$. Then, we construct a total yDT transducer $\bar{M}$ from $M$ by adding to the transition function of $M$ a transition $q(f(x_1, \ldots, x_k)) \to \epsilon$ for every state $q$ and input symbol $f$—where $M$ does not yet provide a transition. By construction, $[\![q]\!]_{\bar{M}}(t) = [\![q]\!]_M(t)$ whenever $[\![q]\!]_M(t)$ is defined. Therefore, for every $t \in \mathcal{L}(A_1)$, $[\![q_1]\!]_{\bar{M}}(t) = [\![q_2]\!]_{\bar{M}}(t)$ iff $[\![q_1]\!]_M(t) = [\![q_2]\!]_M(t)$.

W.l.o.g. we may assume that the output alphabet is given by $\Delta = \{a_1, a_2\}$ in case of outputs as strings or $\Delta = \{a_1, a_2, a_1^-, a_2^-\}$ in case of outputs in the free group. We then consider the tree-to-matrix transducer $(\bar{M}, \alpha)$ where the homomorphism $\alpha$ interprets the output symbols as the corresponding generator matrices of the Sanov subgroup of $\mathcal{M}_2(\mathbb{Z})$ and construct the corresponding polynomial transducer $M'$ according to Lemma 3.1. Then states $q_1$ and $q_2$ are equivalent w.r.t. $M$ at $p_0$ iff $M'$ satisfies the invariants

$$\begin{aligned}
\mathbf{z}_{\langle \mathbf{q_1}, \mathbf{11} \rangle} - \mathbf{z}_{\langle \mathbf{q_2}, \mathbf{11} \rangle} &\doteq 0 \\
\mathbf{z}_{\langle q_1, 12 \rangle} - \mathbf{z}_{\langle q_2, 12 \rangle} &\doteq 0 \\
\mathbf{z}_{\langle q_1, 21 \rangle} - \mathbf{z}_{\langle q_2, 21 \rangle} &\doteq 0 \\
\mathbf{z}_{\langle q_1, 22 \rangle} - \mathbf{z}_{\langle q_2, 22 \rangle} &\doteq 0.
\end{aligned}$$

From that, the claim of the lemma follows. □

## 4 AFFINE TRANSDUCERS

In this section, we show that for total polynomial transducers that are affine, all *affine* invariants for some top-down regular set are decidable in polynomial time. By Lemma 3.5, this result gives rise to decision procedures of equivalence for affine polynomial transducers as well as for linear and *sur* yDT transducers. For stating complexity results, we adopt a *uniform* complexity measure. This means that we assume that all numbers can be stored, retrieved, and compared in a single step. Also, we count arithmetic operations as $O(1)$, see Section 5 for more details on the uniform complexity measure.

Let $M = (Q, \Sigma, \delta, q_0)$ be a total affine transducer, and $A = (P, \Sigma, p_0, \rho)$ a DTT automaton. Assume that $Q = [n]$ for some natural number $n$ and let $H$ denote an affine function $H : \mathbb{Q}^n \to \mathbb{Q}$, given by

$$H(\mathbf{z}) = a_0 + a_1 \cdot \mathbf{z}_1 + \cdots + a_n \cdot \mathbf{z}_n. \tag{4}$$

Our goal is to decide for any given state $p_0$ of $A$ whether or not $H \doteq 0$ is an invariant, i.e., $H([\![t]\!]_M) = 0$ for all $t \in \mathrm{dom}(p_0)$. As the transducer $M$ is *affine*, $[\![\delta(q, f)]\!]$ constitutes a *multi-affine* mapping from $(\mathbb{Q}^n)^k$ to $\mathbb{Q}$. Technically, a multi-affine mapping $F$ is affine in each

argument. This means that the transformation $H'$ corresponding to the $\kappa$th argument and fixed $\mathbf{x}_1, \ldots, \mathbf{x}_{\kappa-1}, \mathbf{x}_{\kappa+1}, \ldots, \mathbf{x}_k$, which is defined by:

$$H'(\mathbf{x}') = F(\mathbf{x}_1, \ldots, \mathbf{x}_{\kappa-1}, \mathbf{x}', \mathbf{x}_{\kappa+1}, \ldots, \mathbf{x}_k)$$

is affine, i.e.,

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

holds for vectors $\mathbf{y}_0, \ldots, \mathbf{y}_k \in \mathbb{Q}^n$ and $\lambda_1, \ldots, \lambda_k \in \mathbb{Q}$. Let us define the (output) semantics of an input symbol $f \in \Sigma$ of arity $k$ as the function $[\![f]\!] : (\mathbb{Q}^n)^k \to \mathbb{Q}^n$ such that

$$[\![f]\!]\ \mathbf{x} = ([\![\delta(1, f)]\!]\ \mathbf{x}, \ldots, [\![\delta(n, f)]\!]\ \mathbf{x}). \tag{5}$$

By definition, this function is again multi-affine. This observation allows us to prove the following theorem:

Theorem 4.1. *Let $\Sigma$ be a fixed ranked alphabet, and $A$ a* DTT *automaton over $\Sigma$. For a total affine transducer $M$ with input alphabet $\Sigma$ and $n$ states, an affine function $H : \mathbb{Q}^n \to \mathbb{Q}$ and a state $p_0$ of $A$, it is decidable in polynomial time (w.r.t. the uniform complexity measure) whether or not $H([\![t]\!]_M) = 0$ holds for all $t \in \mathrm{dom}(p_0)$.*

Proof. Assume that the DTT automaton is given by $A = (P, \Sigma, \rho, p_0)$. The collection of sets of vectors $\{[\![t]\!]_M, t \in \mathrm{dom}(p), p \in P\}$, can be characterized by means of a constraint system. Let $V_p, p \in P$, denote the the collection of least sets with

$$V_p \supseteq [\![f]\!](V_{p_1}, \ldots, V_{p_k}) \tag{6}$$

whenever $\rho(p, f) = (p_1, \ldots, p_k)$ holds. Then for each $p \in P$, the set $\{[\![t]\!]_M \mid t \in \mathrm{dom}(p)\}$ is precisely given by $V_p$.

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

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

Recall that 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 $\mathrm{aff}(V)$. For the affine function $H : \mathbb{Q}^n \to \mathbb{Q}$ 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 $\mathbf{v} \in \mathrm{aff}(V)$;
(3) $H(\mathbf{v}) = 0$ for all $\mathbf{v} \in B$ if $B$ is any subset of $V$ with $\mathrm{aff}(B) = \mathrm{aff}(V)$.

Instead of verifying that $H(\mathbf{v}) = 0$ holds for all elements $\mathbf{v}$ of $V_{p_0}$, it thus 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 $\mathrm{aff}(V_{p_0})$. It is unclear, though, how the least solution $V_{p'}, p' \in P$, of the constraint system (6) can be computed. Instead of computing this least solution, we propose to consider the least solution of the constraint system (6), 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

```
forall (p ∈ P) B_p := ∅;
repeat
    done := true;
    forall (p, p_1, ..., p_k ∈ P, f ∈ Σ^(k) with ρ(p, f) = (p_1, ..., p_k))
        forall ((v_1, ..., v_k) ∈ B_{p_1} × ... × B_{p_k})
            v := [[f]](v_1, ..., v_k);
            if v ∉ aff(B_p)
                B_p := B_p ∪ {v};
                done := false;
until (done = true);
```

Fig. 1. Computing bases for the closures $\mathsf{aff}(\{[\![t]\!] \mid t \in \mathsf{dom}(p)\}), p \in P$.

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:

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

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

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

Let $V_p, p \in P$, and $V_p^\sharp, p \in P$, denote the least solutions of Equation (6) w.r.t. the complete lattices $\mathcal{P}(\mathbb{Q}^n)$ and $\mathcal{A}(\mathbb{Q}^n)$, respectively. Since for each $f \in \Sigma$, $[\![f]\!]$ is affine in each of its arguments, it follows by the transfer lemma of Apt and Plotkin (1986) (see also Aarts et al. (1995)), that

$$\mathsf{aff}(V_p) = V_p^\sharp \qquad (p \in P).$$

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 (6) 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 $V_{p_0}^\sharp = \mathsf{aff}(V_{p_0})$, we finally may check whether or not $H(\mathbf{v}) = 0$ for all $\mathbf{v} \in B_{p_0}$, which completes the procedure.

The algorithm starts with empty sets $B_p, p \in P$. Then it repeatedly performs one round through all transitions $\rho(p, f) = (p_1, \ldots, p_k)$ of $A$ while the flag done is false. For each transition $\rho(p, f) = (p_1, \ldots, p_k)$, the transformation $[\![f]\!]$ is applied to every $k$-tuple $v = (v^{(1)}, \ldots, v^{(k)})$ with $v^{(\kappa)} \in B_{p_\kappa}$. The resulting vectors then are added to $B_p$—whenever they are not yet contained in the affine closure $\mathsf{aff}(B_p)$ 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 for 1. Thus, evaluating a right-hand side $\delta(i, f)$ takes time at most proportional to the number of symbols occurring in $\delta(i, f)$. Concerning the complexity of the algorithm, we note:

- —In each round of the *repeat-until* loop, for each transition of $A$, at most $(n + 1)^m$ tuples are considered ($m$ the maximal arity of an input symbol).
- —The cost of computing the vector $[\![f]\!](\mathbf{v}_1, \ldots, \mathbf{v}_k)$ is in $O(|M|)$.
- —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 $B_p$.

— Of these rounds, the algorithm performs at most $h \cdot (n+1)$ ($h$ and $n$ the number of states of $A$ and $M$, respectively).

Accordingly, a full round of the *repeat-until* loop can be executed in time $O(|A| \cdot (n+1)^m \cdot (|M| + n^3))$. Assuming that $n \leq |M|$, we thus obtain an upper complexity bound of $O(|A| \cdot h \cdot (n+1)^{m+1} \cdot (|M| + n^3)) = O(|A| \cdot h \cdot |M| \cdot (n+1)^{m+3})$ for the algorithm. □

The base algorithm as presented in the proof of Theorem 4.1, and its complexity can be further improved as follows:

- We replace the given iteration by a *worklist* iteration that 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 that 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 $(v - v'), v \in B \backslash \{v'\}$. By maintaining a basis of $L_B$ in *Echelon* form, membership in $\mathrm{aff}(B)$ can be tested in time $O(n^2)$.

Applying these three optimizations, the overall complexity comes down to $O(|A| \cdot |M| \cdot (n+1)^{m+2})$. By using the translation of total *msur* transducers into total affine transducers from Lemma 2.3, we obtain as a corollary:

Corollary 4.2. *Let $M$ denote a total* msur *polynomial transducer with input alphabet $\Sigma$, $n$ states and multiplicative* sur *partition $\Pi$. Let $H : \mathbb{Q}^n \to \mathbb{Q}$ denote a $\Pi$-homogeneous polynomial function, i.e., $H$ is given by a polynomial $r$ in unknowns $\mathbf{z}_1, \ldots, \mathbf{z}_n$, where each monomial of $r$ is $\Pi$-homogeneous. Let $A$ denote a* dtt *automaton with input alphabet $\Sigma$. Then for every state $p_0$ of $A$, it can be decided in time polynomial in the size of $A$, and exponential in the size of $M$ (w.r.t. the uniform complexity measure), whether or not $H \doteq 0$ is an invariant at $p_0$, i.e., $H([\![t]\!]_M) = 0$ holds for all $t \in \mathrm{dom}(p_0)$.*

Proof. W.l.o.g. we may assume that the input alphabet is binary; this is justified, because it is straightforward to construct in polynomial time a transducer $M'$ such that for every tree $t$, $[\![M']\!](\mathrm{fcns}(t)) = [\![M]\!](t)$ if $t \in \mathrm{dom}(M)$ and $[\![M']\!](\mathrm{fcns}(t))$ is undefined otherwise. The tree $\mathrm{fcns}(t)$ is the *first-child/next-sibling* encoded binary tree of the tree $t$: In this tree, a node $u$ is the first child of a node $v$ iff $u$ is the first child of $v$ in $t$, and $u$ is the second child of $v$ iff $u$ is the next sibling of $v$.

Assume that the set of states of $M$ is given by $[n]$. The translation of $M$ into an The affine transducer $M'$ constructed for $M$ has a $\bar{Q}$ of states that consists of subsets of states $S \subseteq [n]$, where $S \subseteq Q'$ for some $Q' \in \Pi$ and $[\![S]\!]_{M'}(t) = \prod_{q \in S} [\![q]\!]_M(t) = \prod_{q \in S} [\![t]\!]_{M,q}$. In particular, each monomial of the polynomial $r$ representing $H$, is of the form $a_S \cdot \prod_{q \in S} \mathbf{z}_q$ for a coefficient $a_S \in \mathbb{Q}$ and some $S \in \bar{Q}$. Thus, $H([\![t]\!]_M) = 0$ for all $t \in \mathrm{dom}(p_0)$ holds iff $H'([\![t]\!]_{M'}) = 0$ for all such $t$ and the *affine* function $H'(\mathbf{z})$ defined by

$$H'(\mathbf{z}) = \sum_{S \in \bar{Q}} a_S \cdot \mathbf{z}_S.$$

Now applying the polynomial algorithm from Theorem 4.1 to the affine transducer $M'$ and the affine invariant $H' \doteq 0$ allows us to decide whether or not the invariant $H \doteq 0$ holds for $M$ at state $p_0$. □

So far, we have verified affine invariants for affine transducers or $\Pi$-homogeneous invariants for *msur* transducers relative to some dtt automaton only. Due to Lemma 3.5, the decision procedures

of invariants for total affine or *msur* transducers give rise to decision procedures for the equivalence of yDT transducers with either unary output alphabet or which are linear or *sur*. We remark that the exponential upper bounds of Theorem 4.3 come with exponential lower bounds, since non-emptiness already for *ssur* yDT transducers is DExpTime-complete (see Theorem 19 of Maneth (2015)).

Theorem 4.3. *Equivalence can be decided in deterministic exponential time for unary* yDT *transducers as well as for* sur yDT *transducers. If the transducers are* linear*, then equivalence can be decided in polynomial time (w.r.t. the uniform complexity measure).*

## 5 LENGTHS OF OCCURRING NUMBERS

Recall that our algorithms consider occurring numbers as atomic values that can be processed in time $O(1)$. For unary yDT transducers or affine transducers that only require multiplication with constants, this assumption is justified, as the lengths of numbers may only grow polynomially. The situation is different, though, for general affine or *msur* transducers.

Let us first consider general affine transducers. To calculate an upper bound to the occurring numbers, we first note that for each state $p$ of $A$, the basis of $\mathrm{aff}(V_p)$ as calculated by our algorithm is of the form $[\![t]\!]_M$ for a tree in $\mathrm{dom}(p)$ of depth at most $(n+1)\cdot h$ if $n$ and $h$ are the numbers of states of $M$ and $A$, respectively. Concerning the lengths of occurring numbers, we prove:

Lemma 5.1. *Assume that $M$ is an affine transducer with $n$ states where the ranks of input symbols are bounded by $m > 1$, and the absolute values of the constants occurring in right-hand sides of rules are bounded by $c \geq 1$. Then*

$$|[\![q]\!]_M(t)| \leq ((n+1)\cdot c)^{m\frac{m^N-1}{m-1}}$$

*if $N$ is the depth of $t$ (a leaf is counted to have depth 1).*

Proof. Let $R[N]$ denote the least upper bound to the absolute values of the outputs $[\![q]\!]_M(t)$ for trees $t$ of depth at most $N$. We claim that $R[N] \leq ((n+1)\cdot c)^{m\cdot\sum_{r=0}^{N-1} m^r}$ holds. The proof is by induction on the depth of $t$. Thus, assume that $t = f\,\mathbf{t}$ with $\mathbf{t} = (\mathbf{t}_1, \ldots, \mathbf{t}_k)$, $k \geq 0$, and assume that the induction hypothesis holds for the $\mathbf{t}_i$. Let $q(f(x_1, \ldots, x_k)) \to T$ be a rule of $M$. Then

$$[\![q]\!]_M(t) = [\![T]\!]_M\,\mathbf{t}.$$

Since $M$ is affine, the right-hand side $T$ can be written as a sum of monomials of the form $a \cdot q_1(x_{i_1}) \cdot \ldots q_r(x_{i_r})$, where the number of such monomials is bounded by $(n+1)^m$, $|a| \leq c$ and the number $r$ is bounded by $m$. Thus,

$$\begin{aligned} R[N] &\leq (n+1)^m \cdot c \cdot R[N-1]^m \\ &\leq (n+1)^m \cdot c \cdot ((n+1)\cdot c)^{m\cdot m\cdot\sum_{r=0}^{N-2} m^r} \\ &\leq ((n+1)\cdot c)^m \cdot ((n+1)\cdot c)^{m\cdot\sum_{r=1}^{N-1} m^r} \\ &\leq ((n+1)\cdot c)^{m\cdot\sum_{r=0}^{N-1} m^r}. \end{aligned}$$

□

Accordingly, the *bit length* $Z(N)$ of numbers occurring in $[\![t]\!]_M$ for trees of depth $N$ is bounded by $O(m^N \cdot \log((n+1)\cdot c))$, where $m > 1$ is the maximal rank of an input symbol. This means that the *bit lengths* of occurring numbers can only be bounded by an exponential in the number of states of $M$ and $A$. Still, violation of an affine invariant can be decided in *randomized* polynomial time.

Theorem 5.2. *Violation of an affine invariant of a total affine transducer relative to a* DTT *automaton is decidable in* randomized polynomial time*, i.e., there is a polynomial probabilistic algorithm that, in case of invariance, always returns* ***false****, while in case of violation of the invariant returns* ***true*** *with probability at least 0.5.*

Proof. Assume that $M$ is a total affine polynomial transducer and $A$ a DTT automaton. Assume that the affine invariant is given by $H \doteq 0$. Our goal is to decide, given a state $p$ of $A$, whether or not $H([\![t]\!]_M) = 0$ for all $t \in \mathsf{dom}(p)$. Now let $z$ denote any prime number. Then the set of integers modulo $z$, $\mathbb{Z}_z$, 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 $\mathbb{Z}_z$. The resulting algorithm allows us to decide whether the invariant holds for all inputs from $\mathsf{dom}(p)$ modulo the prime number $z$ by using polynomially many operations on numbers of length $O(\log(z))$ only. In particular, if a violation is found, then $H \doteq 0$ cannot be an invariant of $M$ for state $p$ over $\mathbb{Q}$ either.

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

Lemma 5.3. *$H \doteq 0$ is an invariant of $M$ for $p$ iff $H \doteq 0$ is an invariant of $M$ for $p$ 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, $H([\![t]\!]_M) = 0$ holds also over $\mathbb{Q}$ for all $t \in \mathsf{dom}(p)$ of depth at most $n \cdot h$. Therefore, $H \doteq 0$ must be an invariant.

Clearly, if $H \doteq 0$ is an invariant of $M$ at $p$, then $H \doteq 0$ is also an invariant of $M$ at $p$ modulo every prime number $z$. Therefore now assume that $H \doteq 0$ is not an invariant of $M$ at $p$. Let $K$ denote the set of all primes $z$ such that $H \doteq 0$ is still an invariant of $M$ at $p$ modulo $z$. By Lemma 5.3, 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., Hardy and Wright (2008)). Therefore, with probability at least 0.75, a prime randomly drawn from this range is not contained in $K$ and therefore witnesses that $H \doteq 0$ is not an invariant of $M$ at $p$. Any number randomly drawn from the given range is a prime with probability at least $4/D$. Therefore, a polynomial number of draws suffice to extract a sample that contains at least one prime number with probability at least 0.75. Since $0.75 \cdot 0.75 \geq 0.5$, the assertion of the theorem follows. □

By replacing the deterministic decision procedure for affine transducers with its randomized counterpart, we arrive at an algorithm for deciding non-equivalence of linear $y$DT transducers that runs in randomized polynomial time—but with numbers of polynomial length only. Our algorithm for deciding non-equivalence of *sur* transducers then runs in *randomized* exponential time using exponentially long numbers.

## 6 GENERAL POLYNOMIAL TRANSDUCERS

In the following, we drop the restriction that the polynomial transducers are necessarily affine or *msur*. Let $M = (Q, \Sigma, \delta, q_0)$ denote a total polynomial transducer with $Q = [n]$ for some $n \geq 1$. Then for every state $q \in Q$ and input symbol $f \in \Sigma$ of some rank $k \geq 0$, the right-hand side $\delta(q, f) = T$ is a polynomial in the terms $j(x_i), j \in [n]$. Accordingly, the $q$th component of the semantic function $[\![f]\!]$ is defined by a polynomial $\mathbf{r}_q^{(f)}$ in the variables $\mathbf{x}_{ij}, i \in [k], j \in [n]$. We remark that techniques based on affine closures are no longer appropriate. Instead, we reason about *invariants* satisfied by the vectors in $\{[\![t]\!]_M \mid t \in \mathsf{dom}(p)\}$. Let $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n)$ a vector of fresh distinct variables, indexed by the states of $M$. Let $\mathbb{Q}[\mathbf{z}]$ denote all polynomials with variables from $\mathbf{z}$ and coefficients in $\mathbb{Q}$. For a polynomial $H \in \mathbb{Q}[\mathbf{z}]$, our goal is to verify whether an equality $H \doteq 0$ holds for all vectors $[\![t]\!]_M, t \in \mathsf{dom}(p_0)$ for some state $p_0$ of the DTT $A$.

The key concept that we introduce here is the notion of an *inductive* invariant of $M$ relative to the DTT automaton $A$. As candidate invariants, we only need conjunctions of equalities $r \doteq 0$,

$r \in \mathbb{Q}[\mathbf{z}]$. Instead of referring to such conjunctions 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 $\langle S \rangle_R = \{\sum_{\kappa=1}^{k} r_\kappa \cdot s_\kappa \mid k \geq 0, r_1, \ldots, r_k \in R, s_1, \ldots, s_k \in S\}$. 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 $\mathbf{v} \in \mathbb{Q}^n$, it holds that $s(\mathbf{v}) = 0$ for all $s \in \langle S \rangle_R$ iff $s(\mathbf{v}) = 0$ for all $s \in S$.

An *inductive invariant* $\mathcal{I}$ of the total polynomial transducer $M$ relative to $A$ is a family of ideals $\mathcal{I}_p \subseteq \mathbb{Q}[\mathbf{z}], p \in P$, such that for all transitions $\rho(p, f) = (p_1, \ldots, p_k)$,

$$\mathcal{I}_p \subseteq \{r \in \mathbb{Q}[\mathbf{z}] \mid r[\mathbf{r}^{(f)}/\mathbf{z}] \in \langle \mathcal{I}_{p_1}(\mathbf{x}_1) \rangle_{\mathbb{Q}[\mathbf{x}]} + \cdots + \langle \mathcal{I}_{p_k}(\mathbf{x}_k) \rangle_{\mathbb{Q}[\mathbf{x}]}\} \tag{7}$$

holds. Thereby, we have used the following notation. For $i = 1, \ldots, k$, $\mathbf{x}_i$ is the vector of the distinct fresh variables $\mathbf{x}_{iq}, q = 1, \ldots, n$, which refers to the $i$th argument of $f$, and $\mathbf{x}$ is the concatenation of these vectors $\mathbf{x}_i$. The vector substitution $[\mathbf{v}/\mathbf{z}]$ of a vector $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ for $\mathbf{z}$ denotes the simultaneous substitution of the expressions $\mathbf{v}_q$ for the variables $\mathbf{z}_q$. 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 (7) for the transition $\rho(p, f) = (p_1, \ldots, p_k)$ 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 from $\mathbf{x}$. The constraint (7) therefore expresses that the weakest precondition of every polynomial in $\mathcal{I}_p$ can be generated from the polynomials provided by $\mathcal{I}$ for the states $p_i$—after the variables from $\mathbf{z}$ therein have been substituted with the corresponding variables from $\mathbf{x}_i$.

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

Theorem 6.1. *Assume that $\mathcal{I}$ is an inductive invariant of the total polynomial transducer $M$ relative to the* DTT *automaton $A$. Then for every state $p$ of $A$ and polynomial $r \in \mathcal{I}_p$, $r([\![t]\!]_M) = 0$ holds for all $t \in \mathrm{dom}(p)$.*

Proof. By structural induction on $t$, we prove that $r([\![t]\!]_M) = 0$ holds. Assume that $\rho(p, f) = (p_1, \ldots, p_k)$ and $t = f(\mathbf{t}_1, \ldots, \mathbf{t}_k)$, where (by induction hypothesis) for every $i = 1, \ldots, k$ and every $r \in \mathcal{I}_{p_i}$, $r([\![\mathbf{t}_i]\!]_M) = 0$ holds. Since $[\![t]\!]_{M,q} = \mathbf{r}_q^{(f)}([\![\mathbf{t}_1]\!]_M, \ldots, [\![\mathbf{t}_k]\!]_M)$, we have that

$$r([\![t]\!]_M) = r[\mathbf{r}^{(f)}/\mathbf{z}]([\![\mathbf{t}_1]\!]_M, \ldots, [\![\mathbf{t}_k]\!]_M).$$

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

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

for suitable polynomials $r_{i\mu_i} \in \mathbb{Q}[\mathbf{x}]$, where for $i = 1, \ldots, k$, $s_{i\mu_i} \in \mathcal{I}_{p_i}$. Therefore for all $\mu_i$, $s_{i\mu_i}([\![\mathbf{t}_i]\!]_M) = 0$, and thus $r([\![t]\!]_M) = r[\mathbf{r}^{(f)}/\mathbf{z}]([\![\mathbf{t}_1]\!]_M, \ldots, [\![\mathbf{t}_k]\!]_M) = 0$. □

We conclude that every inductive invariant $\mathcal{I}$ with $r \in \mathcal{I}_p$ provides us with a *certificate* that $r([\![t]\!]_M) = 0$ holds for all $t \in \mathrm{dom}(p)$. In the next step, we convince ourselves that the reverse implication also holds, i.e., for all polynomials $r$ for which $r([\![t]\!]_M) = 0$ holds for all $t \in \mathrm{dom}(p)$, an inductive invariant $\mathcal{I}$ exists with $r \in \mathcal{I}_p$. To prove this statement, we consider the family $\bar{\mathcal{I}}$ of ideals $\bar{\mathcal{I}}_p, p \in P$, where

$$\bar{\mathcal{I}}_p = \{r \in \mathbb{Q}[\mathbf{z}] \mid \forall\, t \in \mathrm{dom}(p).\ r([\![t]\!]_M) = 0\}.$$

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

THEOREM 6.2. *$\bar{\mathcal{I}}$ is an inductive invariant of the total polynomial transducer $M$ relative to the* DTT *automaton $A$.*

PROOF. For any set $V \subseteq \mathbb{Q}^n$ of vectors, let $\mathcal{I}(V)$ denote the set of polynomials $r$ over $\mathbf{z}$ that vanish on $V$, i.e., with $r(\mathbf{v}) = 0$ for all $\mathbf{v} \in V$. Let $\mathbf{x}$ denote the set of variables $\mathbf{x}_{ij}, i \in [k], j \in [n]$, where for $i \in [k]$, $\mathbf{x}_i = \{\mathbf{x}_{ij} \mid j \in [n]\}$. Let $V_i \subseteq \mathbb{Q}^n$ for $i \in [k]$. Then $\mathcal{I}(V_1 \times \cdots \times V_k)$ denotes the set of all polynomials $r \in \mathbb{Q}[\mathbf{x}]$ such that $r(v_1, \ldots, v_k) = 0$ whenever $(v_1, \ldots, v_k) \in V_1 \times \cdots \times V_k$. We claim that then

$$\mathcal{I}(V_1 \times \cdots \times V_k) = \langle \mathcal{I}(V_1)(\mathbf{x}_1) \rangle_{\mathbb{Q}[\mathbf{x}]} + \cdots + \langle \mathcal{I}(V_k)(\mathbf{x}_k) \rangle_{\mathbb{Q}[\mathbf{x}]} \tag{8}$$

holds. This means that the set of polynomials that vanish on the Cartesian product $V_1 \times \cdots \times V_k$ is exactly given by the ideal of $\mathbb{Q}[\mathbf{x}]$ that is generated by the polynomials in $\mathbb{Q}[\mathbf{x}_i]$ that vanish on the set $V_i$ $(i = 1, \ldots, k)$.

We remark that the ideal of $\mathbb{Q}[\mathbf{x}]$ generated from $\mathcal{I}(V_i)(\mathbf{x}_i)$ is exactly given by $\mathcal{I}(\top^{i-1} \times V_i \times \top^{k-i})$, where $\top = \mathbb{Q}^n$. Accordingly, equality (8) can be rewritten to:

$$\mathcal{I}(V_1 \times \cdots \times V_k) = \sum_{i=1}^{k} \mathcal{I}(\top^{i-1} \times V_i \times \top^{k-i}).$$

Thus, equality (8) is a consequence of the following lemma. Although formulated for $\mathbb{Q}$, the lemma holds (with the same proof) for any field.

LEMMA 6.3. *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 Becker and Weispfenning (1998)). Let $\mathbf{x} = \{\mathbf{x}_1 \ldots, \mathbf{x}_{m_1+m_2}\}$ be 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 $\mathbb{Q}[\mathbf{x}_1, \ldots, \mathbf{x}_{m_1}]$, we have $G_1 \subset \mathbb{Q}[\mathbf{x}_1, \ldots, \mathbf{x}_{m_1}]$, and also $G_2 \subset \mathbb{Q}[\mathbf{x}_{m_1+1}, \ldots, \mathbf{x}_{m_1+m_2}]$. It follows by Buchberger's criterion (see Becker and Weispfenning (1998, 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 $g := \mathrm{nf}(f)$, which (by definition) has no monomial that is divisible by the leading monomial of any polynomial in $G_1 \cup G_2$ and that 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 $g := \mathrm{nf}(f) \neq 0$. 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 $\mathbf{x}_1, \ldots, \mathbf{x}_{m_1}$, $g$ has a nonzero term $c\mathbf{x}_1^{e_1} \cdots \mathbf{x}_{m_1}^{e_{m_1}}$ with $c \in \mathbb{Q}[\mathbf{x}_{m_1+1}, \ldots, \mathbf{x}_{m_1+m_2}]$. 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 $g_{\mathbf{w}} := g(\mathbf{x}_1, \ldots, \mathbf{x}_{m_1}, \mathbf{w}) \in \mathbb{Q}[\mathbf{x}_1, \ldots, \mathbf{x}_{m_1}]$. 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 $V_p = \{[\![t]\!]_M \mid t \in \mathrm{dom}(p)\}$. Then the ideal $\bar{\mathcal{I}}_p$ is exactly given by $\bar{\mathcal{I}}_p = \mathcal{I}(V_p)$. Assume that $r \in \bar{\mathcal{I}}_p$ and $\rho(p, f) = (p_1, \ldots, p_k)$ holds. Then for all tuples of trees $(\mathbf{t}_1, \ldots, \mathbf{t}_k)$ with $\mathbf{t}_i \in \mathrm{dom}(p_i)$ $(i = 1, \ldots, k)$, $r([\![f(\mathbf{t}_1, \ldots, \mathbf{t}_k)]\!]_M) = 0$ holds. Therefore,

$$r[\mathbf{r}^{(f)}/\mathbf{z}](\mathbf{v}_1, \ldots, \mathbf{v}_k) = 0$$

holds for all $(\mathbf{v}_1, \ldots, \mathbf{v}_k) \in V_{p_1} \times \cdots \times V_{p_k}$. Accordingly,

$$\begin{aligned} r[\mathbf{r}^{(f)}/\mathbf{z}] &\in \mathcal{I}(V_{p_1} \times \cdots \times V_{p_k}) \\ &= \langle \mathcal{I}(V_{p_1})(\mathbf{x}_1) \rangle_{\mathbb{Q}[\mathbf{x}]} + \cdots + \langle \mathcal{I}(V_{p_k})(\mathbf{x}_k) \rangle_{\mathbb{Q}[\mathbf{x}]} \qquad \text{by (8)} \\ &= \langle \bar{\mathcal{I}}_{p_1}(\mathbf{x}_1) \rangle_{\mathbb{Q}[\mathbf{x}]} + \cdots + \langle \bar{\mathcal{I}}_{p_k}(\mathbf{x}_k) \rangle_{\mathbb{Q}[\mathbf{x}]}. \end{aligned}$$

As a consequence, $\bar{\mathcal{I}}$ satisfies the constraints (7) and therefore is an inductive invariant of $M$ relative to $A$. □

The inductive invariant $\bar{\mathcal{I}}$ is the *largest* invariant and, accordingly, the *greatest* fixpoint of the inclusions (7). Since the set of polynomial ideals in $\mathbb{Q}[\mathbf{x}]$ has unbounded *decreasing* chains, it is unclear whether $\bar{\mathcal{I}}$ 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 $\mathbf{r}_q^{(f)}$ can be considered as a constant or a polynomial again over the variables $\mathbf{z}$. Thus, the constraints in Equation (7) to be satisfied by an inductive invariant $\mathcal{I}$, can be simplified to:

$$\mathcal{I}_p \subseteq \{r \in \mathbb{Q}[\mathbf{z}] \mid r(\mathbf{r}^{(b)}) = 0\} \qquad \text{if } \rho(p, b) = (), \tag{9}$$

$$\mathcal{I}_p \subseteq \{r \in \mathbb{Q}[\mathbf{z}] \mid r[\mathbf{r}^{(f)}/\mathbf{z}] \in \mathcal{I}_{p_1}\} \text{ if } \rho(p, f) = p_1. \tag{10}$$

According to the second constraint, the invariant $r \doteq 0$ to hold for all tress in $\mathrm{dom}(p)$ is transformed by the monadic input symbol $f$ into the invariant $r[\mathbf{r}^{(f)}/\mathbf{z}] \doteq 0$ to hold for all trees in $\mathrm{dom}(p_1)$.

For a state $p_0$ of the DTT automaton $A$ and a polynomial $H \in \mathbb{Q}[\mathbf{z}]$ to hold for all $t \in \mathrm{dom}(p_0)$, the propagation of these invariants (9) and (10) can be expressed by the following system of constraints:

$$\mathcal{I}_{p_0} \supseteq \langle H \rangle, \tag{11}$$

$$\mathcal{I}_{p_1} \supseteq \{r[\mathbf{r}^{(f)}/\mathbf{z}] \mid r \in \mathcal{I}_p\} \qquad \text{if } \rho(p, f) = p_1. \tag{12}$$

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 $s_1, \ldots, s_u$ such that $J = \langle s_1, \ldots, s_u \rangle$ and, likewise, that each *increasing* chain $J_0 \subseteq J_1 \subseteq \ldots$ of ideals is ultimately stable. Therefore, the system (11) and (12) has a *least solution* that is attained after finitely many fixpoint iterations. We claim:

LEMMA 6.4. *Assume that $\mathcal{I}$ is the least solution of the system of constraints (11)–(12). Then $\mathcal{I}$ is an inductive invariant iff for each transition $\rho(p, b) = ()$ of $A$, $r(\mathbf{r}^{(b)}) = 0$ for all $r \in \mathcal{I}_p$. In this case, it is the* least *inductive invariant $\mathcal{I}'$ with $H \in \mathcal{I}'_{p_0}$. Otherwise, no inductive invariant with this property exists.*

PROOF. We have that $\mathcal{I}$ is a solution of Equation (12) iff $\mathcal{I}$ satisfies the constraints (10). Moreover, $r(\mathbf{r}^{(b)}) = 0$ for all $r \in \mathcal{I}_p$ holds for all $\rho(p, b) = ()$ of $A$ iff $\mathcal{I}$ satisfies the constraints (9). Therefore, $\mathcal{I}$ is an inductive invariant iff these two assumptions are met.

Now assume that $\mathcal{I}$ is the least solution of Equations (11) and (12). If it passes all tests on the transitions $\rho(p, b)$, then it therefore must be the least inductive invariant $\mathcal{I}'$ with $H \in \mathcal{I}'_{p_0}$. If it does not pass all tests, then there cannot be any inductive invariant $\mathcal{I}'$ with $H \in \mathcal{I}'_{p_0}$. This can be seen as follows. Assume for a contradiction that there is an inductive invariant $\mathcal{I}'$ with $H \in \mathcal{I}'_{p_0}$. Since $\mathcal{I}'$ satisfies the constraints in Equation (10), $\mathcal{I}'$ is also a solution of Equation (12). Therefore, $\mathcal{I}_p \subseteq \mathcal{I}'_p$ for all states $p$ of $A$. Now, since already $\mathcal{I}$ does not pass all tests on transitions $\rho(p, b) = ()$, then $\mathcal{I}'$ cannot pass all these tests either. But then $\mathcal{I}'$ does not satisfy the constraints (9) and therefore fails to be an inductive invariant—contradiction. □

Since the least solution of system (11) nad (12) can effectively be computed and the tests required by Lemma 6.4 can also be effectively performed, we obtain:

Theorem 6.5. *Assume that $M$ is a total polynomial transducer with a monadic input alphabet and $A$ a* DTT *automaton. Then it is decidable for a polynomial $H$ and a state $p_0$ of $A$, whether or not $H \doteq 0$ is an invariant at $p_0$.*

Proof. By Lemma 6.4, $H([\![t]\!]_M) = 0$ for all $t \in \mathrm{dom}(p_0)$ holds iff the least solution $\mathcal{I}$ of the constraint system (11) and (12) satisfies Equation (9). By Hilbert's basis theorem, Kleene iteration for system (11) and (12) eventually terminates. Therefore, $\mathcal{I}$ can be computed. To check condition (9), it suffices to check for each state $p$ and each nullary input symbol $b$ with $\rho(p, b) = ()$, that $r(\mathbf{r}^{(b)}) = 0$ holds for each polynomial $r \in G$ for some generating system $G$ of the ideal $\mathcal{I}_p$. Therefore, the statement of the theorem follows. □

In case of non-monadic input symbols, it is no longer clear whether computing the greatest solution of constraint system (7) 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 (7) can be effectively decided for any given candidate invariant $\mathcal{I}$, the set of *all* inductive invariants of $M$ relative to $A$ is recursively enumerable. Accordingly, if a polynomial equality $H \doteq 0$ holds for all vectors $[\![t]\!]_M, t \in \mathrm{dom}(p_0)$, then an inductive invariant certifying this fact will eventually be found in the enumeration. In this way, we obtain a *semi*-decision procedure for the validity of the polynomial equality for $M$ relative to $A$. The fact, on the other hand, that $H \doteq 0$ does not hold for all $[\![t]\!]_M, t \in \mathrm{dom}(p_0)$, is witnessed by a specific tree $t \in \mathrm{dom}(p_0)$ for which $H([\![t]\!]_M) \neq 0$. Since $\mathrm{dom}(p_0)$ is recursively enumerable as well, we obtain another *semi*-decision procedure for the invalidity of a polynomial equality $H \doteq 0$ for $M$ at state $p_0$ of $A$. Putting these two *semi*-decision procedures together, we obtain:

Theorem 6.6. *Assume that $M$ is a total polynomial transducer and $A$ a* DTT *automaton. Then it is decidable whether or not a given polynomial equality $H \doteq 0$ is an invariant of $M$ at some state $p_0$ of $A$.*

Lemma 3.5 together with Theorems 6.5 and 6.6 provide us with decision procedures for (possibly partial) *y*DT transducers with output in a free monoid or free group. We obtain our main technical results:

Theorem 6.7. *Equivalence is decidable for*

*(1) y*DT *transducers with output in a free monoid;*
*(2) y*DT *transducers with output in a free group.*

One particular subcase of Theorem 6.7 is when the input alphabet is monadic. This case, for which we have provided a dedicated algorithm in Theorem 6.5, is known to be equivalent to the sequence

equivalence problem of HDT0L systems. Recently, this problem has been shown to be Ackermann-complete (Benedikt et al. 2017), where the upper bound is attained by the algorithm we have presented. The equivalence problem for $y$DT transducers with non-monadic input alphabets, as shown to be decidable in Theorem 6.7, seems to be significantly more difficult.

## 7 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 difficult 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

$$\bar{\mathcal{I}}_p^{(d)} = \bigcap \{ [\![f]\!]^\sharp(\bar{\mathcal{I}}_{p_1}^{(d-1)}, \ldots, \bar{\mathcal{I}}_{p_k}^{(d-1)}) \mid \rho(p,f) = (p_1, \ldots, p_k) \}, \text{where} \tag{13}$$

$$\begin{aligned} [\![f]\!]^\sharp(I_1, \ldots, I_k) &= \{ r \in \mathbb{Q}[\mathbf{z}] \mid r[\mathbf{r}^{(f)}/\mathbf{z}] \in \langle I_1(\mathbf{x}_1) \rangle_{\mathbb{Q}[\mathbf{x}]} + \cdots + \langle I_k(\mathbf{x}_k) \rangle_{\mathbb{Q}[\mathbf{x}]} \} \\ &= (\langle \mathbf{z}_q - \mathbf{r}_q^{(f)} \mid q = 1, \ldots, n \rangle_{\mathbb{Q}[\mathbf{x} \cup \mathbf{z}]} \\ &\qquad + \langle I_1(\mathbf{x}_1) \rangle_{\mathbb{Q}[\mathbf{x} \cup \mathbf{z}]} + \cdots + \langle I_k(\mathbf{x}_k) \rangle_{\mathbb{Q}[\mathbf{x} \cup \mathbf{z}]}) \cap \mathbb{Q}[\mathbf{z}]. \end{aligned} \tag{14}$$

To prove that the effect of $[\![f]\!]^\sharp$ can be effectively computed, its definition is re-written into the expression (14). The latter expression is built up by means of effective operations on polynomial ideals. Let us briefly verify that the corresponding equality between ideals holds. First, we claim that the ideal $\langle \mathbf{z}_q - \mathbf{r}_q^{(f)} \mid q = 1, \ldots, n \rangle_{\mathbb{Q}[\mathbf{x} \cup \mathbf{z}]}$ contains all differences $r - r[\mathbf{r}^{(f)}/\mathbf{z}]$, $r \in \mathbb{Q}[\mathbf{z}]$. Clearly, the claim is true for $r \in \mathbb{Q}$. Now assume it holds for all polynomials up to degree $d-1$. Consider a monomial $m = \mathbf{z}_q \cdot m'$ in $\mathbb{Q}[\mathbf{z}]$ of degree $d$. Then

$$\begin{aligned} m - m[\mathbf{r}^{(f)}/\mathbf{z}] &= \mathbf{z}_q \cdot (m' - m'[\mathbf{r}^{(f)}/\mathbf{z}]) + m'[r^{(f)}/\mathbf{z}] \cdot \left( \mathbf{z}_q - \mathbf{r}_q^{(f)} \right) \\ &\in \left\langle \mathbf{z}_q - \mathbf{r}_q^{(f)} \middle| q = 1, \ldots, n \right\rangle_{\mathbb{Q}[\mathbf{x} \cup \mathbf{z}]}. \end{aligned}$$

Since with all monomials of a polynomial $r$, also $r$ is contained in the ideal, the claim follows.

As a consequence, every polynomial $r \in \mathbb{Q}[\mathbf{z}]$ with $r[\mathbf{r}^{(f)}/\mathbf{z}] \in \langle I_1(\mathbf{x}_1) \rangle_{\mathbb{Q}[\mathbf{x}]} + \cdots + \langle I_k(\mathbf{x}_k) \rangle_{\mathbb{Q}[\mathbf{x}]}$ is contained in the ideal expression of line (14).

For the reverse inclusion, consider any polynomial $r \in \mathbb{Q}[\mathbf{z}]$ contained in the expression (14). Then $r = g + g'$, where $g \in \langle \mathbf{z}_q - \mathbf{r}_q^{(f)} \mid q = 1, \ldots, n \rangle_{\mathbb{Q}[\mathbf{x} \cup \mathbf{z}]}$ and $g' \in \langle I_1(\mathbf{x}_1) \rangle_{\mathbb{Q}[\mathbf{x} \cup \mathbf{z}]} + \cdots + \langle I_k(\mathbf{x}_k) \rangle_{\mathbb{Q}[\mathbf{x} \cup \mathbf{z}]}$. Accordingly, $r[\mathbf{r}^{(f)}/\mathbf{z}] = g[\mathbf{r}^{(f)}/\mathbf{z}] + g'[\mathbf{r}^{(f)}/\mathbf{z}]$. Thereby, $g[\mathbf{r}^{(f)}/\mathbf{z}] = 0$. The polynomial $g'[\mathbf{r}^{(f)}/\mathbf{z}]$, on the other hand, is contained in $\langle I_1(\mathbf{x}_1) \rangle_{\mathbb{Q}[\mathbf{x}]} + \cdots + \langle I_k(\mathbf{x}_k) \rangle_{\mathbb{Q}[\mathbf{x}]}$—which we wanted to prove.

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

LEMMA 7.1.

*(1) For every $d \geq 0$, $\bar{\mathcal{I}}_p^{(d)} = \{ r \in \mathbb{Q}[\mathbf{z}] \mid \forall\, t \in \mathrm{dom}_d(p).\ r([\![t]\!]_M) = 0 \}$;*

*(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 6.1 and 6.2. Statement 2 follows from statement 1 as the intersection to the right consists of all

polynomials $r \in \mathbb{Q}[\mathbf{z}]$ such that $r([\![t]\!]_M) = 0$ for all $t \in \mathrm{dom}(p)$—which is precisely is the definition of $\bar{\mathcal{I}}_p$. □

From statement 1 of Lemma 7.1, we conclude that there is a counter-example to the validity of $H \doteq 0$ at some state $p_0$ of $A$ iff $H \notin \bar{\mathcal{I}}_{p_0}^{(d)}$. Thus, the semi-algorithm for falsifying the invariant $H \doteq 0$ at $p_0$ can be formulated as:

**for** ($\mathbf{d} \geq \mathbf{0}$) {
  determine $p \mapsto \bar{\mathcal{I}}_p^{(d)}$;
  **if** ($H \notin \bar{\mathcal{I}}_{p_0}^{(d)}$) **return** "not valid";
}

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 $I \subseteq \mathbb{Q}[\mathbf{z}]$, let us denote by $I_d$ the intersection $I \cap \mathbb{Q}_d[\mathbf{z}]$. 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 Colón (2007).

The intersection $I_d$ can be effectively computed as follows. Given a Gröbner basis $G$ for $I$ 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 $I_d$ considered as a vector space is given by the set of polynomials $g \cdot m$, where $g \in G_d$ and $m$ is a monomial such that $\deg(m) + \deg(g) \leq d$. Moreover, the ideal $\langle G_d \rangle_{\mathbb{Q}[\mathbf{z}]}$ is the smallest ideal $I' \subseteq I$ such that $I'_d = I_d$.

Let $\alpha_d$ denote the function that maps each ideal $I$ 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

$$\mathcal{I}_p \subseteq \alpha_d([\![f]\!]^\sharp(\mathcal{I}_{p_1}, \ldots, \mathcal{I}_{p_k})) \tag{15}$$

for every transition $\rho(p, f) = (p_1, \ldots, p_k)$. 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 7.2.

*(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. 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 (15). Accordingly, $\bar{\mathcal{I}}_p \subseteq \mathcal{I}_{p,\bar{d}}$. Hence, $\bar{\mathcal{I}}_p = \mathcal{I}_{p,\bar{d}}$, and the third statement follows. □

In light of the argument for proving statement 3 of Lemma 7.2, we observe that for any $d$, $p \mapsto \mathcal{I}_{p,d}$ represents the largest inductive invariant that can be represented by a Gröbner basis

with maximal total degree $d$. Given Lemma 7.2, the semi-algorithm for verifying the invariant $H \doteq 0$ at $p_0$ therefore looks as follows:

**for** $(\mathbf{d} \geq \mathbf{1})$ {
  determine $p \mapsto \mathcal{I}_{p,d}$;
  **if** $(H \in \mathcal{I}_{p_0,d})$ **return** "valid";
  }

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.

## 8 APPLICATION TO OTHER TYPES OF TRANSDUCERS

In this section, we discuss some further models and extensions where the main theorem provides us with new decidability results.

Tree transducers can be equipped with regular look-ahead. For top-down transducers this increases the expressive power, and the resulting class of translations for instance includes the translations of bottom-up tree transducers, see Engelfriet (1977). A top-down tree-to-string transducer *with regular look-ahead* ($y\text{DT}^R$ transducer) $M$ consists of an ordinary transducer together with a deterministic bottom-up tree automaton $B$ (called the "look-ahead automaotn of $M$). For a $y\text{DT}^R$ transducer, a rule is of the form

$$q(f(x_1 : p_1, \ldots, x_k : p_k)) \to T, \tag{16}$$

where $T$ is as for ordinary $y\text{DT}$ 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 $i \in \{1, \ldots, k\}$. Our result extends to look-ahead, using the technique as in Engelfriet et al. (2009): One changes the input alphabet to indicate (possibly incorrect) runs of the look-ahead automata $B_1, B_2$ of the two given $y\text{DT}^R$ transducers and changes the transducers to check correctness of runs. The input ranked alphabet $\Sigma$ is replaced with pairs $\langle \tau_1, \tau_2 \rangle$ of transitions of $B_1, B_2$. If $\tau_i$ is the transition $(p_1^i, \ldots, p_k^i, f) \mapsto p^i$ (for $i = 1, 2$), then the rule (16) is simulated by

$$q(\langle \tau_1, \tau_2 \rangle(x_1, \ldots, x_k)) \to T.$$

A top-down deterministic automaton $A$ can then check whether the transitions in the input tree indeed form accepting runs of the automata $B_1, B_2$ on some tree $t \in \mathcal{T}_\Sigma$ for which the translation of the $y\text{DT}^R$ transducer is defined. In this way, we have reduced the equivalence problem for $y\text{DT}^R$ transducers to that of ordinary $y\text{DT}$ transducers.

In a *macro tree transducer* (Engelfriet and Vogler 1985; Engelfriet and Maneth 1999), states are ranked, and a state $q$ of rank $l + 1$ has associated with it *parameters* $y_1, \ldots, y_l$ of type output tree. The right-hand side of a rule for state $q$ is a tree over states, output symbols, and parameters that may appear at leaves only; the first child of a state in a right-hand side must be a variable $x_i$ that appears in the left-hand side of the rule. As a result of Engelfriet and Maneth (2002), the class of translations realized by $y\text{DT}^R$ transducers is equal to a class realized by particular macro tree-to-string transducers with look-ahead (for short $y\text{DMT}^R$ transducers). The particular restriction requires that the transducer uses each parameter in a rule at most once. In fact, from the results of Engelfriet and Maneth (1999) we can state the result in terms of $y\text{DMT}^R$ transducers that are *finite-copying in the parameters* ($y\text{DMT}^R_{\text{fcp}}$ transducers). A $y\text{DMT}^R$ 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 $[\![q]\!](s)$ is $\leq k$.

Corollary 8.1. *Equivalence of $y\text{DT}^R$ transducers as well as of $y\text{DMT}^R_{\text{fcp}}$ transducers is decidable.*

In Alur and D'Antoni (2017), deterministic *streaming tree transducers* have been introduced. In the variant for ranked trees, these consist of bottom-up deterministic automata that, while processing the input tree, may update the string values maintained in a finite set of variables. Thereby, the new values at a node are constructed from the variables at the child nodes by concatenation where each variable is used at most once (and after use reset to $\epsilon$). Viewed as a tree-to-string transducer, this model coincides with $y$DT transducers extended with regular look-ahead—subject to the *single-use* restriction. Corollary 4.2 together with the simulation of $y\text{DT}^R$ transducers by means of $y$DT transducers as sketched for Corollary 8.1 provides us with:

Corollary 8.2. *Equivalence of deterministic streaming tree transducers on ranked trees can be decided in* DExpTime *(w.r.t. the uniform cost measure).*

A variation of top-down transducers that has been considered in the context of XML is 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. XML document trees are naturally modeled by unranked trees.

There are several models of top-down tree transducers for unranked trees. In Perst and Seidl (2004), 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 $y$DT transducer and are of the form $q(a(x_1, x_2)) \to T$, where $T$ is a string as in a $y$DT 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 a direct instance of the equivalence problem for $y$DT transducers.

Corollary 8.3. *Equivalence is decidable for deterministic forest transducers with string outputs.*

Another, much earlier model of unranked top-down tree transducers is the *unranked top-down tree transducer* (Maneth and Neven 1999). 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 $k$ children, then a word of length $k$ from $L$ is chosen 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 $k$. Clearly, the translation realized by a deterministic such transducer is a function. As an example, consider the unranked top-down tree transducer with the following rules:

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

where $L$ is the regular language $q^*$ consisting 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 non-deterministic transducers. Thus,

given a deterministic unranked top-down tree transducer $M$, we can construct an equivalent non-deterministic forest transducer $N$. It follows from the explanation above that we may consider $N$ as a non-deterministic top-down tree-to-string transducer.

By an old result (Engelfriet 1978), for any functional DT transducer, an equivalent $\text{DT}^{\text{R}}$ 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 $y\text{DT}$ transducer, one can construct an equivalent $y\text{DT}^{\text{R}}$ transducer. Thus, we can construct for $N$ an equivalent $y\text{DT}^{\text{R}}$ transducer, for which equivalence is decidable by Corollary 8.1.

Corollary 8.4. *Equivalence is decidable for deterministic unranked top-down tree transducers.*

Let us finally discuss the extension of our result to $y$DMT transducers. $y$DMT transducers extend $y$DT transducers with *parameters*, say, $y_1, \ldots, y_l$, in which output values can be accumulated.

*Example 8.5.* Consider the $y$DMT transducer with set $Q = \{q_0, q\}$ of states and initial state $q_0$ and the following transition rules:

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

where the output is considered as a string. This $y$DMT transducer realizes a translation $\tau$ that 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:

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

It is still open whether or not the equivalence problem for $y$DMT transducers is decidable. It turns out, however, that equivalence is at least decidable for *unary* $y$DMT transducers. The reason is that the *semantics* $[\![t]\!]_M$ of an input tree $t$ w.r.t. a given unary $y$DMT transducer $M$ is a vector where each component $[\![t]\!]_{M,q}$ is an *affine function* of the values $\mathbf{y}_1, \ldots, \mathbf{y}_l$ of the parameters and thus can be represented by $l + 1$ numbers. Moreover, for each input symbol $f$ of rank $k \geq 0$ and every state $q$, the coefficients $\mathbf{r}_0^{(f)}, \ldots, \mathbf{r}_l^{(f)}$ of the affine function produced by $[\![f]\!](\mathbf{x}_1, \ldots, \mathbf{x}_k)$ for state $q$ are all polynomials in the coefficients $\mathbf{x}_{i\lambda}, 1 \leq i \leq k, 0 \leq \lambda \leq l$.

*Example 8.6.* Consider the $y$DMT transducer $M$ from Example 8.5, which we extend to a total $y$DMT transducer by adding the rules

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

Then, we obtain:

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

Thus, $M$ can be simulated by a polynomial transducer where the number of states increases by a factor of $l + 1$. Accordingly, we obtain:

Corollary 8.7. *Equivalence is decidable for $y$DMT transducers with unary output alphabet.*

Moreover, by the constructions from Section 3, the results stated in Corollaries 8.1, 8.3, and 8.4 also hold when output is not considered in the free monoid but in the free group.

## 9 CONCLUSION

We introduced the concept of polynomial transducers and showed for these that polynomial invariants for top-down regular subsets of inputs are decidable. Via a translation into polynomial transducers, we deduced that equivalence of deterministic top-down tree-to-string transducers ($y$DT transducers) is decidable—even when the output is interpreted over the free group. The key concept that helped us to arrive at a decision procedure in the general case is *inductive* invariants expressed as polynomial ideals, certifying invariants. While such invariants can be automatically inferred for monadic input alphabets, we were less explicit for non-monadic input alphabets. For non-monadic input alphabets, we proved that *every* polynomial invariant can be certified by means of inductive invariants. Since enumerating all inductive invariants is rather impractical, we presented a more explicit method that allows to systematically construct the best inductive invariant up to a given maximal degree. Together with an explicit enumeration of potential counterexamples, decidability of polynomial invariants of polynomial transducers for top-down regular sets follows.

We also considered important subclasses of $y$DT transducers and the corresponding subclasses of polynomial transducers. W.r.t. a uniform cost measure, we showed that equivalence is decidable in polynomial time for linear $y$DT transducers and in deterministic exponential time for *sur* $y$DT transducers. The latter class is essentially the class of *streaming* tree-to-string transducers for ranked trees as considered by Alur and D'Antoni (2017). Again, our decision procedures can also be applied to the case when output is no longer considered to be in a free monoid but in a free group.

Still, the equivalence problem for $y$DMT transducers (as stated in Engelfriet (1980)) remains open. By a translation into polynomial transducers, we were able to deal with $y$DMT transducers when the output alphabet is *unary*. It remains unclear, however, whether and how the approach can be extended to arbitrary output alphabets. Even the equivalence problem for DMT transducers with tree output remains open.

## ACKNOWLEDGMENT

We thank the anonymous referees for careful reading and for many helpful comments. In particular, the suggestion to introduce *polynomial* transducers is theirs.

## REFERENCES

Chritiene Aarts, Roland Carl Backhouse, Eerke A. Boiten, Henk Doornbos, Netty van Gasteren, Rik van Geldrop, Paul F. Hoogendijk, Ed Voermans, and Jaap van der Woude. 1995. Fixed-point calculus. *Inf. Process. Lett.* 53, 3 (1995), 131–136.

Rajeev Alur and Loris D'Antoni. 2017. Streaming tree transducers. *J. ACM* 64, 5 (2017), 31:1–31:55. DOI: http://dx.doi.org/10.1145/3092842

K. R. Apt and G. D. Plotkin. 1986. Countable nondeterminism and random assignment. *J. ACM* 33, 4 (1986), 724–767.

Thomas Becker and Volker Weispfenning. 1998. *Gröbner Bases. A Computational Approach to Commutative Algebra*, 2nd ed. Springer-Verlag, Berlin.

Michael Benedikt, Timothy Duff, Aditya Sharad, and James Worrell. 2017. Polynomial automata: Zeroness and applications. In *LICS*. 1–12.

Scott Boag, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie, and Jérôme Siméon. 2010. *XQuery 1.0: An XML Query Language (Second Edition)*. Technical Report. W3C Recommendation.

Adrien Boiret and Raphaela Palenta. 2016. Deciding equivalence of linear tree-to-word transducers in polynomial time. In *DLT*. 355–367.

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

Michael Colón. 2007. Polynomial approximations of the relational semantics of imperative programs. *Sci. Comput. Program.* 64, 1 (2007), 76–96.

K. Culik II and J. Karhumäki. 1986. A new proof for the D0L sequence equivalence problem and its implications. In *The Book of L*, G. Rozenberg and A. Salomaa (Eds.). Springer, Berlin, 63–74.

Joost Engelfriet. 1977. Top-down tree transducers with regular look-ahead. *Math. Syst. Theory* 10, 1 (1977), 289–303.
Joost Engelfriet. 1978. On tree transducers for partial functions. *Inf. Process. Lett.* 7, 4 (1978), 170–172.
Joost Engelfriet. 1980. Some open questions and recent results on tree transducers and tree languages. In *Formal Language Theory: Perspectives and Open Problems*, R. V. Book (ed.). Academic Press, New York, 241–286.
Joost Engelfriet and S. Maneth. 1999. Macro tree transducers, attribute grammars, and MSO definable tree translations. *Inf. Comput.* 154, 1 (1999), 34–91.
Joost Engelfriet and S. Maneth. 2002. Output string languages of compositions of deterministic macro tree transducers. *J. Comput. Syst. Sci.* 64, 2 (2002), 350–395.
Joost Engelfriet and S. Maneth. 2003a. A comparison of pebble tree transducers with macro tree transducers. *Acta Inf.* 39, 9 (2003), 613–698.
Joost Engelfriet and S. Maneth. 2003b. Macro tree translations of linear size increase are MSO definable. *SIAM J. Comput.* 32, 4 (2003), 950–1006.
Joost Engelfriet and S. Maneth. 2006. The equivalence problem for deterministic MSO tree transducers is decidable. *Inform. Proc. Letters* 100, 5 (2006), 206–212.
Joost Engelfriet, Sebastian Maneth, and Helmut Seidl. 2009. Deciding equivalence of top-down XML transformations in polynomial time. *J. Comput. Syst. Sci.* 75, 5 (2009), 271–286.
Joost Engelfriet and H. Vogler. 1985. Macro tree transducers. *J. Comp. Syst. Sci.* 31, 1 (1985), 71–146.
Z. Ésik. 1980. Decidability results concerning tree transducers I. *Acta Cybernet.* 5, 1 (1980), 1–20.
Z. Fülöp and H. Vogler. 1998. *Syntax-Directed Semantics; Formal Models Based on Tree Transducers*. Springer-Verlag.
Ferenc Gécseg and Magnus Steinby. 1997. Tree languages. In *Handbook of Formal Languages, Vol. 3*, Grzegorz Rozenberg and Arto Salomaa (Eds.). 1–68.
T. V. Griffiths. 1968. The unsolvability of the equivalence problem for lambda-free nondeterministic generalized machines. *J. ACM* 15, 3 (1968), 409–413.
S. Hakuta, S. Maneth, K. Nakano, and H. Iwasaki. 2014. XQuery streaming by forest transducers. In *ICDE*. 952–963.
G. H. Hardy and E. M. Wright. 2008. *An Introduction to the Theory of Numbers (sixth ed.)*. Oxford University Press, Oxford.
J. Honkala. 2000. A short solution for the HDT0L sequence equivalence problem. *Theor. Comput. Sci.* 244, 1–2 (2000), 267–270.
Simon Holm Jensen, Magnus Madsen, and Anders Møller. 2011. Modeling the HTML DOM and browser API in static analysis of javascript web applications. In *SIGSOFT FSE*. 59–69.
Ralf Küsters and Thomas Wilke. 2007. Transducer-based analysis of cryptographic protocols. *Inf. Comput.* 205, 12 (2007), 1741–1776.
G. Laurence, A. Lemay, J. Niehren, S. Staworko, and M. Tommasi. 2011. Normalization of sequential top-down tree-to-word transducers. In *LATA*. 354–365.
G. Laurence, A. Lemay, J. Niehren, S. Staworko, and M. Tommasi. 2014. Learning sequential tree-to-word transducers. In *LATA*. 490–502.
Aurélien Lemay, Sebastian Maneth, and Joachim Niehren. 2010. A learning algorithm for top-down XML transformations. In *PODS*. 285–296.
A.A. Letichevsky and M.S. Lvov. 1993. Discovery of invariant equalities in programs over data fields. *Applic. Alg. Eng. Commun. Comput.* 4, 4 (1993), 21–29.
Yang Liu, Yun Huang, Qun Liu, and Shouxun Lin. 2007. Forest-to-string statistical translation rules. In *ACL*, Vol. 45. 704.
Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *ACL*. 609–616.
Clara Löh. 2015. *Geometric Group Theory, an Introduction*. Technical Report. Fakultät für Mathematik, Universität Regensburg, Germany.
Andreas Maletti, Jonathan Graehl, Mark Hopkins, and Kevin Knight. 2009. The power of extended top-down tree transducers. *SIAM J. Comput.* 39, 2 (2009), 410–430.
S. Maneth. 2003. *Models of Tree Translation*. Ph.D. thesis. University of Leiden.
Sebastian Maneth. 2015. A survey on decidable equivalence problems for tree transducers. *Int. J. Found. Comput. Sci.* 26, 8 (2015), 1069–1100.
S. Maneth, A. Berlea, T. Perst, and H. Seidl. 2005. XML type checking with macro tree transducers. In *PODS*. 283–294.
S. Maneth and F. Neven. 1999. Structured document transformations based on XSL. In *DBPL*. 80–98.
S. Maneth, T. Perst, and H. Seidl. 2007. Exact XML type checking in polynomial time. In *ICDT*. 254–268.
Simon Marlow and Philip Wadler. 1993. Deforestation for higher-order functions. In *Functional Programming*. 154–165.
K. Matsuda, K. Inaba, and K. Nakano. 2012. Polynomial-time inverse computation for accumulative functions with multiple data traversals. In *PEPM*. 5–14.
Markus Müller-Olm and Helmut Seidl. 2004a. Computing polynomial program invariants. *Inf. Process. Lett.* 91, 5 (2004), 233–244.

Markus Müller-Olm and Helmut Seidl. 2004b. A note on karr's algorithm. In *ICALP*. 1016–1028.
T. Perst and H. Seidl. 2004. Macro forest transducers. *Inf. Process. Lett.* 89 (2004), 141–149.
W. Plandowski. 1994. Testing equivalence of morphisms on context-free languages. In *ESA*. 460–470.
W. C. Rounds. 1970. Mappings and grammars on trees. *Math. Syst. Theory* 4, 3 (1970), 257–287.
K. Ruohonen. 1986. Equivalence problems for regular sets of word morphisms. In *The Book of L*, G. Rozenberg and A. Salomaa (Eds.). Springer, Berlin, 393–401.
Helmut Seidl. 1990. Deciding equivalence of finite tree automata. *SIAM J. Comput.* 19, 3 (1990), 424–437.
Helmut Seidl, Sebastian Maneth, and Gregor Kemper. 2015. Equivalence of deterministic top-down tree-to-string transducers is decidable. In *FOCS*. 943–962.
S. Staworko, G. Laurence, A. Lemay, and J. Niehren. 2009. Equivalence of deterministic nested word to word transducers. In *FCT*. 310–322.
J. W. Thatcher. 1970. Generalized sequential machine maps. *J. Comput. Syst. Sci.* 4, 4 (1970), 339–367.
Janis Voigtländer. 2005. *Tree Transducer Composition as Program Transformation*. Ph.D. thesis. TU Dresden.
Janis Voigtländer and Armin Kühnemann. 2004. Composition of functions with accumulating parameters. *J. Funct. Program.* 14, 3 (2004), 317–363.
W3C 1999. *XSL Transformations (XSLT)*. W3C. Retrieved from http://www.w3.org/TR/xslt.
Philip Wadler. 1990. Deforestation: Transforming programs to eliminate trees. *Theor. Comput. Sci.* 73, 2 (1990), 231–248.

Received February 2017; revised January 2018; accepted January 2018