Journal of Computer and System Sciences 75 (2009) 271–286

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Deciding equivalence of top–down XML transformations in polynomial time

Joost Engelfriet [a], Sebastian Maneth [b,*], Helmut Seidl [c]

[a] *Leiden Institute of Advanced Computer Science, The Netherlands*
[b] *NICTA and University of NSW, Sydney, Australia*
[c] *Institut für Informatik, Technische Universität München, Germany*

ARTICLE INFO

Article history:
Received 15 October 2007
Available online 24 January 2009

Keywords:
XML
Top–down tree transducer
Equivalence
Minimization

ABSTRACT

Many useful XML transformations can be expressed by deterministic top–down tree transducers. A normal form is presented for such transducers (extended with the facility to inspect their input trees). A transducer in normal form has a unique canonical form which can be obtained by a minimization procedure, in polynomial time. Thus, equivalence of transducers in normal form can be decided in polynomial time. If the transducer is total, the normal form can be obtained in polynomial time as well.

1. Introduction

The transformation of XML documents is of fundamental importance for practical XML processing. Transformations are needed, e.g., for insertion of derived formating information or for adaptation of documents to the particular syntax demanded by a given application. Many routine XML-transformations are *simple*, i.e., can be produced by a single top–down traversal over the tree structure underlying the input document. Such transformations include simple filterings, relabelings, insertions, and deletions as well as duplications of elements. Simple transformations can conveniently be expressed by means of *deterministic top–down tree transducers* running over a ranked-tree encoding of the given input document. An example of a top–down XML transformation is shown in Fig. 1; it copies the input document and additionally constructs a table of contents containing the titles $t_1, \ldots, t_n$ of all sections. A top–down tree transducer is a simple functional program: functions recursively generate trees through pattern matching on their single input tree argument. Here we consider a slightly extended model, by allowing the transducer to inspect its input tree, even the parts that it does not transform into output. The resulting deterministic top–down tree transducer *with inspection* is more robust: for instance, the corresponding class of transformations is closed under composition (see [10]).

We are interested in the problem of deciding whether or not two such transducers realize the same transformation. In 1978, Zachar showed that this problem is decidable for deterministic bottom-up (or: frontier-to-root) tree transducers [28]. Only two years later, equivalence has also been shown decidable for deterministic top–down (or root-to-frontier) tree transducers by Ésik [9] (see also [5] and Section IV.9 of [13]). The involved algorithm, however, is based on upper bounds on the difference of sizes of intermediate trees appearing in derivations of the transducers. Since the algorithm explicitly keeps track of very large "difference trees," it seems hard to extract an efficient implementation. Instead, we introduce a new normal form for deterministic top–down tree transducers (with inspection): we prove that every such transducer can be transformed effectively into an equivalent *earliest* transducer, which means that it produces its output in a uniform way and

* Corresponding author.
E-mail addresses: engelfri@liacs.nl (J. Engelfriet), sebastian.maneth@nicta.com.au (S. Maneth), seidl@in.tum.de (H. Seidl).

0022-0000/$ – see front matter
doi:10.1016/j.jcss.2009.01.001

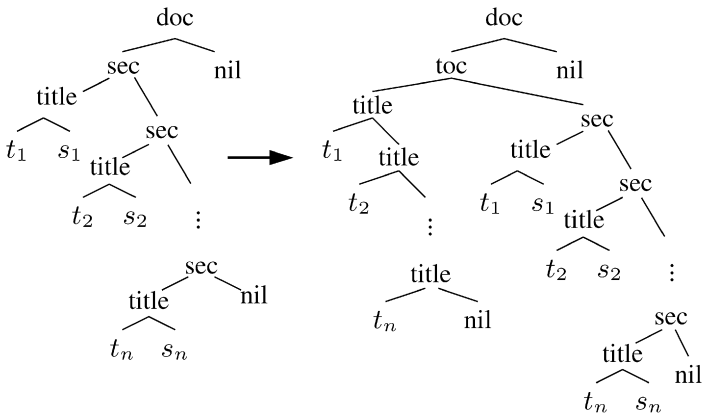

Fig. 1. A typical top–down XML translation.

"as early as possible." We also prove that earliest transducers have a unique canonical form that can be obtained by a kind of *minimization*, in polynomial time. Hence, two transducers are equivalent iff their canonical forms are the same (up to renaming of states). This provides a new way to decide equivalence of deterministic top–down tree transducers, which takes polynomial time for earliest transducers. While the normal form can be achieved for every deterministic top–down tree transducer with inspection, we show that it can be obtained even in polynomial time for total transducers, i.e., transducers whose translation is defined for every input tree. Thus, equivalence of total transducers can be decided in polynomial time. The canonical form can be seen as the generalization of a corresponding canonical form for deterministic finite-state string transducers as considered by Mohri [20].

These methods can also be extended to provide a procedure for deciding equivalence of deterministic top–down tree transducers with regular look-ahead. Such transducers additionally allow to test input trees for membership in arbitrary regular tree languages. For practical purposes, such as query evaluation of XPATH, this is a very useful property as it allows to check for the existence of (bottom-up) tree patterns in the input. Note also that every deterministic *bottom-up* tree transducer can be transformed into an equivalent deterministic top–down tree transducer with look-ahead [4]. Finally, note that for nondeterministic top–down tree transducers the equivalence problem is undecidable, because this already holds for ε-free (one-way) finite-state *string* transducers [14].

The XPATH query language is a popular formalism for selecting nodes from an XML document. A wide range of query and transformation languages, such as XQuery and XSLT, use XPath as their node selection formalism. An XPATH expression is similar to a regular expression and is evaluated on the paths of the XML tree, starting at the root node. The containment and equivalence problems are already coNP complete for a small fragment of XPATH which only uses child, descendant, wildcard, and filter (branching) [18]. In the absence of any one of the operations descendant, wildcard, or filter, containment is in PTIME [2,19,27]. It is possible to express an XPATH query through a tree transducer: every input node is copied to the output, and a new unary "select symbol" is inserted above each node selected by the query. However, even simple queries such as "select all a-nodes that have a b-node descendant" cannot be realized by a top–down transducer in this way (because the transducer does not know of the presence of b-node descendants upon visiting an a-node). To remedy this problem, one can first relabel the input tree by the run of a tree automaton, or, equivalently, add regular look-ahead. Top–down tree transducers with regular look-ahead (which can be tested for equivalence using our methods) can indeed realize the above mentioned fragment of XPATH. Note, however, that the use of nested filters in an XPATH query is similar to a conjunction and will cause the look-ahead tree automaton to be of exponential size in the size of the query.

2. Preliminaries

Top–down tree transducers conventionally work on ranked trees. This means that the number of children of a node is determined by the *rank* of the symbol at that node. We therefore consider ranked alphabets Σ consisting of finitely many symbols; each symbol $a \in \Sigma$ is implicitly equipped with a rank in $\{0, 1, \ldots\}$, where rank 0 indicates that a is the potential label of a leaf. We assume that a ranked alphabet contains at least one symbol of rank 0.

The set $\mathcal{T}_\Sigma$ of ranked *trees* over the ranked alphabet Σ then is defined by

$$t ::= a(t_1, \ldots, t_k)$$

where a ranges over symbols in Σ of rank k. As usual, we also write a for the tree $a()$. Note that, since there is at least one symbol of rank 0, $\mathcal{T}_\Sigma \neq \emptyset$. We represent the nodes of a tree in Dewey notation, i.e., by sequences of numbers (for readability, numbers in the sequence are separated using dots). Formally, the set $V(t)$ of *nodes* of the tree t is inductively defined as: $V(t) = \{\varepsilon\} \cup \{i.v \mid 1 \leqslant i \leqslant k,\ v \in V(t_i)\}$ if $t = a(t_1, \ldots, t_k)$, $a \in \Sigma$ of rank $k \geqslant 0$ and $t_1, \ldots, t_k \in \mathcal{T}_\Sigma$. Thus, the empty sequence ε represents the root of t and $u.i$ represents the ith child of the node u of t. In abuse of notation, we also

use "." to denote concatenation of sequences of numbers. A node v is an ancestor of node w if there is a (possibly empty) sequence of numbers u such that $w = v.u$. The size of t, denoted $\text{size}(t)$, is the number $|V(t)|$ of its nodes. The depth of t, denoted $\text{depth}(t)$, is the maximal number of nodes on a path in t from the root to a leaf.

A *pattern* is a prefix of a tree. Formally the set of all patterns is given by the set of all trees in $\mathcal{T}_{\Sigma\cup\{\top\}}$, where $\top$ is a new symbol of rank zero which is not in Σ. Assume p is a pattern containing exactly k occurrences of $\top$, and $p_1, \ldots, p_k$ is a sequence of patterns. Then the pattern $q = p[p_1, \ldots, p_k]$ is obtained from p by replacing the ith occurrence of $\top$ (in left-to-right order) with p_i. Note that the result q is a tree, i.e., does not contain occurrences of $\top$, iff the $p_1, \ldots, p_k$ are all trees.

Consider the set $\mathcal{P}_\Sigma = \mathcal{T}_{\Sigma\cup\{\top\}} \cup \{\bot\}$ of all patterns enhanced with an extra bottom element $\bot$ (not in $\Sigma \cup \{\top\}$). On this set, we define a partial ordering by $\bot \sqsubseteq p$ for all p, and $p \sqsubseteq p'$ for patterns p, p' iff $p = p'[p_1, \ldots, p_k]$ for suitable patterns $p_1, \ldots, p_k$. The latter means that every non-$\top$ node of p' is also a node of p and has the same label in both patterns. Intuitively, p' is a prefix of p. With respect to this ordering, every set $X \subseteq \mathcal{P}_\Sigma$ has a least upper bound $p = \bigsqcup X$. If X is empty or just contains $\bot$, $p = \bot$. Otherwise, p is a pattern and the set V of non-$\top$ nodes of p consists of all nodes v such that every ancestor of v is in $V(p')$ for all $p' \in X \setminus \{\bot\}$ and has the same label from Σ in all $p' \in X\setminus\{\bot\}$. In particular if $V = \emptyset$, the least upper bound of X is given by the pattern $\top$. Since every subset of $\mathcal{P}_\Sigma$ has a least upper bound, $\mathcal{P}_\Sigma$ is a complete lattice.

While the length of a strictly decreasing chain in $\mathcal{P}_\Sigma$ can be infinite, the length of a strictly *increasing* chain is always finite. More precisely, the number of elements in a strictly increasing chain above a pattern p is bounded by the number of non-$\top$ nodes in p.

3. Deterministic top–down tree transducers

A *deterministic top–down tree transducer* (t-transducer for short) is a tuple $T = (Q, \Sigma, \Delta, \delta, A)$, where

- Q is a finite set of states,
- Σ and Δ are ranked input and output alphabets, respectively, disjoint with Q,
- δ is the (possibly partial) transition function, and
- A is the axiom.

The axiom A has the form $p[q_1(x_0), \ldots, q_r(x_0)]$ for a variable x_0 meant to be bound to the input tree, a pattern $p \in \mathcal{T}_{\Delta\cup\{\top\}}$, and a sequence $q_1, \ldots, q_r$, $r \geqslant 0$, of states in Q.

For every state q in Q and input symbol $a \in \Sigma$ of rank k the transition function δ contains at most one transition, which is of the form

$$q\big(a(x_1, \ldots, x_k)\big) \to p\big[q_1(x_{i_1}), \ldots, q_r(x_{i_r})\big]$$

where $x_1, \ldots, x_k$ are distinct variables, $p \in \mathcal{T}_{\Delta\cup\{\top\}}$ is a pattern, $q_1, \ldots, q_r \in Q$, and x_{i_j} are variables occurring among the $x_1, \ldots, x_k$. For every state q and input symbol a let $\delta(q, a)$ be the right-hand side of the transition for q and a if it is defined, and let $\delta(q, a)$ be undefined otherwise.

Note that the axiom and the right-hand sides of transitions are trees over the ranked alphabet $\Delta \cup Q \cup X$, where each state in Q has rank 1, $X = \{x_i \mid i \geqslant 0\}$ is the set of variables, and each variable has rank 0. Similarly, the left-hand sides of transitions are trees over $\Sigma \cup Q \cup X$.

The transducer is *total* if $\delta(q, a)$ is defined for all $q \in Q$ and $a \in \Sigma$. The *size* of T, denoted $|T|$, is the sum of the size of its axiom and the sizes of the left-hand sides and right-hand sides of its transitions.

The semantics $[\![q]\!]$ of every state q of the transducer is a partial function $\mathcal{T}_\Sigma \to \mathcal{T}_\Delta$ which is defined by recursion over its argument. Assume the argument s of the function $[\![q]\!]$ is of the form $s = a(s_1, \ldots, s_k)$. Assume further that $\delta(q, a) = p[q_1(x_{i_1}), \ldots, q_r(x_{i_r})]$. If the recursive calls $[\![q_j]\!](s_{i_j})$ return results t_j, then the call $[\![q]\!](s)$ returns the value $[\![q]\!](s) = p[t_1, \ldots, t_r]$. If on the other hand, $\delta(q, a)$ is undefined or one of the recursive calls $[\![q_j]\!](s_{i_j})$ is undefined, then the function $[\![q]\!]$ is also undefined for s. In the following, we denote the domain of $[\![q]\!]$ by $\text{DOM}(q)$.

The t-transducer T *realizes* a partial function $[\![T]\!] : \mathcal{T}_\Sigma \to \mathcal{T}_\Delta$. Assume that the axiom of T is given by $A = p[q_1(x_0), \ldots, q_r(x_0)]$. Then the domain of $[\![T]\!]$, denoted $\text{DOM}(T)$ and also called the *domain of* T, is defined by $\text{DOM}(T) = \text{DOM}(q_1) \cap \cdots \cap \text{DOM}(q_r)$. For every $s \in \text{DOM}(T)$, the output $[\![T]\!](s)$ of the transducer T on input s is defined by:

$$[\![T]\!](s) = p\big[[\![q_1]\!](s), \ldots, [\![q_r]\!](s)\big].$$

We call two t-transducers T_1 and T_2 *equivalent* if $[\![T_1]\!] = [\![T_2]\!]$. A partial function that can be realized by a t-transducer is called a *t-translation*.

In this paper we will, without loss of generality, only deal with t-transducers of which *all states are reachable*. A state of a t-transducer T is called *reachable* if it occurs in the axiom of T or in $\delta(q, a)$ for some reachable state q and some input symbol a. Intuitively, this means that it occurs in a (not necessarily successful) computation of T starting with the axiom. The reachable states can be determined in time linear in $|T|$ by depth-first search of the directed graph with the states as nodes and an edge from q to q' if q' occurs in $\delta(q, a)$ for some a, starting with the states in the axiom. Obviously,

the unreachable states of a t-transducer can be removed, together with their transitions. Thus, for every t-transducer T an equivalent t-transducer T' can be constructed in linear time, such that all states of T' are reachable.

If t-transducer T_2 can be obtained from t-transducer T_1 by a (bijective) renaming of states, we will identify T_1 and T_2. Note that, since all states are reachable, this can be checked in linear time.

Top–down tree transducers were introduced by Thatcher and Rounds [23,24] (see also [11] for a survey on tree transducers). Conventionally, a top–down tree transducer has an initial state, instead of an axiom. It should be clear that this choice has no impact on the class of t-translations: to simulate a conventional transducer (with initial state q_0) using our model, simply define the axiom as $q_0(x_0)$. Conversely, to simulate our transducer, with axiom $p[q_1(x_0), \ldots, q_r(x_0)]$, by a conventional one, add the new state q_0 as initial state and, for every input symbol a such that $\delta(q_i, a)$ is defined for all $i \in \{1, \ldots, r\}$, define $\delta(q_0, a)$ as the tree $p[\delta(q_1, a), \ldots, \delta(q_r, a)]$.

Example 1. We define a t-transducer $T_X = (Q, \Sigma, \Delta, \delta, A)$ that realizes the translation of XML documents with section and title markup as described in the Introduction. The transducer has states $Q = \{q_0, t, e, n, id\}$, Σ containing at least the symbols doc, sec, title, and nil, $\Delta = \Sigma \cup \{\text{toc}\}$, axiom $A = q_0(x_0)$, and the following transitions in δ:

$$q_0\big(\text{doc}(x_1, x_2)\big) \to \text{doc}\big(\text{toc}\big(t(x_1), id(x_1)\big), n(x_2)\big),$$
$$t\big(\text{sec}(x_1, x_2)\big) \to \text{title}\big(e(x_1), t(x_2)\big),$$
$$t(\text{nil}) \to \text{nil},$$
$$e\big(\text{title}(x_1, x_2)\big) \to id(x_1),$$
$$n(\text{nil}) \to \text{nil},$$

where the state id has the obvious transitions to realize the identity on $\mathcal{T}_\Sigma$. Note that the right-hand side of the first transition is $p[t(x_1), id(x_1), n(x_2)]$ where p is the pattern $\text{doc}(\text{toc}(\top, \top), \top)$.

Wildcards

Query languages such as XPATH support a wildcard operator for selecting a node with *any* label. Such a mechanism for dealing with arbitrary labels is also present in pattern matching constructs of mainstream programming languages in the form of the "default case." For a fixed, finite set of ranks, this can be obtained in our setting by enhancing the ranked input alphabet Σ with special symbols "$*_k$," representing input labels of rank k that are arbitrary, but not in Σ. Then, a transition of the form $q(*_k(x_1, \ldots, x_k)) \to *_k(q(x_1), \ldots, q(x_k))$ copies any non-Σ symbol from the input to the output tree. Note that in the context of XML we typically work on binary trees (with leaves representing the empty hedge) and henceforth only need one incarnation of the $*$-symbol of rank two.

Deterministic top–down tree automata

A *deterministic top–down tree automaton* (dtta for short) is a t-transducer $M = (Q, \Sigma, \Delta, \delta, A)$ such that $\Delta = \Sigma$, $A = q_0(x_0)$ for some $q_0 \in Q$ called the *initial state*, and every transition in δ is of the form

$$q\big(a(x_1, \ldots, x_k)\big) \to a\big(q_1(x_1), \ldots, q_k(x_k)\big).$$

In what follows, $q_0(x_0)$ will be abbreviated by q_0, and $a(q_1(x_1), \ldots, q_k(x_k))$ by $q_1 \cdots q_k$.

The *language accepted by* the dtta M is $\text{DOM}(M)$, which equals $\text{DOM}(q_0)$. Note that M realizes the identity on its domain, i.e., $[\![M]\!](s) = s$ for every $s \in \text{DOM}(M)$.

We will say that a dtta M is *minimal* if for all states q, q' of M: $\text{DOM}(q) \neq \emptyset$, and if $q \neq q'$ then $\text{DOM}(q) \neq \text{DOM}(q')$. Recall that we only consider t-transducers (and hence dtta's) of which all states are reachable.

The following two facts are well known:

Proposition 2.

(1) *The domain of every t-transducer T can be accepted by some dtta M_T. Moreover, M_T can be constructed from T in exponential time.*
(2) *For every dtta M with $\text{DOM}(M) \neq \emptyset$ an equivalent minimal dtta M' can be constructed in polynomial time. Two minimal dtta's are equivalent iff they are the same; hence, M' is unique.*

The first fact is shown in (the proof of) Theorem 3.1 of [4] by a straightforward subset construction. Thus, the states of M_T are sets of states of T. When M_T arrives in state B at a node of the input tree, B is the set of all states of T that arrive at that node in parallel. Moreover, $\delta(B, a)$ is defined iff $\delta(q, a)$ is defined for all $q \in B$, and so, $\text{DOM}(B)$ is the intersection of all $\text{DOM}(q)$, $q \in B$.

The second fact is well known (see [12] or Section II.11 of [13], and [21]) but is also easy to prove. For the sake of completeness we briefly discuss the proof (also because our formalism differs slightly from those in [12,13,21]). Every dtta can be viewed as a context-free grammar, with the states as nonterminals, and with a production $q \to a(q_1, \ldots, q_k)$ corresponding to the transition $q(a(x_1, \ldots, x_k)) \to a(q_1(x_1), \ldots, q_k(x_k))$. The useless nonterminals can be removed from a context-free grammar (and thus from the dtta) in linear time, cf., e.g., Section 7.4.3 in [15], or [3]. Identifying all states q, q' with $\textsc{dom}(q) = \textsc{dom}(q')$ then gives a minimal dtta equivalent to the given one. To see that these pairs of states can be determined in polynomial time, define the relation $\equiv$ on the set of states of the dtta to be the largest equivalence relation such that if $q \equiv q'$, then

(a) $\delta(q, a)$ is defined iff $\delta(q', a)$ is defined, and
(b) if $\delta(q, a) = q_1 \cdots q_k$ and $\delta(q', a) = q'_1 \cdots q'_k$, then $q_j \equiv q'_j$ for all j.

It is easy to show that $\textsc{dom}(q) = \textsc{dom}(q')$ iff $q \equiv q'$. The equivalence relation $\equiv$ can be computed in polynomial time by a standard fixpoint iteration.

Now consider two minimal dtta's M and M' that are equivalent. Define the relation $q \equiv q'$ as above, with q a state from M and q' a state from M'. It is straightforward to show that $\equiv$ is a bijection between the states of M and M', and that M and M' are the same up to the renaming $\equiv$ of states.

Similar arguments will be used (in greater detail) for top–down tree transducers.

Transducers with inspection

To be able to find, for every transducer, an equivalent transducer in (earliest) normal form, as discussed in the Introduction, we need a slight extension of the t-transducer. If a t-transducer inspects a subtree of the input tree, then it also has to produce output on that subtree; in other words, if a subtree is deleted, it cannot be inspected. In this way the t-transducer differs from the finite-state string transducer, which must always read the whole input string (possibly producing empty output). It is exactly this feature that is responsible for the fact that the t-translations are not closed under composition, see Section I of [23]. We now add to the t-transducer the facility to inspect subtrees that are deleted, by allowing it to run a dtta in parallel with itself. This makes sense from the point of view of XML transformations, because such transformations are usually defined on trees that are valid with respect to a (generalized) DTD. Here we only allow DTD's that can be expressed by a dtta. The general case of an arbitrary regular tree language is treated at the end of this paper (cf. also the transformational systems of [23]).

A *deterministic top–down tree transducer with inspection* (i-transducer for short) is a pair $T = (P, I)$, where

- $P = (Q_P, \Sigma, \Delta, \delta_P, A)$ is a t-transducer, and
- $I = (Q_I, \Sigma, \Sigma, \delta_I, c_0(x_0))$ is a dtta,

with the same input alphabet, and with $Q_P \cap Q_I = \emptyset$. We define the set of states and the transition function of T to be $Q = Q_P \cup Q_I$ and $\delta = \delta_P \cup \delta_I$ respectively. The states in Q_P are called *processing* states, and those of Q_I *inspecting* states, with c_0 being the *initial* inspecting state. Similarly, the transitions in δ_P and δ_I are called processing and inspecting transitions, respectively. In what follows, we will also specify i-transducer T as one tuple $(Q, \Sigma, \Delta, \delta, A, c_0)$, where $Q_P, Q_I, \delta_P, \delta_I$ are assumed to be specified implicitly. The size of T is $|T| = |P| + |I|$.

The *translation realized by* T is the restriction of $[\![P]\!]$ to $\textsc{dom}(I)$, i.e., it is the partial function

$$[\![T]\!] = \big\{ \big(s, [\![P]\!](s)\big) \bigm| s \in \textsc{dom}(P) \cap \textsc{dom}(I) \big\}.$$

The *domain of* T is defined to be $\textsc{dom}(T) = \textsc{dom}(P) \cap \textsc{dom}(I)$; in other words, it is the domain of $[\![T]\!]$. We observe that $\textsc{dom}(T)$ can be accepted by some dtta M_T, which can be constructed in exponential time. In fact, a dtta M_P with $\textsc{dom}(M_P) = \textsc{dom}(P)$ can be constructed in exponential time by Proposition 2, and then M_T can be obtained from M_P and I by an obvious product construction in quadratic time.

Two i-transducers T_1 and T_2 are called *equivalent* if $[\![T_1]\!] = [\![T_2]\!]$. A partial function that can be realized by an i-transducer is called an *i-translation*.

For a given input alphabet Σ, let I_{id} be the (unique) total dtta with set of states $\{id\}$; note that $\textsc{dom}(I_{id}) = \mathcal{T}_\Sigma$ and that I_{id} is minimal. An i-transducer $T = (P, I_{id})$ is "really" a t-transducer. For that reason, every t-transducer will also be considered to be an i-transducer. An i-transducer is *total* if it corresponds to a total t-transducer.

By definition, the i-translations are just the restrictions of the t-translations to the dtta languages. Equivalently, the i-translations are the compositions of the dtta translations (which are the identities on dtta languages) with the t-translations. Thus, every i-translation is the composition of two t-translations. This also holds in the other direction, because the class of i-translations is closed under composition, as shown in [10]. The closedness of the i-translations under composition (as opposed to the t-translations) is important when i-transducers are considered as queries on XML databases, because it allows the use of views: an i-query on an i-view of the database can be replaced by an equivalent i-query on the database.

4. Common prefixes

Consider a processing state q of an i-transducer T with nonempty domain $\mathrm{DOM}(q)$. Define the pattern

$$\mathsf{pref}(q) = \bigsqcup \{ [\![q]\!](s) \mid s \in \mathrm{DOM}(q) \}$$

as the *common prefix* of all outputs possibly produced by q. Since the set of patterns is a complete lattice, the pattern $\mathsf{pref}(q)$ is well defined.

Example 3. Consider the total t-transducer T_1 with the following two transitions:

$$q\big(a(x_1,x_2)\big) \to d\big(q(x_1), d\big(q(x_1), e\big)\big),$$
$$q(e) \to d\big(d(e,e), d(e,e)\big).$$

Obviously, all outputs generated by the state q start with the pattern $d(\top, d(\top, e))$. In fact, the common prefix of all outputs produced by q is the pattern $\mathsf{pref}(q) = d(d(\top,\top), d(\top, e))$.

We will show how to compute the common prefixes $\mathsf{pref}(q)$, $q \in Q_P$, under the assumption that the transducer T is uniform. An i-transducer $T = (P, I) = (Q, \Sigma, \Delta, \delta, A, c_0)$ is called *uniform* if I is a minimal dtta and there is a mapping $\rho : Q_P \to Q_I$ with the following properties (for all $q, c, \bar{q}, \bar{c} \in Q$):

(1) $\rho(q) = c_0$ if q occurs in A;
(2) if $\rho(q) = c$ then, for every $a \in \Sigma$:
  (a) $\delta(q,a)$ is defined iff $\delta(c,a)$ is defined, and
  (b) if, for the same variable x_i, $\bar{q}(x_i)$ occurs in $\delta(q,a)$ and $\bar{c}(x_i)$ in $\delta(c,a)$, then $\rho(\bar{q}) = \bar{c}$.

The fact that $\mathrm{DOM}(c_1) \neq \mathrm{DOM}(c_2)$ for distinct states $c_1, c_2 \in Q_I$ will only play a role in the proof of Theorem 15. It will, however, be frequently used that $\mathrm{DOM}(c)$ is nonempty for every $c \in Q_I$.

Note that since all states of P are reachable, the mapping ρ is unique (when it exists). Moreover, if T is uniform, ρ can easily be computed in linear time (by an obvious variant of depth-first search). The mapping ρ will be called the *relevance map* of the uniform i-transducer T.

Intuitively, uniformity of $T = (P, I)$ means that, during a computation of P and I on an input tree s, starting with the axiom A and the initial state c_0 respectively, the processing states are keeping track of the inspecting state (by uniformity properties (1) and (2)(b)). Since, as will be shown in the next lemma, the dtta I accepts $\mathrm{DOM}(T)$, the processing states "follow" the behavior of the state of I at the current node of s. This means that a processing state q with $\rho(q) = c$ continues its computation, at a certain node of s, iff the inspecting state c does (by uniformity property (2)(a)). In this way, q only processes "relevant" subtrees of s. In fact, $\mathrm{DOM}(c)$ is the set of all input subtrees that are processed by q during the computations of T (starting with A and c_0) on input trees from $\mathrm{DOM}(T)$. Since $\mathrm{DOM}(c) \neq \emptyset$, this also implies that every processing state of T occurs in at least one such computation.

Next we state some easy properties of uniform transducers.

Lemma 4. *Let $T = (Q, \Sigma, \Delta, \delta, A, c_0)$ be a uniform i-transducer with relevance map ρ. Then the following statements hold.*

(1) *If $\rho(q) = c$, then $\mathrm{DOM}(c) \subseteq \mathrm{DOM}(q)$.*
(2) *$\mathrm{DOM}(c_0) = \mathrm{DOM}(T)$.*
(3) *For every $c \in Q_I$ and $s \in \mathcal{T}_\Sigma$, there is a tree $s' \in \mathrm{DOM}(c)$ such that for every q with $\rho(q) = c$, $[\![q]\!](s') = [\![q]\!](s)$ if $s \in \mathrm{DOM}(q)$.*
(4) *If $\rho(q) = c$, then $\mathsf{pref}(q) = \bigsqcup\{[\![q]\!](s) \mid s \in \mathrm{DOM}(c)\}$.*

Proof. (1) We proceed by induction on the structure of s, and show that $s \in \mathrm{DOM}(c)$ implies $s \in \mathrm{DOM}(q)$. Assume that $s = a(s_1, \ldots, s_k) \in \mathrm{DOM}(c)$. Then there exist $c_1, \ldots, c_k$ such that $\delta(c,a) = c_1 \cdots c_k$ and $s_i \in \mathrm{DOM}(c_i)$ for $1 \leqslant i \leqslant k$. By uniformity property (2)(a), $\delta(q,a)$ is defined, say $\delta(q,a) = p[q_1(x_{i_1}), \ldots, q_r(x_{i_r})]$, and by uniformity property (2)(b), $\rho(q_j) = c_{i_j}$ for all $j = 1, \ldots, r$. By induction, $s_{i_j} \in \mathrm{DOM}(q_j)$. Hence $[\![q]\!](s) = p[[\![q_1]\!](s_{i_1}), \ldots, [\![q_r]\!](s_{i_r})]$, and so $s \in \mathrm{DOM}(q)$.

(2) If $A = p[q_1(x_0), \ldots, q_r(x_0)]$, then $\mathrm{DOM}(T) = (\mathrm{DOM}(q_1) \cap \cdots \cap \mathrm{DOM}(q_r)) \cap \mathrm{DOM}(c_0)$. By uniformity property (1), and by statement (1) of this lemma, this equals $\mathrm{DOM}(c_0)$.

(3) The proof is by induction on the structure of s. Let $s = a(s_1, \ldots, s_k)$. Assume first that $\delta(c,a)$ is undefined. Then $\delta(q,a)$ is undefined for every q with $\rho(q) = c$, by uniformity property (2)(a). And so, $[\![q]\!](s)$ is undefined for every such q, i.e., $s \notin \mathrm{DOM}(q)$. Thus we can take s' to be any element of $\mathrm{DOM}(c)$, which is nonempty by minimality of I.

Now assume that $\delta(c,a)$ is defined, say $\delta(c,a) = c_1 \cdots c_k$. By induction, there exist input trees $s'_i \in \mathrm{DOM}(c_i)$, $1 \leqslant i \leqslant k$, such that if $\rho(\bar{q}) = c_i$, then $[\![\bar{q}]\!](s'_i) = [\![\bar{q}]\!](s_i)$ if $s_i \in \mathrm{DOM}(\bar{q})$. Now take $s' = a(s'_1, \ldots, s'_k)$. Obviously $s' \in \mathrm{DOM}(c)$. For every q with $\rho(q) = c$, if $\bar{q}(x_i)$ occurs in $\delta(q,a)$, then $\rho(\bar{q}) = c_i$ by uniformity property (2)(b). From that it easily follows that s' has the desired property.

(4) This is an immediate consequence of statement (3). □

Note that uniformity does not imply that the processing states that arrive at a particular node of the input tree (and hence are mapped to the same inspecting state), all have the same domain.

Example 5. Consider the i-transducer $T = (P, I)$ with the input (and output) alphabet Σ consisting of a nullary input symbol e and a binary input symbol a, with processing transitions $\delta(q_1, a) = q(x_1)$, $\delta(q_2, a) = q(x_2)$, $\delta(q, e) = e$, and axiom $a(q_1(x_0), q_2(x_0))$. The dtta I is the minimal dtta accepting the domain $\{a(e, e)\}$ of P. It has states c_0 and c (with c_0 the initial state), and inspecting transitions $\delta(c_0, a) = cc$ and $\delta(c, e) = \varepsilon$. Thus, T is uniform with relevance map $\rho = \{q_1 \mapsto c_0, q_2 \mapsto c_0, q \mapsto c\}$. Note however, that the domain of $[\![q_1]\!]$ is $\{a(e, t) \mid t \in \mathcal{T}_\Sigma\}$ while the domain of $[\![q_2]\!]$ is $\{a(t, e) \mid t \in \mathcal{T}_\Sigma\}$. The sets of relevant inputs for q_1 as well as for q_2 are given by $\mathrm{DOM}(c_0) = \{a(e, e)\}$.

As an example application of statement (3) of Lemma 4, consider the inspecting state c_0 and the input tree $s = a(e, a(e, e)) \in \mathrm{DOM}(q_1)$. To find a tree $s' \in \mathrm{DOM}(c_0)$, we first observe that $\delta(c_0, a) = cc$. Thus, $s' = a(s'_1, s'_2)$ where s'_1 is obtained from c and $s_1 = e$, and s'_2 from c and $s_2 = a(e, e)$. Since $\delta(c, e)$ is defined, $s'_1 = e$. Since $\delta(c, a)$ is undefined, s'_2 is chosen arbitrarily in $\mathrm{DOM}(c) = \{e\}$, i.e., $s'_2 = e$. Hence $s' = a(e, e)$ satisfies statement (3) of the lemma: $s' \in \mathrm{DOM}(c_0)$ and $[\![q_1]\!](s') = e = [\![q_1]\!](s)$.

According to our definition, a *total* i-transducer T is always uniform (with $\rho(q) = id$ for every processing state q). Consider the i-transducer $T = (P, I)$ where P is the t-transducer T_X from Example 1, and I is the minimal dtta for $\mathrm{DOM}(P)$. If T would be uniform, with relevance map ρ, then its first transition would imply that $\rho(t) = \rho(id)$, by uniformity property (2)(b). But $\delta(id, \mathrm{title}) = \mathrm{title}(id(x_1), id(x_2))$ whereas $\delta(t, \mathrm{title})$ is undefined, contradicting uniformity property (2)(a). Thus, T is not uniform.

We now show that every i-transducer with a nonempty domain is (effectively) equivalent to a uniform transducer.

Lemma 6. *For every i-transducer T with $\mathrm{DOM}(T) \neq \emptyset$, a uniform i-transducer T' can be constructed in exponential time such that $[\![T']\!] = [\![T]\!]$.*

Proof. Let $T = (P, I) = (Q, \Sigma, \Delta, \delta, A, c_0)$ be an i-transducer with $\mathrm{DOM}(T) \neq \emptyset$. By Proposition 2, a dtta accepting $\mathrm{DOM}(T)$ can be constructed in exponential time, and using polynomial time, this dtta can be turned into an equivalent minimal dtta. For this reason, we assume from now on that $\mathrm{DOM}(I) = \mathrm{DOM}(T)$, and that I is minimal (which is a first requirement for uniformity). It should be clear that the construction in the remainder of the proof can be performed in polynomial time.

The idea for the new transducer T' simply consists of incorporating the state of the dtta I into the states of the t-transducer P, when they are running in parallel on the same input tree. Accordingly, $T' = (P', I)$ and the states of P' will be of the form $\langle q, c\rangle$ with $q \in Q_P$ and $c \in Q_I$. We will define the states and transitions of P' inductively, and simultaneously show that $\mathrm{DOM}(c) \subseteq \mathrm{DOM}(q)$ for every state $\langle q, c\rangle$.

We observe here that if $\mathrm{DOM}(c) \subseteq \mathrm{DOM}(q)$, then $\langle q, c\rangle$ satisfies the following property (†): for every $a \in \Sigma$, if $\delta(c, a)$ is defined then $\delta(q, a)$ is defined. In fact, if $\delta(c, a) = c_1 \cdots c_k$, then there is a tree $s = a(s_1, \ldots, s_k)$ such that $s \in \mathrm{DOM}(c)$ (because $\mathrm{DOM}(c_i) \neq \emptyset$ for all c_i, by minimality of I); hence $s \in \mathrm{DOM}(q)$ and so $\delta(q, a)$ is defined.

If $A = p[q_1(x_0), \ldots, q_r(x_0)]$, then the axiom A' of T' is

$$A' = p\big[\langle q_1, c_0\rangle(x_0), \ldots, \langle q_r, c_0\rangle(x_0)\big]$$

where $\langle q_1, c_0\rangle, \ldots, \langle q_r, c_0\rangle$ are new states of P'. Note that $\mathrm{DOM}(c_0) = \mathrm{DOM}(I) = \mathrm{DOM}(T) \subseteq \mathrm{DOM}(q_i)$.

For a new state $\langle q, c\rangle$ of P' and an input symbol $a \in \Sigma$ of rank k, assume that $\delta(c, a)$ is defined and given by $\delta(c, a) = c_1 \cdots c_k$. Then, by (†), the t-transducer P has a transition

$$q\big(a(x_1, \ldots, x_k)\big) \to p\big[q_1(x_{i_1}), \ldots q_r(x_{i_r})\big].$$

Accordingly, the new t-transducer P' has the transition:

$$\langle q, c\rangle\big(a(x_1, \ldots, x_k)\big) \to p\big[\langle q_1, c_{i_1}\rangle(x_{i_1}), \ldots, \langle q_r, c_{i_r}\rangle(x_{i_r})\big]$$

for further states $\langle q_j, c_{i_j}\rangle$, $j = 1, \ldots, r$. It should be clear that $\mathrm{DOM}(c_{i_j}) \subseteq \mathrm{DOM}(q_j)$: If s_i is an arbitrary element of $\mathrm{DOM}(c_i)$, for $i = 1, \ldots, k$, then $s = a(s_1, \ldots, s_k)$ is in $\mathrm{DOM}(c)$. Since we already know that $\mathrm{DOM}(c) \subseteq \mathrm{DOM}(q)$, we obtain that $s \in \mathrm{DOM}(q)$, and hence $s_{i_j} \in \mathrm{DOM}(q_j)$. Note that this argument is correct because all $\mathrm{DOM}(c_i)$ are nonempty.

Obviously, by construction, the resulting transducer T' is uniform with the relevance map ρ that maps every pair $\langle q, c\rangle$ to its second component c. Moreover, it is straightforward to verify by structural induction on input trees s that for every state $\langle q, c\rangle$ of P', if $s \in \mathrm{DOM}(c)$, then $s \in \mathrm{DOM}(\langle q, c\rangle)$ and $[\![\langle q, c\rangle]\!](s) = [\![q]\!](s)$. Since $\mathrm{DOM}(c_0) = \mathrm{DOM}(T)$, this implies by construction of the axiom A' of T', that $[\![T]\!] \subseteq [\![T']\!]$. From the fact that $\mathrm{DOM}(T') \subseteq \mathrm{DOM}(I) = \mathrm{DOM}(T)$, we conclude that T and T' are equivalent. □

The size of T' in the proof of Lemma 6 heavily depends on the size of the dtta accepting $\mathrm{DOM}(T)$, and hence on the number (of combinations) of different states that arrive at a node of the input tree in parallel. In practice we expect that this number is not too large. In fact, the bulk of practical translations are of *linear size increase*, i.e., the size of every output tree is bounded by a constant times the size of the corresponding input tree. It is well known by an old result of Aho

and Ullman [1] that for any linear size increase (deterministic) top–down tree translation there is (effectively) a transducer that is "finite-copying." The latter means that the number of states arriving at any input node is bounded by a constant c (called the "copying number"). For a transducer with copying number c, the size of the dtta accepting $\mathrm{DOM}(T)$ is at most exponential in c, and hence, so is the running time of the construction of Lemma 6. Note that the transducer of Example 1 has copying number 2.

Example 7. We will turn the t-transducer $T_X = (Q, \Sigma, \Delta, \delta, A)$ of Example 1 into an equivalent uniform i-transducer $T'_X = (P', I)$. The minimal dtta I with domain $\mathrm{DOM}(I) = \mathrm{DOM}(T_X)$ has the same states as T_X, with primes to distinguish them, it has initial state q'_0, and the following transitions:

$$\delta_I(q'_0, \mathrm{doc}) = t'n',$$
$$\delta_I(t', \mathrm{sec}) = e't',$$
$$\delta_I(t', \mathrm{nil}) = \varepsilon,$$
$$\delta_I(e', \mathrm{title}) = id'id',$$
$$\delta_I(n', \mathrm{nil}) = \varepsilon,$$

where id' has all transitions to realize the identity on $\mathcal{T}_\Sigma$. The t-transducer P' has states $\langle q, q'\rangle$ for all $q \in Q$, which we will again denote by q, and it has states $\langle id, t'\rangle$ and $\langle id, e'\rangle$, which we will denote by id_t and id_e, respectively. Note that $\rho(q) = q'$, $\rho(id_t) = t' = \rho(t)$, and $\rho(id_e) = e' = \rho(e)$. The axiom of P' is still $q_0(x_0)$, and its transitions are:

$$q_0\big(\mathrm{doc}(x_1, x_2)\big) \to \mathrm{doc}\big(\mathrm{toc}\big(t(x_1), id_t(x_1)\big), n(x_2)\big),$$
$$t\big(\mathrm{sec}(x_1, x_2)\big) \to \mathrm{title}\big(e(x_1), t(x_2)\big),$$
$$t(\mathrm{nil}) \to \mathrm{nil},$$
$$id_t\big(\mathrm{sec}(x_1, x_2)\big) \to \mathrm{sec}\big(id_e(x_1), id_t(x_2)\big),$$
$$id_t(\mathrm{nil}) \to \mathrm{nil},$$
$$e\big(\mathrm{title}(x_1, x_2)\big) \to id(x_1),$$
$$id_e\big(\mathrm{title}(x_1, x_2)\big) \to \mathrm{title}\big(id(x_1), id(x_2)\big),$$
$$n(\mathrm{nil}) \to \mathrm{nil},$$

where, as before, the state id has all transitions to realize the identity on $\mathcal{T}_\Sigma$.

We now turn to the computation of the common prefixes of uniform i-transducers. For a uniform i-transducer T with relevance map ρ, let $\eta(T)$ denote the maximal size of output trees produced for relevant input trees of minimal depth, i.e., $\eta(T) = \max\{\mathrm{size}([\![q]\!](s)) \mid q \in Q_P,\ s \in S_{\rho(q)}\}$ where, for $c \in Q_I$, $S_c = \{s \in \mathrm{DOM}(c) \mid \forall s' \in \mathrm{DOM}(c)\colon\ \mathrm{depth}(s) \leqslant \mathrm{depth}(s')\}$.

We observe that a specific collection of trees $t_q = [\![q]\!](s_{\rho(q)})$, $q \in Q_P$, with $s_c \in S_c$ for every $c \in Q_I$, can be computed in time $\mathcal{O}(|T| \cdot \eta(T))$. To see this, note first that trees $s_c \in S_c$, $c \in Q_I$, can be computed by an obvious variant of the algorithm that computes the useful nonterminals of a context-free grammar: when the algorithm treats a transition $\delta(c, a) = c_1 \cdots c_k$ and trees s_{c_i} have already been computed for $i = 1, \ldots, k$, the tree s_c is set to $a(s_{c_1}, \ldots, s_{c_k})$. It is easy to see that the depth of s_c is minimal. The time taken by the algorithm is linear in the sum of $|I|$ (as it is a variant of the known algorithm) and the time to write down the trees s_c, i.e., the sum of their sizes. However, we do not wish to compute s_c, but the output trees $t_q = [\![q]\!](s_c)$, for every $q \in Q_P$ with $\rho(q) = c$. So, instead, when the algorithm treats a transition $\delta(c, a) = c_1 \cdots c_k$, it computes for each such q the tree $t_q = [\![q]\!](s_c)$ by substituting $t_{\bar{q}}$ for every $\bar{q}(x_i)$ in $\delta(q, a)$. Note that since $\rho(\bar{q}) = c_i$, the tree $t_{\bar{q}} = [\![\bar{q}]\!](s_{c_i})$ was assumed to be computed before. The time taken by this algorithm is linear in the sum of $|I|$ and the sizes of the trees t_q, $q \in Q_P$. Since $\mathrm{size}(t_q) \leqslant \eta(T)$, the time is linear in $|I| + |Q_P| \cdot \eta(T)$, and so it is $\mathcal{O}(|T| \cdot \eta(T))$.

Then we have:

Theorem 8. *Let $T = (P, I) = (Q, \Sigma, \Delta, \delta, A, c_0)$ be a uniform i-transducer.*
The common prefixes $\mathrm{pref}(q)$, $q \in Q_P$, can be computed in time $\mathcal{O}(|T| \cdot \eta(T))$. They are of size at most $\eta(T)$.
If $A = p[q_1(x_0), \ldots, q_r(x_0)]$, then

$$\bigsqcup\big\{[\![T]\!](s) \;\big|\; s \in \mathrm{DOM}(T)\big\} = p\big[\mathrm{pref}(q_1), \ldots, \mathrm{pref}(q_r)\big].$$

Proof. For the complete lattice $\mathcal{P}_\Delta$, we construct the following system of in-equations for the unknown patterns Y_q, $q \in Q_P$:

$$Y_q \sqsupseteq p[Y_{q_1}, \ldots, Y_{q_r}] \quad \text{whenever } \delta(q, a) = p\big[q_1(x_{i_1}), \ldots, q_r(x_{i_r})\big].$$

Here, we define substitution to be strict, meaning that $p[p_1,\dots,p_r]=\bot$ whenever $p_i=\bot$ for some i. Each right-hand side in this constraint system is monotonic in its arguments, and hence the system has a least solution. A closer look reveals that it is distributive for argument sequences of patterns, i.e., for any nonempty set $\mathcal{S}$ of sequences $(p_1,\dots,p_r)$ with $p_j\neq\bot$ for all $j=1,\dots,r$ and least upper bound $(\bar p_1,\dots,\bar p_r)$,

$$p[\bar p_1,\dots,\bar p_r]=\bigsqcup\big\{p[p_1,\dots,p_r]\;\big|\;(p_1,\dots,p_r)\in\mathcal{S}\big\}.$$

Note that it is crucial that we have joint distributivity w.r.t. nonempty sets of sequences of patterns and not just distributivity in each component separately: the reason is that during the computation of the transducer the different components may not be chosen independently of each other.

First, we show that the patterns $\mathsf{pref}(q)$, $q\in Q_P$, are a solution of the system of in-equations. For that, let $\delta(q,a)=p[q_1(x_{i_1}),\dots,q_r(x_{i_r})]$ be a processing transition of T. We claim that:

$$\mathsf{pref}(q)\sqsupseteq p\big[\mathsf{pref}(q_1),\dots,\mathsf{pref}(q_r)\big].$$

Let $\rho(q)=c$, where ρ is the relevance map of T. By (4) of Lemma 4,

$$\begin{aligned}\mathsf{pref}(q)&=\bigsqcup\big\{[\![q]\!](s)\;\big|\;s\in\mathrm{DOM}(c)\big\}\\&\sqsupseteq\bigsqcup\big\{p\big[[\![q_1]\!](s_{i_1}),\dots,[\![q_r]\!](s_{i_r})\big]\;\big|\;s=a(s_1,\dots,s_k)\in\mathrm{DOM}(c)\big\}\\&=\bigsqcup\big\{p\big[[\![q_1]\!](s_{i_1}),\dots,[\![q_r]\!](s_{i_r})\big]\;\big|\;s_1\in\mathrm{DOM}(c_1),\dots,s_k\in\mathrm{DOM}(c_k)\big\}\end{aligned}$$

where $\delta(c,a)=c_1\cdots c_k$ is an inspecting transition of T, which exists by uniformity property (2)(a). By joint distributivity, the least upper bound operation can be pushed inwards:

$$\begin{aligned}\dots&=p\Big[\bigsqcup\big\{\big([\![q_1]\!](s_{i_1}),\dots,[\![q_r]\!](s_{i_r})\big)\;\big|\;s_1\in\mathrm{DOM}(c_1),\dots,s_k\in\mathrm{DOM}(c_k)\big\}\Big]\\&=p\Big[\bigsqcup\big\{[\![q_1]\!](s_1)\;\big|\;s_1\in\mathrm{DOM}(c_{i_1})\big\},\dots,\bigsqcup\big\{[\![q_r]\!](s_r)\;\big|\;s_r\in\mathrm{DOM}(c_{i_r})\big\}\Big]\\&=p\big[\mathsf{pref}(q_1),\dots,\mathsf{pref}(q_r)\big]\end{aligned}$$

where the latter equality follows again from (4) of Lemma 4, because $\rho(q_j)=c_{i_j}$ by uniformity property (2)(b).

In a similar way we can show the second statement of this theorem: By definition of $[\![T]\!]$, (2) of Lemma 4, joint distributivity, uniformity property (1), and (4) of Lemma 4,

$$\begin{aligned}\bigsqcup\big\{[\![T]\!](s)\;\big|\;s\in\mathrm{DOM}(T)\big\}&=\bigsqcup\big\{p\big[[\![q_1]\!](s),\dots,[\![q_r]\!](s)\big]\;\big|\;s\in\mathrm{DOM}(c_0)\big\}\\&=p\Big[\bigsqcup\big\{[\![q_1]\!](s)\;\big|\;s\in\mathrm{DOM}(c_0)\big\},\dots,\bigsqcup\big\{[\![q_r]\!](s)\;\big|\;s\in\mathrm{DOM}(c_0)\big\}\Big]\\&=p\big[\mathsf{pref}(q_1),\dots,\mathsf{pref}(q_r)\big].\end{aligned}$$

Now let y_q, $q\in Q_P$, denote *any* solution of our system of in-equations. We claim that $y_q\sqsupseteq[\![q]\!](s)$ for every $q\in Q$ and every input $s\in\mathrm{DOM}(q)$. From this claim, we deduce that

$$y_q\sqsupseteq\bigsqcup\big\{[\![q]\!](s)\;\big|\;s\in\mathrm{DOM}(q)\big\}=\mathsf{pref}(q).$$

Thus, the patterns $\mathsf{pref}(q)$, $q\in Q_P$, constitute not just some solution of the system of in-equations, but the least solution.

We prove the claim by structural induction on s. Assume that $s=a(s_1,\dots,s_k)\in\mathrm{DOM}(q)$ and $\delta(q,a)=p[q_1(x_{i_1}),\dots,q_r(x_{i_r})]$ is a transition of T. Since y_q, $q\in Q_P$, is a solution of the system of in-equations, we have: $y_q\sqsupseteq p[y_{q_1},\dots,y_{q_r}]$.

By induction hypothesis for the s_i, $y_{q_j}\sqsupseteq[\![q_j]\!](s_{i_j})$, and therefore by monotonicity,

$$y_q\sqsupseteq p[y_{q_1},\dots,y_{q_r}]\sqsupseteq p\big[[\![q_1]\!](s_{i_1}),\dots,[\![q_r]\!](s_{i_r})\big]=[\![q]\!](s).$$

This completes the proof of the claim.

In order to compute the least solution of our system of in-equations, we first compute for every processing state q an output tree $t_q=[\![q]\!](s)$ for some tree $s\in S_{\rho(q)}$. As mentioned before this theorem, such trees t_q can be computed in time $\mathcal{O}(|T|\cdot\eta(T))$. For each q, the tree t_q is a lower bound for the pattern $\mathsf{pref}(q)$, i.e., $\mathsf{pref}(q)\sqsupseteq t_q$. Since $\mathrm{size}(t_q)\leqslant\eta(T)$, the size of $\mathsf{pref}(q)$ is at most $\eta(T)$. Taking t_q as the initial value of the variable Y_q, subsequent fixpoint iteration will compute the least solution, only replacing subtrees of t_q with $\top$. Therefore, the number of updates to the variable Y_q is bounded by $\mathrm{size}(t_q)\leqslant\eta(T)$, and the least solution can be computed in time quadratic in $|T|\cdot\eta(T)$.

In the remainder of this proof we describe an algorithm that computes the least solution in time linear in $|T|\cdot\eta(T)$. Construct a directed graph $G=(V,E)$ and a subset S of V, as follows. The set V of nodes consists of all pairs $\langle q,v\rangle$ with v a node of t_q. The set $S\subseteq V$ consists of all $\langle q,v\rangle$ such that v has an ancestor w with the following property: there is an in-equation $Y_q\sqsupseteq p[Y_{q_1},\dots,Y_{q_r}]$ such that w is a node of $p[t_{q_1},\dots,t_{q_r}]$, with a label different from its label in t_q. Note that if $\langle q,v\rangle\notin S$, then v is a node of $p[t_{q_1},\dots,t_{q_r}]$ for every in-equation $Y_q\sqsupseteq p[Y_{q_1},\dots,Y_{q_r}]$. Finally, the set of edges E consists

of all pairs $(\langle q', v'\rangle, \langle q, v\rangle)$ such that $\langle q, v\rangle \notin S$ and there are an in-equation $Y_q \sqsupseteq p[Y_{q_1}, \ldots, Y_{q_r}]$ and a $j \in \{1, \ldots, r\}$ such that $q' = q_j$ and $v = u_j.v'$, where u_j is the jth node of p labeled with $\top$ (i.e., v is a node of $p[t_{q_1}, \ldots, t_{q_r}]$ that "corresponds to" node v' of t_{q_j}). Intuitively, if $\langle q, v\rangle \in S$ then in the first round of fixpoint iteration (with t_q as initial value of Y_q), node v is removed from t_q (or replaced by $\top$); and an edge $(\langle q', v'\rangle, \langle q, v\rangle)$ means that if v' is removed from $t_{q'}$, then in the next round v is removed from t_q.

The set S can be computed by a depth-first left-to-right traversal of t_q for each in-equation $Y_q \sqsupseteq p[Y_{q_1}, \ldots, Y_{q_r}]$, simultaneously traversing $p[t_{q_1}, \ldots, t_{q_r}]$. Since each such traversal takes time size(t_q), the total time is linear in $|T| \cdot \eta(T)$. Then the set E can be computed in a similar way. Thus, G has size $\mathcal{O}(|T| \cdot \eta(T))$.

Now define p_q to be the pattern such that $p_q \sqsupseteq t_q$, and $v \in V(t_q)$ is a non-$\top$ node of p_q iff $\langle q, v\rangle$ is not reachable from S in G. That p_q is indeed a pattern, can easily be proved: if $\langle q, w\rangle$ is reachable from S and w is an ancestor of v in t_q, then $\langle q, v\rangle$ is reachable from S. We now claim that the patterns p_q, $q \in Q_P$, are the least solution of the in-equations. We leave the straightforward proof to the reader: first show that p_q, $q \in Q_P$, is a solution, and then show that if y_q, $q \in Q_P$, is *any* solution with $y_q \sqsupseteq t_q$ for all q, then $y_q \sqsupseteq p_q$ for all q. The nodes $\langle q, v\rangle$ that are reachable from S can be determined by depth-first search of G, in time linear in its size. Hence, the patterns p_q can be determined in time $\mathcal{O}(|T| \cdot \eta(T))$. □

Consider a total transducer T. For every processing state q and every symbol e of rank 0, T has a transition $q(e) \to t$ for some tree $t \in \mathcal{T}_\Delta$. A rough upper bound to the sizes of such trees is given by the size of T itself, and so $\eta(T) \leqslant |T|$. Hence, according to Theorem 8, the common prefixes for all processing states can be computed for total transducers in *quadratic* time, i.e., in time $\mathcal{O}(|T|^2)$.

In the case of non-total uniform transducers, we do not have at hand the small trees $\delta(q, e)$ as for total transducers. Instead, however, we can rely for a processing state q, on *some* output tree t returned by $[\![q]\!]$ on a relevant input tree $s \in \text{DOM}(\rho(q))$ of minimal depth. Obviously, the depth of such a tree s is at most the number of inspecting states of T, which is at most $|T|$. Accordingly, the size of the output tree $t = [\![q]\!](s)$ is at most exponential in $|T|$. Hence, the value $\eta(T)$ of Theorem 8 can be at most exponential in the size of T, and so the common prefixes of a uniform i-transducer can be computed in *exponential* time.

Example 9. Consider the total transducer T_1 of Example 3 with the transitions $\delta(q, a) = d(q(x_1), d(q(x_1), e))$ and $\delta(q, e) = d(d(e, e), d(e, e))$. The corresponding system of in-equations is

$$Y_q \sqsupseteq d\big(Y_q, d(Y_q, e)\big),$$
$$Y_q \sqsupseteq d\big(d(e, e), d(e, e)\big).$$

Fixpoint iteration (starting with $y^{(0)} = \bot$) terminates after only three rounds:

$$y^{(1)} = d\big(d(e, e), d(e, e)\big),$$
$$y^{(2)} = d\big(d(\top, \top), d(\top, e)\big) = y^{(3)},$$

where $y^{(i)}$ denotes the ith Kleene approximation of the least solution for the variable Y_q.

Clearly, $s_{\rho(q)} = e$ and $t_q = [\![q]\!](e) = d(d(e, e), d(e, e))$. Thus, initializing Y_q with t_q gives the same iteration as above. For the graph $G = (V, E)$, $V = \{\langle q, v\rangle \mid v \in V(t_q)\}$. The sets S and E are obtained by comparing t_q with $p[t_q, t_q]$, where $p = d(\top, d(\top, e))$ is the pattern of $\delta(q, a)$. This gives $S = \{\langle q, 1.1\rangle, \langle q, 1.2\rangle, \langle q, 2.1\rangle\}$ and $E = \{(\langle q, \varepsilon\rangle, \langle q, 1\rangle)\}$. Hence the only reachable nodes of G are those in S, and so p_q is obtained from t_q by replacing nodes 1.1, 1.2, and 2.1 by $\top$, i.e., $p_q = d(d(\top, \top), d(\top, e))$.

5. Earliest transducers

A uniform i-transducer T is called *earliest* if for every processing state q of T there exist input trees s_1 and s_2 such that the roots of $[\![q]\!](s_1)$ and $[\![q]\!](s_2)$ have different labels. In other words, $\text{pref}(q) = \top$ for all processing states q. In particular, by the second statement of Theorem 8, this implies that the pattern p of the axiom of T equals the least upper bound of all outputs produced by T, i.e.,

$$p = \bigsqcup \big\{ [\![T]\!](s) \;\big|\; s \in \text{DOM}(T) \big\}.$$

Example 10. The uniform i-transducer $T'_X = (P', I)$ of Example 7 is not earliest, because the roots of all outputs of q_0, id_e, and n are labeled doc, title, and nil, respectively. However, T'_X can easily be turned into an equivalent earliest transducer $T''_X = (P,'' I)$, as follows.

Replace state q_0 by two states, q_0^1 and q_0^2, and similarly for state id_e. Furthermore, remove state n and remove the transitions for q_0, id_e, and n. Replace the axiom by $\text{doc}(\text{toc}(q_0^1(x_0), q_0^2(x_0)), \text{nil})$, change the transition for id_t and sec into $id_t(\text{sec}(x_1, x_2)) \to \text{sec}(\text{title}(id_e^1(x_1), id_e^2(x_1)), id_t(x_2))$, and add the following transitions for the new states:

$$q_0^1\big(\mathrm{doc}(x_1,x_2)\big) \to t(x_1),$$
$$q_0^2\big(\mathrm{doc}(x_1,x_2)\big) \to id_t(x_1),$$
$$id_e^1\big(\mathrm{title}(x_1,x_2)\big) \to id(x_1),$$
$$id_e^2\big(\mathrm{title}(x_1,x_2)\big) \to id(x_2).$$

Note that P'' does not process the second input subtree of doc any more, but that I still inspects that tree to check that it is nil. This is the reason that we had to add the inspection facility to the top–down tree transducer in order to obtain our normal form result.

We now prove the normal form result mentioned in Abstract and discussed in Introduction.

Theorem 11. *Every i-transducer T with $\mathrm{DOM}(T) \neq \emptyset$ is effectively equivalent to an earliest i-transducer T'.*
If T is uniform, then T' can be constructed in time $\mathcal{O}(|T| \cdot \eta(T))$.
If T is total, then T' can be constructed in time $\mathcal{O}(|T|^2)$.

Proof. By Lemma 6, we can construct for every transducer with nonempty domain an equivalent uniform transducer. Therefore, assume that the i-transducer $T = (P, I)$ is uniform. By Theorem 8, we can compute for every processing state q of T, the pattern $\mathsf{pref}(q)$ which is common to all outputs produced by q. The idea then is to produce this common prefix as early as possible. Together with the state q, we additionally record the node v in the pattern $\mathsf{pref}(q)$ which is to be expanded next. This means that the processing states of the new i-transducer $T' = (P', I)$ are of the form $\langle q, v\rangle$ where $q \in Q_P$, and v is one of the nodes of $\mathsf{pref}(q)$ labeled with $\top$. Note that the inspecting dtta of T' is the same as the one of T. To ensure that all states of P' are reachable, we will define the states and transitions of P' inductively.

If $A = p[q_1(x_0), \ldots, q_r(x_0)]$ is the axiom of T, then the axiom A' of T' is given by:

$$A' = p\big[p_1\big[\langle q_1, v_{1,1}\rangle(x_0), \ldots, \langle q_1, v_{1,l_1}\rangle(x_0)\big], \ldots, p_r\big[\langle q_r, v_{r,1}\rangle(x_0), \ldots, \langle q_r, v_{r,l_r}\rangle(x_0)\big]\big]$$

where $v_{j,1}, \ldots, v_{j,l_j}$ is the left-to-right sequence of nodes in $p_j = \mathsf{pref}(q_j)$ labeled with $\top$. All pairs $\langle q, v\rangle$ in A' are new states in P'.

For a new state $\langle q, v\rangle$ of P' and an input symbol a, assume that

$$\delta(q,a) = p\big[q_1(x_{i_1}), \ldots, q_r(x_{i_r})\big]$$

in P. Let $p_j = \mathsf{pref}(q_j)$ for $j = 1, \ldots, r$. From the proof of Theorem 8 we know that $\mathsf{pref}(q) \sqsupseteq p[p_1, \ldots, p_r]$. Hence, v is a node of $p[p_1, \ldots, p_r]$. Then we define

$$\delta'(\langle q, v\rangle, a) = \text{ the subtree at node } v \text{ of the tree}$$
$$p\big[p_1\big[\langle q_1, v_{1,1}\rangle(x_{i_1}), \ldots, \langle q_1, v_{1,l_1}\rangle(x_{i_1})\big], \ldots, p_r\big[\langle q_r, v_{r,1}\rangle(x_{i_r}), \ldots, \langle q_r, v_{r,l_r}\rangle(x_{i_r})\big]\big]$$

in P', where $v_{j,1}, \ldots, v_{j,l_j}$ is the left-to-right sequence of nodes in p_j labeled with $\top$. All pairs $\langle q', v'\rangle$ in $\delta'(\langle q, v\rangle, a)$ are further states of P'.

Since $\mathrm{DOM}(I) = \mathrm{DOM}(T)$ by (2) of Lemma 4, the following claim suffices to prove that T' and T are equivalent. It can easily be shown by structural induction on input tree s.

Claim: For every processing state $\langle q, v\rangle$ of T' and every input tree $s \in \mathrm{DOM}(q)$, $[\![\langle q, v\rangle]\!](s)$ is defined and equals the subtree of $[\![q]\!](s)$ at node v.

It should be clear that T' is uniform with relevance map ρ' defined by $\rho'(\langle q, v\rangle) = \rho(q)$, where ρ is the relevance map of T. The transducer T' is also earliest: For every processing state $\langle q, v\rangle$ of T', node v in $\mathsf{pref}(q)$ is labeled $\top$. By the definition of pref, this means that there are input trees s_1 and s_2 such that the label of v in $[\![q]\!](s_1)$ is different from the label of v in $[\![q]\!](s_2)$; hence, by the claim above, the roots of $[\![\langle q, v\rangle]\!](s_1)$ and $[\![\langle q, v\rangle]\!](s_2)$ have different labels.

It remains to consider the complexity bounds stated in the theorem. According to Theorem 8, the given bounds are sufficient to compute the common prefixes $\mathsf{pref}(q)$, which are of size at most $\eta(T)$. Every transition of T for a processing state q gives rise to at most $\eta(T)$ transitions of T' for processing states $\langle q, v\rangle$ where the sum of the sizes of all right-hand sides is bounded by the original size times $\eta(T)$. Moreover, each new right-hand side can be produced in time linear in its size. A similar statement holds for the axioms. Hence, the construction of T' from T takes time $\mathcal{O}(|T| \cdot \eta(T))$. □

Example 12. Consider again the total transducer T_1 of Example 3, with axiom $A = q(x_0)$ and transitions

$$q\big(a(x_1,x_2)\big) \to d\big(q(x_1), d\big(q(x_1), e\big)\big),$$
$$q(e) \to d\big(d(e,e), d(e,e)\big).$$

We have seen in Example 9 that $\mathrm{pref}(q) = d(d(\top,\top), d(\top, e))$. Thus, the states of the new transducer T_1' are $\langle q, 1.1\rangle$, $\langle q, 1.2\rangle$, and $\langle q, 2.1\rangle$, which we will denote by 1, 2, and 3, respectively. The axiom of the new transducer T' is

$$A' = d\big(d\big(1(x_0), 2(x_0)\big), d\big(3(x_0), e\big)\big).$$

Let us now construct the transitions of T_1' corresponding to the first transition of T_1. Note that $\delta(q, a) = p[q(x_1), q(x_1)]$ for the pattern $p = d(\top, d(\top, e))$. Let $t = \mathrm{pref}(q)[1(x_1), 2(x_1), 3(x_1)] = d(d(1(x_1), 2(x_1)), d(3(x_1), e))$. Then $\delta'(\langle q, v\rangle, a)$ is the subtree at node v of the tree $p[t, t] = d(t, d(t, e))$. For $v = 1.1$, 1.2, 2.1 these subtrees are: the first subtree of t, the second subtree of t, and t itself, respectively. Thus, the new transitions are

$$1\big(a(x_1, x_2)\big) \to d\big(1(x_1), 2(x_1)\big),$$
$$2\big(a(x_1, x_2)\big) \to d\big(3(x_1), e\big),$$
$$3\big(a(x_1, x_2)\big) \to d\big(d\big(1(x_1), 2(x_1)\big), d\big(3(x_1), e\big)\big).$$

Since $\delta'(\langle q, v\rangle, e)$ is the subtree at node v of $\delta(q, e)$, the transitions of T_1' corresponding to the second transition of T_1 are $i(e) \to e$ for $i = 1, 2, 3$.

6. Minimizing earliest transducers

The key property of earliest transducers is that they produce the respective output trees in a canonical fashion. This means that for two processing states q and q' of an earliest i-transducer, with $\rho(q) = \rho(q')$, the (partial) functions $[\![q]\!]$ and $[\![q']\!]$ are equal if and only if the patterns on the right-hand sides of all corresponding transitions are equal and the corresponding recursive calls in the right-hand sides agree.

Formally, let T be an earliest i-transducer with relevance map ρ. On the set Q_P of processing states of T we define the relation $\equiv$ to be the largest equivalence relation $\sim$ that satisfies the following property $(*)$:

If $q \sim q'$, then

(a) $\rho(q) = \rho(q')$, and
(b) if $\delta(q, a) = p[q_1(x_{i_1}), \ldots, q_r(x_{i_r})]$ and $\delta(q', a) = p'[q'_1(x_{i'_1}), \ldots, q'_{r'}(x_{i'_{r'}})]$, then $p = p'$, $r = r'$, and for all $j = 1, \ldots, r$, $i_j = i'_j$ and $q_j \sim q'_j$.

Note that $\equiv$ is well defined. In fact, the set of equivalence relations on Q_P is a complete lattice with respect to $\supseteq$ (the inverse of inclusion), with intersection as join and $Q_P \times Q_P$ as bottom element. For a given equivalence relation $\sim$, let $f(\sim)$ be the equivalence relation such that $qf(\sim)q'$ iff statements (a) and (b) hold. Since f is monotone, it has a least fixpoint, which is the equivalence relation $\equiv$. Thus, $\equiv$ equals $\bigcup_{i \geqslant 0} f^i(Q_P \times Q_P)$ and can be computed by fixpoint iteration. Clearly, the number of iterations is at most $|Q_P|$, and each iteration step compares at most $|Q_P|^2$ pairs of states. Hence the total number of comparisons is at most $|Q_P|^3$. However, if one keeps track of representatives of the equivalence classes of $f^i(Q_P \times Q_P)$, then it is not difficult to see that at most $\mathcal{O}(|Q_P|^2)$ comparisons are needed. Since each comparison takes at most $\mathcal{O}(|T|)$ time, $\equiv$ can be computed in time $\mathcal{O}(|T|^3)$.

The equivalence relation $\equiv$ and its computation are similar to those in the minimization of deterministic top–down tree automata, cf. [12,13,21] and Section 3.

Theorem 13. *Let $T = (Q, \Sigma, \Delta, \delta, A, c_0)$ be an earliest i-transducer, with relevance map ρ. Then the following holds*:

(1) *The equivalence relation $\equiv$ can be computed in time $\mathcal{O}(|T|^3)$.*
(2) *For processing states q, q' of T with $\rho(q) = \rho(q')$, the following three statements are equivalent*:
  (a) $q \equiv q'$;
  (b) $[\![q]\!] = [\![q']\!]$;
  (c) $[\![q]\!](s) = [\![q']\!](s)$ *for all* $s \in \mathrm{DOM}(\rho(q))$.

Proof. Since assertion (1) was proved above, it remains to prove assertion (2).

(a) $\Rightarrow$ (b). Assume that $q \equiv q'$. It is straightforward to show by structural induction on input trees s that $s \in \mathrm{DOM}(q)$ iff $s \in \mathrm{DOM}(q')$, and that $[\![q]\!](s) = [\![q']\!](s)$ whenever $s \in \mathrm{DOM}(q)$. Hence $[\![q]\!] = [\![q']\!]$.

(b) $\Rightarrow$ (c). It is obvious by (1) of Lemma 4.

(c) $\Rightarrow$ (a). To prove this, it suffices to show that the relation $\sim$, defined as follows, satisfies the property $(*)$ above: For $q, q' \in Q_P$, $q \sim q'$ iff $\rho(q) = \rho(q')$ and $[\![q]\!](s) = [\![q']\!](s)$ for all $s \in \mathrm{DOM}(\rho(q))$.

So, assume that $q \sim q'$, and let $c = \rho(q) = \rho(q')$. Assume further that $\delta(q, a) = p[q_1(x_{i_1}), \ldots, q_r(x_{i_r})]$ and $\delta(q', a) = p'[q'_1(x_{i'_1}), \ldots, q'_{r'}(x_{i'_{r'}})]$ in T. Then, by uniformity property (2)(a), $\delta(c, a)$ is defined. Let $\delta(c, a) = c_1 \cdots c_k$. Now consider arbitrary input trees $s_i \in \mathrm{DOM}(c_i)$, $i = 1, \ldots, k$. Note that since $\mathrm{DOM}(c_i) \neq \emptyset$, there is at least one such tree for every i. Then $s = a(s_1, \ldots, s_k)$ is in $\mathrm{DOM}(c)$. Therefore, since $q \sim q'$,

$$p\big[[\![q_1]\!](s_{i_1}), \ldots, [\![q_r]\!](s_{i_r})\big] = [\![q]\!](s) = [\![q']\!](s) = p'\big[[\![q'_1]\!](s_{i'_1}), \ldots, [\![q'_{r'}]\!](s_{i'_{r'}})\big].$$

We must show that $p = p'$, $r = r'$, and for $j \in \{1, \ldots, r\}$, $i_j = i'_j$ and $q_j \sim q'_j$. For a contradiction, assume that $p \neq p'$. If at some node v the patterns p and p' have different labels in Δ, then $[\![q]\!](s) \neq [\![q']\!](s)$. Since each symbol in the output alphabet Δ has a unique rank, it suffices to consider the case that p' at some node v is labeled by a $d \in \Delta$ and p at v is labeled $\top$. Let v be the jth occurrence of $\top$ in p. This means that $[\![q_j]\!](s_{i_j})$ is a tree of the form $d(\cdots)$. Since this holds for arbitrary $s_{i_j} \in \mathrm{DOM}(c_{i_j})$, we obtain from (4) of Lemma 4 (which is applicable by uniformity property (2)(b)) that $\mathrm{pref}(q_j) = d(\cdots) \neq \top$, which contradicts the earliest property of T. Hence, $p = p'$ and $r = r'$. Next, we show that for all j, $i_j = i'_j$. This follows by a similar argument: assume for a contradiction that $i_j \neq i'_j$. Then the output trees produced by both q_j and q'_j cannot depend on their input trees. If, however, $[\![q_j]\!](s) = t$ for every $s \in \mathrm{DOM}(c_{i_j})$, then $t = \mathrm{pref}(q_j)$ contradicting the earliest assumption on T. Consequently, $i_j = i'_j$ for all j.

Finally we show that for all j, $q_j \sim q'_j$. By uniformity property (2)(b), $\rho(q_j) = \rho(q'_j) = c_{i_j}$. Hence it remains to prove that $[\![q_j]\!](\bar{s}) = [\![q'_j]\!](\bar{s})$ for all $\bar{s} \in \mathrm{DOM}(c_{i_j})$. As before, consider arbitrary input trees $s_i \in \mathrm{DOM}(c_i)$, with $s_{i_j} = \bar{s}$. Then $s = a(s_1, \ldots, s_k)$ is in $\mathrm{DOM}(c)$, and

$$p\big([\![q_1]\!](s_{i_1}), \ldots, [\![q_r]\!](s_{i_r})\big) = [\![q]\!](s) = [\![q']\!](s) = p\big([\![q'_1]\!](s_{i_1}), \ldots, [\![q'_r]\!](s_{i_r})\big).$$

We conclude that $[\![q_j]\!](s_{i_j}) = [\![q'_j]\!](s_{i_j})$. □

From an earliest i-transducer T and the equivalence relation $\equiv$ defined above, we can construct, in linear time, a new i-transducer $T_{\equiv}$ by replacing each processing state by its equivalence class w.r.t. $\equiv$. The resulting transducer is equivalent to T and again an earliest transducer.

In fact, suppose that $q_1 \equiv q_2$ and replace q_1 by q_2 in the axiom or in the right-hand side of some transition of T. By Theorem 13, $[\![q_1]\!] = [\![q_2]\!]$. Hence, by the definition of the semantics of a t-transducer, this does not change $[\![T]\!]$ or any of the $[\![q]\!]$. Also, because $\rho(q_1) = \rho(q_2)$, the same relevance map ρ still satisfies the uniformity properties. Since $[\![q]\!]$ has not changed, neither has $\mathrm{pref}(q)$. Hence the transducer is still earliest. Thus, we can pick a representative of each equivalence class of $\equiv$, iterate this replacement procedure, and finally disregard all states that are not representatives.

We will say that an earliest i-transducer T is *canonical* if for all processing states q, q' of T: if $\rho(q) = \rho(q')$ and $q \neq q'$, then $[\![q]\!] \neq [\![q']\!]$ (i.e., if every equivalence class of $\equiv$ is a singleton). Note that $T_{\equiv}$ is canonical. From the above discussion and Theorem 13 we obtain the following result.

Theorem 14. *For every earliest i-transducer T a canonical i-transducer T' can be constructed in time $\mathcal{O}(|T|^3)$ such that $[\![T']\!] = [\![T]\!]$.*

Thus, by Theorem 11, every i-transducer T with nonempty domain is equivalent to a canonical i-transducer. In the next theorem we prove that (up to renaming of states) that canonical transducer is *unique*. Thus, it is a unique minimal earliest i-transducer realizing $[\![T]\!]$. Here, minimality is meant w.r.t. to the number of processing states q with $\rho(q) = c$, for each inspecting state c.

Theorem 15. *Let T_1, T_2 be equivalent canonical i-transducers. Then T_1 and T_2 are the same* (*up to renaming of states*).

Proof. Since $T_1 = (P_1, I_1)$ and $T_2 = (P_2, I_2)$ are equivalent, their domains are equal. Thus, by (2) of Lemma 4, I_1 and I_2 are equivalent minimal dtta's, and so, by Proposition 2, they are the same. Thus, in what follows we assume w.l.o.g. that T_1 and T_2 have the same inspecting dtta: $T_1 = (P_1, I)$ and $T_2 = (P_2, I)$ with $\mathrm{DOM}(I) = \mathrm{DOM}(T_1) = \mathrm{DOM}(T_2)$. Assume w.l.o.g. that $Q_{P_1} \cap Q_{P_2} = \emptyset$. It remains to show that P_1 and P_2 are the same.

From the transducers T_1, T_2, we construct an i-transducer $T = (P, I)$ where the set of processing states of T is given by $Q_P = Q_{P_1} \cup Q_{P_2}$, the transition function δ_P of T is the union of δ_{P_1} and δ_{P_2}, the relevance map ρ is the union of the relevance maps of T_1 and T_2, and the axiom of T is given by $A = ::(A_1, A_2)$ for a new output symbol :: where $A_\nu = p_\nu[q_1^{(\nu)}(x_0), \ldots, q_{r^{(\nu)}}^{(\nu)}(x_0)]$ is the axiom of T_ν ($\nu = 1, 2$).

Obviously, T is uniform w.r.t. ρ. Moreover, it should be clear that for every state $q \in Q_{P_\nu}$, $[\![q]\!]$ is the same in T and T_ν. So, $\mathrm{pref}(q)$ is also the same, and hence T is earliest.

Let us have a look at the axioms of T_1, T_2, and T. We first observe that $p_1 = p_2$ since:

$$p_1 = \bigsqcup\big\{[\![T_1]\!](s) \mid s \in \mathrm{DOM}(I)\big\} = \bigsqcup\big\{[\![T_2]\!](s) \mid s \in \mathrm{DOM}(I)\big\} = p_2$$

which follows from the facts that T_1, T_2 are earliest (cf. the beginning of Section 5) and equivalent. Then also $r^{(1)} = r^{(2)} =: r$.

Furthermore (for all $j = 1, \ldots, r$), $[\![q_j^{(1)}]\!](s) = [\![q_j^{(2)}]\!](s)$ for all $s \in \mathrm{DOM}(I)$. Let c_0 be the initial state of I. By uniformity property (1), $\rho(q_j^{(1)}) = \rho(q_j^{(2)}) = c_0$. Therefore by Theorem 13, $q_j^{(1)} \equiv q_j^{(2)}$ for all $j = 1, \ldots, r$, where $\equiv$ is the equivalence relation on the states of T as defined for Theorem 13.

We now prove that the equivalence relation $\equiv$ constitutes a bijection between the sets Q_{P_1} and Q_{P_2}. For that, we first observe that for every $q_1 \in Q_{P_1}$ there exists $q_2 \in Q_{P_2}$ such that $q_1 \equiv q_2$, and vice versa. Since all states of T_1 and T_2 are reachable, this can easily be proved by induction on the definition of reachability. In fact, we just proved the base case of this induction, and the induction step is immediate from the definition of $\equiv$ (and uniformity property (2)(a)). Therefore, the

relation $\equiv$ is left- and right-total. Now assume that $q_1 \equiv q_2$ and $q_1 \equiv q_2'$. Then $\rho(q_1) = \rho(q_2) = \rho(q_2')$ and, by Theorem 13, $[\![q_1]\!] = [\![q_2]\!] = [\![q_2']\!]$. Hence $q_2 = q_2'$, because T_2 is canonical. The same argument applied with the roles of T_1 and T_2 exchanged, concludes the proof that $\equiv$ is a bijection between Q_{P_1} and Q_{P_2}.

It should be clear that P_1 and P_2 are the same up to the renaming $\equiv$ of states. We have already seen that their axioms are the same, and their processing transitions are the same by uniformity property (2)(a) and the definition of $\equiv$. □

As a consequence of Theorems 11, 14, and 15, we obtain the main result of this paper.

Theorem 16. *Every i-transducer T with $\mathrm{DOM}(T) \neq \emptyset$ is effectively equivalent to a unique canonical i-transducer $c(T)$. If T is uniform then $c(T)$ can be constructed in time $\mathcal{O}(|T|^3 \cdot \eta(T)^3)$. If T is total then $c(T)$ can be constructed in time $\mathcal{O}(|T|^6)$. If T is earliest, then $c(T)$ can be constructed in time $\mathcal{O}(|T|^3)$.*

We observe that if T is total then so is $c(T)$, because the constructions in Theorems 11 and 14 do not change the inspecting dtta.

Recall from the discussion following Theorem 8 that the structural parameter $\eta(T)$ is at most exponential in $|T|$. Thus, for arbitrary uniform transducers T, $c(T)$ can be constructed in exponential time. This implies, by Lemma 6, that our construction of $c(T)$ from T takes double exponential time in general. This is the best possible because, as shown in the next example, the size of $c(T)$ can be double exponential in the size of T.

Example 17. For every $n \geqslant 1$ we will describe a t-transducer $T_n = (Q, \Sigma, \Delta, \delta, A)$ of size $\mathcal{O}(n)$ such that the domain of T_n contains a single input tree s_0, of depth exponential in n, and $t_0 = [\![T_n]\!](s_0)$ has size double exponential in n. Since $[\![T_n]\!] = \{(s_0, t_0)\}$, the canonical i-transducer $c(T_n)$ has axiom t_0 (cf. the beginning of Section 5), and so the size of $c(T_n)$ is double exponential in the size of T_n.

Let $\Sigma = \{0, 1, \#, e\}$ where e has rank 0 and the other symbols have rank 1. Thus, the trees over Σ are monadic, and can be written (and viewed) as strings in the usual way, e.g., the string $01e$ denotes the tree $0(1(e))$. A unique input tree in the domain of T_n is

$$s_0 = w_0 \# w_1 \# \cdots \# w_{2^n-1} e$$

where w_j is the reverse of the binary representation of the number j, of length n. Thus, for $n = 3$, $s_0 = 000 \# 100 \# 010 \# 110 \# 001 \# 101 \# 011 \# 111e$. To recognize s_0, transducer T_n uses n parallel computations where the ith computation checks the correctness of the ith bits of all w_j.

Thus, T_n has states of the form $\langle p, q, r\rangle$ with $p \in \{1, \ldots, n+1\}$, $q \in \{\#, 1, 0\}$, and $r \in \{0, 1\}$. The first component is a counter that enables T_n to walk from one ith bit to the next ith bit. The second component indicates that, during this walk, the symbol # has not yet been encountered (#), or that it was encountered, and after that no 0 was read (1) or at least one 0 was read (0). At the end of the walk, the third component is checked to be equal to the current ith bit, and, on the basis of the second component, is changed to the expected value of the next ith bit.

Let $\Delta = \{b, a, e\}$ with ranks n, 2, and 0, respectively. The axiom of T_n is $A = b(\langle n+1, 1, 0\rangle(x_0), \ldots, \langle 2, 1, 0\rangle(x_0))$. The ith state $\langle n - i + 2, 1, 0\rangle$ of A is going to check the ith bits. In the transitions of T_n that follow, we use x to stand for x_1, we assume that $r, r' \in \{0, 1\}$ and $i \in \{1, \ldots, n\}$, and we use dots to indicate that the second subtree of a is identical to the first subtree. The latter means that the output tree produced by each state is a full binary tree over $\{a, e\}$ of the same depth as the input tree, and hence of size double exponential in n.

$$\langle n+1, 1, 1\rangle\big(1(x)\big) \to a\big(\langle 1, \#, 0\rangle(x), \ldots\big),$$
$$\langle n+1, 1, 0\rangle\big(0(x)\big) \to a\big(\langle 1, \#, 1\rangle(x), \ldots\big),$$
$$\langle n+1, 0, r\rangle\big(r(x)\big) \to a\big(\langle 1, \#, r\rangle(x), \ldots\big),$$
$$\langle i, \#, r\rangle\big(r'(x)\big) \to a\big(\langle i+1, \#, r\rangle(x), \ldots\big),$$
$$\langle i, \#, r\rangle\big(\#(x)\big) \to a\big(\langle i+1, 1, r\rangle(x), \ldots\big),$$
$$\langle i, 1, r\rangle\big(1(x)\big) \to a\big(\langle i+1, 1, r\rangle(x), \ldots\big),$$
$$\langle i, 1, r\rangle\big(0(x)\big) \to a\big(\langle i+1, 0, r\rangle(x), \ldots\big),$$
$$\langle i, 0, r\rangle\big(r'(x)\big) \to a\big(\langle i+1, 0, r\rangle(x), \ldots\big),$$
$$\langle i, \#, 0\rangle(e) \to e.$$

The last transition means that the input tree is only accepted when all bits wish to turn into 0.

We observe that since a t-transducer with monadic input trees is essentially an alternating finite automaton with universal branching only (when viewed as an acceptor of its domain), the above construction is closely related to the well-known fact that such automata are exponentially more succinct than deterministic finite automata, even for singleton languages.

It follows from Theorem 16 that two i-transducers T_1 and T_2 (with nonempty domains) are equivalent iff $c(T_1)$ and $c(T_2)$ are the same. Thus, as a corollary we (re)obtain the decidability of equivalence.

Theorem 18. *The equivalence of deterministic top–down tree transducers (with inspection) is decidable. If the transducers are total or earliest, then equivalence can be decided in polynomial time.*

By Theorem 16, equivalence of total transducers can be tested in time $\mathcal{O}(n^6)$, and equivalence of earliest transducers in time $\mathcal{O}(n^3)$, where n is the sum of the sizes of the transducers. We also observe that for i-transducers with a monadic output alphabet (which means that all output symbols have rank 0 or rank 1), all constructions in this paper can be done in polynomial time. Hence equivalence of such transducers (of which finite-state string transducers are a special case) can also be tested in polynomial time.

A useful extension of the top–down tree transducer is the top–down tree transducer with *regular look-ahead* [4]. Such a transducer can test its input subtrees for membership in arbitrary regular tree languages, by means of a deterministic bottom-up tree automaton called the "look-ahead automaton." Clearly this also extends the i-transducer, which can only restrict its input subtrees to dtta languages (cf. [10] where the i-transducer is shown to be equivalent to a particular type of top–down tree transducer with regular look-ahead, and cf. [26] for a survey on different types of regular look-ahead).

Alternatively, one can think of a transducer with regular look-ahead as the composition of two translations: The first translation relabels the input tree (each a-labeled node v of the input tree is relabeled by $\langle a, c_1, \ldots, c_k\rangle$ if the look-ahead automaton arrives in state c_i at the ith subtree of v). The second translation is an ordinary top–down tree transducer, running on the relabeled input tree. We can use the decision procedure for equivalence of deterministic top–down tree transducers [9] to decide equivalence of deterministic top–down tree transducers T_1, T_2 with regular look-ahead, but it is more convenient to use Theorem 18: Let rel be the relabeling that adds, to each node of the input tree, the look-ahead states at all children nodes for the look-ahead automata of both transducers. Then we construct for each T_i the i-transducer $T_i' = (P_i, I_i)$ which realizes the transduction $\{(\mathsf{rel}(s), [\![T_i]\!](s)) \mid s \in \mathrm{DOM}(T_i)\}$. The dtta I_i checks whether the input is a correct relabeling $\mathsf{rel}(s)$ of an input tree s of T_i, and the t-transducer P_i simulates T_i on the relabeled input tree. Clearly, T_1 is equivalent to T_2 iff T_1' is equivalent to T_2'.

Corollary 19. *The equivalence problem for deterministic top–down tree transducers with regular look-ahead is decidable.*

This corollary can be used to check whether or not two transducers (possibly with look-ahead) are equivalent on a given regular set R of input trees, i.e., on a generalized DTD, by letting their look-ahead automata (additionally) check membership of the input tree in R.

7. Open problems

In the context of XML there have been attempts to generalize top–down transducers to unranked trees, e.g., [16,17,22, 25]. Such transducers cannot be simulated by ordinary top–down tree transducers on ranked-tree encodings, because they implicitly support concatenation of trees. Is equivalence of such transducers decidable? Can they be transformed into a normal form similar to the one presented here?

Another popular model of tree transducer is the macro tree transducer [8,11,22]. It can be seen as a generalization of top–down tree transducers by adding context-parameters to states. It is a long standing open problem whether or not equivalence for deterministic macro tree transducers is decidable. Recently it has been proved that equivalence is decidable for deterministic macro tree transducers that are of linear size increase [6], i.e., for which the size of every output tree is bounded by a constant times the size of the corresponding input tree. Note that this result is incomparable to Theorem 18: the methods from [6] do not help whenever the transducers produce output whose size is not linearly bounded by the size of the corresponding input. Finally, note that the restriction of macro tree transducers to monadic output (all output symbols and states have rank 0 or rank 1) corresponds to the "top–down tree-to-string transducers" [5,7] for which it also still remains open whether or not equivalence is decidable.

References

[1] A.V. Aho, J.D. Ullman, Translations on a context-free grammar, Inform. and Control 19 (1971) 439–475.
[2] S. Amer-Yahia, S. Cho, L.V.S. Lakshmanan, D. Srivastava, Tree pattern query minimization, VLDB J. 11 (4) (2002) 315–331.
[3] A. Bouajjani, J. Esparza, A. Finkel, O. Maler, P. Rossmanith, B. Willems, P. Wolper, An efficient automata approach to some problems on context-free grammars, Inform. Process. Lett. 74 (2000) 221–227.
[4] J. Engelfriet, Top–down tree transducers with regular look-ahead, Math. Systems Theory 10 (1977) 289–303.
[5] J. Engelfriet, Some open questions and recent results on tree transducers and tree languages, in: R.V. Book (Ed.), Formal Language Theory, Perspectives and Open Problems, Academic Press, New York, 1980.
[6] J. Engelfriet, S. Maneth, The equivalence problem for deterministic MSO tree transducers is decidable, Inform. Process. Lett. 100 (2006) 206–212.
[7] J. Engelfriet, G. Rozenberg, G. Slutzki, Tree transducers, L systems, and two-way machines, J. Comput. System Sci. 20 (1980) 150–202.
[8] J. Engelfriet, H. Vogler, Macro tree transducers, J. Comput. System Sci. 31 (1985) 71–146.
[9] Z. Ésik, Decidability results concerning tree transducers I, Acta Cybernet. 5 (1980) 1–20.

[10] Z. Fülöp, S. Vágvölgyi, Top–down tree transducers with deterministic top–down look-ahead, Inform. Process. Lett. 33 (1989) 3–5.
[11] Z. Fülöp, H. Vogler, Syntax-Directed Semantics – Formal Models based on Tree Transducers, in: W. Brauer, G. Rozenberg, A. Salomaa (Eds.), EATCS Monographs in Theoretical Computer Science, Springer, Berlin, 1998.
[12] F. Gécseg, M. Steinby, Minimal ascending tree automata, Acta Cybernet. 4 (1978) 37–44.
[13] F. Gécseg, M. Steinby, Tree Automata, Akadémiai Kiadó, 1984.
[14] T.V. Griffiths, The unsolvability of the equivalence problem for Λ-free nondeterministic generalized machines, J. ACM 15 (1968) 409–413.
[15] J.E. Hopcroft, R. Motwani, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, second ed., Addison–Wesley, New York, 2001.
[16] S. Maneth, F. Neven, Structured Document Transformations Based on XSL, in: 7th Internat. Workshop on Database Programming Languages, DBPL'99, in: Lecture Notes in Comput. Sci., vol. 1949, Springer, Berlin, 2000, pp. 80–98.
[17] W. Martens, F. Neven, On the complexity of typechecking top–down XML transformations, Theoret. Comput. Sci. 336 (2005) 153–180.
[18] G. Miklau, D. Suciu, Containment and equivalence for a fragment of XPath, J. ACM 51 (2004) 2–45.
[19] T. Milo, D. Suciu, Index structures for path expressions, in: 7th Internat. Conference on Database Theory, ICDT'99, in: Lecture Notes in Comput. Sci., vol. 1540, Springer, Berlin, 1999, pp. 277–295.
[20] M. Mohri, Minimization algorithms for sequential transducers, Theoret. Comput. Sci. 234 (2000) 177–201.
[21] M. Nivat, A. Podelski, Minimal ascending and descending tree automata, SIAM J. Comput. 26 (1) (1997) 39–58.
[22] T. Perst, H. Seidl, Macro forest transducers, Inform. Process. Lett. 89 (2004) 141–149.
[23] W.C. Rounds, Mappings and grammars on trees, Math. Systems Theory 4 (1970) 257–287.
[24] J.W. Thatcher, Generalized^2 sequential machine maps, J. Comput. System Sci. 4 (1970) 339–367.
[25] A. Tozawa, Towards Static Type Inference for XSLT, in: ACM Symp. on Document Engineering, 2001, pp. 18–27.
[26] S. Vágvölgyi, Top–down tree transducers with two-way tree walking look-ahead, Theoret. Comput. Sci. 3 (1992) 43–74.
[27] M. Yannakakis, Algorithms for acyclic database schemes, in: 7th Internat. Conference on Very Large Data Bases, IEEE Computer Society Press, 1981, pp. 82–94.
[28] Z. Zachar, The solvability of the equivalence problem for deterministic frontier-to-root tree transducers, Acta Cybernet. 4 (1978) 167–177.