A learning algorithm for top-down XML transformations — Source link ↗

Aurélien Lemay, Sebastian Maneth, Joachim Niehren

Institutions: university of lille, NICTA

Related papers:

- Deciding equivalence of top--down XML transformations in polynomial time

- Macro Tree Translations of Linear Size Increase are MSO Definable

- Macro Tree Transducers

- Decidability results concerning tree transducers II.

- Top-down tree transducers with regular look-ahead

A Learning Algorithm for Top-Down XML Transformations

Aurélien Lemay, Sebastian Maneth, Joachim Niehren

A Learning Algorithm for Top-Down XML Transformations

Aurélien Lemay
University of Lille
Mostrare project, INRIA & LIFL

Sebastian Maneth
NICTA and UNSW
Sydney, Australia

Joachim Niehren
INRIA Lille
Mostrare project, INRIA & LIFL

ABSTRACT

A generalization from string to trees and from languages to translations is given of the classical result that any regular language can be learned from examples: it is shown that for any deterministic top-down tree transformation there exists a sample set of polynomial size (with respect to the minimal transducer) which allows to infer the translation. Until now, only for string transducers and for simple relabeling tree transducers, similar results had been known. Learning of deterministic top-down tree transducers (DTOPs) is far more involved because a DTOP can copy, delete, and permute its input subtrees. Thus, complex dependencies of labeled input to output paths need to be maintained by the algorithm. First, a Myhill-Nerode theorem is presented for DTOPs, which is interesting on its own. This theorem is then used to construct a learning algorithm for DTOPs. Finally, it is shown how our result can be applied to XML transformations (e.g. XSLT programs). For this, a new DTD-based encoding of unranked trees by ranked ones is presented. Over such encodings, DTOPs can realize many practically interesting XML transformations which cannot be realized on first-child/next-sibling encodings.

Categories and Subject Descriptors: I.2.6 [Learning]: Concept Learning

General Terms: Algorithms

Keywords: Tree transformation, top-down, transducer, learning algorithm, minimization, Myhill-Nerode equivalence.

1. INTRODUCTION

XML is a popular format for data exchange on the web. Many communities have defined standard XML dialects to exchange their data (for instance, XBRL for business data, or SPL for pharmaceutical products). Another common use of XML is as a storage format for data to be displayed on web pages. It allows to separate the logical content from the display content of the data. Different web pages can, dynamically, pull out required information from a larger XML document, and display it appropriately. Technically speaking, this amounts to transforming XML into HTML. The most commonly used programming language for this task is the W3C standard XSLT. More generally, XSLT is used to convert XML into XML. A drawback of XSLT programs is that they are inaccessible to non-experts and are difficult to write and maintain.

Imagine a system that is able to automatically infer an XSLT program from a given set of examples. It would free the web programmer from the tedious task of XSLT programming. In this paper we present a learning algorithm that allows to build such systems. Our algorithm works in a Gold style model in polynomial time and with polynomially many examples [15]: it takes as an input a set of couples of input / output trees of a target transduction and, if the input is rich enough, can infer a representation of the target. Besides *(1) decidable equivalence* in the class of objects to be learned we require that the number of examples needed to to learn an object of size n is *(2) bounded polynomially in n* (polynomial data) and that the learning algorithm *(3) infers the object in time polynomial in n*. Points (2) and (3) can be proved through efficient minimization of the desired object. As an example consider a regular language R and call two strings x and y "R-equivalent" if there is no z such that exactly one of xz and yz is in R. The Myhill-Nerode theorem says that the R-equivalence is of finite index if and only if R is regular. Moreover, there exist small representations of the different R-equivalence classes (which, in essence, require the same space as the minimal DFA for R).

Since XSLT programs are Turing complete [18, 26], polynomial exact learning can only be done for subclasses. The tree translation core of XSLT can conveniently be modeled by tree transducers [2, 17, 19, 21]. A large and well-studied class of tree transducer for which equivalence is decidable [13] is the deterministic top-down tree transducer (DTOP). Only recently an efficient minimization procedure was proved for DTOPs [12]: given a total DTOP, one can construct an equivalent "minimal earliest" transducer. In the partial case, inspection by a domain automaton is needed to generalize this result. Our contribution, starting from the earliest normal form, is to:

1. Present a Myhill-Nerode theorem for deterministic top-down tree transducers.

2. Provide an algorithm for polynomial exact learning of deterministic top-down tree transducers.

3. Show how to apply our results to learning of XML transformations.

In terms of previous work, our result provides a breakthrough in transducer learning: previous work only considered non-copying and non-swapping transducers (such as word transducers, or relabeling tree transducers). In contrast, DTOPs have the power to delete, exchange, and copy their input subtrees. Note that many practical XSLT programs make use of deletion and copying of subtrees.

Myhill-Nerode Theorem for TOPs

A TOP has rules of the form $q(f(x_1, \ldots, x_k)) \to t$ which say that if the transducer is in state q and processes an input node labeled f, then it should output the tree t. The tree t is over output symbols, and may also contain "state calls" of the form $\langle q', x_i \rangle$ at its leaves. Such a call means to insert the result of translating in state q' the i-th subtree of the current input node. Thus, a TOP can be seen as a particular left-linear term rewriting system. Note that a variable x_i may occur many times in t ("copying"), or may not appear at all ("deletion"). If for every state q and input symbol f there is at most one rule, then the transducer is deterministic and realizes a partial function from trees to trees.

How can we define the analog to R-equivalence (mentioned before), for functions τ realized by deterministic TOPs (for short, DTOPs)? Roughly speaking, we will chop τ into pieces, by considering functions from certain input subtrees to certain output subtrees. If there are only finitely many different such functions for τ, then τ can be realized by a DTOP. More precisely, consider an input tree s and a node π of s. The states of a DTOP that process the node π are uniquely determined by the *edge path* from the root of s to π (an edge path is the concatenation of pairs of node label and child-number on the path from the root to π). For a pair of edge paths $p = (u, v)$ we define the *residual* $p^{-1}\tau$ as all pairs (s', t') such that there are s, t with $\tau(s) = t$, u is an edge path in s to the subtree s', and v is an edge path in t to the subtree t'. Two pairs p_1, p_2 of edge paths are τ-equivalent if and only if $p_1^{-1}\tau = p_2^{-1}\tau$. Our Myhill-Nerode theorem says that for particular pairs of edge paths, called io-paths, τ-equivalence is of finite index if and only if τ can be realized by a DTOP.

Let us consider an example. We want to exchange a list of a-nodes with a list of b-nodes. The lists are represented in the first-child-next-sibling encoding, while the empty list is represented by $\#$. Thus, we want to translate

$$root(a(\#, a(\#, \ldots a(\#, \#) \ldots))), b(\#, b(\#, \ldots b(\#, \#) \ldots))$$

into the tree obtained by exchanging the root's two subtrees. Since this translation τ_{flip} is partial, there are exactly 4 different τ_{flip}-equivalence classes; the shortest representatives for these classes are the following pairs of edge paths:

$$(\epsilon, (root, 1)), (\epsilon, (root, 2)), ((root, 2), (root, 1)), ((root, 1), (root, 2)),$$

These io-paths in this order correspond exactly to the states $q_1, \ldots, q_4$ of the unique minimal earliest uniform DTOP M_{flip} below. It starts with the axiom $root(\langle q_1, x_0 \rangle, \langle q_2, x_0 \rangle)$ and has the following rules:

$$
\begin{aligned}
q_1(root(x_1, x_2)) &\to \langle q_3, x_2 \rangle \\
q_2(root(x_1, x_2)) &\to \langle q_4, x_1 \rangle \\
q_3(\#) &\to \# \\
q_3(b(x_1, x_2)) &\to b(\#, \langle q_3, x_2 \rangle) \\
q_4(\#) &\to \# \\
q_4(a(x_1, x_2)) &\to a(\#, \langle q_4, x_2 \rangle)
\end{aligned}
$$

Note that a minimal earliest uniform DTOPs as defined in [12] always comes together with a (minimal) deterministic top-down tree automaton recognizing the domain. In

our example, consider the (q_4, a)-rule. It deletes the first subtree; without domain automaton this means that *any* tree would be accepted here, but we want only the tree $\#$ there.

Learning Algorithm

Using our Myhill-Nerode theorem for DTOPs, we show that for any given top-down tree translation, a *characteristic sample set* can be computed in polynomial time (with respect to the size n of the minimal DTOP). Given a characteristic sample set (or a superset), the learning algorithm correctly infers the desired transducer. The characteristic sample set for the example τ_{flip} of before consists of only four pairs of trees:

$$
\begin{aligned}
root(\#, \#) &\to root(\#, \#) \\
root(a(\#, \#), \#) &\to root(\#, a(\#, \#)) \\
root(\#, b(\#, \#)) &\to root(b(\#, \#), \#) \\
root(a(a(\#, \#), \#), b(b(\#, \#), \#)) &\to \\
& \quad root(b(b(\#, \#), \#), a(a(\#, \#), \#))
\end{aligned}
$$

Note that a DTOP can translate a monadic input tree (of height n) into a full binary tree of height n. This implies that the trees in a characteristic sample set can have exponential size with respect to n. This can be avoided by representing output trees by their minimal DAGs; DAG representation of the output tree of a DTOP can be computed in linear time with respect to the size of the input tree (see [20]).

Inference of XML Transformations

XML documents are naturally modeled by *unranked trees*. There have been several proposal of tree transducers for unranked trees [21, 27]. These models are more expressive than to use a classical ranked DTOP on the "first-child/next-sibling" (FC/NS) encoding of the unranked trees. For instance, consider the translation XML$_{\text{flip}}$ of a root node with n children labeled a followed by m children labeled b, into a root node with first the m b-nodes followed by the n a-nodes. This example can easily be realized by the unranked transducers of [21, 27], however, *cannot* be realized by any ranked DTOP on FC/NS encoded trees. The reason is that a DTOP cannot change the order of nodes on a path.

Unfortunately, the added expressive power of unranked transducers comes at a price: we do not know whether deterministic unranked top-down tree transducers have decidable equivalence. In fact, since such transducers can completely flatten their output, they include the (classical) top-down tree-to-string translations. It is a long-standing open problem in tree transducer theory whether deterministic top-down tree-to-string transducers have decidable equivalence (already mentioned in [10]).

Are there other ranked tree encodings of unranked trees, so that a DTOP can realize XML$_{\text{flip}}$? We claim "yes". In fact, in the context of XML we believe that one should require the presence of input and output DTDs, before running the learning algorithm. We can use these DTDs to construct encodings that overcome restrictions of the FC/NS encoding. For instance, assume the following DTD for the input documents of XML$_{\text{flip}}$:

```
<!ELEMENT   root (a*,b*) >
<!ELEMENT   a EMPTY >
<!ELEMENT   b EMPTY >
```

And the same DTD, with `a*` and `b*` interchanged in the first line, for the output documents. Our idea of DTD-based encoding is to group elements from the same regular subexpression, under a new tree node. In our example, we will

have labels "(a*,b*)" (binary) and "a*", "b*" (unary). With this encoding, the input tree $root(a, a, b)$ is represented as

$$root(\text{"(a*,b*)"}(\text{"a*"}(a, \text{"a*"}(a, \text{"a*"}(\#, \#))), \text{"b*"}(b, \text{"b*"}(\#, \#))))$$

As we have seen before, a simple DTOP similar to M_{flip} can translate this tree into

$$root(\text{"(b*,a*)"}(\text{"b*"}(b, \text{"b*"}(\#, \#)), \text{"a*"}(a, \text{"a*"}(a, \text{"a*"}(\#, \#))))).$$

Thus, if we supply adequately DTD-encoded trees to our learning algorithm, then it can infer a ranked transducer for XML$_{\text{flip}}$. This transducer has twelve states and sixteen rules, but can still be inferred by four examples, as for τ_{flip}. The transducer we obtain can, modulo syntax, be seen as an XSLT program for unranked trees, i.e., XML documents: rules correspond to `apply-templates` with the mode corresponding to the state. Note that the class of unranked tree transformations realized by DTOPs over DTD-encoded trees is strictly included in the unranked top-down translations of [21, 27]; to see this, observe that the latter class contains both the DTD-encoding and the DTD-decoding.

Related Work

In the context of XML, little work deals with learning of queries and transformations. In fact, there is no prior research work on learning of XSLT programs. The "XSLT Inference Tool" (part of the Word 2003 SDK by Microsoft) can infer very restricted types of XSLT programs; it uses only a single example of input and output document in order to infer an XSLT program. The most related work is XLearner [23]. XLearner is a practical system that infers XQuery programs. It uses Angluin's algorithm [1] in order to infer path DFA's, from which it then constructs XPath expressions. For typical XQueries, the system needs a large number of user interactions (in the hundreds). It seems that the classes of XQuery that are learned by XLearner are incomparable to the class of programs the we infer. As mentioned in [23], there exists interesting work on inferring schema mappings, e.g., LSD [7] and Clio [28]. It will be interesting to see if an implementation of our results can be useful for automatic inference of XML schema mappings, and if so, how it compares to the such existing systems. There is a large amount of work on learning of DTDs and Schemas, see, e.g., [3] and the references given there. It is easily possible to combine any DTD inference algorithm with our work, by simply first inferring input (and output) DTDs, and then executing our algorithm to infer a transformation.

For finite-state transducers, algorithms exist for learning of subsequential string transducers [25]. They are based on minimal earliest transducers, which were formally introduced for strings by Mohri [22], see [6] for a survey. A learning algorithm and experimental results for deterministic Mealy machines is presented in [24]. Note that our result, applied to tree translations over monadic trees, also allows to infer minimal string transducers. For tree transducer, the only existing work deals with node selecting queries [4], which, in our context can be seen as simple relabelings (that is, DTOPs without copying and permuting of input variables). Previous work on induction of weighted tree transducers compute optimal weights for the rules of a fixed given tree transducer [16].

2. TOP-DOWN TREE TRANSDUCERS

We fix notations and present the standard definition of top-down tree transducers, together with some basic results. We then define "residuals", which allow us to prove a first version (of one direction) of the Myhill-Nerode theorem.

Trees and Paths. An alphabet is a finite set of symbols. A ranked alphabet is an alphabet F together with a total mapping rank$_F$ from F to non-negative integers. For $k \geq 0$ we denote by $F^{(k)}$ the set $\{f \in F \mid \text{rank}_F(f) = k\}$. We often write $f^{(k)}$ to indicate that f is of rank k. The set of (ordered, finite) trees over F is the set of ground terms over F and denoted by $\mathcal{T}_F$. This is the least set such that for all $k \geq 0$ and $f \in F^{(k)}$, $f(s_1, \ldots, s_k)$ is in $\mathcal{T}_F$ if $s_1, \ldots, s_k \in \mathcal{T}_F$. For a one-node tree $f()$ we simply write f. For a finite set A disjoint from F we define $\mathcal{T}_F(A)$ as $\mathcal{T}_{F'}$ where $F' = F \cup A$ and rank$_{F'}(a) = 0$ for every $a \in A$. Let $s \in \mathcal{T}_F$. The nodes of s are denoted by words over $\mathbb{N}$. The root node is denoted by the empty word ε. For a node $\pi \in \mathbb{N}^*$, its i-th child is denoted $\pi \cdot i$. For any word w, $w \cdot \varepsilon = \varepsilon \cdot w = w$, i.e., ε behaves as identity element.

In what follows, let $s = f(s_1, \ldots, s_k)$ with $f \in F^{(k)}$, $k \geq 0$, and $s_1, \ldots, s_k \in \mathcal{T}_F$. The set $nodes(s)$ of nodes of s is defined recursively as $nodes(f(s_1, \ldots, s_k)) = \{\varepsilon\} \cup \{i.\pi \mid 1 \leq i \leq k, \pi \in nodes(s_i)\}$. For a node $\pi \in nodes(s)$ we define the *label of π*, denoted by $s[\pi]$, as f if $\pi = \varepsilon$ and as $s_i[\pi']$ if $\pi = i.\pi'$, $i \in \mathbb{N}$, and $\pi' \in \mathbb{N}^*$. Similarly, we define the *subtree (of s) at π*, denoted by $\pi^{-1}s$, as s if $\pi = \varepsilon$, and as $\pi'^{-1}s_i$ if $\pi = i.\pi'$, $i \in \mathbb{N}$, and $\pi' \in \mathbb{N}^*$. For symbols $f_1, \ldots, f_n$ of rank zero and trees $s_1, \ldots, s_n$ we denote by $[f_1 \leftarrow s_1, \ldots, f_n \leftarrow s_n]$ the substitution of replacing every f_i-labeled leaf by the new subtree s_i.

An *F-labeled path*, or *F-path*, or *path* for short, is a (possibly empty) word over the alphabet $F_\#$ of labeled positions: $F_\# = \{(f, i) \mid k \geq 1, f \in F^{(k)}, 1 \leq i \leq k\}$. Note that constants in $F^{(0)}$ do not appear in such paths. We say that $u = (f_1, i_1) \cdot \ldots \cdot (f_n, i_n)$ *belongs to s*, denoted $u \models s$, if for all $1 \leq j \leq n$: $s[i_1.i_2 \cdots .i_j] = f_j$. The set of all paths that belong to s is denoted by $paths(s)$. The label at u is denoted by $s[u]$ and is defined as $s[i_1 \cdots .i_n]$. Note that the empty path ε belongs to every tree (and represents the root node). If $u \models s$, then $u^{-1}(s)$ denotes the subtree of s at u. We also need labeled paths that determine the label of the node that they address. We define *F-npaths* U to be words of the form $u \cdot f$ consisting of an F-path u and a symbol $f \in F$. For $U = u \cdot f$ and $s \in \mathcal{T}_F$ we write $U \models s$ if $u \models s$ and $u^{-1}s[\varepsilon] = f$. For $s \in \mathcal{T}_F$ we denote by $npaths(s)$ the set of all npaths U such that $U \models s$.

Transducers. We fix an infinite set $X = \{x_0, x_1, x_2, \ldots\}$ of *input variables*, and, for every $k \geq 0$ define the set $X_k = \{x_1, \ldots, x_k\}$, so that $X_0 = \emptyset$ in particular. For sets A and B, we write $\langle a, b \rangle$ for elements in the Cartesian product $A \times B$.

DEFINITION 1. *A deterministic top-down tree transducer (DTOP) is a tuple $M = (Q, F, G, ax, rhs)$ where Q is a finite set of states, F and G are ranked alphabets of input and output symbols, respectively, $ax \in \mathcal{T}_G(Q \times \{x_0\})$ is the axiom, and rhs is a (possibly partial) function which associates to (q, f) a tree in $\mathcal{T}_G(Q \times X_k)$ with $q \in Q$, $f \in F^{(k)}$, and $k \geq 0$. For every state $q \in Q$, M induces the function $[\![M]\!]_q : \mathcal{T}_F \to \mathcal{T}_G$ which, for every $f \in F^{(k)}$, $k \geq 0$, and $s_1, \ldots, s_k \in \mathcal{T}_F$ is defined recursively as $[\![M]\!]_q(f(s_1, \ldots, s_k)) =$*

$$rhs(q, f)[\langle q', x_i \rangle \leftarrow [\![M]\!]_{q'}(s_i) \mid q' \in Q, 1 \leq i \leq k].$$

The transduction defined by M *is the function* $[\![M]\!] : \mathcal{T}_F \to \mathcal{T}_G$ *defined as* $[\![M]\!](s) = ax[\langle q, x_0\rangle \leftarrow [\![M]\!]_q(s) \mid q \in Q]$ *for every* $s \in \mathcal{T}_F$.

In the rest of the paper, unless specified differently, F and G always denote (arbitrary) input and output alphabets, respectively. A DTOP can be seen as a particular confluent and terminating term rewrite system, with left-linear rules. In fact, it is often intuitive to think of the rewrite rules that are induced by the family of right-hand sides of the transducer. If $t = rhs(q, f)$ for a DTOP M, then $q(f(x_1, \ldots, x_k)) \to t$ is called *the* (q, f)-*rule* (*of* M).

EXAMPLE 1. *The constant transformation that maps all trees in* $\mathcal{T}_F$ *to a constant* $b \in \mathcal{T}_G$ *can be defined by the* DTOP M_1 *with axiom* $ax = b$, *and without states and rules. The same transduction can be defined by the* DTOP M_2 *with a singleton state set* $Q = \{q_0\}$, *axiom* $ax = \langle q_0, x_0\rangle$, *and the rule* $q_0(f(x_1, \ldots, x_k)) \to b$ *for every* $f \in F^{(k)}$ *and* $k \geq 0$.
There exist further transducers that define still the same transduction, but produce the output even later, for instance at the first-child of the root if it exists, or at the root otherwise. For instance, consider the DTOP M_3 *which has two states* $Q = \{q_0, q_1\}$, *axiom* $ax = \langle q_0, x_0\rangle$, *and, for every* $f \in F^{(k)}$, $k \geq 1$, $f' \in F^{(k')}$, $k' \geq 0$, *and* $a \in F^{(0)}$, *has the rules:*

$$
\begin{aligned}
q_0(f(x_1, \ldots, x_k)) &\to \langle q_1, x_1\rangle\ , \\
q_0(a) &\to b\ , \\
q_1(f'(x_1, \ldots, x_{k'})) &\to b\ .
\end{aligned}
$$

Note that the domain of any DTOP can be accepted by a *deterministic top-down tree automaton* (DTTA) (see, e.g., Proposition 2(1) of [12]). A DTTA can be defined as a DTOP which realizes the partial identity, i.e., for which: $rhs(q, f)$ (if it exists) is of the form $f(\langle q_1, x_1\rangle, \ldots, \langle q_k, x_k\rangle)$. Tree languages accepted by DTTAs are path-closed, see, e.g., [14]. A tree language L is *path-closed* if P-closure$(L) \subseteq L$ where P-closure$(L) = \{s \mid npaths(s) \subseteq npaths(L)\}$ and $npaths(L) = \{U \mid U \in npaths(s),\ s \in L\}$.

PROPOSITION 2. *The domain of a* DTOP *is path-closed.*

Consider a DTOP $M = (Q, F, G, ax, rhs)$. We denote by M_x the transducer $M_x = (Q, F', G', ax, rhs \cup rhs')$ where $F' = F \cup \{x^{(0)}\}$, $G' = G \cup \{\langle q, x\rangle^{(0)} \mid q \in Q\}$, and $rhs'(q, x) = \langle q, x\rangle$ for every $q \in Q$. Consider now an input tree $s \in \mathcal{T}_F$ of M and an F-path u that belongs to s. Then the tree $\zeta = [\![M]\!](s[u \leftarrow x])$ shows the computation of the transducer M, where we have "stopped" the translation at the unique node "u". Intuitively, if ζ contains the pair $\langle q, x\rangle$ at some path v, then it means that at v, M is translating the input subtree $s' = u^{-1}s$ in state q; in other words, the output tree $[\![M]\!](s)$ has at v the subtree $[\![M]\!]_q(s')$.

DEFINITION 3. *Let* M *be a* DTOP *and let* u, v *be* F *and* G-*paths, respectively. Then* (u, v) *reaches* q *if there exists* $s \in \mathcal{T}_F$ *such that* $u \models s$ *and* $v^{-1}([\![M_x]\!](s[u \leftarrow x])) = \langle q, x\rangle$.

In the literature, the sequence of states that appear in the tree $[\![M]\!](s[u \leftarrow x])$ is often called "the state sequence" of s at u. It is well known, that we can stop M's computation at any $u \models s$, and then replace every $\langle q, x\rangle$ by the q-translation of $u^{-1}s$. The following proposition is given in Lemma 3.6 of [11], but for total macro tree transducers (mtts); DTOPs

are special mtts without parameters. It is obvious that the statement also holds for partial DTOPs: $[\![M]\!](s)$ is defined if and only if ($[\![M_x]\!]$ is defined and $[\![M]\!]_q(u^{-1}s)$ is defined for every q such that $\langle q, x\rangle$ occurs in $[\![M_x]\!]$).

PROPOSITION 4. *Let* $M = (Q, F, G, ax, rhs)$ *be a* DTOP, u *a path, and* $s \in \mathcal{T}_F$ *such that* $u \models s$. *Then*

$$[\![M]\!](s) = [\![M_x]\!](s[u \leftarrow x])[\langle q, x\rangle \leftarrow [\![M]\!]_q(u^{-1}s) \mid q \in Q].$$

Residuals. We identify the partial function $[\![M]\!]_q$ defined by a transducer M with state q by pairs p of labeled input-output paths, by introducing a notion of residuals of a transduction $[\![M]\!]$ with respect to pair p.

DEFINITION 5. *The residual* $p^{-1}\tau$ *of a partial function* $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ *at a pair* $p = (u, v)$, *where* u *is a* F-*path and* v *a* G-*path, is the relation* $p^{-1}\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ *with:*

$$p^{-1}\tau = \{(u^{-1}(s), v^{-1}(t)) \mid (s, t) \in \tau,\ u \models s, v \models t\}\ .$$

For us, only very particular pairs of paths are sensible. For instance, we not care about p's such that $p^{-1}\tau$ is not a function. This happens if the node v was generated by an input subtree that is disjoint (i.e., in a different subtree) with u. E.g., if $u \neq v$ and τ is the identity function on trees, then $p^{-1}\tau$ is not a function. We also do not care about p's such that $p^{-1}\tau$ is empty. This happens if u is not a node of any $s \in dom(\tau)$, or if v is not a node of $\tau(s)$ for any $s \in dom(\tau)$. Recall now the definition of "p reaches q" given below Proposition 2. If, for a given DTOP M, p reaches q, then $p^{-1}[\![M]\!]$ is a function. This follows from the next lemma, together with the fact that $[\![M]\!]_q$ is a function for every state q of M.

LEMMA 6. *Let* M *be a* DTOP *and* u, v *paths. If* $p = (u, v)$ *reaches* q, *then* $p^{-1}[\![M]\!] = [\![M]\!]_q$.

PROOF. Let F be the input ranked alphabet of M and $s \in \mathcal{T}_F$. Consider an arbitrary tree $\tilde{s} \in \mathcal{T}_F$ with $u^{-1}\tilde{s} = s$. Clearly, such $\tilde{s}$ exists. By Lemma 4,

$$[\![M]\!](\tilde{s}) = [\![M_x]\!](\tilde{s}[u \leftarrow x])[\langle q, x\rangle \leftarrow [\![M]\!]_q(u^{-1}\tilde{s}) \mid q \in Q].$$

Since (u, v) reaches q, we know that the tree $[\![M_x]\!](\tilde{s}[u \leftarrow x])$ is labeled $\langle q, x\rangle$ at v. In particular, this means that $v \models [\![M]\!](\tilde{s})$. Thus, we can apply v^{-1} to the above equation. In the right-hand side, this yields precisely the argument of the substitution, i.e., we obtain

$$v^{-1}[\![M]\!](\tilde{s}) = [\![M]\!]_q(u^{-1}\tilde{s}).$$

By the definition of residual, $v^{-1}[\![M]\!](\tilde{s}) = (p^{-1}[\![M]\!])(u^{-1}\tilde{s})$ if and only if $u \models \tilde{s}$ and $v \models [\![M]\!](s)$. Thus, the left-hand side in the equation equals $(p^{-1}[\![M]\!])(u^{-1}\tilde{s})$. Since $u^{-1}\tilde{s} = s$, we have $(p^{-1}[\![M]\!])(s) = [\![M]\!]_q(s)$. $\square$

Let M be a DTOP with set of states Q. Let p_1, p_2 be pairs of labeled paths such that for $i \in \{1, 2\}$ there exists $q_i \in Q$ which is reached by p_i. On such pairs p_1, p_2 we define the *congruence relation* $\equiv_M$ of M as

$$p_1 \equiv_M p_2 \text{ iff } p_1^{-1}[\![M]\!] = p_2^{-1}[\![M]\!].$$

The next corollary follows obviously from Lemma 6.

COROLLARY 7. *The congruence* $\equiv_M$ *of a* DTOP M *has finite index.*

This corollary of Myhill-Nerode type has the disadvantage that the congruence is defined on objects that depend on the transducer M, rather than only on properties of the transduction $[\![M]\!]$. Therefore, it does not immediately lead to a unique minimal representation of such a transduction. This problem can be overcome for transducers for which the output production is normalized.

3. EARLIEST TRANSDUCERS

DTOPs in general do not have a unique minimal transducer (with respect to the number of states), and therefore also no Myhill-Nerode theorem. This is a problem for us because the learning techniques we want to apply are heavily dependent on such a theorem. However, [12] present the notion of *earliest* DTOP, in order to overcome this problem. Intuitively, in those transducers, the output is produced as soon as possible.

First, we need to introduce the notion of largest common prefix of a set of trees. For two trees $t, t' \in \mathcal{T}_G$ we define their *largest common prefix tree* $t \sqcup t' \in \mathcal{T}_G(\{\bot\})$ as follows:

$$g(t_1, \ldots, t_k) \sqcup g'(t'_1, \ldots, t'_{k'}) =$$
$$\begin{cases} g(t_1 \sqcup t'_1, t_2 \sqcup t'_2, \ldots, t_k \sqcup t'_k) & \text{if } g = g' \\ \bot & \text{otherwise.} \end{cases}$$

Note that the $\sqcup$ operator is associative and commutative. Thus, $\sqcup$ is easily extended to sets $L = \{t_1, \ldots, t_n\}$ of trees by setting $\bigsqcup L$ equal to $t_1 \sqcup t_2 \sqcup \cdots \sqcup t_n$ independently of the ordering of the trees in L.

Let $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ be a partial function and u a path that belongs to a tree in $dom(\tau)$. We define τ's *maximal output* for u as $out_\tau(u) = \bigsqcup\{\tau(s) \mid u \dashv s, s \in dom(\tau)\}$. and the same way $out_\tau(U)$ for node paths U. Note that $out_\tau(u)$ and $out_\tau(U)$ are undefined for all paths that do not belong to any tree in $dom(\tau)$. For a partial function $\tau \neq \emptyset$, $out_\tau(\varepsilon)$ is the "global common prefix" of all output trees. For instance, if $out_\tau(\varepsilon) = g(\bot, \bot)$ then the root node of every tree in the range of τ has label g (of rank 2).

We do not want useless states in an earliest DTOP. A state q of a DTOP M is *productive* if there exists a pair (u, v) that reaches q and there is an $s \in dom([\![M]\!])$ with $u \dashv s$. A DTOP is *productive* if all of its states are productive.

DEFINITION 8. *A DTOP M is* earliest *if it is productive and for every state q of M, $out_{[\![M]\!]_q}(\varepsilon) = \bot$.*

The notion of earliest DTOPs was introduced in [12] for "DTOPs with inspection" (see Section 7). It was proved there in Theorem 11 of [12] that any DTOP can be transformed into an earliest DTOP, that produces the same output on the domain of the original DTOP. $<$

EXAMPLE 2. *The transducer M_1 of Example 1 is earliest, while M_2 and M_3 are not (because $out_{[\![M_2]\!]_{q_0}}(\varepsilon) = out_{[\![M_3]\!]_{q_0}}(\varepsilon) = b \neq \bot$).*

The axiom and rules of an earliest DTOP are in a special form, as shown in the next lemma.

LEMMA 9. *For all earliest DTOPs $M = (Q, F, G, ax, rhs)$:*

(1) ax is of the form $out_{[\![M]\!]}(\varepsilon)\Psi$ for some substitution Ψ mapping $\bot$ nodes to $Q \times \{x_0\}$ and

(2) for every (q, f)-rule of M and every pair of paths (u, v) that reaches q, then $rhs(q, f) = (v^{-1} out_{[\![M]\!]}(u \cdot f))\Psi$ for some substitution Ψ mapping $\bot$ nodes to $Q \times X_k$.

PROOF. For the second part, consider a (q, f)-rule and a pair $p = (u, v)$ that reaches q. We need to prove that the set of paths of $rhs(q, f)$ is the same that the set of paths v' such that $v \cdot v' \dashv out_{[\![M]\!]}(u \cdot f)$. If $v' \dashv rhs(q, f)$, then for all trees $s \in dom([\![M]\!]_q)$ with $f \dashv s$, $v' \dashv [\![M]\!]_q(s)$. By Lemma 6 and Proposition 4 this implies that for every $s \in dom([\![M]\!])$ with $u \cdot f \dashv s$: $v \cdot v' \dashv [\![M]\!](s)$, i.e., $v' \dashv v^{-1} out_{[\![M]\!]}(u \cdot f)$. For the other direction, assume by contradiction that $v' \dashv v^{-1} out_{[\![M]\!]}(u \cdot f)$ and v' does *not* belong to $rhs(q, f)$. By Proposition 4, there must be an ancestor v'' of v' (i.e., $v'' = v' \cdot \tilde{v}$) such that some element $\langle q', x_i \rangle \in Q \times X$ occurs at v'' in $rhs(q, f)$. Then $\tilde{v} \dashv [\![M]\!]_{q'}(s)$ for every $s \in dom([\![M]\!]_{q'})$, and hence M is not earliest; a contradiction. $\square$

We call pairs of paths that reach a state io-paths. They have many important properties for us and are the essential notion used to infer a transducer from a translation.

DEFINITION 10. *An io-path $p = (u, v)$ for a partial function $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ is a pair of an F-path and a G-path such that $out_\tau(u)[v] = \bot$ and $p^{-1}\tau$ is functional.*

EXAMPLE 3. *Consider the function $\tau = \{(f(0, 0), 0), (f(0, 1), 0), (f(1, 0), 0), (f(1, 1), 1)\}$. Clearly $(\varepsilon, \varepsilon)$ is an io-path for τ. There is no other io-path for τ, e.g., $p = ((f, 1), \varepsilon)$ is not an io-path because $p^{-1}\tau = \{(0, 0), (1, 0), (1, 1)\}$ is not functional. Note that τ cannot be realized by any DTOP, in constrast to deterministic bottom-up tree transducers which can realize all finite tree functions.*

LEMMA 11. *Let M be an earliest DTOP. If $p = (u, v)$ is an io-path for $[\![M]\!]$ then p reaches a state q of M.*

The converse follows from Lemma 6. Thus for earliest DTOPs M, reachabilty in M captures io-paths of $[\![M]\!]$.

PROOF. Consider an io-path $p = (u, v)$ for $[\![M]\!]$. Let n be the length of u and let u_i be the prefix of length i of u, i.e., $u_0 = \varepsilon$ and $u_n = u$. For $i \in \{1, \ldots, n\}$, let v_i be such that $out_{[\![M]\!]}(u_i)[v_i] = \bot$, and denote $p_i = (u_i, v_i)$. Note that for each $i \in \{1, \ldots, n\}$ there is a v' (potentially empty) such that $v_i = v_{i-1} \cdot v'$. We prove inductively on i that each p_i reaches a state q_i of M.

Base case $(i = 0)$ is direct from Lemma 9. For the inductive step $(u_i \rightarrow u_{i+1})$, consider p_i and p_{i+1} with the last step of u_{i+1} being (f, j) and let v' be such that $v_{i+1} = v_i \cdot v'$. By induction hypothesis p_i reaches a state q_i. Then, from Lemma 9, $rhs(q_i, f)[v']$ must equal $\langle q, x_l \rangle$ for some state $q \in Q$ and some $l \in \{1, ..., k\}$ where k is the rank of f. If it did not, then it would equal a symbol $g \in G$, which means that for every input tree s with $u \dashv s$, $[\![M]\!](s)[v_{i+1}] = g$ as, either v_{i+1} is not equal to v and g is the next label step in v that follows the position indicated by v_{i+1}, or $v_{i+1} = v$ and that comes from from functionality of $p^{-1}[\![M]\!]$. As before, this contradicts that $out_{[\![M]\!]}(u_{i+1})[v_{i+1}] = \bot$, because there exists s_1, s_2 for which $g_1 = [\![M]\!](s_1)[v_{i+1}] \neq [\![M]\!](s_2)[v_{i+1}] = g_2$. Since p_i reaches q_i, and q appears at v' in $rhs(q_i, f)$, we have by Lemma 4 that p_{i+1} reaches q. $\square$

We can now define the Myhill-Nerode congruence of partial functions $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ between io-paths p_1 and p_2 of τ as follows: $p_1 \equiv_\tau p_2$ iff $p_1^{-1}\tau = p_2^{-1}\tau$.

THEOREM 12. *For an earliest* DTOP *M, the Myhill-Nerode congruence* $\equiv_{\llbracket M \rrbracket}$ *has finite index.*

PROOF. Let M be an earliest DTOP. By Lemma 11, every io-path p reaches some state q of M, and hence, by Lemma 6, $p^{-1}\llbracket M \rrbracket = \llbracket M \rrbracket_q$. Since M has only a finite number of states, the number of classes of io-paths that are equivalent w.r.t. the Myhill-Nerode congruence must be finite. □

This gives us the first part of the Myhill-Nerode theorem. it remains to prove the second part, i.e. that any partial function realized by a DTOP has a Myhill-Nerode congruence of finite index.

4. TOP-DOWN PARTIAL FUNCTIONS

In order to have the second part of the Myhill-Nerode theorem, we want to find a semantic characterization of the partial function defined by a DTOP.

An obvious property that we will often need is that if a path belongs to $out_\tau(u)$, then it also belongs to $out_\tau(u \cdot u')$.

LEMMA 13. *Let U, U' be node-paths such that U' is a descendant of U, and U and U' belong to some trees in $dom(\tau)$. Let V be a G-node-path. If $V \dashv out_\tau(U)$, then $V \dashv out_\tau(U')$.*

PROOF. For arbitrary $A, B \subseteq \mathcal{T}_G$ with $A \subseteq B$ it follows from the definition of $\bigsqcup$ that every G-labeled node in $\bigsqcup B$ also belongs to $\bigsqcup A$. Now, let $A = \{s \in dom(\tau) \mid U' \dashv s\}$ and $B = \{s \in dom(\tau) \mid U \dashv s\}$. Since $A \subseteq B$ we obtain that if V belongs to $out_\tau(U) = \bigsqcup B$ then it also belongs to $out_\tau(U') = \bigsqcup A$. □

We define the parent of a node-path as $parent(u \cdot (f, i) \cdot f') = u \cdot f$ and $parent(\varepsilon \cdot f) = \varepsilon$.

DEFINITION 14. *Let $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ be a partial function and V a G-npath. The set of top-down origins of V with respect to τ, denoted by $td\text{-}origin^\tau(V)$, consists of all F-npaths U such that:*

1. $V \dashv out_\tau(U)$, *and*

2. $V \not\dashv out_\tau(parent(U))$.

The presence of a *td-origin* of a npath in an input tree implies the presence of this npath in the corresponding output tree, as stated by following claim:

CLAIM 15. *If $U \dashv s$, $s \in dom(\tau)$, and $U \in td\text{-}origin(V)$ then $V \dashv \tau(s)$.*

PROOF. The first condition of the definition of td-origins ensures $V \dashv out_\tau(U)$ which is equivalent to $V \dashv \tau(s)$ for all $s \in dom(\tau)$ with $U \dashv s$. □

EXAMPLE 4. *Let $F = G = \{f^{(1)}, a^{(0)}, b^{(0)}\}$ and consider the total function $\tau_4 : \mathcal{T}_F \to \mathcal{T}_G$ such that $\tau_4(s) = f(a)$ if $s = a$, and otherwise $\tau_4(s) = f(b)$. We have $td\text{-}origin^{\tau_4}(f) = \emptyset$ since $f \in out_\tau(\epsilon)$. Furthermore, $td\text{-}origin^{\tau_4}((f, 1) \cdot a) = \{\epsilon \cdot a\}$ and $td\text{-}origin^{\tau_4}((f, 1) \cdot b) = \{\epsilon \cdot b, \epsilon \cdot f\}$.*

Given a tree $s \in \mathcal{T}_F$, we define $td\text{-}origin^\tau_s(V)$ as the set $td\text{-}origin^\tau(V) \cap npaths(s)$.

DEFINITION 16. *A partial function $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ is called top-down if $dom(\tau)$ is path-closed and for every $(s, t) \in \tau$, npath V with $V \dashv t$, and $V \not\dashv out_\tau(\epsilon)$: the set $td\text{-}origin^\tau_s(V)$ is a singleton.*

It should be clear that the total function τ_4 of Example 4 is top-down. Let us now look at a translation that is not top-down.

EXAMPLE 5. *Let $F = G = \{f^{(2)}, a^{(0)}, b^{(0)}\}$ and consider the translation $\tau_5 : \mathcal{T}_F \to \mathcal{T}_G$ satisfying $\tau_5(s) = a$ if $s = f(a, a)$, and otherwise $\tau_5(s) = b$. This translation is not top-down: $out_{\llbracket M \rrbracket}((f, 1)a) = \bot$ and $out_{\llbracket M \rrbracket}((f, 2)a) = \bot$, hence $td\text{-}origin(a) = \emptyset$.*

The following Lemma is proved by induction on the structure of input trees.

LEMMA 17. *For any* DTOP *M, $\llbracket M \rrbracket$ is top-down.*

Note that the converse of Lemma 17 is wrong, since all functions defined by infinite state top-down deterministic transducers are top-down (the above proof continues to work if the set of states is infinite), but not always definable by a DTOP.

Our next objective (Lemma 19) is to show that td-origins of top-down partial functions are organized in a top-down manner. We start with an auxiliary lemma.

LEMMA 18. *Let u, f, v, g be such that $u \cdot f \in td\text{-}origin^\tau(v \cdot g)$. There exist $\tilde{U}$ and $\tilde{g} \neq g$ such that $v \cdot \tilde{g} \dashv out_\tau(u \cdot \tilde{U})$*

PROOF. Assume by contradiction that there are no $\tilde{U}$ and $\tilde{g}$ with $\tilde{g} \neq g$ and $v \cdot \tilde{g} \dashv out_\tau(u \cdot \tilde{U})$. By the definition of out, this implies that $v \cdot g \dashv out_\tau(parent(u \cdot f))$; hence, $u \cdot f$ is *not* in $td\text{-}origin^\tau(v \cdot g)$; a contradiction. □

LEMMA 19. *Let $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ be a top-down partial function and $s \in \mathcal{T}_F$. If V' is a proper descendant of V, $\{U\} = td\text{-}origin_s(V)$, and $\{U'\} = td\text{-}origin_s(V')$, then U' is a descendant of U.*

PROOF. From $\{U\} = td\text{-}origin_s(V)$ we know that $V \dashv out_\tau(U)$ and $V \not\dashv out_\tau(parent(U))$, and similarly for U' w.r.t. V'. Obviously, U' cannot be a proper ancestor of U: if it was then, by Lemma 15, $V' \dashv out_\tau(parent(U))$. Since V is an ancestor of V', this means that $V \dashv out_\tau(parent(U))$; a contradiction.

It remains to show that U' is not disjoint from U. Assume it was. Let v and g such that $v \cdot g = V$, and u and f such that $u \cdot f = U$. By Lemma 18, there exists $\tilde{U}$ and $\tilde{g} \neq g$ such that $v \cdot \tilde{g} \dashv out_\tau(u \cdot \tilde{U})$. Hence, there exists an $s \in dom(\tau)$ such that $u \cdot \tilde{U} \dashv s$ and $v \cdot \tilde{g} \dashv \tau(s)$. Since $V' \in out_\tau(U')$, there exists an $s' \in dom(\tau)$ such that $U' \dashv s'$ and $V' \dashv \tau(s')$. By path-closedness of $dom(\tau)$ there exists an $s'' \in dom(\tau)$ such that $u \cdot \tilde{U} \dashv s''$ and $U' \dashv s''$. By the definition of out we have $V' \dashv \tau(s'')$, and from the fact that $v \cdot \tilde{g} \dashv out_\tau(u \cdot \tilde{U})$ we know that $V \not\dashv \tau(s'')$. This contradicts that V' is a proper descendant of V. □

5. ALIGNMENTS

To have the Myhill-Nerode Theorem it remains to prove the converse of Theorem 12: we need to construct an earliest transducer for any given top-down partial function whose Myhill-Nerode equivalence has finite index. Recall that the Myhill-Nerode equivalence is defined on io-paths. The trickiest part of the construction is to align paths of output trees to paths in input trees. In particular, when we construct the rules, we need to be able to make the correspondence between the variables in the right-hand side with the variables of the left-hand side (the "x_i").

LEMMA 20. *Let τ be a partial function. If τ is top-down and $u \cdot f \in td\text{-}origin^\tau(v \cdot g)$ then $(u,v)^{-1}\tau$ is functional.*

Consider that $v \cdot \bot \sqsubseteq out_\tau(u)$ for some top-down partial function τ. By the definition of *out* (and $\bigsqcup$) this means that at least two distinct output symbols can appear at v, for input trees containing u. This property, called "two-valuedness", allows to align input and output paths. We say that a pair (u,v) is *two-valued for τ* if there exist F-npaths U_1, U_2 and distinct symbols $g_1, g_2 \in G$ such that:

$$v \cdot g_1 \sqsubseteq out_\tau(u \cdot U_1) \quad \text{and} \quad v \cdot g_2 \sqsubseteq out_\tau(u \cdot U_2).$$

LEMMA 21. *If τ is functional and top-down and $v \cdot \bot \sqsubseteq out_\tau(u)$ then (u,v) is two-valued.*

The next two lemmas are central. They show that only one particular alignment between output path and input path is possible. This is first proved for two-valued pairs (Lemma 22) and then lifted to io-paths (Lemma 23).

LEMMA 22. *Let $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ be a top-down partial function and let $(u \cdot (f,i), v)$ be two-valued for τ. If $(u \cdot (f,j), v)^{-1}\tau$ is functional and nonempty, then $i = j$.*

PROOF. Since $(u \cdot (f,i), v)$ is two-valued for τ, there exist paths u_1, u_2 such that $\bot \neq v^{-1} out_\tau(u \cdot (f,i) \cdot u_1)[\varepsilon] \neq v^{-1} out_\tau(u \cdot (f,i) \cdot u_2)[\varepsilon] \neq \bot$. Thus for arbitrary trees $s_1, s_2 \in dom(\tau)$ that contain $u \cdot (f,i) \cdot u_1$ and $u \cdot (f,i) \cdot u_2$, respectively,

$$v^{-1}(\tau(s_1)) \neq v^{-1}(\tau(s_2)) \qquad (\#)$$

To show a contradiction, assume that $i \neq j$. Let $\rho = (u \cdot (f,j), v)^{-1}\tau$. Since ρ is nonempty, there exists a tree $\tilde{s}$ with $u \cdot (f,j) \sqsubseteq \tilde{s}$ and $v \sqsubseteq \tau(\tilde{s})$. Obviously, $\tilde{s}$ contains $u \cdot (f,i)$.

We construct the tree $\tilde{s}_1$ from $\tilde{s}$ by replacing the subtree at $u \cdot (f,i)$ by an arbitrary input tree that contains u_1. Since τ is path-closed $\tilde{s}_1 \in dom(\tau)$, and $v^{-1}(\tau(\tilde{s}_1))$ exists. Since $i \neq j$, $u \cdot (f,j)^{-1}\tilde{s}_1 = u \cdot (f,j)^{-1}\tilde{s}$. The definition and functionality of ρ imply:

$$v^{-1}(\tau(\tilde{s}_1)) = \rho((u \cdot (f,j))^{-1}\tilde{s}_1) = \rho((u \cdot (f,j))^{-1}\tilde{s}).$$

Similarly, we construct a tree $\tilde{s}_2$ which contains $u \cdot (f,i) \cdot u_2$ such that $v^{-1}(\tau(\tilde{s}_2)) = \rho(u \cdot (f,j)^{-1}\tilde{s})$. Hence:

$$v^{-1}(\tau(\tilde{s}_1)) = \rho(u \cdot (f,j)^{-1}\tilde{s}) = v^{-1}(\tau(\tilde{s}_2))$$

in contradiction to $(\#)$. $\square$

LEMMA 23. *Let $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ be a partial function that is top-down. For every io-path (u,v) of τ, $f \in F^{(k)}$, $k \geq 1$, and v' with $v \cdot v' \cdot \bot \sqsubseteq out_\tau(u \cdot f)$, there is a unique $i \in \{1, \ldots, k\}$ such that $(u \cdot (f,i), v \cdot v')$ is an io-path. We denote the number i by $orig^\tau_{u \cdot f, v}(v')$.*

6. CANONICAL TRANSDUCERS

We say that a partial top-down function τ has *finite index*, if its Myhill-Nerode equivalence (on io-paths) is of finite index. We now show how to construct an earliest DTOP $M = min(\tau)$ such that for every $s \in dom(\tau)$, $[\![M]\!](s) = \tau(s)$.

Note that there exist top-down partial functions with finite index, that cannot be defined by any DTOP, because the DTOP cannot delete an input subtree and check its domain at the same time. This property of DTOPs was already noted in [8]. For instance, consider the top-down partial function

$\tau = \{(f(c,a),a),(f(c,b),b)\}$. In order to realize τ, every DTOP would have to delete first subtrees below f-roots, since the output does not depend on them. As a consequence, no such DTOP can check whether the first subtree is a c.

DEFINITION 24. *Let τ be a top-down partial function with Myhill-Nerode equivalence of finite index. We define the DTOP $min(\tau) = (Q, F, G, ax, rul)$ as follows:*

$$Q = \{[p] \mid p \text{ is an io-path of } \tau\}$$

The axiom ax is the term $out_\tau(\varepsilon)\Psi_0$ where the substitution Ψ_0 replaces subtrees at $\bot$-nodes as follows:

$$\Psi_0 = [v' \leftarrow \langle[(\varepsilon,v')], x_0\rangle \mid v' \cdot \bot \sqsubseteq out_\tau(\varepsilon)]$$

For all io-paths $p = (u,v)$ of τ and $f \in F^{(k)}$ where $k \in \mathbb{N}_0$ such that $u \cdot f \in npaths(dom(\tau))$, the following rule is an element of rul:

$$[p](f(x_1, \ldots, x_k)) \to v^{-1}(out_\tau(u \cdot f))\Psi$$

where Ψ is the substitution on trees, that replaces all $\bot$-subtrees as follows:

$$\Psi = [v' \leftarrow \langle[p_{v'}], x_{i_{v'}}\rangle \mid v' \cdot \bot \sqsubseteq out_\tau(u \cdot f)]$$

Here, $i_{v'}$ is the unique index such that $p_{v'} = (u \cdot (f,i_{v'}), v \cdot v')$ is an io-path of τ, i.e., $i_{v'} = orig^\tau_{u \cdot f, v}(v')$.

Note that $min(\tau)$ is well-defined. The set Q is finite, since there $[\![M]\!]$ has finite index by assumption. Pairs (ϵ, v') in the axiom are io-paths of τ, since $v' \cdot \bot$ is assumed. Furthermore, by Proposition 23, the existence and uniqueness of $i_{v'}$ are ensured, such that $p_{v'}$ is an io-path of τ.

LEMMA 25. *If $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ is a top-down partial function whose Myhill-Nerode equivalence has finite index, then $[\![min(\tau)]\!]_{|dom(\tau)} = \tau$.*

LEMMA 26. *The transducer $min(\tau)$ is earliest.*

7. MINIMALITY

We show that $min(\tau)$ is indeed a unique earliest DTOP for τ with a minimal number of states that is compatible with $dom(\tau)$ in a particular sense defined below.

In [12] they minimize "DTOPs with inspection", that is, a DTOP together with a deterministic top-down tree automaton that defines the domain. In this way, the problem of deletion and checking mentioned in the beginning of Section 6 is overcome. It turns out that a minimal earliest DTOP with inspection is *not* unique. In [12] this is solved by imposing uniformity on transducers. Here, we take a similar approach, but express it independently of the representation of the domain by a top-down tree automaton.

DEFINITION 27. *We call an earliest DTOP M compatible with a domain D if it satisfies the following three conditions:*

(C_0) *io-paths $p_1 = (u_1, v_1)$ and $p_2 = (u_2, v_2)$ of $[\![M]\!]$ with different D-restricted domains, i.e., $u_1^{-1}(D) \neq u_2^{-1}(D)$, reach different states of M.*

(C_1) *for all F-paths u either both $out_{[\![M]\!]}(u)$ and $out_{[\![M]\!]_{|D}}(u)$ are undefined, or satisfy $out_{[\![M]\!]}(u) = out_{[\![M]\!]_{|D}}(u)$.*

(C_2) *no superfluous rules, i.e., for all io-paths $p = (u, v)$ of $[\![M]\!]$, all $f \in F$ such that $(u \cdot f)^{-1}(D) = \emptyset$, and all states q of M reached by p, $rhs(q, f)$ is undefined.*

EXAMPLE 6. *Let τ be the identity function with domain restricted to $D = \{f(c, a), f(c, b)\}$. This partial function has two io-paths $p_1 = (\varepsilon, \varepsilon)$ and $p_2 = ((f, 2), \varepsilon)$ with residuals $p_1^{-1}(\tau) = \tau$ and $p_2^{-1}(\tau) = \{(a, a), (b, b)\}$.*

Clearly, τ cannot be defined by any DTOP *without domain inspection, since these must all delete the first subtree and thus cannot test whether it is equal to c. Nevertheless there exists an earliest* DTOP M_0 *with a single state, whose restriction to D defines τ. It has the axiom $ax = f(c, \langle q_0, x_0 \rangle)$ and the rules:*

$$q_0(f(x_1, x_2)) \rightarrow \langle q_0, x_2 \rangle, \qquad q_0(a) \rightarrow a, \qquad q_0(b) \rightarrow b.$$

Transducer M_0, however, fails to satisfies property (C_0). The problem is that the io-paths p_1 and p_2 reach the same state q_0 in M, even though they have different domains: $\varepsilon^{-1}(D) = D$ and $(f, 2)^{-1}(D) = \{a, b\}$.

The minimal earliest DTOP M_1 *that defines τ on D and that is compatible with D, has two states, $ax = f(c, \langle q_0, x_0 \rangle)$ and the rules:*

$$q_0(f(x_1, x_2)) \rightarrow \langle q_1, x_2 \rangle, \qquad q_1(a) \rightarrow a, \qquad q_1(b) \rightarrow b.$$

There exists another earliest DTOP M_2 *with one state, which defines τ on D, that satisfies (C_0) but not (C_1). It has axiom $ax = \langle q_0, x_0 \rangle$ and the following rules:*

$$q_0(f(x_1, x_2)) \rightarrow f(c, \langle q_0, x_2 \rangle), \qquad q_0(a) \rightarrow a, \qquad q_0(b) \rightarrow b.$$

Note that M_2 is earliest, even though it does not produce the maximal possible output when restricted to D. Therefore, (C_1) is violated: $\bot = out_{\llbracket M_2 \rrbracket}(\varepsilon) \neq out_\tau(\varepsilon) = f(c, \bot)$.

There exist a DTOP*s M_3 that defines τ on D and satisfies (C_0) and (C_1), but not (C_2). This transducer has two states, but is not isomorphic to M_1. M_3 is obtained from M_1 by adding the following rule, which permits to accept further trees outside the domain: $q_0(g(x_1)) \rightarrow a$.*

The uniqueness result of minimal earliest compatible transducers in the following Myhill-Nerode theorem for DTOPs is quite intricate.

THEOREM 28. *The following three properties are equivalent for all partial functions $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$:*

1. *τ is top-down and has finite index, i.e., the number of different residuals of io-paths of τ is finite.*

2. *$\llbracket M \rrbracket_{|L(A)} = \tau$ for some* DTTA *A and* DTOP *M.*

3. *There exists a unique earliest* DTOP *M with a minimal number of states that is compatible with $D = dom(\tau)$ and such that $\llbracket M \rrbracket_{|D} = \tau$.*

It τ satisfies the above properties, then we call $min(\tau)$ the unique minimal earliest compatible DTOP *of τ.*

8. CHARACTERISTIC SAMPLES

We show that top-down partial functions with finite index can be characterized by its domain and a finite sample containing examples of input-output pairs.

We fix a path-closed set $D \subseteq \mathcal{T}_F$ and consider the class of top-down partial functions $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ of finite index with domain $dom(\tau) = D$. A sample S for τ is a finite subrelation $S \subseteq \tau$. Our objective is to characterize transducers DTOP $min(\tau)$ by a finite sample of input-output pairs for τ whose cardinality is polynomial in the size of M, under the assumption $D = dom(\tau)$.

In order to identify the states of DTOPs by particular io-paths that reach them, we assume two total orders $<_F$ on F-paths and $<_G$ on G-paths such that paths become smaller when deleting letters. We then lift these to orders to an order of pairs of F- and G-paths lexicographically:

$$(u, v) < (u', v') \Leftrightarrow u < u' \vee (u = u' \wedge v < v').$$

For examples, we assume that $u < u'$ if u has fewer letters than u' or the same numbers of letters while u preceeds u' lexicographically. In analogy for $v < v'$.

DEFINITION 29. *Let $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ be a top-down partial function of finite index and $M = min(\tau)$. Let q be a state of M and $f \in F^{(k)}$ for some $k \in \mathbb{N}_0$.*

- *The io-path of q is the w.r.t. $<$ least io-path of τ that reaches q in M; we denote it by io-path$_q$.*

- *If $rhs(q, f)$ is defined in M then for all paths v' of $rhs(q, f)$ that are labeled by some $\langle q', x_i \rangle$ where $1 \leq i \leq k$, we define the io-path of the transition q, f, v' by io-path$_{q,f,v'} = (u \cdot (f, i), v \cdot v')$ where io-path$_q = (u, v)$.*

A state-io-path of τ is some io-path$_q$ where q is a state of $min(\tau)$, and similarly, a trans-io-path of τ is a io-path$_{q,f,v'}$ for some transition q, f, v' of M.

For instance, the 4 io-paths of the transformation τ_{flip} given in the introduction are precisely the state-io-paths for the states $q_1, \ldots, q_4$ of the DTOP M_{flip} there, with respect to the concrete order we fixed above.

More generally, it should be noticed that the definitions of state- and trans-io-paths depend on the precise ordering. As a consequence, the definitions of characteristic samples below will depend on this ordering too.

DEFINITION 30. *Let $\rho \subseteq \mathcal{T}_F \times \mathcal{T}_G$ a binary relation and $p_1 = (u_1, v_1)$ and $p_2 = (u_2, v_2)$ be pairs of F- and G-paths. We call p_1 and p_2 mergeable w.r.t. ρ and D if $u_1^{-1}(D) = u_2^{-1}(D)$ and there is no $s \in u_1^{-1}(D)$ such that $p_1^{-1}(\rho)(s) \neq p_2^{-1}(\rho)(s)$.*

DEFINITION 31. *Let $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ be a top-down partial function with $dom(\tau) = D$. A relation $S \subseteq \mathcal{T}_F \times \mathcal{T}_G$ is called a characteristic sample for τ w.r.t $<$ if:*

(C) *S is a sample for τ, i.e., $S \subseteq \tau$ is finite.*

(A) *$out_S(\varepsilon) = out_\tau(\varepsilon)$.*

(T) *for all state-io-paths (u, v) of τ and all $f \in \Sigma^{(k)}$, either $out_\tau(u \cdot f)$ and $out_S(u \cdot f)$ are both undefined or $out_\tau(u \cdot f) = out_S(u \cdot f)$.*

(O) *for all state-io-paths (u, v) of τ and all $f \in \Sigma^{(k)}$ such that $out_\tau(u \cdot f)$ is defined, then, for all v' with $v' \cdot \bot \dashv out_\tau(u \cdot f)$, there is a unique i such that $(u \cdot (f, i), v \cdot v')$ is an io-path of S.*

(N) *state-io-paths $p_1 = (u_1, v_1)$ and trans-io-paths $p_2 = (u_2, v_2)$ of τ that are not mergeable w.r.t. S and D are not mergeable w.r.t. τ and D.*

Consistency (C) ensures soundness. The axiom of $min(\tau)$ can be inferred from S by (A). Property (N) requires that S contains sufficient information to separate non-equivalent state- and trans-io-paths of τ.

LEMMA 32. *Let S be a sample for τ that satisfies (N), p_1 a state- and p_2 a trans-io-path for τ. If $p_1^{-1}(\tau) \neq p_2^{-1}(\tau)$ then p_1 and p_2 are not mergeable w.r.t S and D.*

Properties (T) and (O) permit to infer the transitions of $min(\tau)$: (T) allows to build the rigid parts of the right hand sides of the rules, while (O) allows to find the correct alignment, i.e, which variables to put into the right hand side.

LEMMA 33. *Let S be a sample for τ that satisfies (T) and (O), $p = (u, v)$ a state-io-path for τ and $f \in \Sigma(k)$ with $out_\tau(u \cdot f)$ is defined, then for all v' with $v' \cdot \perp \; =\!\mid\; out_\tau(u \cdot f)$ it holds that $orig_{u \cdot f, v}^S(v') = orig_{u \cdot f, v}^\tau(v')$*

PROPOSITION 34. *For every top-down partial function τ with finite index, there exists a w.r.t. $<$ characteristic sample S whose cardinality is polynomial in the size of $min(\tau)$.*

9. LEARNING ALGORITHM

We present our learning algorithm for DTOPs, which identifies $min(\tau)$ for some unknown top-down partial function $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ with finite index, from a given DTTA A with $L(A) = dom(\tau)$ and a characteristic sample S for τ.

As main data structure it relies on so called DTOPs with border-states whose semantics is fixed statically.

DEFINITION 35. *A DTOP with border-states is a tuple $M = (Q_{ok}, Q_{border}, F, G, ax, rhs, sem)$ such that:*

1. *(Q, F, G, ax, rhs) is a DTOP where $Q = Q_{ok} \uplus Q_{border}$,*

2. *sem is a collection of partial functions $sem(q) \subseteq \mathcal{T}_F \times \mathcal{T}_G$ for all $q \in Q_{border}$, and*

3. *$rhs(q, f)$ is undefined for all $q \in Q_{border}$ and $f \in F$.*

The partial functions $[\![M]\!]$ and $[\![M]\!]_q$ can be defined for all states $q \in Q$, by $[\![M]\!]_q = sem(q)$ if $q \in Q_{border}$ and as for DTOPs before if $q \in Q_{ok}$.

The learning algorithm RPNI$_{\text{DTOP}}$ is given in Figure 1. As input it receives a finite partial function $S \subseteq \mathcal{T}_F \times \mathcal{T}_G$ and a DTTA A with $L(A) \subseteq \mathcal{T}_F$. All acceptable inputs satisfy the assumption that there exists a partial function $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ with $dom(\tau) = L(A)$ for which S is characteristic. Such a partial function must then be unique. It is the target that our learning algorithm is supposed to identify.

In order to do so, RPNI$_{\text{DTOP}}$ constructs a sequence of DTOPs with border-states. All border-states will be io-paths of S and all ok-states io-paths of τ. The DTOP at the beginning contains only border-states, and the one at the end only ok-states. RPNI$_{\text{DTOP}}$ tries to merge border-states with ok-states in the same order as used for defining characteristic samples. Intuitively, a merging attempt is successful if no contradicting evidence on the equivalence of the io-paths under consideration can be inferred from S and A.

Let $P(S)$ be the set of io-paths of S. This set is totally ordered by the ordering $<$ that we assumed for defining characteristic sets. We lift this ordering to $P(S) \cup \{start\}$ such that the new constant $start$ becomes the least element.

The data structure maintained by RPNI$_{\text{DTOP}}$ updates a pair (p_0, μ), where $p_0 \in P(S) \cup \{start\}$ memorizes the current io-path of S and while μ memorizes io-paths of τ to which io-paths of S got merged.

More formally, $\mu : P(S) \to P(S)$ is a idempotent function with $\mu(p) \leq p$ for all $p \leq p_0$ and $\mu(p) = p$ otherwise. This

means that only io-paths of S smaller than p_0 got merged, and only with some smaller io-path of S. When started with a characteristic sample S for some partial function τ and a DTTA A with $dom(\tau) = L(A)$, then the current data structure (p_0, μ) of our algorithm is supposed to always satisfy the following invariants:

(I$_1$) all ok-states p of $M(p_0, \mu, S)$ are state-io-paths of τ (so they are pairwise non-equivalent).

(I$_2$) state merging is consistent with the Myhill-Nerode congruence of τ, i.e.: $\mu(p) = \mu(p') \Rightarrow p^{-1}(\tau) = p'^{-1}(\tau)$.

Given a partial function S and a pair (p_0, μ) satisfying (I$_1$), we can define a DTOP with border-states independently of τ

$$M(p_0, \mu, S) = (Q_{ok}, Q_{border}, F, G, ax, rhs, sem)$$

as follows:

$Q_{ok} = \{\mu(p) \mid p \in P(S), \; p \leq p_0\}$,
$ax = out^S(\epsilon)[v' \leftarrow \langle \mu(\epsilon, v'), x_0 \rangle \mid v' \cdot \perp \; =\!\mid\; out^S(\epsilon)]$,
$rhs(p, f) = v^{-1}(out^S(u \cdot f))\Psi$
    where $p = (u, v) \in Q_{ok}, f \in F^{(k)}, k \in \mathbb{N}_0$,
    $\Psi = [v' \leftarrow \langle \mu(q_{v'}), x_{i_{v'}} \rangle \mid v \cdot v' \cdot \perp \; =\!\mid\; out^S(u \cdot f)]$,
    and $1 \leq i_{v'} \leq k$ unique such that $q_{v'} = (u \cdot (f, i_{v'}), v \cdot v')$,
    is an io-path of S,
$Q_{border} = \{p \mid p$ occurs in ax or in some $rhs(p', f)$ where
        $p' \in Q_{ok}$ and $f \in F\} \setminus Q_{ok}$,
$sem(p) = p^{-1}(S)$ for all border-states p.

This definition is correct, since for all $p = (u, v) \in Q_{ok}$ the exists indeed a unique $i_{v'}$ such that $q_{v'} = (u \cdot (f, i_{v'}), v \cdot v')$ is an io-path of S. This follows, since by invariant (I$_1$) all $p \in Q_{ok}$ are io-paths of τ, so that we can apply condition (O) of S being a characteristic samples for τ to p. Furthermore, note that all applications of μ above are well-defined, since all pairs (ϵ, v') in the definition of ax are io-paths of S, as well as all pairs $q_{v'}$ in the definition of $rhs(p, f)$.

PROPOSITION 36. *Let S be a characteristic sample for a top-down partial function τ with finite index, and (p_0, μ) a data structure that satisfies invariants (I$_1$) and (I$_2$). If furthermore, p_0 is greater than all state- and trans-io-paths of τ then $M(p_0, \mu, S) = min(\tau)$.*

Let us inspect algorithm RPNI$_{\text{DTOP}}$ in Figure 1 more precisely. It starts with the empty set of ok-states by initializing p_0 with $start$. Function μ is initialized to the identity function on $P(S)$, here named id. The initial DTOP with border-states thus is $M := M(start, id, S)$. This DTOP has no ok-states; its border-states are the io-paths of S occurring in the axiom. The algorithm then tries to merge border-states with ok-states, while following the total order $<$ on $P(S)$.

The criterion of mergeability w.r.t. S and $L(A)$ from Definition 30 is applied; it ensures (I$_1$) so that $\mu(p)^{-1}(\tau) = p^{-1}(\tau)$ for all io-paths p of τ. Since all ok-states are state-io-paths of τ by (I$_1$) and all border-states are trans-io-paths of τ (see the proof Proposition 36 in the long version), every border-state can be merged with at most one ok-state. Since state merging is done in order, every border-state which cannot be merged with any ok-state must be a state-io-path of τ itself, so it can be safely added to the set of ok-states. The algorithm terminates once all state-io-paths and trans-io-paths of τ were visited.

LEMMA 37. *Algorithm RPNI$_{\text{DTOP}}$ preserves (I$_1$) and (I$_2$).*

```
RPNI_dtop(S, A)
  # where A is a DTTA and S is a
  # characteristic sample of some partial
  # function τ ⊆ T_F × T_G with dom(τ) = L(A)
  p0 := start  # preceeds all io−paths of S
  μ  := id     # identity on io−paths of S
  M := M(p0, μ, S)
  while Q_border(M) ≠ ∅ do
    let p = min_<(Q_border(M))
    let_lazy  Ok = {p' ∈ Q_ok(M) | p and p' can
                   be merged w.r.t. S and L(A)}
    if  Ok ≠ ∅
    then # Ok is a singleton
      let p' such that Ok = {p'}
      μ := μ[p' → p] # merge p' with p
    else # p must be a state−io−path!
      skip # add p to the ok−states
    p0 := p
    M := M(p0, μ, S)
  return M # no border−states remain
```

Figure 1: Learning algorithm for DTOPs.

THEOREM 38. *If S is a characteristic sample for a top-down partial function τ with finite index, and A a DTTA recognizing $dom(\tau)$ then the learning algorithm $\mathrm{RPNI}_{\mathrm{DTOP}}(S, A)$ terminates in polynomial time in the size of S and returns $min(\tau)$.*

PROOF. Since we assume τ to be top-down and of finite index, the DTOP $min(\tau)$ is well defined by Theorem 28. The learning algorithm terminates, once all state- and trans-io-paths have been inspected, and thus after linearly many turns in the while loop, w.r.t. the size of $min(\tau)$ and thus S. As shown by Lemma 37, the algorithm preserves invariants (I_1) and (I_2). Proposition 36 shows that the resulting DTOP is indeed equal to $min(\tau)$.

For each trans-io-path $p = (u, v)$, the algorithm builds $rhs(p, f)$. The computation of $v^{-1} out(u \cdot f)$ is in $O(|S|)$. Let k be the arity of f. We need to infer $i_{v'} = orig^S_{u \cdot f, v}(v')$ for all $\perp$ positions v' of $v^{-1} out(u \cdot f)$. This might require to check for all $i_{v'}$-candidates $i \in \{1, \ldots, k\}$ whether the residual $(u \cdot (f, i), v \cdot v')^{-1} S$ is functional. This can be done in time $O(K * N * |S|)$ where K is the largest arity of symbols in F and N the maximal number of state occurrences in some $rhs(p, f)$, and thus at most $O(K * |M| * |S|)$. Subsequently, this trans-io-path is tested for mergeability with in the worst case all state-io-path. Each merge attempt costs $O(|S|)$, which may sums up to a most $O(|S| * |M|)$ per trans-io-path. Since there are $O(|M| * |F|)$ trans-io-path, the overall costs are in $O(|M|^2 * |F| * K * |S|)$. □

EXAMPLE 7. *Consider the transduction τ_{flip} presented in the Introduction. Algorithm $\mathrm{RPNI}_{\mathrm{DTOP}}$ can infer τ_{flip} from an DTTA for the domain and the following characteristic sample of τ_{flip} (with respect to the order $<$ we fixed for examples):*

$$S = \{ \begin{array}{ll} (root(\#, \#) & , \quad root(\#, \#)) \\ (root(a(\#, \#), \#) & , \quad root(\#, a(\#, \#))) \\ (root(\#, b(\#, \#)) & , \quad root(b(\#, \#), \#)) \\ (root(a(a(\#, \#), \#), b(b(\#, \#), \#)) & , \\ \quad root(b(b(\#, \#), \#), a(a(\#, \#), \#)))\} \end{array}$$

The algorithm starts with $p_0 := start$. The first DTOP is thus $M = M(p_0, id, S)$. The axiom of M identifies the axiom of $min(\tau)$, which is $ax = root(\langle p_1, x_0\rangle, \langle p_2, x_0\rangle)$ where

$p_1 = (\varepsilon, (root, 1))$ *and* $p_2 = (\varepsilon, (root, 2))$ *are the two border-states of the first M. Since $p_1 < p_2$, the algorithm considers p_1 next. This border-state cannot be merged with any ok-state of M (there are none). Hence, p_1 is turned into an ok-state by updating $p_0 := p_1$, so that the current DTOP becomes $M := M(p_1, id, S)$. This new transducer provides us with a new rule for the new ok-state, $p_1(root(x_1, x_2)) \to \langle p_3, x_2\rangle$ where $p_3 = ((root, 2), (root, 1))$ is a new border-state. Note that x_1 cannot be chosen instead of x_2 here, since the residual $((root, 1), (root, 1))^{-1}(S)$ contains $(\#, \#)$ and $(\#, b(\#, \#))$, so that it is not functional.*

The next border-state to consider is p_2 since $p_2 < p_3$. Path p_2 cannot be merged with p_1, since both states translate $root(a(\#, \#), \#))$ differently (p_1 outputs $\#$ while p_2 outputs $a(\#, \#)$). Given that there exists no other ok-state, p_2 is added to the ok-states by updating $p_0 := p_2$. The current transducer becomes $M := M(p_2, id, S)$. It provides the rule since $p_2(root(x_1, x_2)) = \langle p_4, x_1\rangle$ with $p_4 = ((root, 1), (root, 2))$, so p_4 is a new border-state. Again, the choice x_1 on the right-hand side was uniquely determined by S.

Now, the least remaining borderstate is p_4 given that $p_4 < p_3$. Note that p_4 can neither be merged with p_1 nor p_2, since they have different domains. Thus p_4 is turned into an ok-state by updating $p_0 := p_4$. This yields two new rules $p_4(\#) = \#$ and $p_4(a(x_1, x_2)) = a(\#, \langle p_5, x_2\rangle)$ where $p_5 = ((root, 2) \cdot (a, 2), (root, 1) \cdot (a, 2))$ is a new border-state. Next, p_3 is considered and turned into an ok-state. This introduces two new rules $p_3(\#) = \#$ and $p_3(b(x_1, x_2)) = b(\#, \langle p_6, x_2\rangle)$ where $p_6 = ((root, 2) \cdot (b, 2), (root, 1) \cdot (b, 2))$. Border-state p_5 can indeed be merged with p_4, so that the algorithm updates $\mu(p_5) := p_4$. Similarly, $\mu(p_6) := p_3$. Now algorithm $\mathrm{RPNI}_{\mathrm{DTOP}}$ stops, given that no border-states remain. The resulting DTOP is precisely the minimal earliest compatible transducer for τ_{flip}, given in the Introduction (modulo the isomorophism mapping p_i to q_i for all $1 \le i \le 4$).

10. XML TRANSFORMATIONS

XML documents are naturally modeled by *unranked* trees. In an unranked tree, a node may have an arbitrary number of children (independent of its label). It is well-known that any unranked tree can be encoded as a binary tree. A commonly used such encoding is the "first-child/next-sibling (FC/NS) encoding" : the first child of an unranked node becomes the left-child in the binary tree, and the next-sibling of a node becomes the right child (and all other edges of the unranked trees are removed). As explained in the Introduction, the FC/NS encoding is problematic in the context of ranked tree transducers. For instance, a DTOP operating on the FC/NS encoding, cannot change the order of children of the corresponding unranked node. The reason is that in the encodings these nodes are *not* children, but are right-descendants, and a DTOP cannot exchange a node with a descendant node.

The idea of our new encoding is to use the DTDs (or XML Schemas) of the input and output documents in order to group items that are distinguished by the DTD in an own subtree, so that the DTOP can distinguish them. Given a finite set of labels F, a DTD D over F consists of a start symbol in F, denoted $start(D)$ and a mapping D that maps each element f of F to a regular expression $D(f)$ over F. In fact, only "1-unambiguous" regular expressions are permitted in DTDs (and XML Schemas). For our purpose it suffices that the regular expressions are unambiguous, which means

that every sequence can be uniquely parsed with respect to the DTD. A regular expression R over F is either and element of F, or the symbol PCDATA, or is of the form R_1^*, R_1^+, $R_1?$, $(R_1|R_2|,\ldots|R_n)$, or $(R_1, R_2, \ldots, R_n)$, for $n \geq 1$ and regular expressions $R_1, \ldots, R_n$. Note that in the last of these forms the comma "," means concatenation.

The best way to explain our encoding, is to assume that the unranked input tree has been parsed against the DTD already, and we have the parse information at hand. If the sequence w was parsed against R^* (or R^+), then we know the subsequences $w_1, \ldots, w_n = w$ such that each w_i is parsed against R. Similarly, if w was parsed against $(R_1, R_2, \ldots, R_n)$, then we know the $w_1 w_2 \cdots w_n = w$ such that each w_i is parsed against R_i. Every data value (text node) of the XML document is parsed into the special symbol PCDATA (for simplicity, we do not consider EMPTY content). Let D be a DTD over F, R a regular expression over F, and w a sequence of unranked trees that parses as $w_1 w_2 \cdots w_n$ against R. The *ranked D-encoding of w with respect to R*, denoted by $enc_D(R, w)$, is defined as

- $\underline{f}(enc_D(D(f), w'))$ if $R = f \in F$, $n = 1$, and w_1 is an unranked tree with root label f and child sequence w'

- PCDATA if $R = $ PCDATA

- $\underline{R_1^*}(\#, \#)$, if $R = R_1^*$ and $n = 0$

- $\underline{R_1^*}(enc_D(R_1, w_1), enc_D(R_1^*, w_2 \cdots w_n))$ if $R = R_1^*$ and $n \geq 1$

- $\underline{R_1^+}(enc_D(R_1, w_1), \#)$ if $R = R_1^+$ and $n = 1$

- $\underline{R_1^+}(enc_D(R_1, w_1), enc_D(R_1^+, w_2 \cdots w_n))$ if $R = R_1^+$ and $n \geq 2$

- $\underline{R_1?}(\#)$ if $R = R_1?$ and $n = 0$

- $\underline{R_1?}(enc_D(R_1, w_1))$ if $R = R_1?$ and $n = 1$

- $\underline{(R_1|\ldots|R_m)}(enc_D(R_i, w))$, where R_i is the unique expression against which w was parsed

- $\underline{(R_1, \ldots, R_m)}(enc_D(R_1, w_1), \ldots, enc_D(R_m, w_m))$.

In our encoding, PCDATA is of rank zero, $\underline{f}$, while $\underline{R?}$ and $\underline{(R_1|\ldots|R_m)}$ are of rank one, $\underline{R^*}$ and $\underline{R+}$ are binary, and $\underline{(R_1, \ldots, R_m)}$ is of rank m. For an unranked tree $t = f(t_1, \ldots, t_m)$ with $f = start(D)$, the *ranked D-encoding of t* is defined as $\underline{f}(enc_D(D(f), t_1 \ldots t_m))$.

As an example, consider the following DTD D_1 (given in W3C syntax; thus $start(D_1) = $ LIBRARY and, for instance, $D_1(\texttt{YEAR}) = $ PCDATA).

```
<!ELEMENT   LIBRARY BOOK* >
<!ELEMENT   BOOK ((AUTHOR, TITLE, YEAR?) | TITLE) >
<!ELEMENT   AUTHOR #PCDATA >
<!ELEMENT   TITLE #PCDATA >
<!ELEMENT   YEAR #PCDATA >
```

Let t_{unr} be the following unranked tree, where each label is abbreviated by its first letter, i.e., LIBRARY is written as "L" and PCDATA as "P":

$$L(B(A(P), T(P)), B(A(P), T(P), Y(P)), B(T(P)))$$

The ranked D-encoding of the tree t_{unr} is equal to $\underline{L}(enc_{D_1}(t_1 t_2 t_3, B^*))$, where t_1, t_2, t_3 are the three subtrees of L, in that order. We obtain $\underline{L}(\underline{B^*}(e_1, \underline{B^*}(e_2, \underline{B^*}(e_3, \underline{B^*}(\#, \#)))))$,

where the e_i are the encodings of the sequences below the B-nodes. For instance, $e_1 = enc_{D_1}(A(P)T(P), B^*)$ which equals

$$\underline{((A, T, Y?)|T)}(\underline{(A, T, Y?)}(\underline{A}(P), \underline{T}(P), \underline{Y?}(\#)).$$

Example XML Transformation

In order to illustrate the kind of transformations that our learning algorithm can infer, we present a transformation that performs swapping, copying and deletion. We consider a transformation τ that transforms input trees conforming to the following DTD D_1:

```
<!ELEMENT   LIBRARY (BOOK*) >
<!ELEMENT   BOOK (AUTHOR, TITLE, YEAR) >
<!ELEMENT   AUTHOR #PCDATA >
<!ELEMENT   TITLE #PCDATA >
<!ELEMENT   YEAR #PCDATA >
```

and translates them into output trees conforming to the following D_2:

```
<!ELEMENT   LIBRARY (SUMMARY,BOOK*) >
<!ELEMENT   SUMMARY (TITLE*) >
<!ELEMENT   BOOK (TITLE, AUTHOR) >
<!ELEMENT   AUTHOR #PCDATA >
<!ELEMENT   TITLE #PCDATA >
```

The content of book-elements is reorganized by τ: author and title are swapped, year is deleted, and a summary is added, which lists just the titles of the books (and hence, titles of book must be copied by a transducer for τ).

For instance, the tree $L(B(A(P), T(P), Y(P)), B(A(P), T(P), Y(P)))$ is transformed into $L(S(T, T), B(T(P), A(P)), B(T(P), A(P)))$. This transformation can be performed by a DTOP with fourteen states. Let s_i be the tree with i book-nodes and t_i its transformation ($t_i = \tau(s_i)$). For instance, $s_2 = L(B(A(P), T(P), Y(P)), B(A(P), T(P), Y(P)))$. Then the set $S = \{(s_0, t_0), (s_1, t_1), (s_2, t_2), (s_3, t_3)\}$ is characteristic for τ. Hence from this set or any of its supersets, the learning algorithm can infer the minimal earliest transducer for τ.

We detail here the construction of S. First, τ's transformation is realised by the following earliest minimal transducer M (working on encodings):

$$ax = \underline{L}(\underline{S}(\underline{T^*}(\langle q_{L1}, x_0\rangle, \langle q_{L2}, x_0\rangle)), \underline{B^*}(\langle q_{L3}, x_0\rangle, \langle q_{L4}, x_0\rangle))$$

$$
\begin{aligned}
q_{L1}(\underline{L}(x_1)) &\rightarrow \langle q_{T1^*}, x_1\rangle \\
q_{L2}(\underline{L}(x_1)) &\rightarrow \langle q_{T2^*}, x_1\rangle \\
q_{L3}(\underline{L}(x_1)) &\rightarrow \langle q_{B1^*}, x_1\rangle \\
q_{L4}(\underline{L}(x_1)) &\rightarrow \langle q_{B2^*}, x_1\rangle \\
q_{T1^*}(\underline{B^*}(x_1, x_2)) &\rightarrow \langle q_T, x_1\rangle \\
q_{T2^*}(\underline{B^*}(x_1, x_2)) &\rightarrow \langle q_{T^*}, x_2\rangle \\
q_{T^*}(\underline{B^*}(x_1, x_2)) &\rightarrow \underline{T^*}(\langle q_T, x_1\rangle, \langle q_{T^*}, x_2\rangle) \\
q_{T^*}(\#) &\rightarrow \# \\
q_{B1^*}(\underline{B^*}(x_1, x_2)) &\rightarrow \langle q_B, x_1\rangle \\
q_{B2^*}(\underline{B^*}(x_1, x_2)) &\rightarrow \langle q_{B^*}, x_2\rangle \\
q_{B^*}(\underline{B^*}(x_1, x_2)) &\rightarrow \underline{B^*}(\langle q_B, x_1\rangle, \langle q_{B^*}, x_2\rangle) \\
q_{B^*}(\#) &\rightarrow \# \\
q_B(\underline{B}(x_1, x_2, x_3)) &\rightarrow \underline{B}(\underline{T}(\langle q_T, x_2\rangle), \underline{A}(\langle q_A, x_1\rangle)) \\
q_B(\#) &\rightarrow \# \\
q_A(A(x_1)) &\rightarrow \langle q_P, x_1\rangle \\
q_T(T(x_1)) &\rightarrow \underline{T}(\langle q_P, x_1\rangle) \\
q_P(P) &\rightarrow P
\end{aligned}
$$

There are fourteen states and fourteen corresponding io-paths for M which are:

$$
\begin{aligned}
q_{L1} &: (\varepsilon; (\underline{L},1)(\underline{S},1)(\underline{T^*},1)) \\
q_{L2} &: (\varepsilon; (\underline{L},1)(\underline{S},1)(\underline{T^*},2)), \\
q_{L3} &: (\varepsilon; (\underline{L},2)(\underline{B^*},1)), \\
q_{L4} &: (\varepsilon; (\underline{L},2)(\underline{B^*},2)),
\end{aligned}
$$

$$
\begin{aligned}
q_{T1^*} &: ((\underline{L},1); (\underline{L},1)(\underline{S},1)(\underline{T^*},1)) \\
q_{T^*2} &: ((\underline{L},1); (\underline{L},1)(\underline{S},1)(\underline{T^*},2)), \\
q_{B1^*} &: ((\underline{L},1); (\underline{L},2)(\underline{B^*},1)), \\
q_{B2^*} &: ((\underline{L},1); (\underline{L},2)(\underline{B^*},2)), \\
q_{T} &: ((\underline{L},1)(\underline{B^*},1); (\underline{L},1)(\underline{S},1)(\underline{T^*},1)) \\
q_{T^*} &: ((\underline{L},1)(\underline{B^*},2); (\underline{L},1)(\underline{S},1)(\underline{T^*},2)(\underline{T^*},2)) \\
q_{B} &: ((\underline{L},1)(\underline{B^*},1); (\underline{L},2)(\underline{B^*},1)) \\
q_{B^*} &: ((\underline{L},1)(\underline{B^*},2); (\underline{L},2)(\underline{B^*},2)(\underline{B^*},2)) \\
q_{A} &: ((\underline{L},1)(\underline{B^*},1)(\underline{B},1); (\underline{L},2)(\underline{B^*},1)(\underline{B},2)) \\
q_{P} &: ((\underline{L},1)(\underline{B^*},1)(\underline{B},1)(\underline{A},1); (\underline{L},2)(\underline{B^*},1)(\underline{B},2)(\underline{A},1)).
\end{aligned}
$$

A characteristic sample for τ can be obtained from the following sets of input trees. We denote s_i the tree with i books nodes. To build the axiom we need $I_A = \{s_0, s_1\}$. For the rules the following set is sufficient to compute the right-hand sides $I_T = \{s_1, s_2, s_3\}$. The computation of origins is direct from the domains so $I_O = \emptyset$. For instance, states q_A and q_T have distincts domains. The only different io-paths that have the same domain are the ones corresponding to q_{S1^*} and q_{B1^*}, q_{S2^*} and q_{B2^*}, q_{S^*} and q_{B^*}, and also q_S and q_B. All of those can be discriminated by $I_D = \{s_1, s_2\}$. Let $t_i = \tau(s_i)$. The set

$$
S = \{(s_0, t_0), (s_1, t_1), (s_2, t_2), (s_3, t_3)\}
$$

is characteristic for τ. This means that from this set or any superset of it, the learning algorithm will be able to build M, the minimal earliest transducer of τ.

11. CONCLUSIONS

Deterministic top-down tree transducers can be learned from examples. To infer a transducer, our algorithm needs polynomially many examples in the size of the minimal transducer. The algorithm does not infer the domain of a transducer, but starts with a domain (given as a deterministic top-down tree automaton).

Although, the learning algorithm is Gold-style [15], i.e., it uses a given set of examples to infer the transducer, it could be used as core in an interactive learner in Angluin-style [1], similar to [5]. Further research is required to justify this claim. In the context of XML, ranked transducers seem particularly useful if input and output DTDs are given. This allows to construct encodings into ranked trees which group similar items (with respect to the DTD) into an own subtree. A DTOP can then delete, interchange, or copy the groups (here, "group" refers to a regular subexpression of the DTD).

Are there other classes beyond DTOP to which our results could be extended? An interesting and robust class are top-down tree transducers with regular look-ahead [9]. Such a transducer is allowed to first execute a bottom-up finite-state relabeling over the input tree, and then run the top-down translation on the relabeled tree. The XPath filters that are used in XSLT programs naturally correspond to bottom-up look-ahead. How can we minimize DTOP transformations with look-ahead (DTOP ^{R}s)? Through changing the relabelings, equivalence of DTOP ^{R}s can be decided [12]. Thus, for a *given* look-ahead, we can minimize the transducer. But it is unclear how to find both a minimal look-ahead and a corresponding minimal transducer, for a given translation. In fact, it is a hard open problem whether or not it is decidable for a given DTOP ^{R} if it has an equivalent DTOP.

12. REFERENCES

[1] D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.

[2] G. J. Bex, S. Maneth, and F. Neven. A formal model for an expressive fragment of XSLT. *Inform. Systems*, 27:21–39, 2002.

[3] G. Jan Bex, W. Gelade, F. Neven, and S. Vansummeren. Learning deterministic regular expr. for the inference of schemas from XML data. In *WWW*, pages 825–834, 2008.

[4] J. Carme, R. Gilleron, A. Lemay, and J. Niehren. Interactive learning of node selecting tree transducer. *Machine Learning*, 66(1):33–67, 2007.

[5] J. Carme, R. Gilleron, A. Lemay, and J. Niehren. Interactive learning of node selecting tree transducers. *Machine Learning*, 66(1):33–67, January 2007.

[6] C. Choffrut. Minimizing subsequential transducers: a survey. *Theor. Comput. Sci.*, 292(1):131–143, 2003.

[7] A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *SIGMOD Conference*, pages 509–520, 2001.

[8] J. Engelfriet. Bottom-up and top-down tree transformations - a comparison. *Math. Syst. Theory*, 9(3):198–231, 1975.

[9] J. Engelfriet. Top-down tree transducers with regular look-ahead. *Math. Syst. Theory*, 10:289–303, 1977.

[10] J. Engelfriet. Some open questions and recent results on tree transducers and tree languages. In *Formal language theory; perspectives and open problems*. Academic Press, 1980.

[11] J. Engelfriet and S. Maneth. Macro tree transducers, attribute grammars, and MSO definable tree translations. *Inf. Comput.*, 154(1):34–91, 1999.

[12] J. Engelfriet, S. Maneth, and H. Seidl. Deciding equivalence of top-down XML transformations in polynomial time. *J. Comput. Syst. Sci.*, 75(5):271–286, 2009.

[13] Z. Ésik. Decidability results concerning tree transducers I. *Acta Cybernetica*, 5:1–20, 1980.

[14] F. Gécseg and M. Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.

[15] E.M. Gold. Complexity of automaton identification from given data. *Inform. Control*, 37:302–320, 1978.

[16] J. Graehl, K. Knight, and J. May. Training tree transducers. *Computational Linguistics*, 34:391–427, 2008.

[17] W. Janssen, A. Korlyukov, and J. Van den Bussche. On the tree-transformation power of XSLT. *Acta Inf.*, 43(6):371–393, 2007.

[18] S. Kepser. A simple proof for the turing-completeness of XSLT and XQuery. In *Extreme Markup Languages*, 2004.

[19] S. Maneth, A. Berlea, T. Perst, and H. Seidl. XML type checking with macro tree transducers. In *PODS*, pages 283–294. ACM Press, 2005.

[20] S. Maneth and G. Busatto. Tree transducers and tree compressions. In *FOSSACS*, pages 363–377, 2004.

[21] S. Maneth and F. Neven. Structured document transform. based on XSL. In *DBPL*, pages 80–98, 1999.

[22] M. Mohri. Minimization algorithms for sequential transducers. *Theor. Comput. Sci.*, 234:177–201, 2000.

[23] A. Morishima, H. Kitagawa, and A. Matsumoto. A machine learning approach to rapid development of XML mapping queries. In *ICDE*, pages 276–287, 2004.

[24] O. Niese. *An integrated approach to testing complex systems*. Phd thesis, Universität Dortmund, Germany, 2003.

[25] J. Oncina, P. Garcia, and E. Vidal. Learning subseq. transduc. for patt. recog. interpretation tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5):448–458, 1993.

[26] R. Onder and Z. Bayram. XSLT version 2.0 is turing-complete: A purely transformation based proof. In

     CIAA, pages 275–276, 2006.

[27] T. Perst and H. Seidl. Macro forest transducers. *Inf. Process. Lett.*, 89(3):141–149, 2004.

[28] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, and R. Fagin. Translating web data. In *VLDB*, pages 598–609, 2002.