A Learning Algorithm for Top-Down Tree Transducers

Adrien Boiret, Aurélien Lemay, Joachim Niehren

A Learning Algorithm for Top-Down Tree Transducers

Adrien Boiret Aurélien Lemay Joachim Niehren

July 20, 2017

Abstract

A generalization from string to trees and from languages to transformations is given of the classical result that any regular language can be learned from annotated examples: we show how to learn top-down deterministic tree transducers (DTOPs) in Gold's model with polynomial resources from samples of input-output examples, while assuming a top-down schema for the input domain. Until now, similar results were known only for string transducers, sequential tree-to-word transducers, and simple relabeling tree transducers. Learning of DTOPs is 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 with top-down domain inspection, which characterizes the unique normal forms of such machines from Engelfriet, Maneth and Seidl in 2009, in a purely semantical manner. This theorem is then used to construct a learning algorithm for DTOPs, for a given top-down domain inspection.

1 Introduction

XML is a widely used format for exchanging semi-structured data between programs in various languages, for storing semi-structured data in databases, and for writing all kinds of documents. Specific XML schemas where proposed by various communities for representing their semi-structured data: XBRL for business data, SPL for pharmaceutical products, or SBML for reaction networks in systems biology, for example. The XML format is also omnipresent in Web pages, given that HTML5 is the language supported by all today's Web navigators. Indeed, HTML5 documents are XML documents satisfying yet another XML schema (in contrast to the original HTML).

Document processing is an application area of the XML technology, which includes the creation of web pages and of software documentations. Such documents are usually developped on a higher level of abstraction, supported by tools such as context management systems (Wiki, Plone, Wordpress, etc), or

by some other document generator (Docbook). A prime idea there is to separate the document's content and its layout. The pure content can again be represented in XML, but with a different schema that reflects the semantics of the content. Only for displaying the content of a document by a Web browser, the content gets enriched by some layout information. This requires to convert an XML document satisfying the content's schema into another XML document satisfying the schema of HTML5.

As illustrated at the task of Web page publication, the main task to be solved in the context of XML is the conversion of XML documents from one schema into another. Similarly, XML transformations need to be defined for composing two programs in different languages, that support XML formats for input and output, or when exporting the result of a query to an XML database. XML transformations can be defined by programing in general purpose programming languages, or by using dedicated XML transformation languages such as the the W3C standards XSLT 3.0. Here, the acronym XSLT stands for XSL Transformations, where XSL means EXtensible Stylesheet Language.

A major drawback of the programming language approach is that programming is accessible only to programmers, but not by arbitrary Web users. However, 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, or the usage of fixed transformations as provided by content management systems. In this paper we present a learning algorithm that can serve as foundation for building such systems. Under the assumption that the schema of the domain of the target tree transformation is given, our algorithm works in a Gold style model in polynomial time and with polynomially many examples [21]: it takes as an input a finite set of pairs of input and output trees of a target transformation, and, if the input is rich enough, can infer a representation of the target.

In order to make this possible, we need to fix a suitable class of machines for defining tree transformations that we want to learn. Besides *(1) decidable equivalence* in the class of machines to be learned we require that the number of examples needed to to learn an machine of size n is *(2) bounded polynomially in* n (polynomial data) and that the learning algorithm *(3) infers the machine in time polynomial in* n. All three points are closely related to efficient normalization of the machines, which is usually based on a Myhill-Nerode like theorem as for deterministic finite automata (DFA). Given a word language L, two strings x and y are called "L-equivalent" if there is no z such that exactly one of xz and yz is in L. The Myhill-Nerode theorem says that the L-equivalence is of finite index if and only if L is regular. Moreover, there exist unique minimal DFA recognizing L, whose states are exactly the L-equivalence classes.

Since XSLT programs are Turing complete [24, 38], polynomial exact learning with (1), (2), and (3) can only be expected for subclasses. The navigational core of XSLT can conveniently be modeled by tree transducers [2, 23, 29, 30]. For example, macro tree transducers (MTTs) [19, 29] are a very expressive class of top-down tree transducers than can model a substantial amount of XSLT tree transformations. This expressiveness, however, comes at a huge computational

cost: the equivalence problem is a long-standing open question on this class. One of the only progress made in that domain is decidability of a fragment of this class [17, 16], through a non-elementary translation into MSO-definable transformations, and the decidability of (deterministic) top-down tree-to-word transducers with concatenantion [40] which can be simulated by MTTs.

The most well-known and largely-studied class of tree transducer for which equivalence is decidable [20] is that of deterministic top-down tree transducers (DTOPs). Engelfriet, Maneth and Seidl [18] showed recently that DTOPs have a unique normal form, under the condition that the domain of input trees is checked externally. For this, they consider DTOP*s with top-down inspection*, where the schema of input trees is checked by an external deterministic top-down tree automaton. The unique normal form a DTOPs with top-down inspection can be computed by a sequence of syntactic normalization steps. First, these machines are synchronized with the top-down automaton that decsribes their domain (uniform). Second, all of their states are made earliest in the output production. Finally, their number of states is minimized.

Results on symbolic learning algorithms for transducers are few and far between. The OSTIA algorithm [37] allows to learn deterministic (subsequential) string transducers in their earliest minimal normal form [10], in the learning model with polynomial resources. As for trees, if bottom-up tree automata [36, 8] in their minimal deterministic normal form can be learned in this model, very few results exists for the learning of tree transformation. However, these pre-existing results can be seen to have enough of a pattern to hope an extension to the DTOP class: once a Myhill-Nerode theorem is found, some RPNI-like algorithm allows for symbolic learning from a finite sample. The main problem with adapting the normalization procedure of [18] in a Myhill-Nerode theorem is the syntactic nature of the properties considered for normalization: uniformity is defined through the inspection automata, and earliestness is defined state by state in the transducer. It was unclear whether such a normal form can be inferred from a finite sample of input-ouput examples, by generalizing the RPNI algorithm.

In the present paper, we show that this is indeed the case. For this, the following contributions are given:

1. Present a Myhill-Nerode theorem for DTOPs with top-down inspection, which recasts their unique normal forms in a purely semantical terms.

2. For a fixed top-down inspection L, provide a learning algorithm for DTOPs with inspection by L that satisfies Gold's learning model with polynomial resources.

Our results are a breakthrough in transducer learning: previous work only considered non-copying and non-swapping transducers (such as word transducers, sequential tree-to-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. These results can indeed be applied to the learning of XML transformations, as shown in [26], onto which this article extends. In this case, the top-down inspection is defined by either a DTD or an XML SCHEMA.

Compared to previous work of Engelfriet, Maneth, and Seidl [18] we consider in this article DTOPs with general domain inspection, rather than top-down inspection by top-down deterministic tree automata. This enables us to introduce the notion of *compatible* DTOP*s with inspection*, a semantic generalization of *uniform* DTOP*s with top-down inspection*. We then show that any DTOP with regular inspection can be made compatible and earliest (while generalizing the previous analogous result for top-down inspection). The normal form we present for DTOPs here remains restricted to top-down inspection.

Compared to [26], we present a generalized and fully independent proof, rather than a reduction to the results of [18]. At the same time, we generalize many of the intermediate results from top-down to regular inspection. While the additional generality is of interest on its own right and lies the foundation for the results of [5], this approach also yields shorter and simpler proofs, basically due to the usage of the notion of compatibility instead of uniformity.

Related Work

In the context of XML, there has been some work on learning node selection queries [8, 34, 6, 7, 41], but only very few on learning tree transformations. The only work on learning XSLT programs that we are aware of is the "XSLT Inference Tool" (part of the Word 2003 SDK by Microsoft). It can infer very restricted types of XSLT programs from a only a single example input and output document pair. The most related work is XLearner [32]. 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 constructsXPath 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 [32], there exists interesting work on inferring schema mappings, e.g., LSD [12] and Clio [39]. 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 [37]. They are based on minimal earliest transducers, which were formally introduced for strings by Mohri [31], see [10] for a survey. A learning algorithm and experimental results for deterministic Mealy machines is presented in [35]. 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 [8], 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 [22].

Outline

In Section 2 we start with an illustration of the learning algorithm and its relationship to the Myhill-Nerode theorem that we want to develop. In Section 3, we recall some preliminaries on tree languages and tree automata.In Section 4, we introduce the class of top-down tree transducers with general inspection (DTOPI). In Section 5 we define the syntactic alignment computed by a DTOPI, as a relation that matches paths of input trees to the corresponding output paths as produced by a DTOPI. Then, Sections 6 and 7 will introduce for all DTOPI with regular inspection two important normalization steps: compatibility, which ensures that the states of a transducer are as specific as they can be on the input tree they expect, and earliestness, which ensures that a transducer produces its output as early as possible. Section 8 provide a semantic counterpart to the syntactic alignment. It lies the basis for a Myhill-Nerode type theorem for DTOPI with top-down inspection. The minimal normal form that results of this Myhill-Nerode type theorem is presented in Section 9. Finally, Section 10 presents a sample-based learning algorithm of this normal form.

2 Illustration of Ideas

Before starting with a formalization, we illustrate informally, how we want to learn DTOPs from input-output examples. In a first step we need to find a Myhill-Nerode Theorem, i.e., that is unique normal form for DTOPs based on a semantic characterization. For this we will have to give a semantic justification to the unique normal forms for DTOPs proposed by Engelfriet, Maneth and Seidl [18].

2.1 Myhill-Nerode Theorem for DTops

A DTOP 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 $q'\langle x_i \rangle$ 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 DTOP 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 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 paths for input respectively output trees, τ-equivalence is of finite index if and only if τ can be realized by a DTop.

2.2 Example Transformation

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 transform

$$\mathsf{P}(\underbrace{\mathsf{A}(\#, \mathsf{A}(\#, \dots \mathsf{A}(\#, \#)\dots))}_{n}, \underbrace{\mathsf{B}(\#, \mathsf{B}(\#, \dots \mathsf{B}(\#, \#)\dots))}_{m})$$

into the tree obtained by exchanging the P's two subtrees.

$$\mathsf{P}(\underbrace{\mathsf{B}(\#, \mathsf{B}(\#, \dots \mathsf{B}(\#, \#)\dots))}_{m}, \underbrace{\mathsf{A}(\#, \mathsf{A}(\#, \dots \mathsf{A}(\#, \#)\dots))}_{n})$$

Since this transformation τ_{flip} is partial, there are exactly 4 different τ_{flip}-equivalence classes; the shortest representatives for these classes are the following pairs of paths for input respectively output trees:

$$\begin{array}{l} q_1 = (\varepsilon, \mathsf{P}1), \\ q_2 = (\varepsilon, \mathsf{P}2), \\ q_3 = (\mathsf{P}2, \mathsf{P}1), \\ q_4 = (\mathsf{P}1, \mathsf{P}2) \end{array} .$$

These path pairs in this order correspond exactly to the states $q_1, \dots, q_4$ of the unique minimal earliest uniform DTop M_{flip} below. It starts with the axiom $\mathsf{P}(q_1\langle x_0\rangle, q_2\langle x_0\rangle)$ and has the following rules:

$$\begin{array}{lcl} q_1(\mathsf{P}(x_1, x_2)) & \to & q_3\langle x_2\rangle \\ q_2(\mathsf{P}(x_1, x_2)) & \to & q_4\langle x_1\rangle \\ q_3(\#) & \to & \# \\ q_3(\mathsf{B}(x_1, x_2)) & \to & \mathsf{B}(\#, q_3\langle x_2\rangle) \\ q_4(\#) & \to & \# \\ q_4(\mathsf{A}(x_1, x_2)) & \to & \mathsf{A}(\#, q_4\langle x_2\rangle) \end{array}$$

Note that a minimal earliest uniform DTops as defined in [18] 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 transformation 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:

$$
\begin{array}{llll}
[& \mathsf{P}(\#,\#) & / & \mathsf{P}(\#,\#), \\
 & \mathsf{P}(\mathsf{A}(\#,\#),\#) & / & \mathsf{P}(\#,\mathsf{A}(\#,\#)), \\
 & \mathsf{P}(\#,\mathsf{B}(\#,\#)) & / & \mathsf{P}(\mathsf{B}(\#,\#),\#), \\
 & \mathsf{P}(\mathsf{A}(\mathsf{A}(\#,\#),\#),\mathsf{B}(\mathsf{B}(\#,\#),\#)) & / & \mathsf{P}(\mathsf{B}(\mathsf{B}(\#,\#),\#),\mathsf{A}(\mathsf{A}(\#,\#),\#)) \quad]
\end{array}
$$

Note that a DTOP can transform 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 [28]).

Inference of XML Transformations

XML documents are naturally modeled by *unranked trees*. There have been several proposal of tree transducers for unranked trees [30, 29]. 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 transformation XML_{flip} of a root node labeled P 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 [30, 29], 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. The equivalence problem for deterministic top-down tree-to-string transducers was recently proven to be decidable [40] with a co-randomized polynomial algorithm for the linear case.

Are there other ranked tree encodings of unranked trees, so that a DTOP can realize XML_{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_{flip}:

```
<!ELEMENT  P (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 sub-expression, 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 P(A, A, B) is represented as

$$\mathsf{P}(\text{“(A*,B*)”}(\text{“A*”}(\mathsf{A}, \text{“A*”}(\mathsf{A}, \text{“A*”}(\#,\#))), \text{“B*”}(\mathsf{B}, \text{“B*”}(\#,\#)))$$

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

$$\mathsf{P}(\text{“(B*,A*)”}(\text{“B*”}(\mathsf{B}, \text{“B*”}(\#,\#)), \text{“A*”}(\mathsf{A}, \text{“A*”}(\mathsf{A}, \text{“A*”}(\#,\#))))).$$

Thus, if we supply adequately DTD-encoded trees to our learning algorithm, then it can infer a ranked transducer for XML_{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 [30, 29]; to see this, observe that the latter class contains both the DTD-encoding and the DTD-decoding.

3 Preliminaries

A partial function from a set A to a set B is a subset $\tau \subseteq A \times B$ such that for any $a \in A$ there exists at most one $b \in B$ such that $(a, b) \in \tau$. In this case we write $\tau(a) = b$. The domain $dom(\tau)$ is the set $\{a \mid \exists b.\ (a, b) \in \tau\}$.

An *alphabet* is a finite set. A word over an alphabet A is a possibly empty sequence of letters $a_1 \dots a_n \in A^*$. We denote the empty word by ε and the concatenation of two words u and u' as uu'.

A *ranked alphabet* is an alphabet F together with a total function $rank_F : F \to \mathbb{N}_0$. If $rank_F(f) = k$, we say that f is of *rank* k, which means it expects k children exactly. For $k \geq 0$ we denote by $F^{(k)}$ the set $\{f \in F \mid rank_F(f) = k\}$. We often write $f^{(k)}$ to indicate that f is of rank k.

3.1 Trees and Paths

Let F be a ranked alphabet. 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 any $f \in F^{(k)}$, $k \geq 0$, and $s_1, \dots, s_k \in \mathcal{T}_F$, the term $f(s_1, \dots, s_k)$ belongs to $\mathcal{T}_F$. For a one-node tree $f()$, where f is a constant, we simply write f. For a finite set X disjoint from F we define $\mathcal{T}_F(X)$ as $\mathcal{T}_{F \cup X}$ where any $x \in X$ has rank 0. We define an F-path to be a word with alphabet $F \cup \mathbb{N}$ in $\{fi \mid f \in F^{(k)},\ 1 \leq i \leq k\}^*$. Note that constants of F cannot appear in F-paths. We will always identify nodes of a tree $s \in \mathcal{T}_F$ with the unique F-path that leads to them:

- The root of s is reached by the empty word ε.

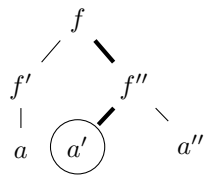


Figure 1: The F-path $u = f2f''1$ and the node that it addresses. The F-npath ua' specifies the label a' of this node in addition.

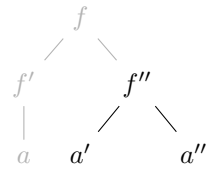


Figure 2: The tree s and its subtree at path $f2$, denoted by $f2^{-1}s$.

- If a node of s is labeled by $f \in F^{(k)}$ and reached by the F-path u, then its i^{th} child is reached by the F-path ufi, where $1 \leq i \leq k$.

An example for a node of a tree s and its corresponding F-path u is given in Fig. 1. In this case, we say that u belongs to s and write:

$$s \models u$$

The set of all paths that belong to s is denoted by $paths(s)$. If $s \models u$, then we denote the label of the node addressed by u with $s[u]$ and the *subtree of* s *at* u by $u^{-1}s$. An example for a subtree is given in Fig. 2.

It should be noticed that an F-path does not state anything about the label of the node to which it leads. We introduce node paths for this purpose as follows. An *F-node-path*, or *F-npath* is a word uf starting with an F-path u and ending with a symbol $f \in F$ of arbitrary arity. Note that uf points to a leaf in some tree if and only if f is a constant. We say that $s \models uf$ if $s \models u$ and $s[u] = f$.

We define two types of substitution on trees. For $s_1, ..., s_n$ not being a subtree of one another, we denote by $\tau = [s_1/t_1, \ldots, s_n/t_n]$ the finite transformation that maps input trees $s_i \in \mathcal{T}_F$ to output trees $t_i \in \mathcal{T}_F$ for all $1 \leq i \leq n$. For any $t \in \mathcal{T}_F$, we write $t\tau \in \mathcal{T}_F$ for the tree obtained from t by substituting subtrees s_i by t_i. For $s_1, ..., s_n$ not being a subtree of one another, we denote by $\alpha = [u_1/s_1, \ldots, u_n/s_n]$ the substitution that maps F-paths u_i to trees s_i for all $1 \leq i \leq n$. For any tree $s \in \mathcal{T}_F$, we write $s\alpha \in \mathcal{T}_F$ for the tree obtained from s by replacing the subtrees at paths u_i by the trees s_i.

Definition 1. *A tree language $L \subseteq \mathcal{T}_F$ is called* path-closed *if, for any two trees $s, s' \in L$ with $s \models u$ and $s' \models u$, it holds that $s[u/u^{-1}s'] \in L$.*

In other words, for any two trees of a path-closed language, one can exchange their subtrees at the same path and remain in the language. For instance, the language $\{f(a,a), f(a',a'), f(a,a'), f(a',a)\}$ is path-closed, as opposed to the language
$\{f(a,a), f(a',a')\}$.

3.2 Tree Automata

We recall the notion of top-down tree automata, their notion of determinism and the relationship with path-closed languages.

Definition 2. *A* (nondeterministic top-down) tree automaton *is a triple* $A = (F, Q, Q_I, \Delta)$ *composed of a finite ranked signature* F*, a finite set* Q *of states, a finite set* $Q_I \subseteq Q$ *of initial states, and a binary relation* $\Delta \subseteq \cup_{k\geq 0}(Q \times F^{(k)}) \times Q^k$ *of transitions.*

Transitions of Δ are denoted as $q \xrightarrow{f} (q_1, \ldots, q_n)$. For all states q we define membership to the language $[\![A]\!]_q$ accepted by A when started at q by induction on the structure on trees, such that for all $k \geq 0$, $f \in F^{(k)}$ and $s_1, \ldots, s_k \in \mathcal{T}_F$:

$$f(s_1, \ldots, s_k) \in [\![A]\!]_q \;\Leftrightarrow_{\text{def}}\; q \xrightarrow{f} (q_1, \ldots, q_k) \text{ and } s_i \in [\![A]\!]_{q_i} \text{ for all } 0 \leq i \leq k$$

We define the language that A accepts by $[\![A]\!] = \cup_{q\in Q_I} [\![A]\!]_q$. A tree language L is called *regular* if $L = [\![A]\!]$ for some tree automaton A.

Definition 3. *We call a top-down tree automaton* A deterministic *or equivalently a* DTTA*, if its set of initial states* Q_I *contains at most one element and if its transition relation* Δ *is a partial function.*

Note that top-down determinism translates the fact that for any given tree, the automaton always has at most one choice for labelling the nodes of a tree when processing it from the root to the leaves. It is well-known that not all regular languages can be recognized by deterministic top-down tree automata (in contrast to deterministic bottom-up tree automata, see e.g. [11]). But since we deal with top-down tree transducers in the present paper, working with top-down automata does suits our needs better than working with bottom-up tree automata.

We now introduce some terminology to reason on the different labellings a node can get in a tree from a top-down tree automaton: For any tree automaton A and path u for the signature of A, we define the set of states Q_u that A assigns to node u as follows by induction on the length of u as follows: we set $Q_\varepsilon = Q_I$ and $Q_{ufi} = \{q_i \mid q \xrightarrow{f} (q_1, \ldots, q_k),\ q \in Q_u\}$. If A is deterministic, then for every u the set Q_u is either empty or a singleton.

We call a tree automaton A *trimmed* if any states $q \in Q$ can be reached from Q_I and satisfies that $[\![A]\!]_q$ is nonempty. Every tree automaton can be made trimmed in linear time by removing all useless states, i.e., those that cannot be reached from Q_I and those for which $[\![A]\!]_q$ is empty. Therefore, we

can assume that all tree automata are trimmed whenever this will be needed in what follows. For any trimmed tree automaton A and path u, note that Q_u is a singleton if and only if $u \in \mathit{paths}(\llbracket A \rrbracket)$.

We next state a simple but fundamental lemma on top-down tree automata.

Lemma 4. *Let A be a top-down tree automaton. Then any path u satisfies $u^{-1}\llbracket A \rrbracket = \cup_{q \in Q_u} \llbracket A \rrbracket_q$.*

Proof. This proof work by recursion on the size of u. For ε, we have $\varepsilon^{-1}\llbracket A \rrbracket = \llbracket A \rrbracket$, which is $\cup_{q_I \in Q_I} \llbracket A \rrbracket_{q_I}$. For starting the induction, we note that $Q_\varepsilon = Q_I$, so that $\varepsilon^{-1}\llbracket A \rrbracket = \cup_{q \in Q_\varepsilon} \llbracket A \rrbracket_q$. For the induction step, let ufi be a path of F. We have $u^{-1}\llbracket A \rrbracket = \cup_{q \in Q_u} \llbracket A \rrbracket_q$ by induction hypothesis. From there, we have to search for all fi-subtrees. If $q \xrightarrow{f} (q_1 \cdots q_n)$ is in Δ, then to build a tree of $u^{-1}\llbracket A \rrbracket$ we have to choose for all j, any tree of $\llbracket A \rrbracket_{q_j}$. Since A is trimmed, these $\llbracket A \rrbracket_{q_j}$ are nonempty. In particular, all $\llbracket A \rrbracket_{q_i}$ is in $ufi^{-1}\llbracket A \rrbracket$. Since this is true for all $q \in Q_u$ and all (q, f) rule, we have $\cup_{q \in Q_{ufi}} \llbracket A \rrbracket_q \subseteq ufi^{-1}\llbracket A \rrbracket$. Conversely, any tree in $u^{-1}\llbracket A \rrbracket$ that contains the path fi starts its run with a (q, f)-rule $q \xrightarrow{f} (q_1 \cdots q_n)$ such that $q \in Q_u$. Its subtree at fi is in q_i. Since $q \in Q_u$, $q_i \in Q_{ufi}$. This gives us $ufi^{-1}\llbracket A \rrbracket \subseteq \cup_{q \in Q_{ufi}} \llbracket A \rrbracket_q$. □

Lemma 5. *Any language accepted by some* DTTA *is* path-closed.

Proof. Let A be a trimmed DTTA, $u \in \mathit{paths}(\llbracket A \rrbracket)$, and $s \in \llbracket A \rrbracket$ such that $s \models u$. We know that Q_u is a singleton $\{q\}$. As seen in Lemma 4, we can replace $u^{-1}s$ by any other tree t in $\llbracket A \rrbracket_q$, and have $s[u/t] \in \llbracket A \rrbracket$. Since $u^{-1}\llbracket A \rrbracket \subseteq \llbracket A \rrbracket_q$, we obtain that $\llbracket A \rrbracket$ is path-closed. □

The converse of Lemma 5 does not hold. But it is true that a language is recognizable by some DTTA if and only if it is regular and path closed.

4 Top-Down Tree Transducers

We study deterministic top-down tree transducers on ranked trees [14], since these can be used to model a subclass of XSLT transformations [29, 25], while being sufficiently restrictive to conserve some good algorithmic properties, such as decidability of equivalence [18].

4.1 Top-Down Tree Transducers

We fix notations and present the standard definition of top-down tree transducers, together with some basic results.

We fix an infinite set $X = \{x_0, x_1, x_2, \dots\}$ of *input variables*, and, for every $k \geq 0$ define the set $X_k = \{x_1, \dots, x_k\}$, so that $X_0 = \emptyset$ in particular.

Definition 6. *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,*
- $Ax \subseteq \mathcal{T}_G(Q \times \{x_0\})$ *is a set with at most one element called the* axiom*,*
- *rhs is a partial function, which for any* $k \geq 0$ *maps elements from* $Q \times F^{(k)}$ *to trees in* $\mathcal{T}_G(Q \times X_k)$*.*

Note that the pair (q, x_i) *will be noted* $q\langle x_i \rangle$*. We define the transformations* $[\![M]\!]_q$ *for all states* q *by mutual recursion and on induction of the size of the tree* $s = f(s_1, \ldots, s_k) \in \mathcal{T}_F$ *:*

$$[\![M]\!]_q(f(s_1, \ldots, s_k)) = rhs(q, f)\,[q'\langle x_i \rangle / [\![M]\!]_{q'}(s_i) \mid q' \in Q,\ 1 \leq i \leq k]$$

The transformation defined by M *is the partial function* $[\![M]\!]$ *from* $\mathcal{T}_F$ *to* $\mathcal{T}_G$ *such that for all* $s \in \mathcal{T}_F$ *for which the expression on the right is defined for some axiom* $ax \in Ax$*:*

$$[\![M]\!](s) = ax[q\langle x_0 \rangle / [\![M]\!]_q(s) \mid q \in Q]$$

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.

As an additional remark, the particular case $Ax = \emptyset$, while necessary to provide stability properties, describes the empty transduction $\tau = \emptyset$. This case will often be ignored in this paper , as for most of these proofs the empty case is trivial and cumbersome. We note $M = (Q, F, G, ax, rhs)$ (with ax instead of Ax) a transducer with exactly one axiom.

In the following two examples, we present two transducers that flip, copy, and delete subtrees.

Example 7. *We consider the transformation* τ_{flip} *which flips pairs of* A*-lists on the left and of* B*-lists on the right. An example for an input-output pair of* τ_{flip} *is given in Fig. 3. The signatures are* $F = G = \{\mathsf{P}^{(2)}, \mathsf{A}^{(1)}, \mathsf{B}^{(1)}, \#^{(0)}\}$*, where* P *is the pair constructor and* # *the end marker for lists. Transformation* τ_{flip} *can be defined by the* DTop M_7 *with axiom* $q_0\langle x_0 \rangle$ *and the following transitions:*

(1) $q_0(\mathsf{P}(x_1, x_2)) \to \mathsf{P}(q_b\langle x_2 \rangle, q_a\langle x_1 \rangle)$
(2) $q_a(\mathsf{A}(x_1)) \to \mathsf{A}(q_a\langle x_1 \rangle)$
(3) $q_a(\#) \to \#$
(4) $q_b(\mathsf{B}(x_1)) \to \mathsf{B}(q_b\langle x_1 \rangle)$
(5) $q_b(\#) \to \#$

The transducer M_7 *has axiom* $q_0\langle x_0 \rangle$*, so it starts in state* q_0 *without producing any initial output. It then applies transition rule (1) to the root, which flips its two children. From there, the state* q_a *copies any* A*-list using rules (2) for* A*'s and (3) for* #*. Similarly, the state* q_b *copies any* B*-list using rules (4) for* B*'s and (5) for* #*.*

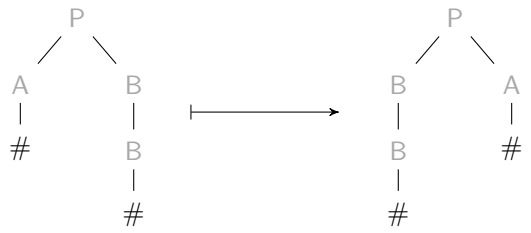


Figure 3: An input-output pair of transformation τ_{flip} flipping pairs of A-lists and B-lists.

We next present an equivalent transducers that produces its output in an earlier manner, while copying and deleting instead of flipping subtrees.

Example 8. *The new* DTop M_8 *starts with the axiom* $\mathsf{P}(q_2\langle x_0\rangle, q_1\langle x_0\rangle)$ *and then applies the following transition rules:*

(0) $ax = \mathsf{P}(q_2\langle x_0\rangle, q_1\langle x_0\rangle)$

(1) $q_1(\mathsf{P}(x_1, x_2)) \to q_a\langle x_1\rangle$ (2) $q_2(\mathsf{P}(x_1, x_2)) \to q_b\langle x_2\rangle$

(3) $q_a(\mathsf{A}(x_1)) \to \mathsf{A}(q_a\langle x_1\rangle)$ (4) $q_a(\#) \to \#$

(5) $q_b(\mathsf{B}(x_1)) \to \mathsf{B}(q_b\langle x_1\rangle)$ (6) $q_b(\#) \to \#$

This transducer M_8 outputs the root right away (possible as it is always labeled P*). The input tree is copied twice. One copy of the tree is read by q_2 which deletes its left son and send the right one to q_b using rule (2). The other is read by q_1 which deletes its right son and send the left one to q_a using rule (1). From there, as in M_7 the state q_a copies any* A*-list using rules (3) for* A*'s and (4) for #. Similarly, the state q_b copies any* B*-list using rules (5) for* B*'s and (6) for #.*

4.2 Top-Down Domains

The domain of a transducer M is the set of input trees s for which $[\![M]\!](s)$ is well-defined, i.e., $dom[\![M]\!]$. It is folklore [15] that the domain of any DTops is accepted by some Dtta, as restated in the following lemma.

Lemma 9. *The domain of any* DTop *is recognizable by some* Dtta*.*

Proof. Let $M = (Q, F, G, \{ax\}, rhs)$ be a DTop and s an input tree (note that if M has no axiom, $dom([\![M]\!]) = \emptyset$, which is obviously recognized by a Dtta). Intuitively, s is in $dom([\![M]\!])$ if M has an axiom, and at no point during the construction of $[\![M]\!](s)$ do we call $[\![M]\!]_q$ on a subtree $u^{-1}s$ such that $s[u] = f$, but $rhs(q, f)$ is undefined. However, unlike for Dttas, a DTop may visit the same subtree $u^{-1}s$ several times and in different states during a computation. In this case, such a subtree must belong to the domain $dom([\![M]\!]_q)$ for all such q. Therefore, we need to reason about the subsets of states of M that visits $u^{-1}S$.

We define the automaton $A = (F, P, p_I, \Delta)$ recognizing $dom(M)$ as follows.

- The states $P = \{Q' \mid Q' \subseteq Q\}$.
- The initial state is $p_I = Q_I = \{q \mid q\langle x_0\rangle \text{ occurs in } ax\}$.
- For $Q' \in P$, and $f \in F^{(k)}$ such that for all $q \in Q'$, $rhs(q, f)$ is defined, then the rule of Δ on (Q', f) is $Q' \xrightarrow{f} (Q_1, \ldots, Q_k)$, where for all i from 1 to k, $Q_i = \bigcup_{q \in Q'} \{q' \mid q'\langle x_i\rangle \text{ occurs in } rhs(q, f)\}$.

We next prove that $[\![A]\!]_{Q'} = \cap_{q \in Q'} dom([\![M]\!]_q)$. This is done by induction on the input tree s. For $s = f(s_1, \ldots, s_k)$, we have $s \in \cap_{q \in Q'} dom([\![M]\!]_q)$ if and only if for every $q \in Q'$, $rhs(q, f)$ is defined, and for every $q'\langle x_i\rangle$ occuring in $rhs(q, f)$, $s_i \in dom([\![M]\!]_{q'})$. In other words, for every $q \in Q'$, $rhs(q, f)$ is defined, and for every i from 1 to k, for all q' such that $q'\langle x_i\rangle$ occurs in a $rhs(q, f)$, $q \in Q'$, then $s_i \in dom([\![M]\!]_{q'})$. For every i, these q' describe exactly the set Q_i in our construction such that $Q' \xrightarrow{f} (Q_1, \ldots, Q_k)$. Since by recursion, $s_i \in \cap_{q' \in Q_i} dom([\![M]\!]_{q'})$ is the same as $s_i \in [\![A]\!]_{Q_i}$, we have that $s \in \cap_{q \in Q'} dom([\![M]\!]_q)$ if and only if for all $q \in Q'$, $rhs(q, f)$ is defined, and for i from 1 to k, $s_i \in [\![A]\!]_{Q_i}$. Since $Q' \xrightarrow{f} (Q_1, \ldots, Q_k)$ is a rule of Δ, so this is equivalent to $s \in [\![A]\!]_{Q'}$.

From there, we justify the choice of Q_I as an initial state. By definition of $[\![M]\!]$, $s \in dom([\![M]\!])$ if and only if for all q such that $q\langle x_0\rangle$ occurs in ax, then $s \in dom([\![M]\!]_q)$. This is equivalent to say that $s \in \bigcap_{q \in Q_I} dom([\![M]\!]_q)$ which, as seen above, is equivalent to $s \in [\![A]\!]_{Q_I}$. Hence, $[\![A]\!] = [\![A]\!]_{Q_I} = dom([\![M]\!])$. □

4.3 Domain Inspection

We note an important weakness of DTOPs: they are not closed under domain restrictions by DTTAs, since they cannot traverse or check those subtrees of the input tree that do not produce any output.

Example 10. *The finite partial function* $\tau_{10} = [f(c, a)/a, f(c, b)/b]$ *cannot be defined by any* DTOP. *This problem is that any* DTOP *must produce the output at the second leaf of the input trees, since the constant that is output depends on which is this leaf. And since nothing else may be output, nothing may be output at the first leaf. Therefore, the first subtree of the input tree cannot be traversed by any* DTOP *defining* τ_{10}*, so it cannot be checked whether the first subtree is a* c *leaf. Nevertheless, there exists a* DTOP M_{10} *such that if* $s \in dom(\tau_{10})$*, then* $[\![M_{10}]\!](s) = \tau_{10}(s)$. M_{10} *has two states* q_1, q_2*, an axiom* $q_0\langle x_0\rangle$ *and the following transition rules:*

$$(1)\quad q_0(f(x_1, x_2)) \to q_1\langle x_2\rangle \qquad (2)\quad q_1(a) \to a \qquad (3)\quad q_1(b) \to b$$

Futhermore, $dom(\tau_{10}) = \{f(c, a), f(c, b)\}$ *is* DTTA*-definable. Therefore, this example shows that the class of* DTOP*s is not closed under top-down inspection, i.e., by domain inspection with* DTTA*s.*

In order to resolve this problem, we follow the approach of [18], and extend DTOPs with domain inspection. In contrast to there, however, we will not only consider domain inspection by DTTAs, but permit more general devices for defining tree languages.

Definition 11. *A* DTOPI *is a* DTOP *with domain inspection, i.e., a pair* $N = (M, D)$ *where* M *is a* DTOP *with input signature* F *and* $D \subseteq \mathcal{T}_F$ *a set of input trees.*

The semantics of a DTOPI is defined by domain restriction, i.e., $[\![N]\!] = [\![M]\!]_{|D}$. Claerly, $dom([\![N]\!]) = dom([\![M]\!]) \cap D$. Note that we admit nonregular tree languages D as inspection domains, so that the domain of the transformation of a DTOPI may be nonregular too.

Definition 12. *A* DTOPI_{reg} *is a* DTOPI *whose inspection domain is regular and a* DTOPI_{td} *is a* DTOPI *whose inspection domain is recognizable by some top-down deterministic tree automata (i.e. it is regular and path-closed).*

Example 13. *We can define the transformation* τ_{flip} *by the* DTOPI $N_{13} = (M_{13}, dom(\tau_{flip}))$ *such that* M_{13} *flips arbitrary pairs, and not only pairs of* A*-list and* B*-lists as done by the* DTOPs M_7 *and* M_8 *from Examples 7 and 8. For this, a single state* q *is sufficient. Furthermore,* M_{13} *has the axiom* $q\langle x_0\rangle$ *and the following transition rules:*

$$\begin{array}{llll} (1) & q(\mathsf{P}(x_1, x_2)) \to \mathsf{P}(q\langle x_2\rangle, q\langle x_1\rangle) & (2) & q(\mathsf{A}(x_1)) \to \mathsf{A}(q\langle x_1\rangle) \\ (3) & q(\mathsf{B}(x_1)) \to \mathsf{B}(q\langle x_1\rangle) & (4) & q(\#) \to \# \end{array}$$

Clearly, $dom(\tau_{flip})$ *is strictly subsumed by* $dom(M_{13})$*, so that external domain inspection is needed to define* τ_{flip} *properly with these more generic rules.*

Proposition 14. *For any* DTOPI $N = (M, D)$*, the domain and inspection domain are related as follows:*

- *if* D *is regular then* $dom([\![N]\!])$ *is regular*
- *if* D *is path-closed then* $dom([\![N]\!])$ *is path-closed*

So if D *is definable by some* DTTA *then* $dom([\![N]\!])$ *is definable some* DTTA *too.*

Proof. Lemma 9 shows that $dom([\![M]\!])$ is always definable by a DTTA, and thus path-closed and regular. Therefore, the Proposition follows from $dom([\![N]\!]) = dom([\![M]\!]) \cap D$, and the fact that both path-closedness and regularity are closed under intersection. □

5 Syntactic Equivalence

We introduce a notion of syntactic alignment computed by DTOPIs, that relate the paths of input trees to the paths of output trees that they produce. We then define a equivalence relation on syntactically aligned pairs of paths, stating that the DTOPIs performs the same transformation there.

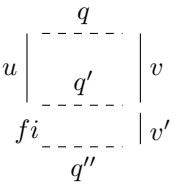
5.1 Syntactic Alignment

We define a notion of syntactic alignment for transducers, to track which paths of output trees are produced by which paths of input trees.

The judgements $u \sim^q_{q'} v$ and $u \sim_q v$ we aim to define describe which input subtrees produce which output subtrees in which states for a DTopI N. Pairs$Succ_N(u, v)$ can thus be understood as the pairs at the "next step" in the computation of an image $[\![N]\!](s)$, where $s \models u$.

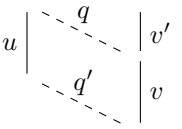
Definition 15. *Let $N = (M, D)$ be a* DTopI*. We define judgements $u \sim^q_{q'} v$ stating that an input path u is aligned to an output path v in state q' when starting from state q, by the following inferences rules where $f^{(k)} \in F$ and $1 \leq i \leq k$:*

$$\frac{}{\varepsilon \sim^q_q \varepsilon} \qquad \frac{u \sim^q_{q'} v \qquad rhs(q', f) \models v' q''\langle x_i \rangle}{ufi \sim^q_{q''} vv'}$$



Furthermore, we define judgements $u \sim_{q'} v$, stating that u is syntactically aligned to v in state q' (or that the pair (u, v) is syntactically aligned in state q'), when starting with the axiom:

$$\frac{ax \models v' q\langle x_0 \rangle \qquad u \sim^q_{q'} v}{u \sim_{q'} v'v}$$



Note that the notion of syntactic alignment for $N = (M, D)$ depends only of the DTop M, so it is independent of the domain inspection D.

Example 16. *We reconsider the two transducers defining the function τ_{flip}. For transducer M_7 from Example 7, which was not earliest, the pair $\varepsilon \sim_{q_0} \varepsilon$, P1 $\sim_{q_b}$ P2, and P2 $\sim_{q_a}$ P1. For the earliest transducer M_8 from Example 8, we have $\varepsilon \sim_{q_b}$ P1, $\varepsilon \sim_{q_a}$ P2, P1 $\sim_{q_a}$ P2A1, and* P1B1 $\sim_{q_b}$ P2B1.

We define the notion of *syntactic successors* of a syntactically aligned pair p in a top-down manner.

Definition 17. *Let $N = (M, D)$ a* DTopI*, $M = (Q, F, G, ax, rhs)$, $p = (u, v)$ a pair syntactically aligned in a state $q \in Q$. For any $f \in F$ such that $rhs(q, f)$ is defined, and any path v' such that $v'^{-1} rhs(q, f) = q'\langle x_i \rangle$ for some $q' \in Q$ we define:*

- *$ind_N(p, f, v') = i$ the index of p, f, v' and*
- *$succ_N(p, f, v') = (ufi, vv')$ the successor of p, f, v'.*

Furthermore, we define $Succ_N(p)$ as the set of all syntactic successors of p with respect to N and some f, v', i.e.:

$$Succ_N(p) = \{succ_N(p, f, v') \mid f \in F,\ q, q' \in Q,\ v'^{-1} rhs(q, f) = q'\langle x_i \rangle\}$$

Corollary 18. *Let $N = (M, D)$ a* DTopI*, $p = (u, v)$ a pair syntactically aligned in a state $q \in Q$. Then all pairs of $Succ_N(p)$ are syntactically aligned in some state $q' \in Q$.*

Proof. Let $M = (Q, F, G, ax, rhs)$. Since $p = (u, v)$ is syntactically aligned in $q \in Q$, there exists v_0, v_1 and a state $q_0 \in Q$ such that $v = v_0 v_1$, $u \sim_q^{q_0} v_1$ and $v_0^{-1} ax = q_0\langle x_0\rangle$. A pair $(ufi, vv') = succ_N(p, f, v')$ if $rhs(q, f)$ is defined, and $v'^{-1} rhs(q, f) = q'\langle x_i\rangle$. This means that $ufi \sim_{q'}^{q_0} v_1 v'$, and thus $ufi \sim_{q'} vv'$. □

The following proposition states the general relevance of syntactic alignment for the transformation defined by a transducer.

Proposition 19. *Let $N = (M, D)$ be a* DTopI*, q a state of M, and (u, v) a pair of input-output paths. If $u \sim_q v$ and $s \in dom([\![N]\!])$ is an input tree with $s \models u$, then $v^{-1}([\![N]\!](s)) = [\![M]\!]_q(u^{-1}s)$.*

Proof. We first show for all u, v, q, q'' that if $u \sim_{q''}^{q} v$ then any $s \in dom([\![M]\!]_q)$ with $s \models u$ satisfies $v^{-1}([\![M]\!]_q(s)) = [\![M]\!]_{q''}(u^{-1}s)$. The proof is by induction on the definition of judgements $u \sim_{q''}^{q} v$.

- In the first case, the judgement $u \sim_{q''}^{q} v$ is derived by the initial rule:

$$\frac{true}{\varepsilon \sim_q^q \varepsilon}$$

  Hence, $u = v = \varepsilon$ and $q = q''$. The output tree $[\![M]\!]_q(s)$ is trivially well-defined for all $s \in dom([\![M]\!]_q)$. Furthermore, it is equal to $\varepsilon^{-1}([\![M]\!]_q(s)) = [\![M]\!]_q(\varepsilon^{-1}s)$.

- In the second case, the syntactic alignment $u \sim_{q''}^{q} v$ is defined as follows, where $u = u'fi$, $f^{(k)} \in F$, $1 \leq i \leq k$, and $v = v'v''$:

$$\frac{u' \sim_{q'}^{q} v' \qquad rhs(q', f) \models v''q''\langle x_i\rangle}{u'fi \sim_{q''}^{q} v'v''}$$

  The induction hypothesis applied to $u' \sim_{q'}^{q} v'$ shows that $v'^{-1}[\![M]\!]_q(s) = [\![M]\!]_{q'}(u'^{-1}s)$. Since $s \models u$, we have $u'^{-1}s = f(s_1, \ldots, s_k)$ for some $s_1, \ldots, s_k$. The recursive definition of $[\![M]\!]_{q'}$ yields:

$$[\![M]\!]_{q'}(u'^{-1}s) = rhs(q', f)\,[\tilde{q}\langle x_j\rangle \leftarrow [\![M]\!]_{\tilde{q}}(s_j) \mid \tilde{q} \in Q,\ 1 \leq j \leq k]\,.$$

  This gives us $v''^{-1}[\![M]\!]_{q'}(u'^{-1}s) = [\![M]\!]_{q''}(s_i)$. Therefore:

$$v^{-1}[\![M]\!]_q(s) = v''^{-1}[\![M]\!]_{q'}(u'^{-1}s) = [\![M]\!]_{q''}(s_i) = [\![M]\!]_{q''}(u^{-1}s)$$

In order to prove the proposition, we recall that for any $s \in dom([\![N]\!])$:

$$[\![N]\!](s) = [\![M]\!](s) = ax[\tilde{q}\langle x_0\rangle \leftarrow [\![M]\!]_{\tilde{q}}(s) \mid \tilde{q} \in Q].$$

The judgement $u \sim_q v$ must be derived as follows where $v = v'v''$:

$$\frac{ax \models v'q\langle x_0\rangle \qquad u \sim^q_{q'} v''}{u \sim_{q'} v'v''}$$

Since $ax \models v'q\langle x_0\rangle$, we have $v^{-1}N(s) = v''^{-1}[\![M]\!]_q(s)$. From $u \sim^q_{q'} v''$ the above claim shows $v''^{-1}[\![M]\!]_q(s) = [\![M]\!]_{q''}(u^{-1}s)$ and thus $v^{-1}N(s) = [\![M]\!]_{q''}(u^{-1}s)$. □

5.2 Trimmed Transducers

The notion of syntactic alignments leads to a proper notion of trimmed DTOPIs:

Definition 20. *A* DTOPI N *is called trimmed if it does not contain any useless states and rules, where:*

- *a state q of N is called useless if there is no pair $p = (u, v)$ aligned in q such that $s \models u$ for some tree $s \in dom([\![N]\!])$, and*
- *a transition $rhs(q, f)$ is called useless if there exists no pair (u, v) aligned in q such that $s \models uf$ for some $s \in dom([\![N]\!])$.*

A trimmed version of a DTOPI N is the DTOPI $trim(N)$ that is obtained from N by removing useless states, useless transitions, and all terms with useless states from the set of axioms.

Lemma 21. *Any* DTOPI_{reg} *is equivalent to some trimmed* DTOPI_{reg} *and any* DTOPI_{td} *is equivalent to some trimmed* DTOPI_{td}.

Proof. If N is a $\text{DTOPI}_{\text{reg}}$ then $trim(N)$ is an equivalent trimmed $\text{DTOPI}_{\text{reg}}$, and if N is a DTOPI_{td} then $trim(N)$ is an equivalent trimmed DTOPI_{td}. □

The computation of $trim(N)$ from N requires to identify the useless states and rules of N, which is less obvious. The actual construction is given in the appendix.

5.3 Origins of Output Constructors

The notion of syntactic alignment allows to define the "origin" of any constructor of an output tree produced by a DTOP, i.e., the unique path of the input tree, at which the DTOP produced that label.

Proposition 22. *Let M be a* DTOP *and $s \in dom(M)$ an input tree. Then for any output path vg such that $[\![M]\!](s) \models vg$, either $ax \models vg$ or there exist a unique decomposition $v = v'v''$, an input path $u'f$, and a state q' such that:*

- $s \models u'f$,
- $u' \sim_{q'} v'$, *and*

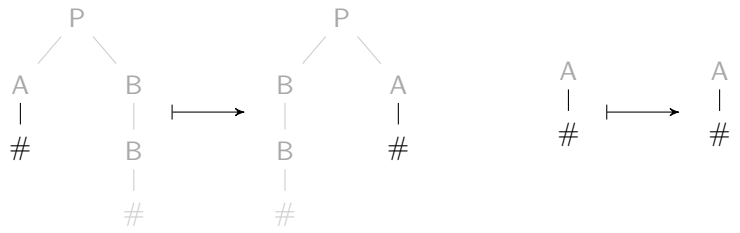


Figure 4: A pair of τ_{flip} on the left, and a pair of its residual at $(\mathsf{P}1, \mathsf{P}2)$ on the right.

- $rhs(q', f) \models v''g$.

The intuition is that each constructor of the output tree is created by the DTop in a production step at a unique input node, where the correspondence between input and output nodes is captured by the notion of syntactic alignments.

The formal proof is given in the appendix. It is not difficult but a little cumbersome since it requires an equivalent bottom-up definition of syntactic alignments.

5.4 Syntactic Equivalence

Residuals play a central role in Myhill-Nerode theorems, as known from the cases of deterministic finite word automata [33] and of subsequential transducers [37]. Therefore, we would like to define a notion of *residuals* of tree transformations, independent of the transducers that might compute it, that state what transformation remains to be done at the current "event" of a top-down transduction process. Such events are pairs of paths $p = (u, v)$, stating that path u of the input tree was read, for producing the output tree until path v.

Definition 23. *The* residual $p^{-1}\tau$ *of a partial function* $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ *at a pair* $p = (u, v)$ *of an* F*-path and a* G*-path, is the relation* $p^{-1}\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ *with:*

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

Example 24. *For the transformation* τ_{flip} *and the pair of path* $p = (\mathsf{P}1, \mathsf{P}2)$*, the residual* $p^{-1}(\tau_{flip})$ *is the identity on* A*-lists (see Fig 5.4).*

In general, each transformation can have an infinite number of different residuals for the infinitely many possible pairs p of paths. However, we only consider very particular pairs of paths. For instance, we do 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. For example, for τ_{flip}, the residual of $(\mathsf{P}1, \mathsf{P}1)$ is not functional. We also do not care about pairs $p = (u, v)$ for which the residual $p^{-1}\tau$ is empty. This happens if

u does not belong to any input tree $s \in dom(\tau)$, or if v is not a node of any $\tau(s)$ where $s \in dom(\tau)$. For example, for τ_{flip}, this happens for the pairs with $u = \mathsf{P1B1}$ or $v = \mathsf{P1A1}$.

The next lemma shows for any DTOPI $N = (M, D)$ that if a pair of paths p is aligned to q in a transducer N as in Definition 15 then $p^{-1}[\![N]\!]$ is a partial function depending on state q and on the residual of the domain.

Lemma 25. *Let $N = (M, D)$ be a* DTOPI *with state q. If $p = (u, v)$ satisfies the syntactic alignement $u \sim_q v$, then $p^{-1}[\![N]\!] = [\![M]\!]_{q|u^{-1}dom([\![N]\!])}$.*

Proof. Proposition 19 gives us $v^{-1}[\![N]\!](s) = [\![M]\!]_q(u^{-1}s)$. By definition of $p^{-1}[\![N]\!]$, $v^{-1}[\![N]\!](s) = p^{-1}[\![N]\!](u^{-1}s)$. We then have $p^{-1}[\![N]\!](u^{-1}s) = [\![M]\!]_q(u^{-1}s)$ for all $s \in dom([\![N]\!])$. By definition, $u^{-1}dom([\![N]\!]) = \{u^{-1}s \mid s \in dom([\![N]\!]), s \models u\}$. We then have $p^{-1}[\![N]\!](t) = [\![M]\!]_q(t)$ for all $t \in u^{-1}dom([\![N]\!])$. □

Definition 26. *Let $N = (M, D)$ be a* DTOPI*. We define the* congruence relation $\equiv_N$ *on pairs p_1 and p_2 of labeled paths that are syntactically aligned by N as follows:*

$$p_1 \equiv_N p_2 \text{ iff } p_1^{-1}[\![N]\!] = p_2^{-1}[\![N]\!]$$

Corollary 27. *The syntactic congruence $\equiv_N$ of a* DTOPI$_{reg}$ *$N = (M, D)$ has finite index.*

Proof. Let $p = (u, v)$ be a pair of paths such that $u \sim_q v$. Lemma 25 implies that $p^{-1}[\![M]\!] = [\![M]\!]_{q|u^{-1}dom([\![N]\!])}$. The domain $dom([\![N]\!])$ is $dom([\![M]\!]) \cap D$. Since both sets are regular it follows that $dom([\![N]\!])$ is regular to. Let A be a trimmed nondeterministic tree automaton (with state set R) that recognizes $dom([\![N]\!])$. Lemma 4 shows that $u^{-1}dom([\![N]\!]) = \cup_{r \in R_u}[\![A]\!]_r$. Hence, $p^{-1}[\![M]\!]$ is characterized by a state q of M and a subset R_u of states of A. Since there are finitely many choices for both, there exists only finitely many possible values of $p^{-1}[\![M]\!]$ for all aligned paths p. □

This corollary is a kind of Myhill-Nerode theorem, but has the disadvantage that the congruence relation $\equiv_N$ is defined on objects that depend on the transducer N, rather than only on the transformation $[\![N]\!]$. Therefore, it does not immediately lead us to a unique minimal normal form of the transformation. For example, the two transducers presented for τ_{flip} in Examples 7 and 8 both have no redundant states, but their equivalences are incomparable: neither is a refinement of the other.

6 Compatible Transducers

Equivalent DTOPIs may check the membership of an input tree to the domain of the transformation in many different manners. In the one extreme case, where no output is to be produced, the job can be entirely done by the domain inspection. In the other extreme case, the domain can be entirely checked by

the underlying DTop. In general case, the DTop and the domain inspection have to share the job in some way or another.

Example 28. *We first consider a* DTopI *which mostly leaves the membership test of the input tree to the domain inspection. It is the* DTopI $(M_{13}, dom(\tau_{flip}))$ *from Example 13. This* DTopI *defines the transformation* τ_{flip} *which flips any pair of* A*-lists and* B*-lists. Its* DTop $(M_{13}, dom(\tau_{flip}))$ *has a single state* q *and the pairs* $(\mathsf{P}1, \mathsf{P}2)$ *and* $(\mathsf{P}2, \mathsf{P}1)$ *are both aligned in* q*. Even though aligned in the same state, the residuals of these pairs differ in their domains. The domain of the residual* $(\mathsf{P}1, \mathsf{P}2)^{-1}\tau$ *is the set of* A*-lists, that is* $\mathsf{P}1^{-1}D$ *where* $D = dom(\tau_{flip})$*, while the domain of the residual* $(\mathsf{P}2, \mathsf{P}1)^{-1}\tau$ *is the set of all* B*-lists, that is* $\mathsf{P}2^{-1}D$*.*

This example illustrates that the residual of a pair $p = (u, v)$ aligned in state q by a DTopI $N = (M, D)$ may still depend on $u^{-1}dom(\llbracket N \rrbracket)$ as stated in Lemma 25, and not only on $\llbracket M \rrbracket_q$ and D as one might hope for. In a canonical DTopI this should not be the case.

Definition 29. *We call a* DTopI $N = (M, D)$ *compatible if* $D = dom(\llbracket N \rrbracket)$*, and if* $dom(p^{-1}\llbracket N \rrbracket)$ *coincides for all pairs* p *that are syntactically aligned in the same state of* N*.*

The notion of compatible DTopI is a semantic counterpart of the syntactic notion of *uniform* DTopI by Engelfriet, Maneth and Seidl [18]. It is not only much simpler but also more general: While the notion of uniform DTopI depends on the Dtta that defines the inspection domain and is thus restricted to top-down inspection, the notion of compatibility applies to general DTopI.

Each state of a compatible DTopI N indeed corresponds to an equivalence class of the syntactic equivalence $\equiv_N$, which is determined by the state to which the pairs in this equivalence class are aligned:

Lemma 30. *Let* N *be a compatible* DTopI*. If two pairs* p *and* p' *are syntactically aligned in the same state, then* $p \equiv_N p'$*.*

Proof. Let $p = (u, v)$ and $p' = (u', v')$ be both syntactically aligned in the same state of N, say q. Lemma 25 then shows that $p^{-1}\llbracket N \rrbracket = \llbracket M \rrbracket_{q|u^{-1}dom(\llbracket N \rrbracket)}$ and $p'^{-1}\llbracket N \rrbracket = \llbracket M \rrbracket_{q|u'^{-1}dom(\llbracket N \rrbracket)}$. By compatibility, the residuals of the domain are the same: $u^{-1}dom(\llbracket N \rrbracket) = u'^{-1}dom(\llbracket N \rrbracket)$. Therefore, the residuals of the transducer are the same: $p^{-1}\llbracket N \rrbracket = p'^{-1}\llbracket N \rrbracket$, that is $p \equiv_N p'$. □

We next show that any $\text{DTopI}_{\text{reg}}$ can be made compatible. The intuition is that a compatible transducer should check as many domain restrictions as possible by itself, rather than delegating this job to the domain inspection. In order to do so, it should run in parallel with its DTop some Dtta that tests membership to the path-closure of the inspection domain, i.e., to the least path-closed tree language subsuming the inspection domain

Example 31. *We reconsider the* DTop M_{13} *from Example 28 which defines* τ_{flip}*. When making* M_{13} *compatible, we will obtain the* DTopI (M_7, D) *where*

$D = dom(\tau_{flip})$. The single state q of M_{13} will be split into the 3 different states q_0, q_a and q_b of M_7. In order to see how this works, we consider the following top-down tree automaton A recognizing D:

$$p_0 \xrightarrow{P} (p_a, p_b) \qquad p_a \xrightarrow{A} (p_a) \qquad p_a \xrightarrow{\#} () \qquad p_b \xrightarrow{B} (p_b) \qquad p_b \xrightarrow{\#} ()$$

Note that this tree automaton is top-down deterministic, which simplifies the example a little bit. The general construction, however, can be done also be lifted to nondeterminstic top-down tree automata recognizing D. It should also be noticed that the result of the construction depends of which tree automaton was chosen.

The states of the compatible DTopI *that we obtain with* DTTA *A will be the pairs of a state of M_7 and a state of A, that is the pairs $q_0 = (q, p_0)$, $q_a = (q, p_a)$, and $q_b = (q, p_b)$. The transition rules will be obtained by pairing transitions of the* DTop *of M_7 and the tree automaton A in the obvious manner. Indeed, the resulting* DTopI *is (M_7, D), which is compatible.*

We next prove that any $\text{DTopI}_{\text{reg}}$ or DTopI_{td} can be made compatible.

Proposition 32. *There exists an algorithm that given a* DTop *M and a top-down tree automaton A computes in time $O(|M|\ 2^{|M|+|A|})$ a compatible* DTopI *equivalent to the* DTopI *$(M, [\![A]\!])$. Furthermore, if A was top-down deterministic then the resulting* DTopI *is a DTopI_{td}.*

Proof. Let $D = [\![A]\!]$, $N = (M, D)$ a $\text{DTopI}_{\text{reg}}$, and $D' = dom([\![N]\!])$ By Lemma 9, we can construct in time $O(|A|\ 2^{|M|})$ a top-down tree automaton A' that recognizes D'. Note that if A is top-down deterministic, then A' is as well. In the general case, however, A' may be nondeterministic. This is a problem since it may be impossible to run A deterministically in a top-down manner, so that no DTop may not be able to check membership to D' exactely. What a DTop may still do is to compute at any path the set of states that A' reaches, while ignoring the dependencies between the states of siblings.

The first idea is to build a DTop M' that runs M while computing the set of reachable states of A' in parallel. The states of M' are pairs (q, P) where q is a state of M and P is a subset of states of A'. The axiom of M' is obtained from the axiom of M by replacing $q\langle x_0\rangle$ by $(q, P_I)\langle x_0\rangle$ where P_I is the set of initial states of A. The rules $rhs'((q, P), f)$ are obtained from the rules $rhs(q, f)$ of M, by replacing any leaf of the form $q'\langle x_i\rangle$ for some q' by $(q', P')\langle x_i\rangle$, where $P' = \{p_i \mid p \in P,\ p \xrightarrow{f} (p_1, \ldots, p_i, \ldots, p_n)$ a rule of $A'\}$. It is not difficult to see that $N' = (M', D')$ is equivalent to N. We now argue that N' is compatible. We claim that if $u \sim_{(q,P)} v$ in M' then P is the set of states reached by A over u, starting at P_I in the axiom and progressing step by step in the rules. Hence, in this case we have $u^{-1}D' = u^{-1}[\![A']\!] = \cup_{p\in P}[\![A']\!]_p$ by Lemma 4 . This shows that the dependence of the residual $u^{-1}D'$ on u is limited to a dependence on P and thus on the state of M' to which (u, v) is aligned. So if also $u' \sim_{(q,P)} v'$ then ${u'}^{-1}D' = \cup_{p\in P}[\![A']\!]_p$ and thus $u^{-1}D' = {u'}^{-1}D'$ as required.

However, the construction of M' may require double-exponential time, since it requires exponential time in the size of A', which itself may be exponential in the size of M. We thus need to improve the construction. For this we note that $D' = [\![A]\!] \cap [\![A'']\!]$ where A'' is the tree automata that recognizes $dom([\![M]\!])$ from Lemma 9. We note that A'' is top-down deterministic and of size at most $O(2^{|M|})$. Unfortunately, we cannot always make A top-down deterministic. So, rather than computing states reachable by the intersection of A and A'', the second idea is sufficient to compute the unique state reached by A'' and the subset of states reached by A. We thus construct a DTOPI M'' that runs M in parallel with computing the reachable states of A and the state reached by A''. The states of M'' are thus triples (q, p'', P) where q is a state of M, p'' is a state of A'' and P a subset of states of A. The construction of M'' can be done similarly to before but now in time $O(|M|\ |A''|\ 2^{|A|})$. Clearly the DTOPI (M'', D') is equivalent to (M', D'), and it is not difficult to see that it is compatible too. □

Corollary 33. *Any* DTOPI *is equivalent to some trimmed compatible* DTOPI.

Proof. This follows from Proposition 32, since for any compatible DTOPI N, the DTOPI $trim(N)$ is compatible too, and trivially trimmed. □

7 Earliest Transducers

We introduce earliest transducers in order to normalize the output production and thereby to find some kind of unique minimal transducers for a given transformation. The idea is to produce the output as early as possible, as first proposed for subsequential transducers by Choffrut [9, 10] and extended to any DTOPI_{td} by Engelfriet, Maneth and Seidl [18]. Our approach is yet more general in that it applies to any $\text{DTOPI}_{\text{reg}}$, i.e. we capture regular domain inspection in addition. This generalization requires a more flexible notion of earliest DTOPIs, that is independent from the notion of uniform transducers. We do so by considering the output production of aligned pairs (here the inspection domain intervenes), and not only the production of the state to which the pair is aligned.

We have to define what it means for a DTOPI to maximize its output production. The definition will be based on the notion of largest common tree prefixes. For two trees $t, t' \in \mathcal{T}_G$ we define their *largest common prefix tree* $t \sqcap t' \in \mathcal{T}_G(\{\bot\})$ as follows:

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

The $\sqcap$ operator is associative and commutative, so that it can be easily lifted to finite sets of trees $D = \{t_1, \ldots, t_n\}$, by defining $\bigsqcap D = t_1 \sqcap t_2 \sqcap \cdots \sqcap t_n$ independently of the ordering of the trees in D.

Let τ be a partial function and u a input path u in $\mathit{paths}(\mathit{dom}(\tau))$. We define τ*'s maximal output at* u as:

$$\mathit{out}_\tau(u) = \bigsqcap\{\tau(s) \mid s \models u, s \in \mathit{dom}(\tau)\}$$

For any partial function $\tau \neq \emptyset$, we call $\mathit{out}_\tau(\varepsilon)$ the "global common prefix" of the range of τ. Similarly, we define the maximal output at a npath uf by $\mathit{out}_\tau(uf) = \bigsqcap\{\tau(s) \mid s \models uf, s \in \mathit{dom}(\tau)\}$. Note that $\mathit{out}_\tau(u)$ is undefined if there does not exist any tree $s \in \mathit{dom}(\tau)$ such that $s \models u$.

Example 34. *For τ_{flip}, $\mathit{out}_{\tau_{\mathit{flip}}}(\varepsilon) = f(\bot, \bot)$, as every tree in the range of τ_{flip} has the form $f(s_1, s_2)$ for some input trees s_1 and s_2. For the input path $u = f1a1a$, we have $\mathit{out}_{\tau_{\mathit{flip}}}(u) = f(\bot, a(a(\bot)))$, since all inputs having this path u must be of the form $f(a(a(s_1), s_2)$ for some input trees s_1 and s_2.*

We now consider *earliest* transducers, that always produce output constructors as soon as possible.

Definition 35. *A* DTOPI $N = (M, D)$ *is* earliest *if for any pair p that is syntactically aligned by M, the residual $p^{-1}[\![N]\!]$ satisfies $\mathit{out}_{p^{-1}[\![N]\!]}(\varepsilon) = \bot$.*

Example 36. *We reconsider the transducers defining τ_{flip}. It can be defined by the* DTOP M_8 *which is earliest, and by the* DTOP M_7 *which is not. In order to see the later, note that M_7 aligns the pair of paths $p = (\varepsilon, \varepsilon)$, while $\mathit{out}_{p^{-1}\tau_{\mathit{flip}}}(\varepsilon) = \mathit{out}_{\tau_{\mathit{flip}}}(\varepsilon) = \mathsf{P}(\bot, \bot)$. This shows that M_7 does not output P at the root as soon as possible. For similar reasons, the* DTOPI $N_{13} = (M_{13}, \mathit{dom}(\tau_{\mathit{flip}}))$ *is not earliest. The* DTOP M_{13}*, however, is earliest, since the range of $[\![M_{13}]\!]$ contains $\#$, so that any early output of $\mathsf{P}(\ldots, \ldots)$ would not be correct.*

Note that without domain inspection, earliest DTOPs are less expressive than DTOPs in general. The next example shows that domain inspection is needed in order to make some DTOPs earliest.

Example 37. *The identity function with domain $\{f(c, a),\ f(c, b)\}$ can be computed by some* DTOP*. However, if we want this* DTOP *to be earliest, then its axiom must produce $f(c, \bot)$ right away. It remains to represent the residual at the pair $(\varepsilon, f2)$, which is the partial function $\{(f(c, a), a), (f(c, b), b)\}$. This residual can be recognized by some* DTOPI*, as shown in Example 10, but not by any* DTOP *without inspection. Therefore, the above partial identify function is not definable by any earliest* DTOP *without inspection, i.e., domain inspection may be required for making* DTOP*s earliest.*

Our aim is to restate Theorem 11 of [18], that shows that every DTOPI_{td} is equivalent to some earliest DTOPI_{td}[1], but also extend it to the more general $\text{DTOPI}_{\text{reg}}$ case.

[1] This may take doubly exponential time in the worst case, but only quadratic time if the given transducer is total.

In order to do so, we start with a lemma that shows that the axioms and transition rules of earliest DTOPIs have a specific form depending on the largest common outputs of the transformation and its residuals at syntactically aligned paths.

Lemma 38. *Let $N = (M, D)$ be an earliest* DTOPI*, with $\tau = [\![N]\!]$, and $M = (Q, F, G, \{ax\}, rhs)$. Then:*

(1) if $\tau \neq \emptyset$, $out_\tau(\varepsilon) = ax[q\langle x_0\rangle/\bot \mid q \in Q]$

(2) for every (q, f) such that $rhs(q, f)$ is defined and any pair p syntactically aligned in q, we have $out_{p^{-1}\tau}(f) = rhs(q, f)[q\langle x_i\rangle/\bot \mid q \in Q, x_i \in X]$

Proof. The proof basically relies on the definitions of syntactic alignment and earliestness, and Proposition 19. It should be noticed that both statements would go wrong without assuming trimmedness. Note that useless rules in M may have any form without preventing N from being earliest.

(1) For any $s \in dom(\tau)$, we have $\tau(s) = ax[q\langle x_0\rangle/[\![M]\!]_q(s) \mid q \in Q]$. This means that for all v such that $ax \models v$, for all $s \in dom(\tau)$, $\tau(s) \models v$. Hence, $out_\tau(\varepsilon) \models v$. This would also be true for a npath vf, for $f \in F$. For v such that $v^{-1}ax = q\langle x_0\rangle$, then $\varepsilon \sim_q v$. Since N is earliest, $out_{(\varepsilon,v)^{-1}\tau}(\varepsilon) = \bot$. Hence, $v^{-1}out_\tau(\varepsilon) = \bot$.

(2) Let $p = (u, v)$ syntactically aligned in q, and a tree $s \in dom(\tau)$ such that $s \models u$. Proposition 19 gives that $v^{-1}\tau(s) = [\![M]\!]_q(u^{-1}s)$. If $s \models ufi$, i.e. $u^{-1}s = f(s_1 \dots s_k)$, then by definition of $[\![M]\!]_q$, $[\![M]\!]_q(f(s_1, ..., s_n)) = rhs(q, f)[q'\langle x_i\rangle/[\![M]\!]_{q'}(s_i) \mid q' \in Q, x_i \in X]$. This means that if for all paths v' such that $rhs(q, f) \models v'$, for all $s' = f(s_1, ..., s_n) \in u^{-1}dom([\![M]\!]_q)$, then $[\![M]\!]_q(s') \models v'$. From Lemma 25, we know that for all $s' \in dom(p^{-1}\tau)$, $p^{-1}\tau(s') = [\![M]\!]_q(s)$. Hence, $out_{p^{-1}\tau}(f) \models v'$. This would also be true for a npath $v'f$, for $f \in F$. Furthermore, if $v'^{-1}rhs(q, f) = q'\langle x_i\rangle$, then (ufi, vv') is syntactically aligned in state q'. Since N is earliest, $out_{(ufi,vv')^{-1}\tau}(\varepsilon) = \bot$. That is to say, $v'^{-1}out_{(u,v)^{-1}\tau}(fi) = \bot$. Hence, if $v'^{-1}rhs(q, f) = q'\langle x_i\rangle$, $v'^{-1}out_{p^{-1}\tau}(f) = \bot$.

□

We next show that any $\text{DTOPI}_{\text{reg}}$ can be made in an equivalent earliest trimmed compatible $\text{DTOPI}_{\text{reg}}$. This result is an extension of what can be found in [18], which demonstrated a similar result for the particular case of DTOPI_{td}.

Proposition 39. *Every* DTOPI_{reg} $N = (M, D)$ *is equivalent to some compatible earliest* DTOPI_{reg} $N' = (M', D')$*. Every* DTOPI_{td} $N = (M, D)$ *is equivalent to some compatible earliest* DTOPI_{td} $N' = (M', D')$*.*

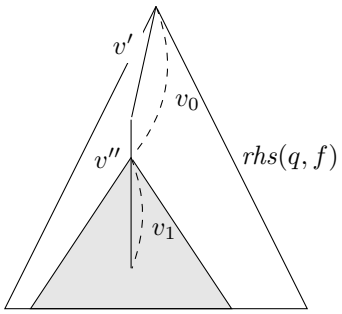


Figure 5: Updating the advance of a state $[q, v']$ after reading f

Proof. Let $N = (M, D)$ where $\tau = [\![N]\!]$ and $M = (Q, F, G, \{ax\}, rhs)$. Corollary 33 tells us that we can suppose w.l.o.g that N is a trimmed compatible transducer. For any state q of N we define the transformation $[\![N]\!]_q$ by $[\![N]\!]_q = [\![M]\!]_{q|u^{-1}\,dom([\![N]\!])}$ where (u, v) is some pair aligned in q. Such a pair p exists for all q since N is trimmed. Which pair p aligned in q is chosen does not matter since N is compatible.

We prove both cases by the same construction of $N' = (Q', F, G, \{ax'\}, rhs')$ from N. We define the inspection domain of N' by $D' = dom(\tau)$. Proposition 14 shows that, (1) D' is regular if D was, and (2) that D' is DTTA-recognizable if D was. In order to prove the proposition, it is thus sufficient to construct a DTOP M' such that $N' = (M', D')$ is compatible, earliest, and $[\![N']\!] = \tau$.

The idea behind the construction of M' is to produce states of M that produces their output "in advance". If state q of M is not earliest (i.e. if $out_{[\![N]\!]_q}(\varepsilon)$ is not $\bot$), we want to create states $[q, v']$ where $v^{-1}out_{[\![N]\!]_q}(\varepsilon) = \bot$, such that if (u, v) are aligned in q for N, (u, vv') are aligned in $[q, v']$ in N'.

Since N' must be earliest, the axiom and rules of M' must have a special form as stated by Lemma 38. In particular,

$$ax' = out_{[\![N]\!]}(\varepsilon)\Phi$$

for some substitution Φ that maps F-paths leading to $\bot$-leafs to $Q' \times \{x_0\}$. To know how to replace a $\bot$-leaf under path v, we say that if $v^{-1}out_\tau(\varepsilon) = \bot$, then there exists v_0, v_1 such that $v = v_0v_1$, and $v_0^{-1}ax = q\langle x_0\rangle$ for some state $q \in Q$. It is easy to see that $v_1^{-1}[\![N]\!]_q = (v_0v_1)^{-1}\tau = \bot$. Then in M', we choose $v^{-1}ax' = [q, v_1]\langle x_0\rangle$.

Motivated by Lemma 38, if $rhs(q, f)$ exists, we define the rule $rhs'([q, v'], f)$ of M' as follows:

$$rhs'([q, v'], f) = {v'}^{-1}out_{[\![N]\!]_q}(f)\Phi$$

for some substitution Φ that maps F-paths leading to $\bot$-leafs to $Q' \times X$. For a path v'' such that $(v'v'')^{-1}out_{[\![N]\!]_q}(f) = \bot$, we want to know what state to call under v'' in $rhs'([q, v'], f)$. We say that in $rhs(q, f)$, there is a path v_0 such that v_0 is a prefix of $v'v''$, and $v_0^{-1}rhs(q, f) = q'\langle x_i\rangle$ (see Figure 7). When

the earliest transducer M' produces v'' on top of its advance of v', the original transducer M only produces v_0 in rule $rhs(q, f)$. This leaves an advance of v_1, which means $v_1^{-1} out_{[\![N]\!]_{q'}}(\varepsilon) = \bot$. Therefore, the new advance is $v_1 = v_0^{-1} v' v''$. We then have $v''^{-1} rhs'([q, v'], f) = [q', v_1]\langle x_i \rangle$.

For the correctness of the construction, we prove that we indeed constructed a transducer that produces its output "ahead" of N. We will prove by induction that $[\![N']\!]_{[q,v']} = v'^{-1}[\![N]\!]_q$.

For a tree $s = f(s_1, ..., s_n)$, we will show that for all output paths v''', if we have $[\![N']\!]_{[q,v']}(s) \models v'''$, then $[\![N]\!]_q(s) \models v'v'''$. We differentiate two cases.

If $rhs'([q, v'], f) \models v'''$, then $v'^{-1} out_{[\![N]\!]_q}(f) \models v'''$, which means $out_{[\![N]\!]_q}(f) \models v'v'''$. By definition of out, this implies $[\![N]\!]_q(s) \models v'v'''$.

If $rhs'([q, v'], f) \not\models v'''$, there are paths $v'', v_{[q',v_1]}$ such that $v''' = v''v_{[q',v_1]}$ and $v''^{-1} rhs'([q, v'], f) = [q', v_1]\langle x_i \rangle$. This means that $[\![N']\!]_{[q',v_1]}(s_i) \models v_{[q',v_1]}$. We use the same notations as above and in Figure 7: there exists v_0 a prefix of $v'v''$, such that $v_0^{-1} rhs(q, f) = q'\langle x_i \rangle$, and $v_0 v_1 = v'v''$. By induction hypothesis, we have $[\![N']\!]_{[q',v_1]}(s_i) = v_1^{-1}[\![N]\!]_{q'}(s_i)$. Since $[\![N']\!]_{[q',v_1]}(s_i) \models v_{[q',v_1]}$, we have $[\![N]\!]_{q'}(s_i) \models v_1 v_{[q',v_1]}$. Since $v_0^{-1} rhs(q, f) = q'\langle x_i \rangle$, and from the definition of $[\![N]\!]_q$, we have $[\![N]\!]_q(s) \models v_0 v_1 v_{[q',v_1]}$. Since $v_0 v_1 = v'v''$ and $v''' = v''v_{[q',v_1]}$, we have $[\![N]\!]_q(s) \models v'v''v_{[q',v_1]}$, and thus $[\![N]\!]_q(s)v'v'''$.

Note that this also proves that N' is earliest: for the state $[q, v']$, we have that $v'^{-1} out_{[\![N]\!]_q}(\varepsilon) = \bot$. Since $[\![N]\!]'_{[q,v']} = v'^{-1}[\![N]\!]_q$, we have that $out_{[\![N']\!]_{[q,v']}}(\varepsilon) = v'^{-1} out_{[\![N]\!]_q}(\varepsilon) = \bot$. This is true for all states of M'. This means that N' is compatible: if two pairs p, p' are syntactically aligned in $[q, v']$ have the same residual $v'^{-1}[\![N]\!]_q$. Furthermore, N' is earliest: if a pair p is syntactically aligned in $[q, v']$, its residual is $p^{-1}\tau = v'^{-1}[\![N]\!]_q$. Since $v'^{-1} out_{[\![N]\!]_q}(\varepsilon) = \bot$, we have $out_{p^{-1}\tau}(\varepsilon) = \bot$. □

In contrast to Proposition 39, there exists DTopI with inspection by path-closed domains, that cannot be made earliest. Indeed, if the domain is path-closed but not regular, the finiteness statement from Corollary 27 may not hold. This can be seen in the following counter-example.

Example 40. *We consider the partial identity function with the path-closed nonregular domain* $D = \{a(a(a(...(\#)))) \mid 2^n$ *symbols* $a, n \geq 0\}$. *This partial function is definable by some* DTopI (M, D) *where* $[\![M]\!]$ *is the total identity function. However, it cannot be defined by any earliest* DTopI *with inspection by some path-closed domain. Indeed, suppose that such an earliest transducer reads the* $2^k + 1$*'th symbols a for some k. It then has to produce* 2^k *symbols a at once. But no* DTop *can do this for all k, since it would need a different state for all k, of which there are infinitely many.*

Note that earliest and compatibility are not enough to obtain a normal form on $\text{DTopI}_{\text{reg}}$: the earliest transducer constructed in the proof of Proposition 39 depends heavily on initial choice of a DTopI defining the transformation.

Example 41. *We consider the partial function which maps maps* $f(a, a)$ *to a and* $f(b, b)$ *to b. This partial function can be defined by two different earliest*

DTOPI_{reg} *by domain* $D = \{f(a,a), f(b,b)\}$*, which is not path-closed so that it is not definable by any* DTTA*. The first* DTOP *outputs the subtree at path* $f1$ *and the second* DTOP *outputs the subtree at path* $f2$*. For the first transducer, the pair* $(f1, \varepsilon)$ *is syntactically aligned but not the pair* $(f2, \varepsilon)$*, while it is the converse for the second transducer.*

This example shows that the same transformation can be defined by two different earliest compatible DTOPIs with the same regular inspection domain, so that the same output is produced from two different input paths. In this case, the syntactically aligned pairs differ for these two earliest DTOPIs. As we will see later on, this problem cannot appear for earliest DTOPI_{td}. For this reason, the normal form and learning algorithm that we will develop are restricted to the class DTOPI_{td}.

8 Semantic Equivalence

We introduce a semantic notion of aligned paths that applies to transformations rather than transducers. The intuition of this semantic alignment is that a pair is semantically aligned if it is susceptible to be a syntactically aligned pair in an earliest transducer. This leads us to a semantic equivalence relation $\equiv_{[\![N]\!]}$ which depends only on the transformation and not on the transducer.

We will show for any DTOPI_{td} that semantic and syntactic alignments are identical. This will leads us to a Myhill-Nerode type Theorem in the more restricted case of top-down inspection.

8.1 Semantic Alignments

We introduce a notion of semantically aligned pairs. We make this notion to identify potential candidates for being syntactically aligned pairs in an earliest DTOPI. In essence, if $p = (u, v)$ is to be a syntactically aligned pair, it should at have a functional residual, and for it to be a syntactically aligned pair in an earliest transducer, it should additionally verify that v is as much of the output as one can guess from reading u in the input.

Definition 42. *A pair* $p = (u, v)$ *is said to be* (semantically) aligned for a partial function τ *if the residual* $p^{-1}\tau$ *is a partial function, and* $v^{-1}out_\tau(u) = \bot$*.*

We now prove a useful equivalence, to define semantically aligned pairs in another equivalent way.

Lemma 43. *For any pair* $p = (u, v)$ *and transformation* τ*:*

$$v^{-1}out_\tau(u) = \bot \text{ if and only if } out_\tau(u) \models v \text{ and } out_{p^{-1}\tau}(\varepsilon) = \bot.$$

Proof. We first prove the implication from the left to the right. For this we assume $v^{-1}out_\tau(u) = \bot$. Then clearly, $out_\tau(u) \models v$. Furthermore, there must exist two trees $s_1, s_2 \in dom(\tau)$ such that $s_1 \models u$, $s_1 \models u$, and $v^{-1}\tau(s_1) \sqcap v^{-1}\tau(s_2) = \bot$. By definition, $p^{-1}\tau$ contains the pairs $(u^{-1}s_1, v^{-1}\tau(s_1))$ and

$(u^{-1}s_2, v^{-1}\tau(s_2))$. This means that $out_{p^{-1}\tau}(\varepsilon) \leqslant v^{-1}\tau(s_1) \sqcap v^{-1}\tau(s_2)$, and thus $out_{p^{-1}\tau}(\varepsilon) = \bot$.

We next prove the inverse implication. We assume $out_\tau(u) \models v$ and $out_{p^{-1}\tau}(\varepsilon) = \bot$. There must exist two trees $s'_1, s'_2 \in u^{-1}dom(\tau)$, such that $(p^{-1}\tau)(s'_1) \sqcap (p^{-1}\tau)(s'_2) = \bot$. By definition, this means that there exists two trees $s_1, s_2 \in dom(\tau)$ such that $u^{-1}s_1 = s'_1$, $u^{-1}s_2 = s'_2$, and $v^{-1}\tau(s_1) = (p^{-1}\tau)(s'_1)$, $v^{-1}\tau(s_2) = (p^{-1}\tau)(s'_2)$. This means that $\tau(s_1) \sqcap \tau(s_2) \models v\bot$. Hence, $out_\tau(u) \leqslant v\bot$. However, since we assumed $out_\tau(u) \models v$, we have $out_\tau(u) \models v\bot$, and thus, $v^{-1}out_\tau(u) = \bot$. □

While the definitions of $v^{-1}out_\tau(u)$ and $out_{p^{-1}\tau}(\varepsilon)$ seem similar, they are not equivalent without the supposition that $out_\tau(u) \models v$. The following example show that the inverse of the Lemma 43 would not hold without assuming so.

Example 44. *Let $F = G$ and τ be the identity transformation on $\mathcal{T}_F$, i.e. $\tau(s) = s$ for all $s \in \mathcal{T}_F$. All pair (u, u) are semantically aligned and have same residual which is τ. We next consider pairs $p = (u, v)$ where $u = \varepsilon$ and $v \neq \varepsilon$. First note that $out_\tau(u) = out_\tau(\varepsilon) = \bot$. Hence, $v^{-1}out_\tau(u)$ is undefined since we assumed $v \neq \varepsilon$. However, $out_\tau(u) = \bot$, so that $out_\tau \not\models v$, i.e. p is not semantically aligned. Nevertheless, the residual $p^{-1}\tau$ is the partial function which maps all the trees $s \in \mathcal{T}_F$ that satisfy $s \models v$ to their subtree $v^{-1}s$. The image of this partial function is the set $\mathcal{T}_F$, so that $out_{p^{-1}\tau}(\varepsilon) = \bot$. This shows that the inverse of the Lemma 43 would not hold without assuming $out_\tau(u) = \bot$.*

Note that Lemma 43 implies for all pairs p semantically aligned for τ that their residual is nonempty, since $out_{p^{-1}\tau}(\varepsilon) = \bot$.

Lemma 45. *For any earliest* DTOPI *N, any two paths that are syntactically aligned in some state of N are semantically aligned for $[\![N]\!]$.*

Proof. Let p be a pair of paths that are syntactically aligned in some state of N. Since p is syntactically aligned, Lemma 25 shows that $p^{-1}[\![N]\!]$ is a partial function, and that $out[\![N]\!](u) \models v$. Thus Lemma 43 yields $v^{-1}out_{[\![N]\!]}(u) = \bot$. Furthermore, Since N is earliest, it follows that $out_{p^{-1}[\![N]\!]}(\varepsilon) = \bot$. Hence, p is semantically aligned for $[\![N]\!]$. □

In the case of $\text{DTOPI}_{\text{reg}}$ in general, however, not all semantically aligned pairs of a transformation τ are realized into syntactically aligned pairs by an earliest DTOPI computing τ.

Example 46. *We reconsider the partial function from Example 41, i.e. the function $[f(a, a)/a,\ f(b, b)/b]$. It semantically aligns the pairs $(\varepsilon, \varepsilon)$, $(f1, \varepsilon)$, and $(f2, \varepsilon)$. Two earliest* DTOPI*s with regular inspection defining this partial function were given in Example 41. The first produces the output at the input path $f1$ so the paths $(\varepsilon, \varepsilon)$ and $(f1, \varepsilon)$ are syntactically aligned, but not $(f2, \varepsilon)$. The second* DTOPI *produces its output at path $f2$. It aligns $(\varepsilon, \varepsilon)$ and $(f2, \varepsilon)$ syntactically, but not $(f1, \varepsilon)$. This shows that not all semantic alignments need to be realized syntactically by all* DTOPI*s with regular inspection.*

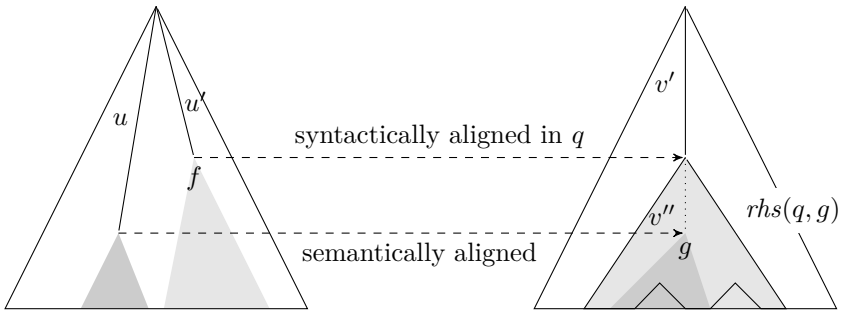


Figure 6: Input path u' that produces the node at output path $v = v'v''$.

8.2 Semantic Equivalence

For any transformation τ, we define an equivalence relation $\equiv_\tau$ between pairs p_1 and p_2 of paths that are semantically aligned by τ, as follows:

$$p_1 \equiv_\tau p_2 \text{ iff } p_1^{-1}\tau = p_2^{-1}\tau.$$

Lemma 47. *For N an earliest* DTOPI *and p_1, p_2 two pairs syntactically aligned in some state of N, syntactic equivalence $p_1 \equiv_N p_2$ implies semantic equivalence $p_1 \equiv_{[\![N]\!]} p_2$.*

Proof. If $p_1 \equiv_N p_2$ then p_1 and p_2 are syntactically aligned, so they are also semantically aligned by Lemma 45, since N is earliest. Furthermore, syntactic equivalence requires $p_1^{-1}[\![N]\!] = p_2^{-1}[\![N]\!]$, so that semantic equivalence follows. □

We now endeavour to obtain a Myhill-Nerode type Theorem for the DTOPIs with top-down inspection, that is for the class of DTOPI_{td}. **Most of the results that will follow would fail for more general regular inspection**.

We wish to prove that the semantic equivalence $\equiv_{[\![N]\!]}$ has finite index, and know from Corollary 27 that the syntactic equivalence $\equiv_N$ has finite index. Therefore, we will show that the classes of syntactic and semantic equivalence classes coincide for any earliest DTOPI_{td}.

Theorem 48. *For any earliest* DTOPI_{td} *N, every semantically aligned pair p is syntactically aligned.*

Proof. Suppose that $p = (u, v)$ is a semantically aligned pair that is not syntactically aligned. We now consider a tree s such that $s \models u$. We consider the syntactic alignment that produce the node under v when the transducer N computes $[\![N]\!](s)$, as described by Proposition 22. This is the alignment $p' = (u', v')$ where the node under v is not produced yet, but will be after reading the node under u' in s. Formally, this means that p' is syntactically aligned in state q, such that $s \models u'f$ for some $f \in F$, $[\![N]\!](s) \models v'g$ for some $g \in G$, v' is a prefix of v such that $v = v'v''$, and $rhs(q, f) \models v''g$.

Since N is earliest, and p' is syntactically aligned, it follows that p' is semantically aligned by Lemma 45, and thus we have that $v'^{-1} out_\tau(u') = \bot$. Since

p is semantically aligned, we have that $v^{-1}out_\tau(u) = \bot$. We will prove this situation to be impossible by distinguishing 4 cases: $u = u'$, u is a prefix of u', u' is a prefix of u, or the last possible case: u and u' are disjoint.

u equals u' Assume that $u' = u$. We have $v^{-1}out_\tau(u) = \bot$, and $v'^{-1}out_\tau(u) = \bot$. Since v' is a prefix of v, this implies that $v' = v$. Hence, $p = p'$. This means that p is syntactically aligned, which is in contradiction with our assumption.

u is a prefix of u' Suppose that u is a strict prefix of u'. Then by the recursive definition of syntactic alignments, there exists a syntactically aligned pair (u, v'') for some prefix v'' of v', and therefore, of v. Since we supposed N earliest, (u, v'') is semantically aligned, which means $v''^{-1}out_\tau(u) = \bot$. Since (u, v) is semantically aligned, we also have $v^{-1}out_\tau(u) = \bot$. This means that $v'' = v$, and thus that p is syntactically aligned, which is in contradiction with our assumption.

u' is a prefix of u Suppose that u' is a strict prefix of u. Hence $u'f$ is a prefix of u too. From the assumption that $rhs(q, f) \models v''g$, we have that $out_\tau(u'f) \models v'v''g$. Since $u'f$ is a prefix of u, we have $out_\tau(u) \models vg$. This means that $v^{-1}out_\tau(u) \neq \bot$, which is in contradiction with the assumption that p is semantically aligned.

u and u' are disjoint This case leads to a contradiction to the path-closedness of the domain. Since $dom(\llbracket N \rrbracket)$ is recognized by a DTTA, it is path-closed. This means that we can change s by replacing $u^{-1}s$ by any tree $s' \in u^{-1}dom(\llbracket N \rrbracket)$ while staying in the domain. However, since $p'^{-1}\llbracket N \rrbracket$ is functional, and $u'^{-1}s$ did not change, we have that $v'^{-1}\llbracket N \rrbracket(s[u/s']) = v'^{-1}\llbracket N \rrbracket(s)$. Notably, $v^{-1}\llbracket N \rrbracket(s[u/s']) = v^{-1}\llbracket N \rrbracket(s)$. Hence, $p^{-1}\llbracket N \rrbracket(u^{-1}s) = p^{-1}\llbracket N \rrbracket(s')$. Since this is true for all $s' \in u^{-1}dom(\llbracket N \rrbracket)$, we have that $p^{-1}\llbracket N \rrbracket$ is constant. However, since p is a semantically aligned, this is a contradiction, as it would prevent $out_{p^{-1}\llbracket N \rrbracket}(\varepsilon) = \bot$.

Since all cases are impossible, our assumption is impossible. Hence, there is no semantic aligned pair p that is not syntactically aligned. □

This theorem leads us directly to our desired Myhill-Nerode type Theorem for semantically aligned pairs:

Corollary 49. *For any* DTOPI$_{td}$ N*, the number of equivalence classes of the semantic equivalence relation* $\equiv_{\llbracket N \rrbracket}$ *is finite.*

Proof. This is an immediate consequence of Theorem 48 on the coincidence of syntactic and semantic alignments for earliest DTOPI$_{\text{td}}$, and the fact that the number of equivalence classes for syntactic aligned pairs is finite, as stated in Corollary 27. □

8.3 Semantic Successors

In analogy to Definition 17 of syntactic successors, we now define a notion of semantic successors. However, as shown in Example 46, there may be several semantic successors for a single triple (p, f, v'):

Definition 50. *Let $N = (M, D)$ be a* DTOPI *and $\tau = [\![N]\!]$. For any semantically aligned pair $p = (u, v)$ of τ, input symbol $f \in F$ and output path v' with $v'^{-1} out_{p^{-1}\tau}(f) = \bot$, we define the sets of semantic indexes and semantic successors as follows:*

- $Ind_\tau(p, f, v') = \{i \mid (ufi, vv') \text{ semantically aligned}\}$,
- $Succ_\tau(p, f, v') = \{(ufi, vv') \mid i \in Ind_\tau(p, f, v')\}$.

Furthermore, we define $Succ_\tau(p)$ as the set of all semantic successors of p with respect to τ and some f and v', that is:

$$Succ_\tau(p) = \{p' \in Succ_\tau(p, f, v') \mid f \in F,\ v'^{-1} out_{p^{-1}\tau}(f) = \bot\}$$

Proposition 48 give valuable informations on semantically aligned pairs: since they are exactly the syntactically aligned pairs of an equivalent earliest compatible transducer, all the properties of syntactically aligned pairs can be lifted to semantically aligned pair. The following corollary shows a useful property that transducers with top-down inspection share, that allows for a characterization of their normal form, and later, their learning algorithm.

Corollary 51. *For any earliest* DTOPI_{td} *N, if (ufi, v) is a semantically aligned pair of $[\![N]\!]$, then there is no index j different from i such that (ufj, v) is a semantically aligned pair of $[\![N]\!]$.*

Proof. Let ufi and ufj be disjoint input paths. If both (ufi, v) and (ufj, v) were semantically aligned, then for all s such that $s \models uf$, $v^{-1}[\![N]\!](s)$ depends functionally of both $ufi^{-1}s$ and $ufj^{-1}s$. As seen in the proof of Proposition 48, this leads to a contradiction. □

We can thus equate syntactic and semantic successors.

Lemma 52. *For any earliest* DTOPI_{td} *N defining $\tau = [\![N]\!]$, any semantically aligned pair $p = (u, v)$ of τ, input symbol $f \in F$, and output path v' such that $v'^{-1} out_{p^{-1}\tau}(f) = \bot$.*

- $Ind_\tau(p, f, v') = \{ind_N(p, f, v')\}$
- $Succ_\tau(p, f, v') = \{succ_N(p, f, v')\}$

Proof. Since p is semantically aligned, Proposition 48 ensures that it is also syntactically aligned in some state q of N. Lemma 38 then gives us that $out_{p^{-1}[\![N]\!]}(f) = rhs(q, f)[q\langle x_i\rangle/\bot \mid q \in Q, x_i \in X]$. This means that $v'^{-1} out_{p^{-1}\tau}(f) = \bot$ if and only it $v'^{-1} rhs(q, f) = q'\langle x_i\rangle$ for some state q'. This in turns means that

$(ufi, vv') \in Succ_\tau(p, f, v')$ if and only if $(ufi, vv') = succ_N(p, f, v')$. Since Corollary 51 indicates that i is the only index such that $(ufi, vv') \in Succ_\tau(p, f, v')$, we also get that $i \in Ind_\tau(p, f, v')$ if and only if $i = ind_N(p, f, v')$. □

In the case of top-down inspection, we thus have *unique* semantic indexes and successors that we can denote by $ind_\tau(p, f, v')$ and $succ_\tau(p, f, v')$.

Another important consequence of top-down inspection is that $Succ_\tau$ can be obtained from $Succ_N$, so that $Succ_\tau$ inherits the inductive nature of $Succ_N$.

Lemma 53. *Let τ be a* DTOPI$_{td}$ *transformation, $p = (u, v)$ be a semantically aligned pair of τ. Then either $u = \varepsilon$, or there exists a semantically aligned pair p' of τ such that $p \in Succ_\tau(p')$.*

Proof. Let N an earliest DTOPI$_{\text{td}}$ such that $[\![N]\!] = \tau$. Proposition 48 implies that since p is a semantically aligned pair, then there is a state q in N such that $u \sim_q v$. The recursive nature of syntactically aligned pairs presented in Definition 15 implies that if $u = u'fi$, then there exists v', v'' such that $v = v'v''$, $u' \sim_{q'} v'$, and $v''^{-1} rhs(q', f) = q\langle x_i \rangle$. By applying Proposition 48 again we get that p' is semantically aligned, and p is a successor of p'. □

The combination of Proposition 48 and 22 has an interesting consequence: the image of any output in a tree of $\tau(s)$ can be semantically linked to a unique input path, i.e. any output node has a unique top-down origin.

Proposition 54. *Let τ be a* DTOPI$_{td}$ *transformation and $s \in dom(\tau)$ an input tree. Then for any output path vg such that $\tau(s) \models vg$, either $out_\tau(\varepsilon) \models vg$ or there exist a unique decomposition $v = v'v''$, an input path $u'f$, such that:*

- *$s \models u'f$,*
- *(u', v') semantically aligned pair, and*
- *$out_{p'^{-1}\tau}(f) \models v''g$.*

Proof. Let N be an earliest DTOPI$_{\text{td}}$ such that $[\![N]\!] = \tau$. Proposition 54 ensures that for any output path vg such that $\tau(s) \models vg$, either $ax \models vg$ or there exist a unique decomposition $v = v'v''$, an input path $u'f$, such that:

- $s \models u'f$,
- (u', v') is syntactically aligned in some state q of N, and
- $rhs(q, f) \models v''g$.

Since N is earliest, Lemma 38 ensures that $out_\tau(\varepsilon) = ax$. This means that $out_\tau(\varepsilon) \models vg$ if and only if $ax \models vg$. Otherwise, Proposition 48 ensures that (u', v') is semantically aligned. Finally, since N is earliest, Lemma 38 ensures that $out_{p'^{-1}\tau}(f) \models vg$ if and only if $rhs(q, f) \models vg$.

The uniqueness of such a pair is also ensured by Proposition 48: Suppose that there is another pair $p'' = (u'', v'')$ such that:

- $s \models u'f$,
- (u', v') semantically aligned pair, and
- $out_{p'^{-1}\tau}(f) \models v''g$.

Then by Proposition 48, (u'', v'') is syntactically aligned in some state q'. Since N is earliest, Lemma 38 ensures that $out_{p''^{-1}\tau}(f) \models vg$ if and only if $rhs(q', f) \models vg$. This means that p'' fits the criteria of Proposition 54, which is a contradiction, as we know p' is the unique pair to fit those criteria. □

9 Unique Normal Forms

The finiteness index Theorem 49 for the semantic equivalence relation allows us to define a normal form for any transformation definable by some DTOPI$_{\text{td}}$. Up to having taken a semantic approach to top-down inspection, this normal form coincides with the one from [18]. But in contrast to there, it will here be obtained from a Myhill-Nerode type characterization. We will then show that the normal form of a transformation τ definable by a DTOPI$_{\text{td}}$ is indeed the minimal earliest trimmed compatible DTOPI$_{\text{td}}$ defining τ, and that this normal form is unique up to state renaming.

Given a transformation τ definable by some DTOPI$_{\text{td}}$, the equivalence relation $\equiv_\tau$ has only a finite number of classes. Therefore, we can define the normal DTOPI$_{\text{td}}$ $can(\tau)$ as follows. Note that in this part, we will denote the equivalence class of a pair p in $\equiv_\tau$ by $[p]_\tau$.

Definition 55. *Let τ a transformation definable by some* DTOPI$_{td}$*. We the define its canonical* DTOPI$_{td}$ *$can(\tau)$ as $N = (M, dom(\tau))$, where $M = (Q, F, G, ax, rhs)$ such that:*

- *Q is the set of classes $[p]_\tau$ of τ such that $p^{-1}\tau \neq \emptyset$.*
- *$ax = out_\tau(\varepsilon)[v/[(\varepsilon, v)]_\tau\langle x_0\rangle \mid v^{-1}out_\tau(\varepsilon) = \bot]$.*
- *For fixing the rules, we define for all $p = (u, v)$ such that $[p]_\tau \in Q$ and for all $f \in F$ such that $uf^{-1}dom(\tau) \neq \emptyset$:*

$$rhs([p]_\tau, f) = out_{p^{-1}\tau}(f)[v'/[(ufi, vv')]_\tau\langle x_i\rangle \mid v'^{-1}out_{p^{-1}\tau}(f) = \bot \text{ and } i = ind_\tau(p, f, v')]$$

Our next objective is to show that $can(\tau)$ is the unique minimal earliest compatible trimmed DTOPI that defines τ. In order to do so, we will prove that $[\![can(\tau)]\!] = \tau$, and that $can(\tau)$ is compatible, trimmed and earliest. Then, to prove its minimality, we will consider another earliest compatible trimmed DTOPI$_{\text{td}}$ that defines τ, and note that it uses equivalent states. Since $can(\tau)$ has no redundant states (no two states equivalent), this will prove that $can(\tau)$ is the unique minimal earliest compatible trimmed DTOPI that defines τ.

Proposition 56. *For any transformation τ definable by some* DTOPI_{td}*, $can(\tau)$ is a* DTOPI_{td} *defining τ that is compatible, trimmed, and earliest.*

Proof. Suppose that $\tau = [\![N]\!]$ for some $\text{DTOPI}_{\text{reg}}$ $N = (M, D)$. Let $can(\tau) = N' = (M', dom(\tau))$. We will first prove that N' defines τ. Then, we will show it is also compatible, trimmed, and earliest.

The first step is to show that $[\![N']\!]_{[p]_\tau} = p^{-1}\tau$. We prove for the input tree $s = f(s_1, ..., s_n)$ that:

$$[\![N']\!]_{[p]_\tau}(s) = out_{p^{-1}\tau}(f)[v'/[\![N']\!]_{[(ufi,vv')]_\tau}(s_i) \mid v'^{-1}out_{p^{-1}[\![N]\!]}(f) = \bot \text{ and } i = ind_\tau(p, f, v')]$$

This is done by induction on the input tree s. By induction hypothesis, $[\![N']\!]_{[(ufi,vv')]_\tau}(s_i) = (ufi, vv')^{-1}\tau(s_i) = v'^{-1}p^{-1}\tau(s)$. We then prove $[\![N']\!]_{[p]_\tau}(s) = p^{-1}\tau(s)$. All paths v'' such that $out_{p^{-1}\tau}(f) \models v''$ are both in $p^{-1}\tau(s)$ and in $[\![N']\!]_{[p]_\tau}(s)$. Plus, for v' such that $v'^{-1}out_{p^{-1}\tau}(f) = \bot$, $v'^{-1}[\![N']\!]_{[p]_\tau}(s) = v'^{-1}p^{-1}\tau(s)$.

We add the axiom on top of these production to show that $\tau = [\![N']\!]$. For a tree $s \in dom(\tau)$ we have that $[\![N']\!](s) = out_\tau(\varepsilon)[v/[\![N']\!]_{[(\varepsilon,v)]_\tau}(s) \mid v^{-1}out_\tau(\varepsilon) = \bot]$. We can then prove $[\![N']\!](s) = \tau(s)$. All paths v' such that $out_\tau(\varepsilon) \models v'$ are both in $\tau(s)$ and in $[\![N']\!](s)$. Plus, for v such that $v^{-1}out_\tau(\varepsilon) = \bot$, $v^{-1}[\![N']\!](s) = v^{-1}\tau(s)$.

To show that $[\![N']\!]$ is compatible and earliest, we prove that if a pair $p = (u, v)$ is syntactically aligned, then $u \sim_{[p]_\tau} v$. This can be proven by induction on the length of u. If $u = \varepsilon$, then $\varepsilon \sim_{[p']_\tau} v$ if and only if $v^{-1}ax = [p']_\tau\langle x_0\rangle$. From Definition 55, this means $v^{-1}out_\tau(\varepsilon) = \bot$, and that $[(\varepsilon, v)]_\tau = [p']_\tau$. If $u = u_0fi$, and $u_0 \sim_{[p_0]_\tau} v_0$, and $u \sim_{[p']_\tau} v$, then there exists v_0, v_1 such that $v_1 rhs'([p_0]_\tau, f) = [p']_\tau\langle x_i\rangle$. By induction, we know that $[(u_0, v_0)]_\tau = [p_0]_\tau$. From Definition 55, $v_1^{-1}rhs'([p_0]_\tau, f) = [p']_\tau\langle x_i\rangle$ means that $v_1^{-1}out_{(u_0,v_0)^{-1}\tau}(\varepsilon) = \bot$, and that $[(u_0fi, v_0v_1)]_\tau = [p']_\tau$.

To show that $[\![N']\!]$ is trimmed, we prove that every state $[p]_\tau$ and every rule $rhs'([p]_\tau, f)$ is useful. If $[p]_\tau$ is a state of Q', then p is a semantically aligned pair of τ. Proposition 48 ensures that p is syntactically aligned in some state of N'. As seen above in this proof, this means that p is syntactically aligned in $[p]_\tau$. Hence, $[p]_\tau$ is useful. Furthermore, since $rhs'([p]_\tau, f)$ exists if and only if $p = (u, v)$ and $uf^{-1}dom(\tau) \neq \emptyset$, every rule is also useful. □

Now that we have shown that all DTOPI_{td} have an equivalent earliest trimmed compatible DTOPI_{td}, we will prove that for all DTOPI_{td}, $can(\tau)$ is the unique minimal earliest trimmed compatible DTOPI_{td} to define τ, up to state renaming. To this end, we will prove that two equivalent earliest trimmed compatible DTOPI_{td} have the same syntactically aligned pairs, which means they use equivalent states. Then, we will consider clean transducers with no two equivalent states and prove that the only clean earliest trimmed compatible DTOPI_{td} to define τ is $min(\tau)$, up to states renaming.

We start by proving that two equivalent trimmed earliest compatible DTOPI$_{\text{td}}$ have the same syntactically aligned pairs.

Theorem 57. *Let $N = (M, D)$ and $N' = (M', D)$ be two equivalent trimmed earliest compatible* DTOPI$_{td}$*. For all p syntactically aligned pair of N, p is a syntactically aligned pair of N'.*

Proof. Let $M = (Q, F, G, ax, rhs)$ and $M' = (Q', F, G, ax', rhs')$. This proof is made by induction on the size of u. If $u = \varepsilon$ then for some state q of N, $ax \models vq\langle x_0\rangle$. Since M is earliest, Lemma 38 gives that $v^{-1}out_{[\![N]\!]}(\varepsilon) = \bot$. For the same reason, since $v^{-1}out_{[\![N']\!]}(\varepsilon) = \bot$, we have that for some q' of N', $ax' \models vq'\langle x_0\rangle$. Hence (u, v) if a syntactically aligned pair of N'.

If $u = u'fi$, then there exists v', v'' such that $v = v'v''$, $u' \sim_{q_0} v'$, and for some state q of N, $rhs(q_0, f) \models vq\langle x_i\rangle$. By induction, there exists a state q'_0 of N' such that $u' \sim_{q_0} v'$. This means that $[\![N]\!]_{q_0} = [\![N']\!]_{q'_0}$, and since N and N' are both trimmed and compatible, if $rhs(q_0, f)$ is defined, then $rhs'(q'_0, f)$ is defined. Since M is earliest, Lemma 38 gives that $v''^{-1}out_{[\![N]\!]_{q_0}}(f) = \bot$. For the same reason, since $v''^{-1}out_{[\![N']\!]_{q'_0}}(f) = \bot$, we have that for some q' of N', for some index j, $ax' \models vq'\langle x_j\rangle$. The fact that $i = j$ is due to the fact that both are equal to $ind_\tau((u', v'), f, v'')$. □

This important theorem allows us to ensure that a DTOPI$_{\text{td}}$ will have as few states as possible when it has exactly one state per semantic class $[p]_\tau$. To ensure this, we define clean transducers as transducers with no redundant states.

Definition 58. *We say a compatible* DTOPI *N is* clean *if it is trimmed, and for q and q' two distinct states of N, $[\![N]\!]_q \neq [\![N']\!]_{q'}$.*

Note that just like for trimmed DTOPI, it is easier to prove the existence of a clean DTOPI equivalent to some DTOPI N than to actually compute it.

Lemma 59. *For N an earliest compatible* DTOPI *there exists an equivalent clean earliest compatible* DTOPI*.*

Proof. The existence of a trimmed DTOPI has already been argued in Lemma 21: if a state or a rule is useless, it can be deleted without changing the semantics of N, or its earliest compatible nature. Similarily, if there exists two equivalent states q, q' such that $[\![N]\!]_q = [\![N]\!]_{q'}$, then one can delete q' and its rules $rhs(q', f)$, nd replace every occurence of $q'\langle x_i\rangle$ by $q\langle x_i\rangle$ in ax and rhs. Since both states are equivalent, this substitution can be done without changing the semantics of N, or its earliest compatible nature. We can thus delete redundant states until none are left, and end up obtaining a clean earliest compatible DTOPI equivalent to N. □

We show that our definition of clean implies a minimal number of states: since two equivalent earliest compatible trimmed DTOPI$_{\text{td}}$ have identical syntactically aligned pairs, it is easy to show that they use equivalent states.

Lemma 60. *For $N = (M, D)$ a trimmed earliest compatible* DTOPI$_{td}$*, and $N' = (M', D)$ an equivalent clean earliest compatible* DTOPI$_{td}$*. There exists an onto function ϕ from the states of N to the states of N' such that for all q state of N, $[\![N']\!]_{\phi(q)} = [\![N]\!]_q$.*

Proof. If q is a state of N, a trimmed DTOPI, there exists (u, v) such that $u \sim_q v$. From Theorem 57 we conclude that there exists a state q' of N' such that $u \sim_{q'} v$. For such a q', we would have $[\![N]\!]_q = [\![N']\!]_{q'}$. We note $\phi(q)$ the only state q' of the clean DTOPI$_{\text{td}}$ such that $[\![N]\!]_q = [\![N']\!]_{q'}$. This function is onto through a symmetrical reasoning: if q' is a state of N', a trimmed DTOPI$_{\text{td}}$, there exists (u, v) such that $u \sim_{q'} v$. From Theorem 57 we conclude that there exists a state q of N such that $u \sim_q v$. For such a q, we would have $[\![N]\!]_q = [\![N']\!]_{q'}$, hence $\phi(q) = q'$. □

In the following theorem, we will prove that there only exists one clean earliest compatible DTOPI$_{\text{td}}$, up to state renaming, that is to say that if two DTOPI$_{\text{td}}$ N and N' are equivalent, clean, earliest and compatible, then there exists a one-to-one function ϕ from the states of N to the states of N' such that if every occurrence of every state q of N is replaced by its image $\phi(q)$, we obtain exactly N'.

Lemma 61. *If N and N' are two equivalent clean earliest compatible* DTOPI$_{td}$*, then $N = N'$, up to state renaming.*

Proof. From Lemma 60, we know that there exists a one-to-one correspondence ϕ between the states of N and the states of N' such that $[\![N]\!]_q = [\![N']\!]_{\phi(q)}$. We now show that this one-to-one correspondence is indeed a state rewriting between N and N'. For that, it just remains to prove that the rules of q and $\phi(q)$ are identical up to state renaming. First of all, since $[\![N]\!]_q = [\![N']\!]_{\phi(q)}$ and both N and N' are trimmed, we know that for all input letter f, there exist a rule $rhs(q, f)$ if and only if there is a tree of root f in $dom([\![N]\!]_q)$, if and only if there is a tree of root f in $dom([\![N']\!]_{\phi(q)})$, if and only if there exist a rule $rhs'(\phi(q), f)$. Furthermore, since both N and N' are earliest, from Lemma 38, we know that $rhs(q, f)\Psi = out_{[\![N]\!]_q}(f)$ and $rhs'(\phi(q), f)\Psi' = out_{[\![N']\!]_{\phi(q)}}(f)$ for some Ψ, Ψ'. Since $[\![N]\!]_q = [\![N']\!]_{\phi(q)}$, all that remains to show is that if $v^{-1}rhs(q, f) = q'\langle x_i\rangle$, then $v^{-1}rhs'(\phi(q), f) = \phi(q')\langle x_i\rangle$. Since $[\![N]\!]_q = [\![N']\!]_{\phi(q)}$, we have that if $v^{-1}rhs(q, f) = q'\langle x_i\rangle$, then $v^{-1}rhs'(\phi(q), f) = q''\langle x_j\rangle$. Since N is trimmed, there exists (u_0, v_0) such that $u_0 \sim_q v_0$ in N (and hence $u_0 \sim_{\phi(q)} v_0$ in N'). This means that for N, $i = ind_{[\![N]\!]}((u_0, v_0), f, v)$, and $u_0 fi \sim_{q'} v_0 v$. Hence for N', $i = ind_{[\![N']\!]}((u_0, v_0), f, v)$, and $u_0 fi \sim_{q''} v_0 v$. This means that if $v^{-1}rhs(q, f) = q'\langle x_i\rangle$, then $v^{-1}rhs'(\phi(q), f) = q''\langle x_j\rangle$, with $q'' = \phi(q')$ and $i = j$. □

This finally proves our normal form theorem:

Theorem 62. *For $N = (M, D)$ a* DTOPI$_{reg}$*, there exists a unique equivalent compatible earliest* DTOPI$_{td}$ *with a minimal number of states, up to state renaming.*

Proof. The existence of such a DTOPI is proven by Lemma 59, its uniqueness by Lemma 61. □

10 Learning from Examples

We next show how to learn transducers of the class DTOPI_{td} for a given input domain. Since DTTAs are themselves learnable [36], this is a reasonable assumption to make.

10.1 Learning Model

We fix ranked alphabet F and G. Since we suppose the domain of our transformation to be previously known, we define the class of all DTOPI that share the same domain D:

Definition 63. *For any top-down domain $D \subseteq \mathcal{T}_F$, we define the transformation class $\text{DTOPI}_{td}(D)$ that contains all transformations τ from $\mathcal{T}_F$ to $\mathcal{T}_G$ definable by some DTOPI_{td} with $dom(\tau) = D$.*

A *sample* is a finite partial function $S \subseteq \mathcal{T}_F \times \mathcal{T}_G$. A sample S is called *compatible* with D if $dom(S) \subseteq D$. A sample S for a transformation τ is a finite subset of $S \subseteq \tau$.

Definition 64. *We say that the class of $\text{DTOPI}_{td}(D)$ is learnable if there are:*
- *an algorithm $learn_D$ defining a partial function that maps samples compatible with D to $\text{DTOPI}_{td}(D)$ in normal form, and*
- *a function $char$ that maps $\text{DTOPI}_{td}(D)$ N in normal form to samples of transformation $[\![N]\!]$.*

We require for any $\text{DTOPI}_{td}(D)$ N and any sample S for $[\![N]\!]$ containing $char(N)$ that $learn_D(S) = N$.

There are several parameters to consider when describing the complexity of learning algorithms:
- Sample complexity describes the number of examples in $char(N)$ as a function of the size of N.
- Time complexity describes the complexity of the learning algorithm $learn_D$ as a function of the size of its input sample.

We say a class is learnable *with polynomial ressources* if the the number of examples in $char(N)$ is polynomial as a function of the size of N, and the learning algorithm $learn_D$ is in polynomial time as a function of the size of its input sample.

Theorem 65. *For any D definable by some DTTA, the class $\text{DTOPI}_{td}(D)$ is learnable with polynomial resources.*

Proof. The proof captures the rest of this section. It will follow from Propositions 73 and 79. □

Since the exact domain D of the target $\text{DTOPI}_{\text{td}}(D)$ $N = (M, D')$ is assumed to be known, and since the target transducer will be in normal form, know have that $D = D'$, so that we only have to learn the DTOP M.

Finally, we will assume that $D \neq \emptyset$, since the case $D = \emptyset$ is easy to treat.

10.2 Characteristic Samples

The purpose a characteristic sample to provide enough information to describe a $\text{DTOPI}_{\text{td}}(D)$ N in normal form, where D is recognized by a DTTA. Since normal forms can be characterized in terms of the finitely many equivalence classes of residuals of $[\![N]\!]$, the objective is to present the required information on finitely many pairs p of paths such that the residuals $p^{-1}[\![N]\!]$ represent all relevant classes.

The first question is by which pair p to represent a residual $p^{-1}[\![N]\!]$. The idea is to choose the least pair p' that defines the same residual as p with respect to the following total order. If $p = (u, v)$ and $p' = (u', v')$ the we define $p < p'$ if and only if:

- if $|u| < |u'|$,
- if $|u| = |u'|$ and $u <_{lex} u'$,
- if $u = u'$ and $|v| < |v'|$, or
- if $u = u'$, and $|v| = |v'|$ and $v <_{lex} v'$.

This order is interesting for two reasons. The first one is that contrary to simple simple lexical order, which can produce infinite sets with no minimals (e.g. the language a^*b if $a <_{lex} b$), this order has a well-defined notion of minimals in sets. Furthermore, it has interesting properties, chief amongst them being stability by composition.

Lemma 66. *If $p = (u, v)$, $p' = (u', v')$ two pairs such that $p <_{lex} p'$, then for every pair (u'', v''), $(uu'', vv'') <_{lex} (u'u'', v'v'')$.*

To properly define the notion of a sample S containing enough information to learn a $\text{DTOPI}_{\text{td}}(D)$ N, we will establish what semantically aligned pairs of τ are of relevance, and what S should teach on them. Our first move is to define minimal pairs and their boundary, i.e. all the semantically aligned pairs a sample should have information about in order to learn N.

Definition 67. *For any transformation τ and any semantically aligned pair p of τ, we define $minp_\tau(p) = min\{p' \mid p' \equiv_\tau p\}$. By extension, we define the set of minimal semantically aligned pairs of τ as:*

$$minp(\tau) = \{p \mid p \text{ least semantically aligned pair for } \tau \text{ with residual } p^{-1}\tau\}$$

Note that for any $\text{DTOPI}_{\text{td}}(D)$ N the set of residuals of $[\![N]\!]$ is finite by Theorem 49. This means $minp(\tau)$ is finite too.

To find all minimal pairs, our algorithm will explore aligned pairs starting at the axiom, and continue repeatedly with all new pairs that are detected. Hence, the aligned pairs that will be explored do not only contain those in $minp(\tau)$, but also their successors, as well as all the aligned pairs of the axiom.

Definition 68. *For any transformation τ we define the boundary of the minimal semantically aligned pairs of τ as:*

$$\begin{aligned} minp^+(\tau) =& \{(\varepsilon, v) \mid v^{-1} out_\tau(\varepsilon) = \bot\} \cup \\ & \{p' \in Succ_\tau(minp(\tau))\} \end{aligned}$$

The following lemma shows for transformations τ defined by DTOPI_{td}'s, all elements of $minp(\tau)$ are in fact part of the axiom, or successors of of some element of $minp(\tau)$:

Lemma 69. *Let N be a DTOPI_{td}, and $[\![N]\!] = \tau$. Then $minp(\tau) \subseteq minp^+(\tau)$*

Proof. We consider $p = (u, v) \in minp(\tau)$. We will prove that it is either of form $(\varepsilon, v) \mid v^{-1} out_\tau(\varepsilon) = \bot$ or a successor of $p' \in minp(\tau)$.

If $u = \varepsilon$, then since all elements of $minp(\tau)$ are semantically aligned, this means that $v^{-1} out_\tau(\varepsilon) = \bot$.

If $u = u'fi$ for some u', f, i, then Lemma 53 shows the existence of a pair $p' = (u', v')$ such that p is a successor of p'. Let v'' such that $v = v'v''$. Suppose $p' = (u', v') \notin minp(\tau)$. Then there exists $p_0 = (u_0, v_0)$ such that $p_0 < p'$ and $p_0 \equiv_\tau p'$. Notably, $(u_0 fi, v_0 v'') < (u'fi, v'v'')$ and $(u_0 fi, v_0 v'') \equiv_\tau (u'fi, v'v'')$. This is in contradiction with the fact that $p \in minp(\tau)$. Hence, $p' \in minp(\tau)$. □

The set $minp(\tau)$ can be seen as having one unique representative for each class of $\equiv_\tau$. In this sense, it can be seen as representing the states of $can(\tau)$. As a matter of fact, we will define $repcan(\tau)$ the state renaming of $can(\tau)$ where each state $[p]_\tau$ is represented by the unique pair $p' = minp_\tau(p)$.

Definition 70. *Let N a $\text{DTOPI}_{td}(D)$, and $\tau = [\![N]\!]$. We define $repcan(\tau)$ the representative of $can(\tau)$ as the* DTOP $M = (Q, F, G, ax, rhs)$ *where:*

- $Q = minp(\tau)$
- $ax = out_\tau(\varepsilon)[v \leftarrow minp_\tau(\varepsilon, v)\langle x_0\rangle \mid v^{-1} out_\tau(\varepsilon) = \bot]$
- *For $p = (u, v) \in minp(\tau)$, f such that $uf^{-1}D \neq \emptyset$,*

$$\begin{aligned} rhs(p, f) = out_{p^{-1}\tau}(f)[v' \leftarrow & minp_\tau(ufi, vv')\langle x_i\rangle \\ & \mid v'^{-1} out_{p^{-1}\tau}(f) = \bot,\ i = ind_\tau(p, f, v')] \end{aligned}$$

We remember that Lemma 52 ensures that for $p = (u, v)$ semantically aligned, and f, v' such that $v'^{-1} out_{p^{-1}\tau}(f) = \bot$, $ind_\tau(p, f, v')$ is unique, and $(ufi, vv') = succ_\tau(p, f, v')$. This $repcan(\tau)$ is the actual target of our algorithm.

We now consider our sample S, and establish what kind of information it needs to contain for our learning algorithm to be able to retro-engineer $repcan(\tau)$. To this end we will notably have to be able to identify equivalent pairs for $\equiv_\tau$. In order to show that $p \not\equiv_\tau p'$ by a sample S for τ, we will require that p and p' are in contradiction with respect to S in the following sense:

Definition 71. *Given a sample S for a transformation τ, we say that two semantically aligned pairs p and p' are in contradiction with respect to S and write $p \nparallel_S p'$ if $p^{-1}S \cup p'^{-1}S$ is not functional.*

In this case, S contains a counter example for $p \not\equiv_\tau p'$. We now use this definition to formalize what it means for a sample S to be characteristic for τ.

Definition 72. *Let N be a* $\text{DTOPI}_{td}(D)$ *and $\tau = [\![N]\!]$. A sample S for τ is called* characteristic *if:*

(1) $out_S(\varepsilon) = out_\tau(\varepsilon)$

(2) for all $p \in minp(\tau)$ and $f \in F$: $out_{p^{-1}S}(f) = out_{p^{-1}\tau}(f)$,

(3) for $p = (u, v) \in minp(\tau)$, $f^{(k)} \in F$, for all $j \in \{1, \dots, k\}$, if $(ufj, vv')^{-1}\tau$ is not functional, then $(ufj, vv')^{-1}S$ is not functional.

(4) for all $p \in minp^+(\tau)$, and all $p' \in minp(\tau)$ such that $p^{-1}D = p'^{-1}D$, and $p \not\equiv_\tau p'$: $p \nparallel_S p'$.

Points (1) and (2) ensure that S contains enough information to build the axiom and rules of our target transducer, as seen in Lemma 38. Point (3) ensures we never explore pairs that are not semantic alignments. Point (4) ensures we are able to tell which aligned pairs are equivalent to which minimal pairs.

Note that if a sample S for τ is characteristic, then any larger sample for τ is characteristic too. It remains to show that for all $\text{DTOPI}_{\text{td}}(D)$ N, there exists a characteristic sample for $[\![N]\!]$.

Proposition 73. *Let D be definable by a* DTTA. *For any* $\text{DTOPI}_{td}(D)$ *N, where $[\![N]\!] = \tau$, there exists a characteristic sample for τ with a number of examples polynomial in the number of equivalence classes in $\equiv_\tau$.*

Proof. We will show that a polynomial number of examples is required for each point (1-4) of Definition 72.

For point (1), we want to ensure that $out_S(\varepsilon) = out_\tau(\varepsilon)$. First, we need at least one example in to ensure that $out_S(\varepsilon)$ is defined. Note that such an example always exists, as we supposed $dom(\tau) = D \neq \emptyset$. We fix $s_\varepsilon \in dom(\tau)$ arbitrarily and add $(s_\varepsilon, \tau(s_\varepsilon))$ to S. Then, since $S \subseteq \tau$ will be guaranteed, the only concern is that $out_S(\varepsilon)$ is bigger than $out_\tau(\varepsilon)$. To this end, we will provide one example per equivalence class $[p]_\tau$ of $\equiv_\tau$, where $p = (\varepsilon, v)$ is an

aligned pair of τ. For each such class $[p]_\tau$, there exists a tree $s_{[p]_\tau}$ such that $v^{-1}\tau(s_{[p]_\tau}) \sqcap v^{-1}\tau(s_\varepsilon) = \bot$. If $(s_\varepsilon, \tau(s_\varepsilon)) \in S$ and for all equivalence class $[p]_\tau$ of $\equiv_\tau$, where $p = (\varepsilon, v)$ is an aligned pair of τ, $(s_{[p]_\tau}, \tau(s_{[p]_\tau})) \in S$, then $out_{pS}(\varepsilon) = out_\tau(\varepsilon)$.

Point (2) works in a similar fashion. We want to ensure that for all $p = (u, v) \in minp(\tau)$, for all input letter f, $out_{p^{-1}S}(f) = out_{p^{-1}\tau}(f)$. First, we need at least one example such that $s_{p,f} \models uf$ to ensure that $out_{p^{-1}S}(f)$ is defined. We call this example $(s_{p,f}, \tau(s_{p,f}))$. Then, since $S \subseteq \tau$, the only concern is that $out_{p^{-1}S}(f)$ is bigger than $out_{p^{-1}\tau}(f)$. We shall ensure that there exists enough examples in S so that for all v' such that $v'^{-1}out_{p^{-1}\tau}(f) = \bot$, $v'^{-1}out_{p^{-1}S}(f) = \bot$. To this end, we will provide one example per equivalence class $[p']_\tau$ of $\equiv_\tau$, where $p' \in Succ_\tau(p)$. For each such class $[p']_\tau$, there exists a tree s such that $p'^{-1}\tau(s) \sqcap p'^{-1}\tau(v^{-1}s_{p,f}) = \bot$. We then choose a tree $s_{p,f,[p']_\tau}$, such that $u^{-1}s_{p,f,[p']_\tau} = s$. If $(s_{p,f}, \tau(s_{p,f})) \in S$ and for all $v'^{-1}out_{p^{-1}\tau}(f) = \bot$, $p' \in Succ_\tau(p)$, $(s_{p,f,[p']_\tau}, \tau(s_{p,f,[p']_\tau})) \in S$, then $out_{p^{-1}S}(f) = out_{p^{-1}\tau}(f)$.

Point (3) is ensured by providing an explicit counterexample for every pair $p' = (ufj, vv')$ we want to prove is not functional. If $p'^{-1}\tau$ is not functional, then there exists $s_{p'}$, $s'_{p'}$ input trees such that $ufj^{-1}s_{p'} = ufj^{-1}s'_{p'}$ but either $vv'^{-1}\tau(s_{p'}) \not= vv'^{-1}\tau(s'_{p'})$ or $\tau(s_{p'}) \models vv'$ but $\tau(s'_{p'}) \not\models vv'$. Hence, if S contains $(s_{p'}, \tau(s_{p'}))$ and $(s'_{p'}, \tau(s'_{p'}))$, then $p'^{-1}S$ is not functional. Note that there is a polynomial number of those pairs: for $p = (u, v) \in minp(\tau)$, for f an input letter, all paths v' such that $v'^{-1}out_{p^{-1}\tau}(f) = \bot$ are in $rhs(q, f)$ where q is the state of N that computes $p^{-1}\tau$. Since $j \leq rank(f)$, this leaves a polynomial number of pairs to consider.

Point (4) works in a similar fashion. For $p = (u, v)$, $p' = (u', v')$ two non-equivalent semantic alignments, if $p^{-1}D = p'^{-1}D$, then there exists s an input tree such that $p{-}1\tau(s) \neq p'{-}1\tau(s)$. We take two input trees $s_{p,p'}$, $s'_{p,p'}$ such that $u^{-1}s_{p,p'} = u'^{-1}s'_{p,p'} = s$. Hence, if S contains $(s_{p,p'}, \tau(s_{p,p'}))$ and $(s'_{p,p'}, \tau(s'_{p,p'}))$, then $p'^{-1}S$ is not functional. Note that there is a polynomial number of those cases to consider, since $minp(\tau)$ and $minp^+([\![N]\!])$ are of polynomial size themselves.

By taking all examples needed to ensure points (1-4), we built a characteristic sample for τ in polynomial size. □

Example 74. *For the transduction τ_{flip} of Example 7, there are four residuals. The first is the residual of the minimal pair* $(\mathsf{P}1, \mathsf{P}2)$*, the identity on lists of* A*, the second is the residual of the minimal pair* $(\mathsf{P}2, \mathsf{P}1)$*, the identity on lists of* B*, and the others are the residuals of the two semantic aligned pairs from the earliest axiom,* $(\varepsilon, \mathsf{P}1)$ *and* $(\varepsilon, \mathsf{P}2)$*.*

$$minp(\tau_{flip}) = \{(\varepsilon, \mathsf{P}1),\ (\varepsilon, \mathsf{P}2),\ (\mathsf{P}1, \mathsf{P}2),\ (\mathsf{P}2, \mathsf{P}1)\}$$

The boundary contains the aligned pairs from the axiom, and those directly extending the pairs in $minp(\tau_{flip})$:

$$minp^+(\tau_{flip}) = minp(\tau_{flip}) \cup \{(\mathsf{P}1\mathsf{A}1, \mathsf{P}2\mathsf{A}1),\ (\mathsf{P}2\mathsf{B}1, \mathsf{P}2\mathsf{B}1)\}$$

To satisfy point (1), a characteristic sample would need enough information to deduce $out_{\tau_{flip}}(\varepsilon)$. *As seen in Proposition 73, this means that we need a first example* $(s_\varepsilon, \tau_{flip}(s_\varepsilon))$, *and another example for all equivalence classes of aligned pairs of form* (ε, v). *In* τ_{flip} *there are two,* $[(\varepsilon, \mathsf{P}1)]_{\tau_{flip}}$ *and* $[(\varepsilon, \mathsf{P}2)]_{\tau_{flip}}$. *We choose* $s_\varepsilon = \mathsf{P}(\#, \#)$. *For* $[(\varepsilon, \mathsf{P}1)]_{\tau_{flip}}$, *we choose* $s_{[(\varepsilon,\mathsf{P}1)]_{\tau_{flip}}} = \mathsf{P}(\#, \mathsf{B}(\#))$. *For* $[(\varepsilon, \mathsf{P}2)]_{\tau_{flip}}$, *we choose* $s_{[(\varepsilon,\mathsf{P}2)]_{\tau_{flip}}} = \mathsf{P}(\mathsf{A}(\#), \#)$.

To satisfy point (2), a characteristic sample would need enough information to deduce $out_{p^{-1}\tau_{flip}}(f)$ *for all relevant pairs* p *and* f. *As seen in Proposition 73, this means that for all* $p = (u, v) \in minp(\tau_{flip})$, f *an input letter, we choose an example* $(s_{p,f}, \tau_{flip}(s_{p,f}))$.

Then, for all equivalence classes $[p']_{\tau_{flip}}$ *where* $p' \in Succ_{\tau_{flip}}(p)$ *we pick another example* $s_{p,\mathsf{P},[p']_{\tau_{flip}}}$ *to ensure* $v'^{-1} out_{p^{-1}S}(\mathsf{P}) = \bot$.
For $p = (\varepsilon, \mathsf{P}1)$, *the only letter that can be read is* f, *which leads to the only successor* $p' = (\mathsf{P}2, \mathsf{P}1)$. *We choose* $s_{p,\mathsf{P}} = \mathsf{P}(\#, \#)$, *and* $s_{p,\mathsf{P},[p']_{\tau_{flip}}} = \mathsf{P}(\#, \mathsf{B}(\#))$.
For $p = (\varepsilon, \mathsf{P}2)$, *the only letter that can be read is* f, *which leads to the only successor* $p' = (\mathsf{P}1, \mathsf{P}2)$. *We choose* $s_{p,\mathsf{P}} = \mathsf{P}(\#, \#)$, *and* $s_{p,\mathsf{P},[p']_{\tau_{flip}}} = \mathsf{P}(\mathsf{A}(\#), \#)$.
For $p = (\mathsf{P}1, \mathsf{P}2)$, *two letters can be read:* A *which leads to the only successor* $p' = (\mathsf{P}1\mathsf{A}1, \mathsf{P}2\mathsf{A}1)$, *and* $\#$, *which leads to no successor. For* A, *we choose* $s_{p,\mathsf{A}} = \mathsf{P}(\mathsf{A}(\#), \#)$, *and* $s_{p,\mathsf{A},[p']_{\tau_{flip}}} = \mathsf{P}(\mathsf{A}(\mathsf{A}(\#)), \#)$. *For* $\#$, *we only need to choose* $s_{p,\#} = \mathsf{P}(\#, \#)$
For $p = (\mathsf{P}2, \mathsf{P}1)$, *two letters can be read:* B *which leads to the only successor* $p' = (\mathsf{P}2\mathsf{B}1, \mathsf{P}1\mathsf{B}1)$, *and* $\#$, *which leads to no successor. For* B, *we choose* $s_{p,\mathsf{B}} = \mathsf{P}(\#, \mathsf{B}(\#))$, *and* $s_{p,\mathsf{B},[p']_{\tau_{flip}}} = \mathsf{P}(\#, \mathsf{B}(\mathsf{B}(\#)))$. *For* $\#$, *we only need to choose* $s_{p,\#} = \mathsf{P}(\#, \#)$.

To satisfy point (3), a characteristic sample would need enough information to deduce which pairs (ufi, vv') *extending a minimal pair* p *are semantic alignments. As seen in Proposition 73, this means that for all* $p = (u, v) \in minp(\tau_{flip})$, *and an extension* $p' = (ufi, vv')$ *that is not a semantic alignment, we need two examples to ensure* $p'^{-1}S$ *is not functional. In* τ_{flip}, *there are only two such pairs we need to consider,* $(\mathsf{P}1, \mathsf{P}1)$ *and* $(\mathsf{P}2, \mathsf{P}2)$. *For* $(\mathsf{P}1, \mathsf{P}1)$, *we choose the pair of examples* $(\mathsf{P}(\#, \#), \mathsf{P}(\#, \#))$ *and* $(\mathsf{P}(\#, \mathsf{B}(\#)), \mathsf{P}(\mathsf{B}(\#), \#))$.
For $(\mathsf{P}2, \mathsf{P}2)$, *we choose* $(\mathsf{P}(\#, \#), \mathsf{P}(\#, \#))$ *and* $(\mathsf{P}(\mathsf{A}(\#), \#), \mathsf{P}(\#, \mathsf{A}(\#)))$.

To satisfy point (4), a characteristic sample would need enough information to differentiate pairs $p \in minp(\tau_{flip}) \cup minp^+(\tau_{flip})$ *from their non-equivalent counterpart of* $p' \in minp(\tau_{flip})$ *of same domain. As seen in Proposition 73, this means that for all such pair of alignments* p, p', *we need one example to ensure* $p \not\upharpoonright_S p'$. *In* τ_{flip}, *only* $(\varepsilon, \mathsf{P}1)$ *and* $(\varepsilon, \mathsf{P}2)$ *are of same domain but not equivalent. We choose the example* $(\mathsf{P}(\#, \mathsf{B}(\#)), \mathsf{P}(\mathsf{B}(\#), \#))$.

Hence, a complete characteristic sample would be the transformation:

$$
\begin{aligned}
S = [\ & \mathsf{P}(\#, \#)/\mathsf{P}(\#, \#)\,, \\
& \mathsf{P}(\mathsf{A}(\#), \#)/\mathsf{P}(\#, \mathsf{A}(\#))\,, && \mathsf{P}(\#, \mathsf{B}(\#))/\mathsf{P}(\mathsf{B}(\#), \#)\,, \\
& \mathsf{P}(\mathsf{A}(\mathsf{A}(\#)), \#)/\mathsf{P}(\#, \mathsf{A}(\mathsf{A}(\#)))\,, && \mathsf{P}(\#, \mathsf{B}(\mathsf{B}(\#)))/\mathsf{P}(\mathsf{B}(\mathsf{B}(\#)), \#)\ ]
\end{aligned}
$$

10.3 Learning Algorithm

We describe the algorithm $learn_D$. The goal is to create a minimal leftmost earliest $\text{DTOPI}_{\text{td}}(D)$, which means creating earliest states with no redundancy, their rules, and the axiom. The idea is to try to fold any new aligned pair we find to an existing state. If no equivalent state can be found, we create a new one.

In this algorithm we build a $\text{DTOPI}_{\text{td}}(D)$ $learn_D(S) = (M, D)$, where $M = (Q, F, G, ax, rhs)$. For simplicity's sake, our states will be pair (u, v). Those states will be divided in two disjoint sets: Q_{safe} for pairs that minimally represent an equivalence class, and therefore represent a state in $learn_D(S)$, and Q_{temp}, for pairs that have not yet been examined by the algorithm, and are still susceptible to be equivalent to an existing pair in Q_{safe}. Rules of rhs are only created for states of Q_{safe}, but leaves in these rules or the axiom can temporarily be pairs $p\langle x_i\rangle$, where p is still an "unapproved" pair of Q_{temp}.

From Definition 72, we know that if S is characteristic, $p^{-1}\tau$ is functional if and only if $p^{-1}S$ is functional. Furthermore, $p \equiv_\tau p'$ if and only if $p^{-1}D = p'^{-1}D$ and $\neg p \not\parallel_S p'$. Procedure *integrate-state* describes how, given a DTOPI N and a sample S, to test if a pair $p \in Q_{temp}$ is equivalent to an existing state in Q_{safe} and, if it is not, how to create a new state and its rules. Note that in these rules, $p_{fi}\langle x_i\rangle$ can appear for pairs that are not yet confirmed to be original states. These pairs are added to Q_{temp}. The creation of the axiom works in a similar manner to the way we create a rule in Procedure *integrate-state*. From there, the full algorithm goes as described in Figure 7.

Example 75. *We try to learn a transducer for τ_{flip}, with the characteristic sample we found in Example 74.*

Our end goal in this part is to prove the correctness of our algorithm, i.e. that if N is a $\text{DTOPI}_{\text{td}}(D)$, $\tau = [\![N]\!]$, S is a characteristic sample for τ and $min(\tau) = (M_\tau, D)$, then $learn_D(S)$ is equal to $repcan(\tau)$.

To this end, we will show that at each intermediary step of the algorithm, we will learn a "partially unfolded" version of $repcan(\tau)$: before calling *integrate-state* on a pair p, all states $p' < p$ should be in Q_{safe}, all pairs $(\varepsilon, v) \in minp^+$ or $Succ_\tau(Q_{safe})$ should have appeared in Q_{temp}, but all such pair smaller than p should already be integrated. This leads to a transducer that has some definitive states in Q_{safe}, some unexplored temporary states in Q_{temp}, and some calls in ax and rhs possibly pointing to a safe state or a temporary state, depending on their lexical order relative to p.

To formalize this notion of partially unfolded rules, we first define what pairs should be replaced, what pairs should still be unexplored, and we define the *p-truncated* version of $repcan(\tau)$.

Definition 76. *Let N be a $\text{DTOPI}_{td}(D)$, $\tau = [\![N]\!]$, and p an aligned pair of τ. For $p' \in minp^+(\tau)$, we call $minrep^p_\tau(p')$ the p-truncated representative of p':*

- *$minrep^p_\tau(p') = p''$ the unique element of $minp(\tau)$ such that $p'' \equiv_\tau p'$ if $p' < p$,*

```
// let F and G be ranked signatures
// and D ⊆ T_F the language of some DTTA

fun learn_D(S) // S ⊆ D × T_G finite partial function
  Q_temp := {(ε, v) | v^{-1} out_S(ε) = ⊥}
  ax = out_S(ε) [v ← (ε, v)⟨x_0⟩ | (ε, v) ∈ Q_temp]
  Q_safe := ∅
  rhs := ∅
  M = (Q_safe ∪ Q_temp, F, G, ax, rhs)
  proc integrate-state(p) = // fusion temporary state p with
       // some safe state if possible or make p safe
       // and create its transition rules.
    Q_eq = {p' ∈ Q_safe | p^{-1}D = p'^{-1}D and not p ∦_S p'}
    (u, v) = p
  in
    case Q_eq // Q_eq may contain at most 1 element
    of {p'} then replace all occurrences of p in M by p'
                          Q_temp := Q_temp\{p}
    of ∅ then
      Q_safe := Q_safe ∪ {p}
      Q_temp := Q_temp \ {p}
      for f ∈ F where uf^{-1}D ≠ ∅ do
        V' = {v' | v'^{-1} out_{p^{-1}S}(f) = ⊥}
        fun i(v') // where v' ∈ V'
          unique index i s.t. (ufi, vv')^{-1}S is functional
               // exists by Corollary 51 since
               // D is defined by a DTTA.
        end
        fun state(v') // where v' ∈ V'
            (ufi(v'), vv')
        end
        fun call(v') // where v' ∈ V'
            state(v')⟨x_{i(v')}⟩
        end
      in
        Q_temp := Q_temp ∪ {state(v') | v' ∈ V'}
        rhs(p, f) := out_{p^{-1}S}(f) [v' ← call(v') | v' ∈ V']
      end
    end
  end
in
    while Q_temp ≠ ∅ do
      p = min(Q_temp)
    in
      integrate-state(p)
    end
    return (M, D)
end
```

Figure 7: Learning algorithm of DTOPs with top-down inspection.

Action	Q_{safe}	***rhs***	Q_{temp}
Initialization	$\emptyset$	$\emptyset$	$(\varepsilon, \mathsf{P1})$; $(\varepsilon, \mathsf{P2})$
integrate-state $(\varepsilon, \mathsf{P1})$	$(\varepsilon, \mathsf{P1})$	$(\varepsilon, \mathsf{P1})(\mathsf{P}(x_1, x_2)) \to (\mathsf{P2}, \mathsf{P1})\langle x_2 \rangle$	$(\varepsilon, \mathsf{P2})$; $(\mathsf{P2}, \mathsf{P1})$
integrate-state $(\varepsilon, \mathsf{P2})$	$(\varepsilon, \mathsf{P1})$ $(\varepsilon, \mathsf{P2})$	$(\varepsilon, \mathsf{P1})(\mathsf{P}(x_1, x_2)) \to (\mathsf{P2}, \mathsf{P1})\langle x_2 \rangle$ $(\varepsilon, \mathsf{P2})(\mathsf{P}(x_1, x_2)) \to (\mathsf{P1}, \mathsf{P2})\langle x_1 \rangle$	$(\mathsf{P1}, \mathsf{P2})$; $(\mathsf{P2}, \mathsf{P1})$
integrate-state $(\mathsf{P1}, \mathsf{P2})$	$(\varepsilon, \mathsf{P1})$ $(\varepsilon, \mathsf{P2})$ $(\mathsf{P1}, \mathsf{P2})$	$(\varepsilon, \mathsf{P1})(\mathsf{P}(x_1, x_2)) \to (\mathsf{P2}, \mathsf{P1})\langle x_2 \rangle$ $(\varepsilon, \mathsf{P2})(\mathsf{P}(x_1, x_2)) \to (\mathsf{P1}, \mathsf{P2})\langle x_1 \rangle$ $(\mathsf{P1}, \mathsf{P2})(\mathsf{A}(x_1)) \to \mathsf{A}((\mathsf{P1A1}, \mathsf{P2A1})\langle x_1 \rangle)$	$(\mathsf{P2}, \mathsf{P1})$; $(\mathsf{P1A1}, \mathsf{P2A1})$
integrate-state $(\mathsf{P2}, \mathsf{P1})$	$(\varepsilon, \mathsf{P1})$ $(\varepsilon, \mathsf{P2})$ $(\mathsf{P1}, \mathsf{P2})$ $(\mathsf{P2}, \mathsf{P1})$	$(\varepsilon, \mathsf{P1})(\mathsf{P}(x_1, x_2)) \to (\mathsf{P2}, \mathsf{P1})\langle x_2 \rangle$ $(\varepsilon, \mathsf{P2})(\mathsf{P}(x_1, x_2)) \to (\mathsf{P1}, \mathsf{P2})\langle x_1 \rangle$ $(\mathsf{P1}, \mathsf{P2})(\mathsf{A}(x_1)) \to \mathsf{A}((\mathsf{P1A1}, \mathsf{P2A1})\langle x_1 \rangle)$ $(\mathsf{P1}, \mathsf{P2})(\#) \to \#$ $(\mathsf{P2}, \mathsf{P1})(\mathsf{B}(x_1)) \to \mathsf{B}((\mathsf{P2B1}, \mathsf{P1B1})\langle x_1 \rangle)$ $(\mathsf{P2}, \mathsf{P1})(\#) \to \#$	$(\mathsf{P1A1}, \mathsf{P2A1})$; $(\mathsf{P2B1}, \mathsf{P1B1})$
integrate-state $(\mathsf{P1A1}, \mathsf{P2A1})$	$(\varepsilon, \mathsf{P1})$ $(\varepsilon, \mathsf{P2})$ $(\mathsf{P1}, \mathsf{P2})$ $(\mathsf{P2}, \mathsf{P1})$	$(\varepsilon, \mathsf{P1})(\mathsf{P}(x_1, x_2)) \to (\mathsf{P2}, \mathsf{P1})\langle x_2 \rangle$ $(\varepsilon, \mathsf{P2})(\mathsf{P}(x_1, x_2)) \to (\mathsf{P1}, \mathsf{P2})\langle x_1 \rangle$ $(\mathsf{P1}, \mathsf{P2})(\mathsf{A}(x_1)) \to \mathsf{A}((\mathsf{P1}, \mathsf{P2})\langle x_1 \rangle)$ $(\mathsf{P1}, \mathsf{P2})(\#) \to \#$ $(\mathsf{P2}, \mathsf{P1})(\mathsf{B}(x_1)) \to \mathsf{B}((\mathsf{P2B1}, \mathsf{P1B1})\langle x_1 \rangle)$ $(\mathsf{P2}, \mathsf{P1})(\#) \to \#$	$(\mathsf{P2B1}, \mathsf{P1B1})$
integrate-state $(\mathsf{P2B1}, \mathsf{P1B1})$	$(\varepsilon, \mathsf{P1})$ $(\varepsilon, \mathsf{P2})$ $(\mathsf{P1}, \mathsf{P2})$ $(\mathsf{P2}, \mathsf{P1})$	$(\varepsilon, \mathsf{P1})(\mathsf{P}(x_1, x_2)) \to (\mathsf{P2}, \mathsf{P1})\langle x_2 \rangle$ $(\varepsilon, \mathsf{P2})(\mathsf{P}(x_1, x_2)) \to (\mathsf{P1}, \mathsf{P2})\langle x_1 \rangle$ $(\mathsf{P1}, \mathsf{P2})(\mathsf{A}(x_1)) \to \mathsf{A}((\mathsf{P1}, \mathsf{P2})\langle x_1 \rangle)$ $(\mathsf{P1}, \mathsf{P2})(\#) \to \#$ $(\mathsf{P2}, \mathsf{P1})(\mathsf{B}(x_1)) \to \mathsf{B}((\mathbf{P2}, \mathbf{P1})\langle x_1 \rangle)$ $(\mathsf{P2}, \mathsf{P1})(\#) \to \#$	$\emptyset$

Figure 8: Example of a run of the algorithm of Figure 7

- $minrep^p_\tau(p') = p'$ *itself if* $p' \geqslant p$

Definition 77. *Let* N *be a* $\text{DTOPI}_{td}(D)$, $\tau = [\![N]\!]$, *with its canonical* DTOP $repcan(\tau) = (Q, F, G, ax, rhs)$, *and* p *an aligned pair of* τ. *We define the* p*-truncated form of* $repcan(\tau)$, $repcan_p(\tau) = (Q_{safe(p)} \cup Q_{temp(p)}, F, G, ax_p, rhs_p)$, *where:*

- *The* p*-truncated safe states* $Q_{safe(p)} = \{p' \in Q \mid p' < p\}$
- *The* p*-truncated temporary states*
  $Q_{temp(p)} = \big(\{(\varepsilon, v) \mid v^{-1}ax = q\langle x_0\rangle\} \cup Succ_\tau(Q_{safe(p)})\big) \cap \{p' \mid p' \geqslant p\}$
- *The* p*-truncated axiom*
  $ax_p = ax\ [v \leftarrow minrep^p_\tau(\varepsilon, v)\langle x_0\rangle \mid v^{-1}ax = q\langle x_0\rangle]$
- *The* p*-truncated rules* rhs_p *are defined so that if* $p' = (u', v') \in Q_{safe(p)}$, *and* f *a letter such that* $rhs(p', f)$ *is defined, then*

$$rhs_p(p', f) = rhs(p', f)\ [v'' \leftarrow minrep^p_\tau(u'fi, v'v'')\langle x_i\rangle \mid v''^{-1}rhs(p', f) = q\langle x_i\rangle]$$

Note that the p-truncated form "develops" as p grows, to finally become $repcan(\tau)$ itself if $p > max(minp^+(\tau))$.

Corollary 78. *Let* N *be a* $\text{DTOPI}_{td}(D)$, $\tau = [\![N]\!]$, $N_\tau = (M_\tau, D)$ *the canonical form of* N, p *semantically aligned for* τ *such that* $p > max(minp^+(\tau))$. *Then* $repcan_p(\tau) = repcan(\tau)$.

Proof. By construction, if $p > max(minp^+(\tau))$, for all pairs $p' \in minp^+(\tau)$, we have $minrep^p_\tau(p') = minp_\tau(p')$. Furthermore, $Q_{safe(p)} = minp(\tau)$, $Q_{temp(p)} = \emptyset$ hence $repcan_p(\tau)$ and $repcan(\tau)$ have same states. Finally, the only difference between the definition of the axiom and rules of $repcan_p(\tau)$ and $repcan(\tau)$ is that the former uses $minrep^p_\tau$ and the latter uses $minp_\tau$. Since they are both identical for all pairs of $minp^+(\tau)$, we have that $repcan_p(\tau)$ and $repcan(\tau)$ have same axioms and rules. □

We prove the correctness of $learn_D$ if the sample S is characteristic. To this end, we show that right before each call to *integrate-state*(p), the transducer M created by $learn_D$ is exactly M_p.

Proposition 79. *Let* N *be a* $\text{DTOPI}_{td}(D)$, $\tau = [\![N]\!]$, *and* S *a characteristic sample of* τ. *Then* $learn_D(S) = repcan(\tau)$.

Proof. We prove the following invariant: if $Q_{temp} \neq \emptyset$ and $p = min(Q_{temp})$, then $Q_{safe} = Q_{safe(p)}$, $Q_{temp} = Q_{temp(p)}$, $ax = ax_p$ and $rhs = rhs_p$.

After the initialization, $Q_{safe} = \emptyset$ and $Q_{temp} = \{(\varepsilon, v) \mid v^{-1}out_S(\varepsilon) = \bot\}$. Since S is characteristic, this means $Q_{temp} = \{(\varepsilon, v) \mid v^{-1}out_\tau(\varepsilon) = \bot\}$. If $Q_{temp} \neq \varepsilon$, we call $p = min(Q_{temp})$. p is the smallest aligned pair of τ. This

means that $Q_{safe(p)} = minp \cap \{p' \mid p' < p\}$ is empty, and therefore $Q_{safe(p)} = Q_{safe}$. This gives us that $Q_{temp(p)} = \{(\varepsilon, v) \mid v^{-1}out_\tau(\varepsilon) = \bot\} \cap \{p' \mid p' \geqslant p\}$. Since all aligned pairs are bigger than p, we have $Q_{temp(p)} = Q_{temp}$.
Since p is the smallest aligned pair, $minrep^p_\tau$ is the identity function. This means that $ax_p = out_\tau(\varepsilon)\ [v \leftarrow (\varepsilon, v)\langle x_0\rangle \mid v^{-1}out_\tau(\varepsilon) = \bot]$. The current axiom in learn is $ax = out_S(\varepsilon)\ [v \leftarrow (\varepsilon, v)\langle x_0\rangle \mid v^{-1}out_S(\varepsilon) = \bot]$. Since S is characteristic, $ax_p = ax$. As for rules, none have been created yet, which means $rhs = rhs_p = \emptyset$.

For the inductive case, we consider $p = (u, v)$ the minimal element of Q_{temp}. We have $Q_{safe} = Q_{safe(p)}$, $Q_{temp} = Q_{temp(p)}$, $ax = ax_p$ and $rhs = rhs_p$. We call the new values after *integrate-state*(p) Q'_{safe}, Q'_{temp}, ax' and rhs'. If $Q'_{temp} \neq \emptyset$, we call p' its minimum. p is added to Q_{safe} if and only if for all $p'' \in Q_{safe}$, $p \not\parallel_S p''$. Since S is characteristic, $Q_{safe} \subseteq minp(\tau)$ and $p \in minp^+(\tau)$, this means that for all $p'' \in Q_{safe(p)}$, $p \not\equiv_\tau p''$. Hence, p is added to Q_{safe} if and only if $p \in minp(\tau)$. For the same reason, states are added to Q_{temp} if and only if $p \in minp(\tau)$. If S is characteristic, then $V' = \{v' \mid v'^{-1}out_\tau(f) = \bot\}$, and for each $v' \in V'$, then $i(v')$ is the unique i such that $(ufi, vv')^{-1}\tau$ is functional, i.e. $i(v') = ind_\tau(p, f, v')$. This means that the new temporary states of Q'_{temp} are $\{state(v') \mid v' \in V'\} = Succ_\tau(p)$. In all cases p is removed from Q_{temp}. This means that regardless of weather p was added or not, $Q'_{safe} = minp(\tau) \cap \{p'' \mid p'' \leqslant p\}$ and $Q'_{temp} = \{(\varepsilon, v) \mid v^{-1}out_\tau(\varepsilon) = \bot\} \cap \{p'' \mid p'' > p\}$. This means that Q'_{safe} and Q'_{temp} are $Q_{safe(p'')}$ and $Q_{temp(p'')}$ for some pair p'' right after p. Since $p' = min(Q_{temp})$, there is no element of $minp^+(\tau)$ between p and p'. Thus, $Q'_{safe} = Q_{safe(p')}$, and $Q'_{temp} = Q_{temp(p')}$.
If p is not a new state, then $minrep^p_\tau$ is different from $minrep^{p'}_\tau$ only for p, which has to be replaced by the only $p'' \in Q_{safe}$ such that $p'' \equiv_\tau p$. As *integrate-state*(p) replaces every state call $p\langle x_i\rangle$ by $p''\langle x_i\rangle$, we have $ax' = ax_{p'}$ and $rhs' = rhs_{p'}$. However, if p is a new state, then $minrep^{p'}_\tau = minrep^p_\tau$, and thus $ax' = ax_{p'}$, but new rules have to be added. *integrate-state*(p) adds new rules. For $p = (u, v)$, we have that for each f such that $uf^{-1}D \neq \emptyset$, we create the rule $rhs'(p, f) = out_{p^{-1}S}(f)\ [v' \leftarrow call(v') \mid v' \in V']$. As previously mentioned, $V' = \{v' \mid v'^{-1}out_\tau(f) = \bot\}$, and $i(v') = ind_\tau(p, f, v')$. This means that $rhs'(p, f) = out_{p^{-1}\tau}(f)\ [v' \leftarrow (ufi, vv')\langle x_i\rangle \mid (ufi, vv') = succ_\tau(p, f, v'), v' \in V']$. For all (ufi, vv') in $Succ_\tau(p)$, we know that $(ufi, vv') > p$, and thus $p'' = minrep^{p'}_\tau(p'')$. Hence, $rhs' = rhs_{p'}$.

It remains to show that the last step that eventually empties Q_{temp} leads to $repcan(\tau)$. If Q_{temp} starts as $\emptyset$, this means $out_S(\varepsilon)$ has no $\bot$-leaf. Since S is characteristic, this means $out_\tau(\varepsilon)$ has no $\bot$-leaf. This is only possible if τ is a constant transduction that sends all trees of D to the same image t. In this case, $learn_D(S)$ produces a transducer with no states and no rules, and an axiom $ax = t$, which is indeed the canonical form of τ. In all other cases, we consider p the last pair to be integrated by *integrate-state*(p). Since it is the last considered pair, we have that $p = max(minp^+(\tau))$. As seen in this proof, after this last *integrate-state*(p), we have $learn_D(S) = repcan_{p'}(\tau)$, where p' is the pair right after p in lexical order. Thus, Corollary 78 gives us that

$learn_D(S) = repcan(\tau)$. □

11 Conclusions

Deterministic top-down tree transformations can be semantically characterized by studying semantic alignment, i.e. the dependencies between input and output paths when producing the output as early as possible. This notion of semantically aligned pairs allows for a definition of residual transformations. From there, there exists a Myhill-Nerode theorem on DTOPs. The normal form of [18] coincides with the minimal normal form that comes from our Myhill-Nerode theorem. This normal form can be inferred by a Gold-style learning algorithm, polynomial in data and time.

For an extension of this result to wider classes, a first step would be to allow for DTOPs with regular domains: if semantic alignments and the notion of compatible and earliest transducers extend to regular inspections, the Myhill-Nerode theorem presented here relies on properties that would not extend to top-down tree transformations with regular domains. Another wider still interesting and robust class are top-down tree transducers with regular look-ahead [15]. 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. This extension, however, is more ambitious, for two main reasons. First, no normal form is known on this class. For string transducers with look-ahead [13], this normal form is composed of a look-ahead as coarse as possible, paired with the minimal transducer on the relabelling. We know this method cannot extend to tree transducers as is, since there is not a unique look-ahead as coarse as possible in the general case. Furthermore, even if a normal form is found, Gold-style learning algorithms encounter particular hurdles when dealing with look-ahead. For the learning problem on string transducers with look-ahead [4] , the normal form provided by the Myhill-Nerode algorithm is difficult to infer in polynomial time: polynomial inference is possible for another, different, more ad-hoc normal form.

As another possible avenue of extension, most Gold-style learning algorithms can be used as core in an interactive learner in Angluin-style [1], similar to [8]. One might wonder if the algorithm presented in this paper can be changed into an Angluin-style learning algorithm, with a polynomial number of learner-teacher interactions.

References

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

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

[3] Geert J. Bex, Wouter Gelade, Frank Neven, and Stijn Vansummeren. Learning deterministic regular expressions for the inference of schemas from XML data. In *17-th International Conference on World Wide Web*, 2008.

[4] Adrien Boiret, Aurélien Lemay, and Joachim Niehren. Learning rational functions. In Hsu C. Yen, Oscar H. Ibarra, Hsu C. Yen, and Oscar H. Ibarra, editors, *Developments in Language Theory*, volume 7410 of *Lecture Notes in Computer Science*, pages 273–283. Springer, 2012.

[5] Adrien Boiret, Aurélien Lemay, and Joachim Niehren. Learning Top-Down Tree Transducers with Regular Domain Inspection. In *International Conference on Grammatical Inference 2016*, Delft, Netherlands, October 2016.

[6] Angela Bonifati, Radu Ciucanu, and Aurélien Lemay. Interactive path query specification on graph databases. In Gustavo Alonso, Floris Geerts, Lucian Popa, Pablo Barceló, Jens Teubner, Martín Ugarte, Jan Van den Bussche, and Jan Paredaens, editors, *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015.*, pages 505–508. OpenProceedings.org, 2015.

[7] Angela Bonifati, Radu Ciucanu, and Slawek Staworko. Learning join queries from user examples. *ACM Trans. Database Syst.*, 40(4):24, 2016.

[8] Julien Carme, Rémi Gilleron, Aurélien Lemay, and Joachim Niehren. Interactive learning of node selecting tree transducers. *Machine Learning*, 66(1):33–67, January 2007.

[9] Christian Choffru. *Contribution a l'etude de quelques familles remarquables de fonctions rationnelles.* PhD thesis, Universite de Paris VII, 1978.

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

[11] Hubert Comon, Max Dauchet, Remi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. Tree automata techniques and applications. October 2007.

[12] AnHai Doan, Pedro Domingos, and Alon Halevy. Reconciling schemas of disparate data sources: A Machine-Learning approach. In *Proceedings of the ACM SIGMOD Conference*, pages 509–520, 2001.

[13] C. C. Elgot and G. Mezei. On relations defined by generalized finite automata. *IBM J. of Res. and Dev.*, 9:88–101, 1965.

[14] J. Engelfriet. Bottom-up and top-down tree transformations. a comparision. *Mathematical System Theory*, 9:198–231, 1975.

[15] J. Engelfriet. Top-down tree transducers with regular look-ahead. *Mathematical System Theory*, 10:198–231, 1977.

[16] J. Engelfriet and S. Maneth. The equivalence problem for deterministic MSO tree transducers is decidable. In R. Ramanujam and S. Sen, editors, *Proceedings of the 25th Conference on Foundations of Software Technology and Theoretical Computer Science – FSTTCS'2005*, volume 3821 of *LNCS*, pages 495–504. Springer-Verlag, 2005.

[17] Joost Engelfriet and Sebastian Maneth. Macro tree translations of linear size increase are MSO definable. *SIAM J. Comput.*, 4(32):950–1006, 2003.

[18] Joost Engelfriet, Sebastian Maneth, and Helmut Seidl. Deciding equivalence of top-down XML transformations in polynomial time. . *Comput. System Sci.*, 75(5):271–286, 2009.

[19] Joost Engelfriet and Heiko Vogler. Macro tree transducer. . *Comput. System Sci.*, 31:71–146, 1985.

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

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

[22] Jonathan Graehl, Kevin Knight, and Jonathan May. Training tree transducers. *Comput. Linguist.*, 34(3):391–427, 2008.

[23] Wim Janssen, Alexandr Korlyukov, and Jan Van den Bussche. On the tree-transformation power of XSLT. *Acta Inf.*, 43(6):371–393, January 2007.

[24] Stephan Kepser. A simple proof for the Turing-Completeness of XSLT and XQuery. In *Extreme Markup Languages®*, 2004.

[25] Pavel Labath and Joachim Niehren. A uniform programmning language for implementing XML standards. In Giuseppe F. Italiano, Tiziana Margaria-Steffen, Jaroslav Pokorný, Jean-Jacques Quisquater, and Roger Wattenhofer, editors, *SOFSEM 2015: Theory and Practice of Computer Science - 41st International Conference on Current Trends in Theory and Practice of Computer Science, Pec pod Snvevzkou, Czech Republic, January 24-29, 2015. Proceedings*, volume 8939 of *Lecture Notes in Computer Science*, pages 543–554. Springer, 2015.

[26] Aurélien Lemay, Sebastian Maneth, and Joachim Niehren. A Learning Algorithm for Top-Down XML Transformations. In Acm, editor, *29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 285–296, Indianapolis, United States, 2010. ACM Press.

[27] Aurélien Lemay, Sebastian Maneth, and Joachim Niehren. A learning algorithm for Top-Down XML transformations. In *29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 285–296. ACM-Press, 2010.

[28] S. Maneth and G. Busatto. Tree transducers and tree compressions. In I. Walukiewicz, editor, *Foundations of Software Science and Computation Structures - FOSSACS'04*, volume 2987 of *LNCS*, pages 363–377, Barcelona, Spain, 2004. Springer-Verlag.

[29] Sebastian Maneth, Alexandru Berlea, Thomas Perst, and Helmut Seidl. XML type checking with macro tree transducers. In *24th ACM Symposium on Principles of Database Systems*, pages 283–294, New York, NY, USA, 2005. ACM-Press.

[30] Sebastian Maneth and Frank Neven. Structured document transformations based on XSL. In Richard C. H. Connor, Alberto O. Mendelzon, Richard C. H. Connor, and Alberto O. Mendelzon, editors, *DBPL*, volume 1949 of *Lecture Notes in Comput. Sci.*, pages 80–98. Springer, 1999.

[31] Mehryar Mohri. Minimization algorithms for sequential transducers. *Theor. Comput. Sci.*, 234(1-2):177–201, 2000.

[32] Atsuyuki Morishima, Hiroyuki Kitagawa, and Akira Matsumoto. A machine learning approach to rapid development of XML mapping queries. In Z. Meral Özsoyoglu, Stanley B. Zdonik, Z. Meral Özsoyoglu, and Stanley B. Zdonik, editors, *ICDE*, pages 276–287. IEEE Computer Society, 2004.

[33] A. Nerode. Linear automaton transformation. In *Proc. Amer. Math. Soc.*, volume 9, pages 541–544, 1958.

[34] Joachim Niehren, Jérôme Champavère, Rémi Gilleron, and Aurélien Lemay. Query Induction with Schema-Guided Pruning Strategies. *. Mach. Learn. Res.*, 2013.

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

[36] J. Oncina and P. Garcia. Inferring regular languages in polynomial update time. In *Pattern Recognition and Image Analysis*, pages 49–61, 1992.

[37] J. Oncina, P. Garcia, and E. Vidal. Learning subsequential transducers for pattern recognition and interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:448–458, 1993.

[38] Ruhsan Onder and Zeki Bayram. XSLT version 2.0 is Turing-Complete: A purely transformation based proof. In Oscar H. Ibarra, Hsu C. Yen, Oscar H. Ibarra, and Hsu C. Yen, editors, *CIAA*, volume 4094 of *Lecture Notes in Comput. Sci.*, pages 275–276. Springer, 2006.

[39] Lucian Popa, Yannis Velegrakis, Renée J. Miller, Mauricio A. Hernández, and Ronald Fagin. Translating web data. In *VLDB*, pages 598–609. Morgan Kaufmann, 2002.

[40] Helmut Seidl, Sebastian Maneth, and Gregor Kemper. Equivalence of deterministic top-down tree-to-string transducers is decidable. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 943–962. IEEE Computer Society, 2015.

[41] Slawek Staworko and Piotr Wieczorek. Characterizing XML twig queries with examples. In Marcelo Arenas and Martín Ugarte, editors, *18th International Conference on Database Theory, ICDT 2015, March 23-27, 2015, Brussels, Belgium*, volume 31 of *LIPIcs*, pages 144–160. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

A Remaining Proofs

A.1 Trimming a DTopI

We describe the construction of a trimmed compatible DTopI from a DTopI. The existence of such a DTopI is argued in Proposition 21.

Proposition 21. *Any* DTopI_{reg} *is equivalent to some trimmed* DTopI_{reg} *and any* DTopI_{td} *is equivalent to some trimmed* DTopI_{td}.

Proof. Let $N = (M, D)$ be a $\text{DTopI}_{\text{reg}}$, $M = (Q, F, G, ax, rhs)$. We call $D' = dom([\![N]\!])$, and consider a trimmed nondeterministic tree automaton $A = (F, P, P_I, \Delta)$ such that $[\![A]\!] = D'$.

We want to find for each state $q \in Q$ the set P_q of states of P reached by q:

$$P_q = \{p \in P \mid \exists (u, v) \mid u \sim_q v \text{ and } u \text{ reaches } p\}$$

.

We can compute P_q recursively, as shown by this equivalent mutually recursive definition. The sets P_q are the smallest subsets of P such that:

- if $q\langle x_0\rangle$ occurs in ax, then $P_I \subseteq P_q$.
- if $p \in P_q$, $p \xrightarrow{f} (p_1, ..., p_i, ..., p_n)$ a rule of A, and $q'\langle x_i\rangle$ occurs in $rhs(q, f)$, then $p_i \in P_{q'}$.

The equivalence between those two definitions can be proven by recursion on the length of the input path u required to access a state $q \in Q$ at the same time as a state $p \in P$. For $u = \varepsilon$, a pair (u, v) is aligned in q if and only if $v^{-1}ax = q\langle x_0\rangle$. Conversely, ε reaches exactly P_I the set of initial states of A. For the recursion, $p \in P_q$ means that there exists a pair (u, v) such that $u \sim_q v$ and u reaches p. If $q'\langle x_i\rangle$ occurs in $rhs(q, f)$, there exists v' such that ${v'}^{-1} rhs(q, f) = q'\langle x_i\rangle$. This means $ufi \sim_{q'} vv'$. Meanwhile in A, if there is a rule $p \xrightarrow{f} (p_1, ..., p_i, ..., p_n)$, and u reaches p, then ufi reaches p_i, which means $p_i \in P_{q'}$.

From there, selecting useful states and rules is immediate. A state q is useful if and only if $P_q \neq \emptyset$. A rule $rhs(q, f)$ is useful if and only if there exists a state $p \in P_q$ with a rule $p \xrightarrow{f} (p_1, ..., p_i, ..., p_n)$. □

A.2 Origins of Output Constructors

We want to prove Proposition 22, that states that when computing an image of a DTop, each constructor of the output tree is produced either by the axiom or when rewriting the subtree at a unique input path.

In order to do this, we will need an alternative characterization of syntactic alignments. It deals with the fact that the definition of judgements $u \sim^q_{q'} v$ extends paths u to the right by repeated concatenation. The alternative

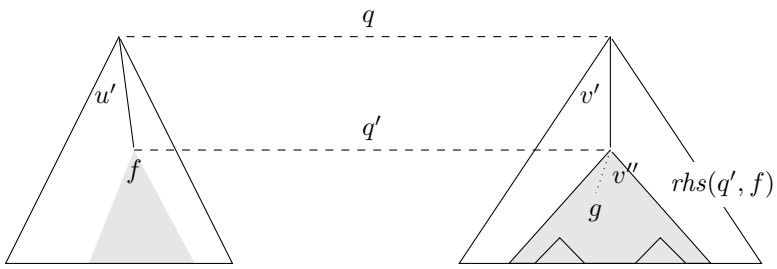
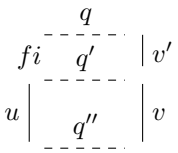


Figure 9: Constructor g at the output node $v = v'v''$ is produced by $[\![M]\!]_q$ from the f-rooted subtree located at node u' of the input tree. This is justified by the syntactic alignment $u' \sim^q_{q'} v'$ and $rhs(q', f) \models v''g$.

characterization is given by the following equivalent judgements $u \dot{\sim}^q_{q'} v$ that repeatedly concatenates to the left instead:

$$\frac{}{\varepsilon \dot{\sim}^q_q \varepsilon} \qquad \frac{rhs(q,f) \models v'q'\langle x_i\rangle \qquad u \dot{\sim}^{q'}_{q''} v}{fiu \dot{\sim}^q_{q''} v'v}$$



Lemma 80. *$u \sim^q_{q'} v$ if and only if $u \dot{\sim}^q_{q'} v$.*

Proof. We can show that $u \sim^q_{q'} v$ if and only if there exists a sequence $q_0, ..., q_n$ such that $q_0 = q$, $q_n = q'$, $u = f_0i_0...f_{n-1}i_{n-1}$, $v = v_0...v_{n-1}$, and for all j from 0 to $n-1$, $rhs(q_j, f_j) \models v_jq_{j+1}\langle x_{i_j}\rangle$. This comes from a simple proof by induction. The first definition adds an additional state after q_n, an input letter after $f_{n-1}i_{n-1}$, and output path after v_{n-1}. The second definition adds an additional state before q_0, an input letter before f_0i_0, and output path after v_0. □

We next show that any constructor of an output tree produced by $[\![M]\!]_q$ comes from a syntactic alignment starting in state q. This is illustrated in Fig. 9 and formalized in Lemma 81.

Lemma 81. *Let M be a* DTop *and $s \in dom([\![M]\!]_q)$ an input tree for some state q of M. Then for any output path v and letter $g \in G$ such that $[\![M]\!]_q(s) \models vg$ there exist a decomposition $v = v'v''$, an input path $u'f$, and a state q' such that:*

- *$s \models u'f$,*
- *$u' \sim^q_{q'} v'$, and*
- *$rhs(q', f) \models v''g$.*

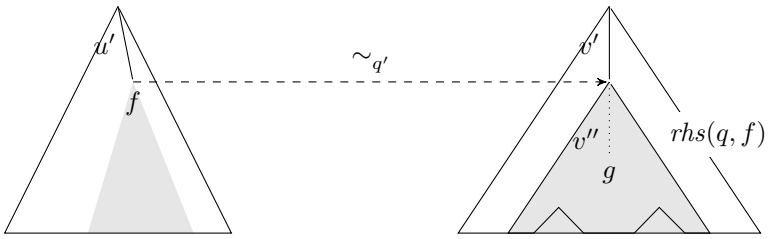


Figure 10: The f-subtree at input path u' produces the g-constructor at output path $v'v''$ by M. This is justified by the syntactic alignment starting with the axiom $u' \sim_{q'} v'$ and $rhs(q', f) \models v''g$.

Proof. The proof is by induction on the structure of s. Let $s = f'(s_1, \ldots, s_k)$ for some $f'^{(k)} \in F$ and trees $s_1, \ldots, s_k$. The definition of $[\![M]\!]_q$ yields:

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

- Case $rhs(q, f') \models vg$. In this case, we choose $u' = v' = \varepsilon$, $f = f'$, and $q' = q$. Clearly, $s \models u'f$, $u' \sim_q^q v'$, and $rhs(q', f) \models v''g$.
- Case $rhs(q, f') \models v'_0\tilde{q}\langle x_i\rangle$ for some decomposition $v = v'_0v''_0$, $1 \leq i \leq k$, and state $\tilde{q'}$. Hence, ${v'_0}^{-1}[\![M]\!]_q(s) = [\![M]\!]_{\tilde{q}}(s_i)$, so that $s_i \in dom([\![M]\!]_{\tilde{q}})$ and $[\![M]\!]_{\tilde{q}}(s_i) \models v''_0g$. By induction hypothesis, there exist a decomposition $v'' = v'_1v''$, an input path u'_1f, and a state q' such that $s_i \models u'_1f$, $u'_1 \sim_{q'}^{\tilde{q}} v'_1$, and $rhs(q', f) \models v''g$. Hence, $s \models f'iu'_1f$ so we can choose $u' = f'iu'_1$ in order to satisfy the first condition $s \models u'f$. Lemma 80 yields $u'_1 \dot{\sim}_{q'}^{\tilde{q}} v'_1$, so that we can apply the inference rule:

$$\frac{rhs(q, f') \models v'_0\tilde{q}\langle x_i\rangle \qquad u'_1 \dot{\sim}_{q'}^{\tilde{q}} v'_1}{f'iu'_1 \dot{\sim}_{q'}^{q} v'_0v'_1}$$

  Lemma 80 the other way around yields $u' \sim_{q'}^q v'_0v'_1$, so we can choose $v' = v'_0v'_1$ to show the second condition. The third condition $rhs(q', f) \models v''g$ is also satisfied as shown above.

□

We next lift the previous Lemma to how a constructor of the output tree may be produced by $[\![M]\!]$ instead of $[\![M]\!]_q$. There are two cases. Either the output constructor is produced by the axiom, or else it has its origin at some node of the input tree, as illustrated in Fig. 10 and formalized in the next proposition.

Proposition 22. *Let M be a* DTop *and $s \in dom(M)$ an input tree. Then for any output path vg such that $[\![M]\!](s) \models vg$, either $ax \models vg$ or there exist a unique decomposition $v = v'v''$, an input path $u'f$, and a state q' such that:*

- $s \models u'f$,

- $u' \sim_{q'} v'$, *and*
- $rhs(q', f) \models v''g$.

Proof. If $ax \models vg$ then the first case is satisfied. Otherwise, by Definition of $[\![M]\!](s)$, the axiom produces some call $q\langle x_0\rangle$ at a prefix of v. This means that there is a decomposition $v = v'_0 v''_0$ such that $ax \models v'_0 q\langle x_0\rangle$, ${v'_0}^{-1}[\![M]\!](s) = [\![M]\!]_q(s)$, and that $[\![M]\!]_q(s) \models v''_0 g$. We can now apply Lemma 81 to $[\![M]\!]_q(s) \models v''_0 g$, which shows that there exist a decomposition $v''_0 = v'_1 v''$, an input path $u'f$, and a state q' such that $s \models u'$, $u' \sim^q_{q'} v'_1$ and $rhs(q', f) \models v''g$. In this case, we can apply the following inference rule:

$$\frac{ax \models v'_0 q\langle x_0\rangle \qquad u' \sim^q_{q'} v'_1}{u' \sim_{q'} v'_0 v'_1}$$

With $v' = v'_0 v'_1$ we have the second condition $u' \sim_{q'} v'$, while the first condition $s \models u'$ and the third condition $rhs(q', f) \models v''g$ were show already above. □