JMLR: Workshop and Conference Proceedings 57:54–65, 2016

Learning Top-Down Tree Transducers with Regular Domain Inspection

Adrien Boiret[1,3] adrien.boiret@inria.fr
Aurélien Lemay[1,3] aurelien.lemay@univ-lille3.fr
Joachim Niehren[2,3] joachim.niehren@inria.fr
[1] *Université de Lille*
[2] Inria *Lille - Nord Europe*
[3] Links *project of* Inria *&* Cristal *(*Umr *9189)*

Abstract

We study the problem of how to learn tree transformations on a given regular tree domain from a finite sample of input-output examples. We assume that the target tree transformation can be defined by a deterministic top-down tree transducer with regular domain inspection ($\text{DTopI}_{\text{reg}}$). An rpni style learning algorithm that solves this problem in polynomial time and with polynomially many examples was presented at Pods'2010, but restricted to the case of path-closed regular domains. In this paper, we show that this restriction can be removed. For this, we present a new normal form for $\text{DTopI}_{\text{reg}}$ by extending the Myhill-Nerode theorem for DTop to regular domain inspections in a nontrivial manner. The rpni style learning algorithm can also be lifted but becomes more involved too.

Keywords: Tree Transducers, Myhill-Nerode Theorem, Normal Form, Model Learning

1. Introduction

Trees are used to structure data in various application domains, subsuming NoSQL databases, document processing, and Web data management. The most prominent and frequently used formats for data trees today are Json and xml. Whenever data trees are used, a major practical problem is how to define tree transformations from one schema into another. In the xml world, the W3C standard languages xslt and XQuery were developed for this purpose, while in the context of Json, such transformations are usually defined in Javascript.

The definition of tree transformations in such programming languages requires considerable programming expertise and coding time, for instance for publishing semi-structured data on the Web, or exchanging it between web services. Therefore, one might hope that symbolic machine learning tools can automatically infer tree transformations from a small number of input-output examples. In particular it would be nice to have algorithms able to learn tree transformations, while satisfying Gold's learning model in polynomial time and

. This work was partially supported by a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020

data (Gold, 1978). The approach of the present paper is to extend the Ostia learning algorithm for learning subsequential transducers (Oncina et al., 1993) to a learning algorithm for tree transformations. This means that we want to identify the unique minimal earliest normal form (Choffrut, 2003) of a tree transformation in a suitable class.

Since full-fledged programming languages for defining tree transformations such as xslt or Javascript tend to be Turing complete, we consider more restricted subclasses of tree transformations definable by tree transducers, for which the equivalence problems is decidable. Deterministic top-down tree transducers (DTops) are such a class, as shown by Engelfriet et al. (2009). Furthermore, DTops are subsumed by the more powerful class of macro tree transducers (Engelfriet and Vogler, 1985; Maneth et al., 2005), which were widely studied as formal core of a fragment of xslt. Even though macro tree transducers do not support xslt's variables that xslt uses to encode compositions, the decidability of their equivalence problem remains a long standing open question. One way to obtain decidable restrictions is to consider macro-tree transducers which cannot copy recursively. This class is closely related to the class of MSO-definable transformations (Engelfriet and Maneth, 2003, 2006).

When not wanting to restrict the copying power, the alternative is to remove macros from tree transducers. This leads to the usage of the DTop class, which extends subsequential transducers from words to trees by adding copying and flip operations. Furthermore, DTops have the advantage of admitting a unique normal form (Engelfriet et al., 2009). As shown by Lemay et al. (2010), these normal forms can be obtained from a Myhill-Nerode theorem. Furthermore, this leads to a learning algorithms in Gold's model with polynomial resources, as in the case of subsequential transducers. We assume that the domain of the target transformation is given as a parameter of the learning algorithm. The main limitation of the previous approach is that the domain must be a path-closed regular tree language, i.e. it must be definable by a top-down deterministic tree automaton. This limitation is inhibiting for many xml transformations, whose domain is given by a RelaxNG schema, since these are made to support general regular tree languages. Therefore, we study the question whether this restriction can be removed.

In this paper, we present a learning algorithm for DTop with regular domain inspection ($\text{DTopI}_{\text{reg}}$), which generalizes on the learning algorithm for DTop from (Lemay et al., 2010). In a first step, we show a semantic equivalence relation with finite index on these $\text{DTopI}_{\text{reg}}$. This requires a novel and nontrivial argument. Due to this result, we can obtain a new normal form for $\text{DTopI}_{\text{reg}}$ that we express with a new Myhill-Nerode theorem. We then lift the learning algorithm from (Lemay et al., 2010) so that it can account for regular domain inspections.

Running Example

Throughout this article we will consider an example based on the possible representation in data trees of articles, both physical and digital, in an XML style. A digital article has a website, and an URL. A physical article has a collection, and a book it is in. This representation presents some redundancy: the website can be infered from the URL, or the collection from the book. Furthermore, each article possesses a full identifier, that contains the type of article (digital/physical), but also the full information of the URL or book. Note

that since this paper deals with finite alphabet, the actual data will be represented with leaves. Specifically, websites and collections will be represented by the letters $\{a, a'\}$ while the additional information for the URL or book will be represented by the letters $\{b, b'\}$. The digital articles are of form: $article(fullid(digital, A, B), website(A), url(A, B))$, for $A \in \{a, a'\}$ and $B \in \{b, b'\}$. The physical articles are of form: $article(fullid(physical, A, B), collection(A), book(A, B))$. Note that this domain is regular but not path-closed.

We consider the transformation τ_{ref} that will document the type and website or collection. So, the image of $article(fullid(digital, a, b'), website(a), url(a, b'))$ will be $digital(a)$, while the image of $article(fullid(physical, a', b), collection(a'), book(a', b))$ will be $physical(a')$. We also consider the mapping of this function from lists of articles of form $cons(s_1, cons(s_2(...\ cons(s_n, nil)...)$ where $s_1, ..., s_n$ are article, to the list of their image by τ_{ref}. We call this transformation τ_{map}.

2. Top-Down Tree Transducers

Let F be a finite ranked alphabet. Given a finite set Q, we write $\mathcal{T}_F(Q)$ the set of all terms over F with variables in Q. The set of all ground terms $\mathcal{T}_F$ is the set of all terms over Σ without variables. For any $k \in \mathbb{N} \cup \{0\}$, we write $F^{(k)}$ for the set of symbols of F of rank k.

An F-path is a word with alphabet $F \cup \mathbb{N}$ in $\{fi \mid f \in F^{(k)},\ 1 \leq i \leq k\}^*$. An F-path u reaches a node of a tree s such that the empty path ε reaches the root of s, and if u reaches a node labelled by $f \in F^{(k)}$, ufi reaches its i-th child (for $1 \leq i \leq k$). We write $s \models u$ if the path u reaches a node of s and denote $u^{-1}s$ the subtree under u. For example, in the tree $s = f(g(a, b), h(a))$, the path $f1g2$ leads to the only leaf labeled by b. For u an F-path and a tree language $L \subseteq \mathcal{T}_F$, we define the residual $u^{-1}L$ as the set of all trees $u^{-1}s$ such that $s \in L$ and $s \models u$.

A tree language on F is a subset of $\mathcal{T}_F$. A tree language is called regular if it can be defined by a tree automaton (see for example Comon et al., 2007). An interesting property is that a tree language L is regular if and only if it has a finite number of distinct residuals. Given a partial function $\tau \subseteq \mathcal{T}_F \times \mathcal{T}_G$ and a pair $p = (u, v)$ of an F-path u and a G-path v, we define the p residual of τ, denoted by $p^{-1}\tau$, as the set of the pairs $(u^{-1}s, v^{-1}t)$ such that $(s, t) \in \tau$.

We study deterministic top-down tree transducers on ranked trees (Engelfriet, 1975). For this, 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 $X_k = \{x_1, \dots, x_k\}$. In particular, note that $X_0 = \emptyset$.

Definition 1 *A* deterministic top-down tree transducer *(*DTop*) is a tuple $M = (Q, F, G, ax, rhs)$ where Q is a finite set of* states*, F and G are ranked alphabets of* input and output symbols*, respectively, $ax \in \mathcal{T}_G(Q \times \{x_0\})$ is called the* axiom*, and rhs is a partial function from $Q \times F^{(k)}$ to $\mathcal{T}_G(Q \times X_k)$ for symbols of rank k. If $rhs(q, f) = t$, we note the corresponding rule $q(f(x_1, ..., x_k)) \to t$.*

Pairs (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 $s = f(s_1, \dots, s_k) \in \mathcal{T}_F$: $[\![M]\!]_q(f(s_1, \dots, s_k)) = \text{rhs}(q, f)[q'\langle x_i\rangle \leftarrow [\![M]\!]_{q'}(s_i) \mid q' \in Q,\ 1 \leq i \leq k]$ where $[q'\langle x_i\rangle \leftarrow [\![M]\!]_{q'}(s_i)]$ means that every occurrence of $q'\langle x_i\rangle$ is replaced by the tree $[\![M]\!]_{q'}(s_i)$.

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

The domain $dom([\![M]\!])$ of a transducer M is the set of trees s for which $[\![M]\!](s)$ is well-defined. It is folklore (Engelfriet, 1977) that this domain is accepted by some deterministic top-down tree automaton, and is thus regular. However, all DTop are not closed under domain specification. Furthermore, we want to specify a regular domain, which is more general than top-down languages.

In order to resolve this problem, we follow the approach of Engelfriet et al. (2009); Lemay et al. (2010), and extend DTops with domain inspection, with the difference that we will not only restrict ourselves to domain inspection by deterministic top-down tree automata.

Definition 2 *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* $D \subseteq \mathcal{T}_F$.

The semantics of a DTopI is defined by domain restriction, i.e., $[\![N]\!] = [\![M]\!]_{|D}$. Note that $dom([\![N]\!]) = dom([\![M]\!]) \cap D$. Hence, if D is a regular language, then $dom([\![N]\!])$ is also regular. A DTopI (M, D) such that D is regular is called a $\text{DTopI}_{\text{reg}}$.

Example 1 *Consider* τ_{ref} *presented earlier, with its domain* D *the set of all possible articles. It is described by the* DTopI $N_{ref} = (M_{ref}, D)$*, with three states* q_0, q_1, q_2*, where the axiom is* $ax = q_0\langle x_0\rangle$*, and the rules:*

$$\begin{array}{ll} (1) & q_0(article(x_1, x_2, x_3)) \to q_1\langle x_2\rangle \\ (2) & q_1(website(x_1)) \to digital(q_2\langle x_1\rangle) \\ (3) & q_1(collection(x_1)) \to physical(q_2\langle x_1\rangle) \end{array} \qquad \begin{array}{ll} (4) & q_2(a) \to a \\ (5) & q_2(a') \to a' \end{array}$$

To complete this DTopI *into* $N_{map} = (M_{map}, D')$ *that defines* τ_{map}*, the mapping of* τ_{ref} *on* D'*, the language of all lists of articles, we add a state* q_{ls}*, and build an axiom* $ax' = q_{ls}\langle x_0\rangle$*. We add the rules:*

$$(6) \quad q_{ls}(cons(x_1, x_2)) \to cons(q_0\langle x_1\rangle, q_{ls}\langle x_2\rangle) \qquad (7) \quad q_{ls}(nil) \to nil$$

3. Syntactic Alignments and Equivalence

We define a notion of syntactic alignment for DTopI, that states which paths of output trees are produced by which paths of input trees. This will allow the formulation of a first Myhill-Nerode like result.

Definition 3 *Let* $N = (M, D)$ *be a* DTopI*. We define judgements* $u \sim_q v$ *stating that an input path* u *is aligned to an output path* v *in state* q*, by the following inferences rules:*

- *If* $v^{-1}ax = q\langle x_0\rangle$*, then* $\varepsilon \sim_q v$.
- *If* $u \sim_q v$*, and* $rhs(q, f)$ *is defined, for* v' *such that* $v'^{-1}rhs(q, f) = q'\langle x_i\rangle$*, we have* $ufi \sim_{q'} vv'$.

To elaborate on our example τ_{ref}, in N_{ref}, $\varepsilon \sim_{q_0} \varepsilon$. After reading the first node *article* without production, we get that $article2 \sim_{q_1} \varepsilon$. The resulting residual $(article2, \varepsilon)^{-1}\tau_{ref}$ sends $collection(a)$ to $physical(a)$, for example.

We note that if $u \sim_q v$, then for $s \in dom([\![N]\!])$ with $s \models u$, we have $v^{-1}([\![N]\!](s)) = [\![M]\!]_q(u^{-1}s)$. In other words, $(u,v)^{-1}[\![N]\!] = [\![M]\!]_{q|_{u^{-1}dom([\![N]\!])}}$. If N is a $\text{DTOPI}_{\text{reg}}$, there are a finite number of states q and of residuals $u^{-1}dom([\![N]\!])$. This leads to this first Myhill-Nerode like result:

Proposition 4 *Let N be a DTOPI_{reg}. There is a finite number of distinct residuals $(u,v)^{-1}[\![N]\!]$ such that (u,v) is a pair syntactically aligned in N.*

We also note an interesting property: each node in the output comes from the axiom or exactly one node of the input.

Lemma 5 *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 $rhs(q', f) \models v''g$.*

4. Earliest Compatible Transducers

In order to normalize a DTOPI, we take two steps that Engelfriet et al. (2009); Lemay et al. (2010) make to normalize DTOPI with top-down inspection. The first step (compatible DTOPI) ensures that the inspection coincides exactly with the domain, and that each state corresponds to a specific residual of the domain. The second step (earliest DTOPI) ensures that a DTOPI produces its output as early as possible, a classical normalization step for transducers as in Choffrut (2003).

Definition 6 *A DTOPI $N = (M, D)$ is compatible if $D = dom([\![N]\!])$ and $dom(p^{-1}[\![N]\!])$ coincide for all pairs of paths p that are syntactically aligned in the same state of N.*

We saw in Section 3 that if $u \sim_q v$, then $(u,v)^{-1}[\![N]\!] = [\![M]\!]_{q|_{u^{-1}dom([\![N]\!])}}$. In a compatible DTOPI, we have that if (u', v') is also syntactically aligned in q, then $(u', v')^{-1}[\![N]\!] = (u,v)^{-1}[\![N]\!]$. In our example, N_{ref} and N_{map} are already compatible, but we can imagine an equivalent transducer that is not: by merging all four states of N_{map} together into one state q, we obtain a new DTOP M'_{map}. If M'_{map} on its own would be able to process trees that are not lists of articles, it remains true that a DTOPI $N'_{map} = (M'_{map}, dom(\tau_{map}))$ would be equivalent to N_{map}. However, its only state q would define all four residuals of N_{map}.

The second step of normalization (earliest DTOPI) ensures that a DTOPI produces its output as fast as possible. This is defined formally with the notion of largest common tree prefixes. For two trees $t, t' \in \mathcal{T}_G$ we define $g(t_1, \ldots, t_k) \sqcap g'(t'_1, \ldots, t'_{k'})$ their *largest common prefix tree* $t \sqcap t' \in \mathcal{T}_G(\{\bot\})$ inductively as $g(t_1 \sqcap t'_1, t_2 \sqcap t'_2, \ldots, t_k \sqcap t'_k)$ if $g = g'$, or $\bot$ otherwise. Note that $\sqcap$ operator can be easily lifted to non-empty sets of trees. For a transformation τ and an input path u that occurs in some tree $s \in dom(\tau)$, we can define the maximal output of τ at u, denoted $out_\tau(u)$, as $\bigsqcap\{\tau(s) \mid s \models u, s \in dom(\tau)\}$. This allows to define earliest transducers:

Definition 7 *A* DTopI $N = (M, D)$ *is* earliest *if for all pairs p syntactically aligned in some state of M, the residual $p^{-1}[\![N]\!]$ satisfies $out_{p^{-1}[\![N]\!]}(\varepsilon) = \bot$.*

In the example, N_{ref} and N_{map} are already earliest. But if we change the domain of N_{map} to apply only on *non empty* lists of articles, then $(\varepsilon, \varepsilon)^{-1}\tau_{map}$ would always start by a root *cons*, and we would need to change N_{map} to be earliest. To do this, the axiom should produce *cons* right away. The new axiom becomes $cons(q_{ls1}\langle x_0\rangle, q_{ls2}\langle x_0\rangle)$, where q_{ls1}, q_{ls2} are two new states:

$$(8)\quad q_{ls1}(cons(x_1, x_2)) \to q_1\langle x_1\rangle \qquad (9)\quad q_{ls2}(cons(x_1, x_2)) \to q_{ls}\langle x_2\rangle$$

Results from Engelfriet et al. (2009); Lemay et al. (2010) prove that all DTopI with a regular path-closed domain is equivalent to a compatible earliest DTopI. We show that this extends to $\text{DTopI}_{\text{reg}}$.

Proposition 8 *Every* DTopI_{reg} $N = (M, D)$ *is equivalent to some compatible earliest* DTopI_{reg} $N' = (M', D')$.

5. Semantic Alignments and Equivalence

We now introduce a semantic notion of aligned paths, defined on transformations rather than on transducers. This leads us to our Myhill-Nerode like theorem.

Definition 9 *A pair $p = (u, v)$ is said to be* (semantically) aligned for a partial function τ *if $v^{-1}out_\tau(u) = \bot$ and the residual $p^{-1}\tau$ is a partial function.*

Semantically aligned pairs can be seen as candidates to be syntactically aligned pairs in earliest $\text{DTopI}_{\text{reg}}$. However, not all are syntactically aligned.

For example in τ_{ref}, $(article2, \varepsilon)$ is both semantically and syntactically aligned. However, due to the redundancy of information under the three attributes of *article*, pairs $(article1, \varepsilon)$ and $(article3, \varepsilon)$ are semantically aligned with a functional residual, but are not syntactically aligned for N_{ref}.

We now present the Myhill-Nerode type theorem our normal form will be based on. If Proposition 4 shows that every DTopI has a finite number of residuals from syntactically aligned pairs, we now prove that every DTopI tranformation has a finite number of residuals from semantically aligned pairs.

Theorem 10 *Let N be a* DTopI_{reg}. *There is a finite number of distinct residuals $(u, v)^{-1}[\![N]\!]$ such that (u, v) is a pair semantically aligned in $[\![N]\!]$.*

Proof Since this theorem depend entirely on the semantics of N, Proposition 8 allows us to assume that N is earliest w.l.o.g. Therefore, all its syntactically aligned pairs are semantically aligned, and Proposition 4 ensures that they define a finite number of residuals. We will prove that there are only finitely many other semantically aligned pairs' residuals.

Imagine a semantically aligned pair $p = (u, v)$ that is not syntactically aligned and take a tree s with $s \models u$. Consider the syntactic alignment that produces the node under v

when N computes $[\![N]\!](s)$. 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 in state q (see Proposition 5).

We start by asserting that if u and u' are on the same branch of s, then $u = u'$ and $v = v'$. Indeed, if u is shorter than u', then N does not produce the node under v as early as possible, but we supposed N earliest. Conversely, if u' is shorter than u, then N produces the node under v faster than possible, which leads to a contradiction. Since we know that p is not syntactically aligned and thus not p', we conclude that u and u' are disjoint.

Now, by definition, $p^{-1}[\![N]\!]$ is functional. Since p' is syntactically aligned, $p'^{-1}[\![N]\!]$ is functional. However, v' is a prefix of v. This means that if $[\![N]\!](s) \models v$, then $v^{-1}[\![N]\!](s)$ depends functionally on what is under u and u'. This is not a contradiction in itself, but it limits what the residual $p^{-1}[\![N]\!]$ can be. Indeed, if we replace the subtree under u in s with another one without changing the subtree under u', then $v^{-1}[\![N]\!](s)$ does not change.

The idea goes as follow: consider A to be the DBTA that recognizes $dom([\![N]\!])$. We note r the state of A such that the tree $u^{-1}s$ is labeled by state r in a run of A. We note $[\![A]\!]_r$ the set of all trees labeled by r. We can substitute $u^{-1}s$ by any tree in $u \in [\![A]\!]_r$ without changing $u'^{-1}(s)$, and thus, without changing $v'^{-1}[\![N]\!](s)$. Hence, $p^{-1}[\![N]\!]$ is constant on each $[\![A]\!]_r$.

Furthermore, its possible values are limited. We pick one tree s_r for each state r of A, and prove that all images of $p^{-1}[\![N]\!]$ are subtrees of some $[\![N]\!]_q(s_r)$. Since v' is a prefix of v, we have that $p^{-1}[\![N]\!](u^{-1}s)$ is a subtree of $p'^{-1}[\![N]\!](u'^{-1}s)$. Since p' is syntactically aligned in a state q, this means $p^{-1}[\![N]\!](u^{-1}s) = [\![N]\!]_q(u'^{-1}s)$. As we also mentioned, if r' is the state of A such that $u'^{-1}s \in [\![A]\!]_{r'}$, then we can replace $u'^{-1}s$ by any tree of $[\![A]\!]_{r'}$ without changing $u^{-1}s$. Hence, $p^{-1}[\![N]\!](u^{-1}s)$ is a subtree of $[\![N]\!]_q(s_{r'})$.

To summarize, if p is semantically aligned, but not syntactically aligned in any state q, then $p^{-1}[\![N]\!]$ has at most one value per L_r (finite number of states). Each of these values is a subtree of a $[\![N]\!]_q(s_r)$ (finite number of q, t_r and v''). This means there is only a finite number of options available for functions $p^{-1}[\![N]\!]$. Hence there are finitely many different $p^{-1}[\![N]\!]$. ■

6. Unique Normal Forms

If in Engelfriet et al. (2009); Lemay et al. (2010) compatible earliest DTopI were enough to establish a unique minimal normal form, this is not the case for $\text{DTopI}_{\text{reg}}$. This comes from the fact that as shown in previous examples, some information redundancy can offer a choice on several ways to produce some of the output.

As we already saw in Section 5, in τ_{ref}, $(article3, \varepsilon)$ is semantically aligned, but not syntactically aligned in N_{ref}. We could create an alternate N'_{ref} that visits $(article3, \varepsilon)$ instead of $(article2, \varepsilon)$ by changing some of the rules of N_{ref}:

(1) $q_0(article(x_1, x_2, x_3)) \to q_1\langle x_3\rangle$
(2) $q_1(url(x_1, x_2)) \to digital(q_2\langle x_1\rangle)$
(3) $q_1(book(x_1, x_2)) \to physical(q_2\langle x_1\rangle)$
(4) $q_2(a) \to a$
(5) $q_2(a') \to a'$

Also, note that some residuals $p^{-1}\tau$ are functional, but cannot be defined by some DTopI, that is to say that there is no DTopI N' such that $[\![N']\!] = p^{-1}\tau$. Such a residual cannot

be used in a DTopI. For example, in τ_{ref}, if $(article1, \varepsilon)$ is semantically aligned, its residual is however impossible to define with a DTopI: it would require to visit both its first and second attribute to produce the same output branch.

When there is a choice, we decide to always choose the leftmost option.

Definition 11 *An earliest compatible* DTopI *is* leftmost *if for every syntactically aligned pair $p = (ufi, v)$, there is no $j < i$ such that the residual $(ufj, v)^{-1}\tau$ is definable by some* DTopI.

For a pair $p = (u, v)$, we define $ind_\tau^{\text{LEFT}}(p, f, v')$ as the smallest index i such that $(ufi, vv')^{-1}\tau$ is definable by some DTopI.

We now characterize a *pre-canonical* form $can'(\tau)$ for a transformation τ.

Definition 12 *Let τ be a transformation defined by a* DTopI_{reg}. *We define its* pre-canonical form $can'(\tau)$ *as the pair $(M, dom(\tau))$, where $M = (Q, F, G, ax, rhs)$ such that $Q = \{[p]_\tau \mid p^{-1}\tau$ is definable by some* DTopI$\}$, *its axiom is $ax = out_\tau(\varepsilon)[v \leftarrow [(\varepsilon, v)]_\tau\langle x_0\rangle \mid v^{-1}out_\tau(\varepsilon) = \bot]$, for all $p = (u, v)$ such that $[p]_\tau \in Q$ and $f \in G$ such that $uf^{-1}dom(\tau) \neq \emptyset$, $rhs([p]_\tau, f) = out_{p^{-1}\tau}(f)[v' \leftarrow [(ufi, vv')]_\tau\langle x_i\rangle \mid v'^{-1}out_{p^{-1}\tau}(f) = \bot$ and $i = ind_\tau^{\text{LEFT}}(p, f, v')]$, where $t[v \leftarrow t']$ means that the subtree of t under v is replaced by t'.*

Each state $[p]_\tau$ produces $p^{-1}\tau$ in an earliest manner. Furthermore, the choice of $x_{ind_\tau^{\text{LEFT}}(p,f,v')}$ in the rules of $can'(\tau)$ ensures that the resulting transducer is leftmost. To make sure no state of our normal form is inaccessible, we finally "trim" $can'(\tau)$, to obtain $can(\tau)$, a canonical normal form for τ.

Definition 13 *Le N a* DTopI_{reg}, *where $\tau = [\![N]\!]$ and $can'(\tau) = (M', D)$, where $M' = (Q', F, G, ax', rhs')$. We define its* canonical form $can(\tau)$ *of states $Q = \{q \in Q' \mid \exists (u, v)$ syntactically aligned in $q\}$, axiom $ax = ax'$ and rules $rhs = rhs'_{|Q\times F}$.*

Proposition 14 *For any transformation τ definable by some* DTopI_{reg}, $can'(\tau)$ *and* $can(\tau)$ *are* DTopI_{reg} *defining τ that are compatible, earliest, and leftmost.*

7. Learning from Examples

We next show how to infer the canonical $\text{DTopI}_{\text{reg}}$ under the condition that the domain is known a priori. We place ourselves in a Gold setting (Gold, 1978), i.e. learning from sample where a sample is a finite partial function in $S \subseteq \mathcal{T}_F \times \mathcal{T}_G$, i.e. a set of examples of the transformation. Like most Gold-model learning algorithm, we need to prove the existence of a *learning function*, which deduces a machine from a sample, and conversely, of a *characteristic sample function*, which maps each machine to a minimal sample required by the learning function to find the proper machine.

Theorem 15 *For the class of* DTop *transformations with regular inspection represented by* DTopI_{reg}, *there exist:*

- *$learn_D$ a partial function that maps samples compatible with a domain D to a* DTopI_{reg} *in normal form, and*
- *char a function that maps a* DTopI_{reg} *N in normal form to a sample of $[\![N]\!]$.*

such that for any DTOPI_{reg} N*, and any sample* S *of* $[\![N]\!]$ *such that* $char(N) \subseteq S$*, then* $learn_D(S) = N$.

This theorem shows that the class of DTop transformations with regular inspection represented by $\text{DTOPI}_{\text{reg}}$ is *learnable* in a Gold setting. We also want to consider the complexity in time and data, i.e. the number of examples in $char(N)$, and the complexity of the learning algorithm $learn_D$ as a function of the size of its input sample. This data complexity is not polynomial in the size of N, but in the (potentially exponentially bigger) number of residuals of $[\![N]\!]$.

The learning algorithm is a generalisation of the one by Lemay et al. (2010), which is itself inspired by the RPNI algorithm, or more directly, by the OSTIA algorithm (Oncina et al., 1993) to learn deterministic string transducers. The main difference between this algorithm and previous implementations of RPNI-like algorithms comes from the fact that regular languages allow for some redundancy in the information; as mentioned in Section 6, not all explored residuals will end up producing a state in the final DTOPI in normal form.

We do not describe in detail the algorithm, but rather gives the main ideas and run it on τ_{map}. To make notations more compact, we note $s_{\phi,A,B}$ the physical article of collection A and book A, B, and $s_{d,A,B}$ the digital article of website A and URL A, B. For example, $s_{\phi,a',b} = article(fullid(\ physical, a', b), collection(a'),\ book(a', b))$, and $s_{d,a,b'} = article(fullid(digital, a, b'),\ website(a),\ url(a, b'))$. Furthermore, when we say that a sample S 'contains' a tree $s \in dom(\tau_{map})$, we want to say that it contains the pair $(s, \tau_{map}(s))$.

The main idea, as in most RPNI-like algorithms, is to deal with two sets of states that we call 'definitive' states and 'candidate' states, where each state being represented by pairs $p = (u, v)$. Definitive states are created whenever we encounter a pair that defines a yet unencountered residual $p^{-1}\tau$. They are the only states to have rules. Candidate states on the other hand have to be inspected by the algorithm and will be either consider equivalent to a definitive state, and merged with it, or transformed into a definitive state. Note that, like in Lemay et al. (2010) but unlike RPNI, candidate states are built 'on the fly', and not derived from an initial automaton (such as the prefix tree for RPNI), but the main loop is otherwise similar.

The algorithm starts by building the axiom of the target transducer, and a first set of candidate states. For this consider $out_S(\varepsilon)$ as the skeleton of the axiom, and take any v such that $v^{-1}out_S(\varepsilon) = \bot$. The $p = (\varepsilon, v)$ thus obtained are all candidate states, with $ax = out_S(\varepsilon)[v \leftarrow (\varepsilon, v)\langle x_0\rangle \mid v^{-1}out_S(\varepsilon) = \bot]$.

In τ_{map}, we only need two examples to prove that $out_{\tau_{map}}(\varepsilon) = \bot$. Indeed, if our sample S contains *nil* and $cons(s_{d,a,b}, nil)$, for example, then it knows that τ_{map} can begin by *cons* or *nil*. We create the axiom $ax = p_0\langle x_0\rangle$ with the new candidate state $p_0 = (\varepsilon, \varepsilon)$.

The algorithm then loop by checking every candidate state $p = (u, v)$ in proper order. It first looks for a definitive state p' which is equivalent on the sample, i.e. such that p and p' share the same domain ($p^{-1}D = p'^{-1}D$) and that do not contradict each other on the sample, as there is no tree t such that $p^{-1}S(t) \neq p'^{-1}S(t)$. If p and p' are equivalent on the sample, then they are merged, which simply means that p is replaced by p' in every right-hand side.

Otherwise, p is made a new definitive state. This implies in particular to build all the rules with p, f as left hand side with all symbols f that can be present in the domain (i.e. $uf^{-1}D \neq \emptyset$). These rules are built just like the axiom was. First, we consider the tree

$t_{p,f} = u^{-1}out_S(u.f)$. For each v' with $v'^{-1}t_{p,f} = \bot$, one needs to find the state and the variable x_i to put there. This relies on finding the leftmost index $ind_\tau^{\text{LEFT}}(p, f, v')$. However, testing if a residual is definable by a DTopI is difficult, while finding all indexes that lead to a functional residual. To solve this problem, we do not make the choice right away. Instead of choosing a single index, we create some 'proto'-rules that can contain sets of calls to candidate states.

(p_0): In our example, we start with $p_0 = (\varepsilon, \varepsilon)$. Since p_0 is the first candidate state, it is necessarily original, i.e. there exists no definitive state p such that $p_0^{-1}\tau_{map} = p^{-1}\tau_{map}$. We create its two output rules, for symbols *cons* and *nil*. For *nil*, (nil, nil) is enough to see that $out_{p_0^{-1}\tau_{map}}(nil) = nil$ and to create the rule $p_0(nil) \to nil$. For *cons*, We use examples $cons(s_{d,a,b}, nil)$, $cons(s_{d,a,b}, nil)$ and $cons(s_{\phi,a,b}, cons(s_{\phi,a,b}, nil))$ to show that $out_{p_0^{-1}\tau_{map}}(cons) = cons(\bot, \bot)$. We now need to see which *consi* in the input is functional for *cons*1 and *cons*2 in the output. Examples $cons(s_{d,a,b}, nil)$ and $cons(s_{\phi,a,b}, nil)$ indicate that a change under *cons*1 can occur in the output with no change under *cons*2 in the input. Similarly, $cons(s_{d,a,b}, nil)$ and $cons(s_{\phi,a,b}, cons(s_{\phi,a,b}, nil))$ show that $(cons2, cons1)$ cannot be a functional residual. We create two candidate pairs $p_1 = (cons1, cons1), p_2 = (cons2, cons2)$, and the rule: $p_0(cons(x_1, x_2)) \to cons(p_1\langle x_1\rangle, p_2\langle x_2\rangle)$.

(p_1): p_1 has not the same domain residual as p_0 and thus becomes the second definitive state. It can only read the label *article* and examples $cons(s_{d,a,b}, nil)$, $cons(s_{\phi,a,b}, nil)$ show that $out_{p_1^{-1}\tau_{map}}(article) = \bot$. However, all three attributes of *article* provide functional residuals. We do not pick one for now and create all three candidates $p_{art1} = (cons1article1, cons1)$, $p_{art2} = (cons1article2, cons1)$ and $p_{art3} = (cons1article3, cons1)$. We have the proto-rule with several choices: $p_1(article(x_1, x_2, x_3)) = \{p_{art1}\langle x_1\rangle, p_{art2}\langle x_2\rangle, p_{art3}\langle x_3\rangle\}$.

(p_2): p_2 has the same residual domain as p_0. Also, since $p_0^{-1}\tau_{map} = p_2^{-1}\tau_{map}$, there will not be any counterexample to differentiate p_0 from p_2. p_2 is merged with p_0 and the rule now reads: $p_0(cons(x_1, x_2)) \to cons(p_1\langle x_1\rangle, p_0\langle x_2\rangle)$.

(p_{art1}): p_{art1} has an original residual and creates a definitive state. When reading *fullid* we have $out_{p_{art1}^{-1}\tau_{map}}(fullid) = \bot$. However, if S contains $cons(s_{d,a,b}, nil)$ and $cons(s_{d,a',b}, nil)$ it proves that neither the first nor the third attribute of *fullid* give a functional residual. If S contains $cons(s_{d,a,b}, nil)$ and $cons(s_{\phi,a,b}, nil)$, we conclude that neither does the second attribute. Hence p_{art1} has no next step available: we create the proto-rule $p_{art1}(fullid(x_1, x_2, x_3)) \to \emptyset$.

In the rest of the algorithm p_{art2} and p_{art3} become definitive states, and create four successors, all equivalent to $p_{data} = (cons1article2website1, cons1digital1)$. We add the following rules:

(1) $p_{art2}(website(x_1)) \to digital(p_{data}\langle x_1\rangle)$
(2) $p_{art2}(collection(x_1)) \to physical(p_{data}\langle x_1\rangle)$
(3) $p_{art3}(url(x_1, x_2)) \to digital(p_{data}\langle x_1\rangle)$
(4) $p_{art3}(book(x_1, x_2)) \to physical(p_{data}\langle x_1\rangle)$
(5) $p_{data}(a) \to a$
(6) $p_{data}(a') \to a'$

The algorithm continues until all candidate states have been checked, and either been merged or added to the definitive states. A final step is needed to obtain a proper DTop. The first problem is that some states of the proto-DTop have some empty call sets. These are states for which a position in a right-hand side has found no candidate states. They are

simply recursively erased. Here, p_{art1} needs to be deleted. The proto-rule from p_1 is now: $p_1, article(x_1, x_2, x_3) = \{p_{art2}\langle x_2\rangle, p_{art3}\langle x_3\rangle\}$.

The final step to obtain a DTop consist essentially to go through all rules of the proto-DTop, check the positions of the right-hand side that has several states, and keep only the leftmost one. In the example, the proto-rule $rhs(p_1, article)$ has a choice between $p_{art2}\langle x_2\rangle$ and $p_{art3}\langle x_3\rangle$, we pick $p_{art2}\langle x_2\rangle$ and the rule is now: $p_1, article(x_1, x_2, x_3) = p_{art2}\langle x_2\rangle$.

The final result needs to be trimmed to obtain $can(\tau)$: in our example the state p_{art3} is now inaccessible and can be deleted. The final result is a DTopI identical to N_{map} up to state renaming.

It remains to describe the size of the required sample. For every pair p that is eventually going to pass as a candidate, we need enough of a sample to differentiate it from every definitive state it is not equivalent to. Furthermore, for every pair p that is eventually going to become a definitive state, we need enough of a sample to find $out_{p^{-1}\tau}(f)$ for each label it can read, as well as enough information to find the indexes that allows to create the next functional residuals. The construction is essentially similar to the one described in Lemay et al. (2010), with the difference that the number of such tests is bounded by the number of different semantic residuals. If Theorem 10 shows that this number is finite, it can be exponentially bigger than $can(\tau)$. Hence, it is worth noting that the polynomiality of the size of the characteristic sample is in relation to the number of residuals of τ, not in the size of $can(\tau)$.

8. Conclusions

We proved a Myhill-Nerode type theorem for top-down tree transducers with regular domain. This theorem led to a leftmost normal form. We have devised a RPNI-like learning algorithm for Gold-style sample-based learning.

A possible extension of this result would be to use this algorithm as a core in an Angluin-style (Angluin, 1987) student-teacher interactive learning model, similar to Carme et al. (2007).

Are there other classes beyond DTopI to which our results could be extended? An interesting and robust class is the class of top-down tree transducers with regular look-ahead (Engelfriet, 1977). 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. Although equivalence has been proven to be decidable (Engelfriet et al., 2009), it is unclear how to find a minimal look-ahead for a given transformation. The corresponding result on words for normal form Elgot and Mezei (1965) and a fortiori a learning algorithm (Boiret et al., 2012) are nontrivial and do not adapt easily to tree transformations.

References

D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2): 87–106, 1987. doi: 10.1016/0890-5401(87)90052-6.

A. Boiret, A. Lemay, and J. Niehren. Learning rational functions. In *DLT 2012*, p. 273–283.

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

C. Choffrut. Minimizing subsequential transducers: a survey. *TCS*, 292(1):131–143, 2003.

H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 2007.

C. C. Elgot and J. E. Mezei. On relations defined by generalized finite automata. *IBM J. Res. Dev.*, 9(1):47–68, January 1965.

J. Engelfriet. Bottom-up and top-down tree transformations - A comparison. *Mathematical Systems Theory*, 9(3):198–231, 1975.

J. Engelfriet. Top-down tree transducers with regular look-ahead. *Mathematical Systems Theory*, 10:289–303, 1977.

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

J. Engelfriet and S. Maneth. The equivalence problem for deterministic MSO tree transducers is decidable. *Inf. Process. Lett.*, 100(5):206–212, 2006.

J. Engelfriet and H. Vogler. Macro tree transducers. *J. Comput. Syst. Sci.*, 31(1):71–146, 1985.

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

E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978.

A. Lemay, S. Maneth, and J. Niehren. A learning algorithm for top-down XML transformations. In *PODS 2010*, p. 285–296.

S. Maneth, A. Berlea, T. Perst, and H. Seidl. XML type checking with macro tree transducers. In *SIGMOD 2005*, p. 283–294.

J. Oncina, P. García, and E. Vidal. Learning subsequential transducers for patt. recognition interpretation tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5):448–458, 1993.