Equivalence of Symbolic Tree Transducers

Vincent Hugot, Adrien Boiret, Joachim Niehren

Equivalence of Symbolic Tree Transducers

Vincent Hugot[23], Adrien Boiret[23], and Joachim Niehren[13]

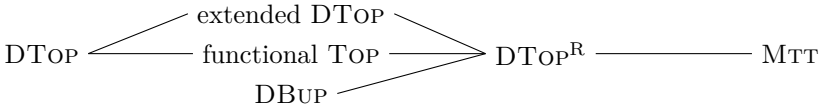
[1] Inria, Lille, France
[2] University of Lille, France
[3] Links project (Inria & Cristal Lab - Umr Cnrs 9189)

Abstract. Symbolic tree transducers are programs that transform data trees with an infinite signature. In this paper, we show that the equivalence problem of deterministic symbolic top-down tree transducers (DTop) can be reduced to that of classical DTop. As a consequence the equivalence of two symbolic DTop can be decided in NExpTime, when assuming that all operations related to the processing of data values are in PTime. This result can be extended to symbolic DTop with lookahead and thus to deterministic symbolic bottom-up tree transducers.

1 Introduction

Data trees are widely used in various domains of computer science. They represent programs in compiler construction or program analysis, syntactic sentence structure in computational linguistics, all or part of the database instances in semi-structured databases, and structured documents in document processing. The most widely used current formats for data trees are json (the Java Script Object Notation) and xml (the eXtensible Markup Language).

We are interested in deciding the equivalence of programs that define transformations on data trees. For instance, we may consider xslt programs defining xml transformations or Linux installation scripts written in `bash` that change the file system tree. Our approach is to compile a subclass of such programs into classes of tree transducers for which equivalence is decidable. Here we present a partial landscape of classical classes of tree transducers without data values [7, 8, 12], where inclusion is read from left to right:



The class DTop^{R} of deterministic top-down tree transducers with regular lookahead by a deterministic bottom-up tree automaton is particularly well behaved [8]. It is closed under composition, which makes it suitable for compilation of programs, and its equivalence problem is decidable in NExpTime, by PTime

This work has been partially supported by CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020 and the ANR project Colis, contract number ANR-15-CE25-0001-01

reduction to the equivalence problem of the class DTop, for which equivalence is decidable in NExpTime [14, 11, 17]. Furthermore, the class DTop^{R} subsumes three other classes of tree transducers with pairwise incomparable expressiveness, which capture different aspects of programs: extended DTop with nested pattern matching, functional top-down tree transducers (functional Top) that relax the determinism requirement of DTop, and deterministic bottom-up tree transducers DBup that operate the other way around. The more general class of macro tree transducers (Mtt) is much more expressive (and includes lookaheads), but has a long-standing open equivalence problem [9, 10, 18] and fails to be closed under composition (though its linear size increase subclass has better properties).

In contrast to classical machines that operate on ranked trees over finite signatures, what we need for program verification are generalised machines that operate on data trees with infinite signatures. Most typically the data values which label the nodes of data trees may be strings over some finite alphabet or natural numbers. For dealing with data trees, the classical classes of tree transducers were extended to symbolic classes [20, 19, 15], and similarly for other kinds of finite state machines. The general idea is to use patterns for describing infinitely many data values in a finite manner, and to allow the transducers to apply transformations on the data values themselves.

We first illustrate by an example that the class of symbolic extended DTop is relevant in practice. For this, we consider the following extended DTop, which performs a routine cleanup and statistics task on a list of log files in a file system, as illustrated in the example of Figure 1.

$$q\langle \text{NIL}\rangle \to \text{CONS}\langle\ \text{FILE}\langle \text{“log”}, \text{“”}\rangle,\ \text{NIL}\rangle \tag{1}$$

$$q\langle \text{CONS}\langle x_1, x_2\rangle\rangle \to \text{CONS}\langle q_{\text{name}}\langle x_1\rangle, q\langle x_2\rangle\rangle \tag{2}$$

$$q_{\text{name}}\langle \text{FILE}\langle \text{“log”}, x_2\rangle\rangle \to \text{FILE}\langle \text{“stats.1”},\ f_{\text{stats}}@x_2\rangle \tag{3}$$

$$q_{\text{name}}\langle \text{FILE}\langle x_1 : \text{“stats.”}(\text{“0”}..\text{“9”})^+,\ x_2\rangle\rangle \to \text{FILE}\langle f_{\text{incr}}@x_1,\ f_{\text{id}}@x_2\rangle \tag{4}$$

Such transducers have nested patterns with variables $x_1, x_2, \ldots$ for matching subtrees, expressions for matching data values such as $\text{“stats.”}(\text{“0”}..\text{“9”})^+$ or cons, and applications of externally defined functions, such as $f_{\text{stats}}@x_2$, where f_{stats} is a string transformation that produces statistics from its input string (log contents). It should be noted that symbolic functional Top are insufficient for this example since rules such as (3) and (4), with nested patterns, cannot be expressed in a top-down manner. In contrast, symbolic DBup offer an alternative solution for this concrete example.

Veanes and Bjørner [20, 19] started the study of symbolic transducers. They showed that equivalence is decidable for symbolic functional Top, if the corresponding problems on data pattern and transformations are. In this paper, we notice that the landscape of tree transducers above remains unchanged when turning classes of classical tree transducers symbolic. Therefore, we can show that the equivalence problem is decidable for the symbolic counterparts of all classes in the landscape except for Mtt. To see this, note that any symbolic DTop is a symbolic functional Top, so equivalence for symbolic DTop is decidable.

Furthermore, equivalence for symbolic DTOP$^{\mathrm{R}}$s can be reduced to equivalence of symbolic DTOP as in the classical case.

We then start studying the complexity of equivalence for classes of symbolic tree transducers. Our main result is that equivalence for symbolic DTOP$^{\mathrm{R}}$s is in NEXPTIME, under the assumption that operations on patterns and data transformations can be performed in PTIME. If not, one needs to multiply the worst case exponential time with the maximal time needed for such operations. We obtain this result from a novel reduction from the equivalence of symbolic DTOP to the equivalence of classical DTOP, using a weakened version of the *origin-equivalence* in [3]. This reduction allows us to conclude that the equivalence problem of symbolic DTOP is indeed in NEXPTIME as for classical DTOP (and not in 2NEXPTIME as a naive analysis would lead to believe). Due to the modularity of the construction, the equivalence testers obtained for DTOP$^{\mathrm{R}}$s are easy to prove correct, to analyse, and to implement.

2 Tree Automata and Transducers

Some familiarity with formal languages and automata theory, as covered for instance in [5], is assumed.

Given a set S, we denote its cardinality by $|S|$ and its powerset by 2^S. The set of Boolean values is written $\mathbb{B} = \{0, 1\}$. $\mathbb{N}$ is the set of natural integers, including zero. We write $m..n$ the integer interval $[m, n] \cap \mathbb{N}$. We will denote tuples $(a_0, a_1, \ldots, a_n)$ by $a_0\langle a_1, \ldots, a_n\rangle$ or simply as a_0 if $n = 0$.

Let $X = \{x_1, x_2, x_3, \ldots\}$ be a set of **variables**. For $K > 0$, we shall often use the subsets $X_K = \{x_1, \ldots, x_K\}$. A **ranked alphabet** Σ is a (potentially infinite) set disjoint from X paired with a function $ar_\Sigma : \Sigma \to \mathbb{N}$ (or just ar where Σ is clear from context). The set of **ranked trees over Σ with variables in X**, denoted by $\mathcal{T}_\Sigma(X)$, is the least set that contains X and all $a\langle t_1, \ldots, t_n\rangle$ where $a \in \Sigma,\ n = ar_\Sigma(a),\ t_1, \ldots, t_n \in \mathcal{T}_\Sigma(X)$. $\mathcal{T}_\Sigma(\varnothing)$ is the set of **ground** Σ-trees, also written $\mathcal{T}_\Sigma$. Notions of position, substitution, etc., are all defined as usual.

We next recall the definitions of deterministic top-down tree automata (DTTA) and deterministic top-down tree transducers (DTOP).

Definition 1. A **quasi** DTTA is a tuple $A = (\Sigma, Q, q_{\mathrm{ini}}, rhs)$ such that Σ is a ranked alphabet, Q a finite set, $q_{\mathrm{ini}} \in Q$, and rhs is a partial function that maps pairs $(q, a) \in Q \times \Sigma$ to tuples of $Q^{ar(a)}$. A DTTA is a quasi DTTA for which Σ is finite, and thus so is rhs.

The elements of Q are called the states of A, q_{ini} its initial state. The rules of A have the form $q\langle a\langle x_1, \ldots, x_n\rangle\rangle \to rhs(q, a)$, where $rhs(q, a)$ is defined and $n = ar(a)$. Each state q of a quasi DTTA A recognises a tree language $[\![q]\!]_A \subseteq \mathcal{T}(\Sigma)$, (or just $[\![q]\!]$ when A is clear from the context) defined by induction on the trees: we have $a\langle t_1, \ldots, t_n\rangle \in [\![q]\!]$ iff there is a rule $q\langle a\langle x_1, \ldots, x_n\rangle\rangle \to q_1, \ldots, q_n$ and $t_k \in [\![q_k]\!]$ for all $k \in 1..n$. The semantics of the automaton is $[\![A]\!] = [\![q_{\mathrm{ini}}]\!]$.

Bottom-up tree automata (BUTA) are defined similarly and as usual, with rules of the form $a\langle q_1, \ldots, q_n\rangle \to q$.

Definition 2. A **quasi** DTop is a tuple $M = (\Sigma, \Delta, Q, ax, rhs)$ such that Σ and Δ are ranked alphabets, Q is a finite set, $q_{\text{ini}} \in Q$, and rhs is a partial function that maps pairs $(q, a) \in Q \times \Sigma$ to $\mathcal{T}_\Delta(Q \times X_{ar(a)})$. A DTop is a quasi DTop for which Σ and Δ are finite, and thus rhs as well.

Σ and Δ are called input and output alphabets; the other components are as in automata. Each state $q \in Q$ has as semantics a partial function $[\![q]\!]$ from $\mathcal{T}_\Sigma$ to $\mathcal{T}_\Delta$, defined by induction on terms $t = a\langle t_1, \ldots, t_n\rangle \in \mathcal{T}_\Sigma$ such that:

$$[\![q]\!](t) \quad = \quad rhs(q, a)\big[q'\langle x_k\rangle \leftarrow [\![q']\!](t_k) \;\big|\; q' \in Q,\ k \in 1..\,ar(a)\big]. \tag{5}$$

The transformation defined by M is the partial function $[\![M]\!] = [\![q_{\text{ini}}]\!]$.

3 Symbolic Tree Automata and Transducers

In this section, we recall the definitions of symbolic Dtta and symbolic DTop as in [15]. Symbolic machines are finite representations of potentially infinite quasi Dtta and quasi DTop. They use **descriptors** to stand for the potentially infinite sets and functions. Given a set S, we call a set D paired with a function $[\![.]\!] : D \to S$ as set of descriptors of elements of S. For instance, we can use the set E of regular expressions $e \in E$ over an alphabet A as descriptors of regular languages $[\![e]\!] \subseteq A^*$. Outside of the definitions, we shall often assimilate the descriptors and their semantics.

Definition 3. A **symbolic** Dtta is a tuple $A = (\Sigma, \Phi, Q, q_{\text{ini}}, rhs)$ such that $(\Phi, Q, q_{\text{ini}}, rhs)$ is a quasi Dtta with a finite set of rules, Φ is an alphabet of descriptors for subsets of the alphabet Σ, with $ar(a) = ar(\varphi)$ for any $a \in [\![\varphi]\!]$ and $\varphi \in \Phi$. For all $a \in \Sigma$ and $q \in Q$, there exits at most one $\varphi \in \Phi$ such that $rhs(q, \varphi)$ is defined and $a \in [\![\varphi]\!]$.

The elements of $\varphi \in \Phi$ are called (descriptors for) guards. A symbolic Dtta A is a finite representation of a (potentially) infinite quasi Dtta A' such that for every rule r of form, $q\langle\varphi\langle x_1, \ldots, x_n\rangle\rangle \to q_1, \ldots, q_n$ of A, and for every $a \in [\![\varphi]\!]$, there is a rule $q\langle a\langle x_1, \ldots, x_n\rangle\rangle \to q_1, \ldots, q_n$ in A'. The semantics of A is defined as that of A': for all $q \in Q$, $[\![q]\!]_A = [\![q]\!]_{A'}$. Symbolic Buta are defined similarly.

Definition 4. A symbolic Dtta is **effective** if it satisfies the following conditions, which we always assume: **(1)** The set of guards Φ is closed under conjunction and negation, i.e., there exists an algorithm computing some function $\wedge : \Phi \times \Phi \to \Phi$ such that $[\![\varphi \wedge \varphi']\!] = [\![\varphi]\!] \cap [\![\varphi']\!]$ for all $\varphi, \varphi' \in \Phi$, and an algorithm computing some function $\neg : \Phi \to \Phi$ such that $[\![\neg\varphi]\!] = \Sigma \setminus [\![\varphi]\!]$. **(2)** There exists an algorithm deciding membership $a \in [\![\varphi]\!]$ given a guard $\varphi \in \Phi$ and a label $a \in \Sigma$,

Definition 5. A **symbolic** DTop is a tuple $M = (\Sigma, \Delta, \Phi, \mathcal{F}, Q, q_{\text{ini}}, rhs)$ such that $(\Phi, \mathcal{F}, Q, q_{\text{ini}}, rhs)$ is a quasi DTop with a finite set of rules, $\mathcal{F}$ is an alphabet of descriptors for partial functions from the input alphabet Σ to the output alphabet Δ, with $ar([\![f]\!](a)) = ar(f)$ for every $a \in \mathrm{dom}(f)$ and $f \in \mathcal{F}$. The same conditions on Σ, Φ and rhs apply as for symbolic Dtta above.

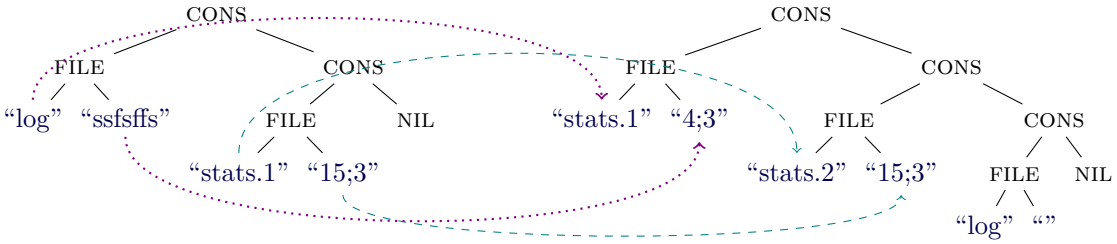


Fig. 1. Log cleanup and statistics: input tree on the left, output on the right.

The elements of $f \in \mathcal{F}$ are called (descriptors for) data transformations. A symbolic DTop M is a finite representation of a (potentially) infinite quasi DTop $M' = (\Sigma, \Delta, Q, ax, rhs')$, such that for every rule $q\langle\varphi\langle x_1, \ldots, x_n\rangle\rangle \to rhs(q, \varphi)$ of M, and for every $a \in [\![\varphi]\!]$, there is a rule $q\langle a\langle x_1, \ldots, x_n\rangle\rangle \to rhs(q, \varphi)[f \leftarrow [\![f]\!](a) \mid f \in \mathcal{F}]$ in M'. The semantics of M is defined as that of M': for all $q \in Q$, $[\![q]\!]_M = [\![q]\!]_{M'}$.

Definition 6. A symbolic DTop is **effective** (which is assumed in the remainder) if the underlying DTTA is, and **(1)** There is an algorithm that computes the value of the data transformation $[\![f]\!](a)$ for a given $f \in \mathcal{F}$ and $a \in \Sigma$ and returns $\perp$ if it is not defined. **(2)** There is an algorithm that decides whether the image of a data transformation $[\![f]\!](\Sigma)$ is empty for a given $f \in \mathcal{F}$.

In **symbolic** DTop$^{\mathrm{R}}$, a symbolic BUTA on the transducer's input signature Σ first annotates the tree with its states P, and then the symbolic DTop transforms the annotated tree on $\Sigma \times P$.

Consider the logs of an application on a Unix-flavoured system. The log is a text file named "log", here containing "s" for every successful login, and "f" for every failure. Every week, the old log is discarded and replaced by statistics: the number of successful and failed logins (Figure 1). For this, we denote by f_{stats} the function counting the numbers n, m of occurrences of "s" and "f" in a string (here a log's contents), and outputting the string n";"m. A fresh "log" is then created. The older log's statistics are named "stats.1", "stats.2", etc. so that higher numbers indicate older stats. To model this with a symbolic DTop$^{\mathrm{R}}$, we represent the contents of the logs folder by a list (CONS and NIL being the usual constructors) of files, each file being a tree of the form FILE⟨"filename", "contents"⟩. The input and output alphabets are thus strings, along with FILE, CONS and NIL. The guards will be a small subset of regular expressions on strings plus descriptors matching CONS and NIL. The lookahead's purpose is to check whether the filename matches a stat file or not (which cannot be done in a top-down transducer's rule, in contrast to the extended rules (3) and (4)) and to annotate the FILE nodes with its findings. Guards are regular expressions. The descriptor matching a specific string is the string itself; $*$ matches everything; "stats."("0".."9")$^+$ is the descriptor matching stats filenames. The lookahead (LA) rules are

$$\text{"stats."}(\text{"0"}..\text{"9"})^+\langle\rangle \to p_{\text{stats}} \qquad \text{FILE}\langle p_{\text{stats}}, p\rangle \to p_{\text{stats file}}$$

$$\text{“log”}\langle\rangle \to p \qquad \text{FILE}\langle p, p\rangle \to p$$
$$(\text{“s”} \mid \text{“f”})^*\langle\rangle \to p \qquad \text{CONS}\langle \_, p\rangle \to p \qquad \text{NIL}\langle\rangle \to p$$

The label functions $\mathcal{F}$ are taken as, for instance, the class of rational functions, which we can implement with word transducers with lookahead [6], satisfying all requisite properties. We represent a constant function by the string it produces, the identity by f_{id}, and $f_{\text{increment}}$ for the function taking strings of the form “stats.”k, where k is the decimal representation of an integer, and yielding “stats.”$(k+1)$. We start in state q:

$$q\langle \text{NIL} : p\rangle \to \text{CONS}\langle\ \text{FILE}\langle\text{“log”}, \text{“”}\rangle,\ \text{NIL}\rangle \tag{1'}$$
$$q\langle \text{CONS} : p\langle x_1, x_2\rangle\rangle \to \text{CONS}\langle q_{\text{name}}\langle x_1\rangle, q\langle x_2\rangle\rangle \tag{2'}$$
$$q_{\text{name}}\langle \text{FILE} : p\langle x_1, x_2\rangle\rangle \to \text{FILE}\langle q_{\text{log}}\langle x_1\rangle, q_{\text{stats}}\langle x_2\rangle\rangle \tag{3a}$$
$$q_{\text{name}}\langle \text{FILE} : p_{\text{stats file}}\langle x_1, x_2\rangle\rangle \to \text{FILE}\langle q_{\text{incr}}\langle x_1\rangle, q_{\text{id}}\langle x_2\rangle\rangle \tag{4a}$$
$$q_{\text{incr}}\langle \text{“stats.”}(\text{“0”}..\text{“9”})^+ : p_{\text{stats}}\rangle \to f_{\text{increment}} \tag{3b}$$
$$q_{\text{id}}\langle * : p\rangle \to f_{\text{id}} \quad q_{\text{stats}}\langle * : p\rangle \to f_{\text{stats}} \quad q_{\text{log}}\langle \text{“log”} : p\rangle \to \text{“stats.1”}\ . \tag{4b}$$

4 Domain and Composition

To study problems like computation or equivalence on symbolic DTop, it is worth considering those problems as extensions of their counterparts for DTop. Indeed, most difficulties that previous papers [20, 19, 15] encountered are already relevant for the composition, normalization, or equivalence problems in the finite-labelled case [11, 16]. Most of those difficulties come from dealing with the domain of a transducer's transformation. Since these problems have been solved in DTop and proofs in symbolic DTop are essentially identical, we shall only present the results, a reference for the proofs in DTop, and the additional conditions required of Φ and $\mathcal{F}$ for them to carry over to the symbolic case.

The first important results concern automata and their expressive power. Symbolic Dtta which, as said before, we always assume to be effective, and have the classical properties of Dtta (e.g. in [5]).

Lemma 7. **(1)** *The class of languages described by symbolic* Dtta *is closed under Boolean set operations.* **(2)** *If equivalence is decidable on Φ, then equivalence is decidable on symbolic* Dtta*.*

The second important result concerns the domains of symbolic DTop. Several or no states can explore a particular subtree. We use previous results on DTop [11] to see that this domain can be recognized by an automaton.

Lemma 8. *Let M be a symbolic* DTop*. Then we can build a symbolic* Dtta *A such that $[\![A]\!] = \text{dom}([\![M]\!])$.*

As pointed out in [15], and contrary to a claim in [20], symbolic DTop are not closed under composition. To find a class closed under composition, a

solution presented in the DTop case [11] is to consider transducers *with domain inspection*: a **symbolic DTop with inspection** is a pair $N = (M, A)$ of a symbolic DTop M and a symbolic Dtta A. Its semantic is $[\![N]\!] = [\![M]\!]_{|[\![A]\!]}$, the function of M restricted to the language of A. We know that DTop with inspection are closed under composition [11]. This result extends to symbolic DTop if the set of functions of $\mathcal{F}$ is itself closed under composition, and the images of guards through functions of $\mathcal{F}$ form a suitable set of guards (i.e. they satisfy the requirements for effectiveness).

Lemma 9. *Let $\mathcal{F}$ be closed under composition, and N, N' be two effective symbolic* DTop *with inspection using functions of $\mathcal{F}$. Then we can build a symbolic* DTop *with inspection N'' such that $[\![N'']\!] = [\![N]\!] \circ [\![N']\!]$.*

The main intuition behind the generalisation of the classical results [1] is presented in several papers [20, 19, 15]; roughly, a rule in N'' is the image of the right-hand side of a rule of N' by a state of N. To obtain closure by composition for symbolic DTop – or indeed for DTop, as the problem is fundamentally unchanged by the alphabets – necessitates the use of either very strong restrictions, as in [15], or the use of domain inspection, which we prefer here.

5 Deciding Equivalence

In this section, we show that, given a few basic properties on label transformations (mostly that equivalence is decidable for label transformations) the equivalence problem for symbolic DTop is decidable, regardless of linearity, by reducing that problem to equivalence for DTop, which is known to be decidable. This method does not involve any external SMT solver, unlike [20, 19].

There are two basic observations behind this reduction. First, although Σ, Φ and $\mathcal{F}$ may well be infinite, only a finite number of predicates and transformations are actually used in a symbolic DTop (or indeed, any pair of symbolic DTop). These finite subsets of Φ and $\mathcal{F}$ will serve as finite input and output signatures in our reduction. Second, for two symbolic DTop to be equivalent, they generally need to use the same input label to produce some output label: two symbolic DTop using the same function on different input nodes will, in general, produce a different output (see Fig.2).

To be more specific, we introduce a notion of *origin* similar to the *syntactic alignments* of [2] and the *origins* of [3], and a weakened version of the *origin equivalence* in [3]. We assimilate, in a given tree, the nodes to their addresses according to the Dewey notation. For instance, in Figure 1, the node 1 is labelled by file, and the node 12 in the input is labelled by "ssfsffs". Let us consider a symbolic DTop M, as well as a tree t in its domain. For each node π of the tree $[\![M]\!](t)$, the node at π is created by examining a node μ of t using a rule of M. This input node is unique (see for example Prop. 52 of [2]) and we call it the **origin node** of π for $[\![M]\!](t)$. In the symbolic case, we can also track the function $f|_\varphi$ used to transform the label of μ into the label of π through a rule of guard φ, and call it the **origin function** of π. In Figure 1, the origin of 12 in the output is 12 in the input, and the origin function is f_{stats}, via rule (4b).

Definition 10. Two equivalent symbolic DTop M, N are *weak-origin-equivalent* iff for all $t \in \mathrm{dom}[\![M]\!]$, for any node π of $[\![M]\!](t)$, the origin nodes of π for M and N are identical, or its origin functions for M and N are constant of same value.

Lemma 11. *If two symbolic* DTop *are equivalent, they are weak-origin-equivalent.*

Intuitively, if M and N are not weak-origin equivalent then, for some tree t, an output node π comes from two different nodes μ and λ in the input using non-constant functions. By changing the label of μ without changing the label of λ, we change $[\![M]\!](t)$ and not $[\![N]\!](t)$, thus proving that M and N are not equivalent.

We now present the reduction from equivalence of symbolic DTop to that of DTop. Let $M = (\Phi, \Sigma, \mathcal{F}, \Delta, P, p_{\mathrm{ini}}, R)$ and $N = (\Phi, \Sigma, \mathcal{F}, \Delta, Q, q_{\mathrm{ini}}, S)$ be two symbolic DTop. We build their DTop **representations**, the DTop $\underline{M}$ and $\underline{N}$. Strictly speaking we should write $\underline{M}^{M,N}$ and $\underline{N}^{M,N}$, as the construction is specific to the pair of transducers under consideration, and the same applies to the representation of each component of the transducers – $\underline{\Phi}$, $\underline{\Sigma}$, etc – which we define below. In this section we assimilate descriptors φ, f and their semantics $[\![\varphi]\!], [\![f]\!]$ to lighten the notations.

We make the following additional **equivalence-testing assumptions**: for all $\varphi, \psi \in \Phi$, it is decidable whether $\varphi = \psi$. For all $f, g \in \mathcal{F}$ and all $\varphi \in \Phi$, it is decidable whether there exists some $c \in \Delta$ such that $f(\varphi) = \{c\}$, and this c is computable; and it is decidable whether $f|_\varphi = g|_\varphi$.

The finite information relevant to the behaviour of symbolic DTop is which guards are satisfied. Thus we let $\Pi = \{\, \mathrm{gd}(r) \mid r \in R \cup S \,\} \subseteq \Phi$ be the subset of guards actually used by either of the two transducers. The **finite alphabet $\underline{\Sigma}$ representing Σ** is defined as $\underline{\Sigma} = 2^{\Pi}$. The **representation of $a \in \Sigma$** is

$$\underline{a} \quad = \quad \{\, \pi \in \Pi \mid a \in \pi \,\} \in \underline{\Sigma}\,. \tag{6}$$

The **representation of a guard $\varphi \in \Pi$** is $\underline{\varphi} = \{\, \Pi' \subseteq \Pi \mid \varphi \in \Pi' \,\} \subseteq \underline{\Sigma}$. The **representation of an input tree $t \in \mathcal{T}(\Sigma)$** is defined inductively as

$$\underline{a(t_1, \ldots, t_n)} \quad = \quad \left\{\, (\underline{a}, b)(u_1, \ldots, u_n) \;\middle|\; b \in \mathbb{B},\ u_i \in \underline{t_i}, \forall i \,\right\} \in \mathcal{T}(\underline{\Sigma} \times \mathbb{B})\,, \tag{7}$$

where the addition of the bit b, called **obit** (origin bit), will be used to store just enough information about origins to ensure weak origin equivalence between M and N. Accordingly, the **representation of a label transformation f restricted to φ, with obit b** is defined as

$$\underline{f|_\varphi, b} \quad = \quad \begin{cases} c & \text{if } f(\varphi) = \{c\}, \text{ and} \\ (f|_\varphi, b) & \text{otherwise.} \end{cases} \tag{8}$$

The **representation of a rule $r \in R \cup S$**, of the form $r = q\langle\varphi\langle x_1, \ldots, x_n\rangle\rangle \to t$, is given by the set $\underline{r}$ of all classical rules

$$q\langle(\rho, b)\langle x_1, \ldots, x_n\rangle\rangle \to t\big[f \leftarrow (\underline{f|_\varphi}, b) \;\big|\; f \in \mathcal{F}\big]\,, \tag{9}$$

for all $b \in \mathbb{B}$, $\rho \in \underline{\varphi}$. Letting $\underline{R} = \bigcup_{r \in R} \underline{r}$ and $\underline{S} = \bigcup_{s \in S} \underline{s}$, we finally have $\underline{M} = (\underline{\Sigma}, \underline{\Delta}, P, p_{\mathrm{ini}}, \underline{R})$ and $\underline{N} = (\underline{\Sigma}, \underline{\Delta}, Q, q_{\mathrm{ini}}, \underline{S})$.

This representation is built so that the following holds: let t be a tree of $\mathrm{dom}(\llbracket M \rrbracket)$, a node π in $\llbracket M \rrbracket(t)$, its origin node μ and its origin function $f|_\varphi$. Then for any tree $u \in \underline{t}$, the node π in $\llbracket \underline{M} \rrbracket(u)$ is c if $f(\varphi) = \{c\}$, $(f|_\varphi, b)$ otherwise, with b the obit under μ in u. This leads to the following result.

Theorem 12. *Let M, N be two symbolic* DTop*, as above. Then $\llbracket M \rrbracket = \llbracket N \rrbracket$ if and only if $\llbracket \underline{M} \rrbracket = \llbracket \underline{N} \rrbracket$.*

Proof. First, we consider domain equality: the domain of $\underline{M}$ is the set of all the representations of trees of $\mathrm{dom}(\llbracket M \rrbracket)$. Hence $\underline{M}$ and $\underline{N}$ are of same domain if and only if M and N are of same domain. Let us now assume that the domains are the same. Suppose M and N are not equivalent; let t be an input tree such that $u = \llbracket M \rrbracket(t) \neq \llbracket N \rrbracket(t) = v$. We consider an address π that exists both in u and v but where the label at π differs in u and v. In M the origin node of π is μ; the origin function is $f|_\varphi$. In N the origin node of π is λ; the origin function is $g|_\psi$. Since the label at π differs in u and v, there must be a difference of origin node or functions.

If $f|_\varphi \neq g|_\psi$, they can't be constants of same value, otherwise the label at π would be the same in u and v. This means that their representations will differ in $\underline{M}$ and $\underline{N}$. Hence for any $t' \in \underline{t}$, the label at π would differ in $\llbracket \underline{M} \rrbracket(t')$ and $\llbracket \underline{N} \rrbracket(t')$. Hence $\llbracket \underline{M} \rrbracket \neq \llbracket \underline{N} \rrbracket$.

If $f|_\varphi = g|_\psi$, then $\mu \neq \lambda$, and $f|_\varphi$ can't be a constant, otherwise the label at π would be the same in u and v. We pick t' a representation of t where the obit under μ is 1, and the obit under λ is 0. The label at π in $\llbracket \underline{M} \rrbracket(t')$ would be $(f|_\varphi, 1)$, but the label at π in $\llbracket \underline{N} \rrbracket(t')$ would be $(f|_\varphi, 0)$. Hence $\llbracket \underline{M} \rrbracket \neq \llbracket \underline{N} \rrbracket$.

Conversely, suppose $\underline{M}$ and $\underline{N}$ are not equivalent; let t' be an input tree such that $u' = \llbracket \underline{M} \rrbracket(t') \neq \llbracket \underline{N} \rrbracket(t') = v'$. We consider an address π' that exists both in u' and v' but where the label at π' differs in u' and v'. In $\underline{M}$ the origin node of π' is μ', and in $\underline{N}$ the origin node of π' is λ'. Thus, μ' and λ' are also the origin node of π' in M and N for all t such that $t' \in \underline{t}$. If $\mu' \neq \lambda'$, we remark that the label of the nodes at π' cannot be the same constant c. This combined with the disparity of node origins means that M and N are not weak-origin-equivalent, and thus not equivalent. If $\mu' = \lambda'$, then the label at π' in u' and v' are representations of different functions (or constants) $f|_\varphi$ and $g|_\varphi$. There is at least one value $a \in \varphi$ such that $f(a) \neq g(a)$. We pick a tree t such that $t' \in \underline{t}$ and the label at μ' in t is labeled a. The node π' in $\llbracket M \rrbracket(t)$ is labeled $f(a)$, while the node π' in $\llbracket N \rrbracket(t)$ is labeled $g(a)$. Hence $\llbracket M \rrbracket \neq \llbracket N \rrbracket$. □

Corollary 13. *Under the equivalence-testing assumptions above, the equivalence problem for symbolic* DTop *is reducible to the equivalence problem on* DTop*, in* ExpTime *in the worst case, plus, at worst, an exponential number of operations in Φ and $\mathcal{F}$.*

Proof. Given that the construction of DTop representations in Thm. 12 is effective, we can build them and decide the equivalence of the representations. □

The exact complexity of this algorithm relies on the complexity of the various operations related to the equivalence-testing assumptions (computing intersec-

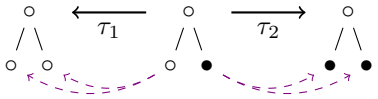


Fig. 2. Using obits to deduce origins. We represent the obits with colours: ∘ are nodes of obit 0, • are nodes of obit 1. We apply two transformations τ_1 and τ_2 on an input tree (here in the middle): τ_1 replaces its right leaf by a copy of the left one, while τ_2 replaces its left leaf by a copy of the right one. Without the obits, these two transformations would look identical. However, as seen above, the obits allow a distinction between τ_1 and τ_2 for some input tree.

tions, deciding function equivalence) in these sets, but also on the precise number of intersections and negations we have to perform in Φ. To compute these representations, we build all guards φ such that for some label a, $\varphi = (\bigcap_{\pi \in \underline{a}} \pi) \setminus (\bigcup_{\pi' \notin \underline{a}} \pi')$, and decide function equivalence on these φ for the functions used in $\underline{M}$ and $\underline{N}$. In the case where guards are all disjoint, the reduction to DTop is actually polynomial. In practice, it can be expected that few intersections actually need to be computed. The representations $\underline{a}$ can also be made more parsimonious by taking into account only tests that actually apply during the run, which can be done as in the construction of Lem. 8.

In any case, we can express the number of states and rules in $\underline{M}$ and $\underline{N}$ independently of Φ and $\mathcal{F}$: the states are unchanged, and the number of rules increases, at worst, exponentially.

Lemma 14. *For M, N two symbolic* DTop*, their* DTop *representations $\underline{M}$ and $\underline{N}$ are* DTop *with an exponential number of rules and the same number of states.*

Since the problem of DTop equivalence is NExpTime [13], a naïve approach to calculating the complexity of symbolic DTop equivalence would yield a 2NExpTime algorithm, plus an exponential number of operations in Φ and $\mathcal{F}$. However, upon finer analysis, the complexity of DTop equivalence is tied to the height of a counter-example between two non-equivalent transducers. This height is, in the worst case scenario, exponential in the number of states in the studied DTop. Since representations do not create new states, the height of the counter-examples is unchanged, and the exponentials do not compound.

Theorem 15. *The equivalence problem for symbolic* DTop *is in* NExpTime*, plus, at worst, an exponential number of operations in Φ and $\mathcal{F}$.*

5.1 Extension to symbolic $\mathrm{DTop}^{\mathrm{R}}$

We want to extend our results from symbolic DTop to the wider class of symbolic $\mathrm{DTop}^{\mathrm{R}}$. This class is relevant for several reasons. The first is that it is more expressive than the class of symbolic DTop with inspection, and subsumes other relevant classes, such as single-valued symbolic DTop [20, 19]. It also possesses interesting properties. Notably, just as it was the case for $\mathrm{DTop}^{\mathrm{R}}$ [8], the class of symbolic $\mathrm{DTop}^{\mathrm{R}}$ is closed under composition.

We want to study the equivalence problem of symbolic DTOP$^{\mathrm{R}}$. For DTOP, the addition of a regular lookahead does not prevent the equivalence problem from being decidable, as it is polynomially reduced to equivalence on plain DTOP (see [17]). This result can be carried over to the symbolic case, with the same method: annotating with both lookaheads.

Lemma 16. *One can polynomially reduce the equivalence problem of symbolic* DTOP^{R} *to the equivalence problem of symbolic* DTOP *with inspection.*

This reduction in polynomial time can be combined with our previous results (Cor. 13 and Thm. 15) to provide the following complexity result:

Theorem 17. *Under the assumptions of Cor. 13, the equivalence problem for symbolic* DTOP^{R} *is decidable in* NEXPTIME*, plus, at worst, an exponential number of operations in Φ and $\mathcal{F}$.*

This result is quite useful, as several DTOP classes are fragments of the class of DTOP$^{\mathrm{R}}$, and their decidability and complexity results can thus be transposed to the symbolic case. Notably, DBUP and nondeterministic functional TOP can be expressed as DTOP$^{\mathrm{R}}$.

Corollary 18. *Under the assumptions of Thm. 13, the equivalence problem for deterministic symbolic bottom-up tree transducers and nondeterministic symbolic functional top-down tree transducers is decidable.*

6 Conclusion

The algorithm presented here provides a novel approach to deciding equivalence for symbolic DTOP, and supports non-linear symbolic DTOP, by reduction to DTOP equivalence. Note that decidability of equivalence for DTOP$^{\mathrm{R}}$ [17] works in a comparable way: rather than finding a normal form, the two regular lookaheads are "harmonized" into one, then the problem is reduced to DTOP equivalence. The methods presented in this paper also apply to symbolic DTOP$^{\mathrm{R}}$ without a critical jump in complexity.

Our method does not involve the computation of a normal form, which is a rather classical technique to decide transducer equivalence [4, 11], with applications to learning. It is interesting to see if normal forms could be defined for symbolic DTOP. This looks challenging, however, as it seems more general than finding normal forms for DTOP$^{\mathrm{R}}$, which remains an open problem.

A first possible extension of our model would be to allow the lookahead to have registers, i.e. to memorize some data from the bottom of the tree to annotate the upper part of the tree with it. Under reasonable restrictions, it is likely that we might adapt our methods to reduce the equivalence problem for these objects to the same problem on DTOP$^{\mathrm{R}}$, thus providing a decidability result.

Furthermore, we would like to find out whether this kind of reduction can be applied to more general classes of transducers such as macro tree transducers (with linear size increase), for which equivalence is decidable [10]. If so, then the

decidability results can fairly easily be lifted to symbolic generalisations of the class, at the cost of a few exponential blowups in complexity.

As a final mention, the inversion problem is interesting for symbolic transformations on words and trees, and is relevant to the applications we consider.

References

1. B. S. Baker. Composition of top-down and bottom-up tree transductions. *Information and Control*, 41(2):186–213, 1979.
2. A. Boiret. *Normalization and Learning of Transducers on Trees and Words*. PhD thesis, Lille University, France, 2016.
3. M. Bojańczyk. Transducers with origin information. In *ICALP 2014*, volume 8573 of *LNCS*, pages 26–37. Springer, 2014.
4. C. Choffrut. Minimizing subsequential transducers: a survey. *Theor. Comput. Sci.*, 292(1):131–143, 2003.
5. 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. release October, 12th 2007.
6. C. C. Elgot and J. E. Mezei. On relations defined by generalized finite automata. *IBM J. Res. Dev.*, 9(1):47–68, Jan. 1965.
7. J. Engelfriet. Bottom-up and top-down tree transformations - A comparison. *Mathematical Systems Theory*, 9(3):198–231, 1975.
8. J. Engelfriet. Top-down tree transducers with regular look-ahead. *Mathematical Systems Theory*, 10:289–303, 1977.
9. J. Engelfriet. Some open questions and recent results on tree transducers and tree languages. pages 241–286. Academic Press, 1980.
10. J. Engelfriet and S. Maneth. Macro tree translations of linear size increase are MSO definable. *SIAM J. Comput.*, 32(4):950–1006, 2003.
11. 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.
12. J. Engelfriet and H. Vogler. Macro tree transducers. *J. Comput. Syst. Sci.*, 31(1):71–146, 1985.
13. Z. Ésik. On functional tree transducers. In *FCT*, pages 121–127, 1979.
14. Z. Ésik. Decidability results concerning tree transducers II. *Acta Cybern.*, 6(3):303–314, 1983.
15. Z. Fülöp and H. Vogler. Forward and backward application of symbolic tree transducers. *Acta Inf.*, 51(5):297–325, 2014.
16. A. Lemay, S. Maneth, and J. Niehren. A learning algorithm for top-down XML transformations. In *PODS*, pages 285–296. ACM, 2010.
17. S. Maneth. Equivalence problems for tree transducers: A brief survey. In Z. Ésik and Z. Fülöp, editors, *AFL*, volume 151 of *EPTCS*, pages 74–93, 2014.
18. H. Seidl, S. Maneth, and G. Kemper. Equivalence of deterministic top-down tree-to-string transducers is decidable. In *FOCS*, pages 943–962, 2015.
19. M. Veanes and N. Bjørner. Foundations of finite symbolic tree transducers. *EATCS*, 105:141–173, 2011.
20. M. Veanes and N. Bjørner. Symbolic tree transducers. In *PSI*, volume 7162 of *LNCS*, pages 377–393. Springer, 2011.