ELSEVIER

Theoretical Computer Science 205 (1998) 207–229

Theoretical Computer Science

On the equivalence problem for letter-to-letter top-down tree transducers[1]

Yves Andre*, Francis Bossut

LIFL, URA 369, CNRS University of Lille 1, 59655 Villeneuve d'Ascq Cedex, France

Received March 1994; revised November 1995
Communicated by J. Sakarovitch

Abstract

Letter to letter top-down tree transducers are investigated in this paper. Informally, trees which appear in the rules of such transducers are reduced to one letter in the right-hand side as in the left one. With an encoding of the tree transformations induced by such transducers into recognizable forests, we recently established the decidability of equivalence for linear top-down transducers. Here, in order to capture the non-linearity of top-down transducers, we introduce new classes of tree automata with equivalence constraints between direct subterms for which equivalence is decidable. We then show that the equivalence problem for non-linear top-down transducers can be reduced to the equivalence problem of automata with equivalence constraints.

1. Introduction

From a general point of view, tree automata and tree transducers model computations on structured objects. Consider a concrete algorithm A taking terms from T_Σ as input and producing terms of T_Δ, where Σ and Δ are finite sets of operators. Abstracting from the meaning of the operators, A turns into a symbolic algorithm which is a tree transducer transforming elements of T_Σ into elements of T_Δ.

Finite state tree transducers, which are a generalization of generalized sequential machines in the word case, were introduced by Rounds and Thatcher [13, 16]. This generalization to trees is interesting from the syntax-directed translation point of view. Let us give some examples. In compiler construction finite state transducers can be used to express simple transformations of abstract syntactical trees. Attribute grammars with only synthesized attributes correspond closely to deterministic top-down transducers. Also, subclasses of functional programs behave like tree transducers on their arguments.

* Corresponding author. E-mail: andrc@lifl.lifl.fr.

[1] Supported in part by the PRC-GDR "Mathématiques et Informatique" and the Basic Research Working Group ESPRIT 6317 ASMICS II.

0304-3975/98/$19.00
PII S0304-3975(97)00080-7

Naturally, the question arises whether or not results obtained for transformations in the word case can be transferred to the tree case. The situation is more complex. First we have to distinguish two main classes of tree transducers: top-down transducers which process the input trees from the root to the leaves and bottom-up transducers for which, on the contrary, the computations begin at the leaves and finish at the root (a comparison between these two classes can be found in [7]). As noteworthy characteristics, let us point out that in the top-down case, since multiple occurrences of variables in patterns are allowed, a transducer is able to compute different images of a given subtree. In the bottom-up case, since some variables can be missing, the image of a correctly parsed subtree can be deleted.

Two transducers are called equivalent if they define the same translation. It is well-known that equivalence is, in general, undecidable for non-deterministic tree transducers and that it is decidable for deterministic transducers, in the bottom-up case [17] as in the top-down one [9]. In [8], Engelfriet showed that it is decidable whether or not a tree transducer is functional. More recently, Seidl proved that this problem is decidable in polynomial time [15]. As a corollary, equivalence is decidable for functional tree transducers. Using a decomposition of finite-valued bottom-up transducers into a finite number of single-valued transducers, Seidl established the decidability of equivalence for finite-valued bottom-up transducers [14, 15].

This paper is devoted to the equivalence problem for non-deterministic letter-to-letter transducers. Informally, trees which appear in the rules of these transducers are reduced to one letter in the right-hand side as in the left one. Here, we prove the decidability of equivalence for non-deleting top-down transducers. In previous works [3, 2], we obtained the decidability of equivalence for linear top-down transducers by encoding the tree translations into recognizable forests. To extend this result, we need to refine the encoding procedure and, moreover, in order to capture the non-linearity of the investigated transducers, we introduce new classes of tree automata. More precisely, equivalence constraints on direct subterms can be mixed to the transitions of these new automata. We follow, in this way, recent works of Bogaert and Tison [5], Lugiez and Moysset [12], and Caron et al. [6]. Especially, decidability of emptiness for these new classes is based on the decidability of emptiness for tree automata with equalities or disequalities on direct subterms [5].

2. Preliminaries

In this section, we just recall definitions and properties used in the following. We refer the reader to [11] for tree rewriting systems and to [7, 10] for tree transducers.

A *ranked alphabet* is a pair (Σ, ρ) where Σ is a finite alphabet and ρ is a mapping from Σ to $\mathbb{N}$. Usually, we will write Σ for short. For any σ of Σ, $\rho(\sigma)$ is called the *rank* of σ.

For any integer n, Σ_n denotes the subset of Σ of letters of rank n. For any $k \geqslant 1$, X_k denotes the set of variables $\{x_1, \ldots, x_k\}$.

Given a ranked alphabet Σ and a denumerable set X of variables, $T_\Sigma(X)$ denotes the set of all *terms* (*trees*) over Σ and indexed by X. For any term t, we denote by $\mathcal{V}(t)$ the set of variables which appear in t. In the particular case $T_\Sigma(\emptyset)$, we will write T_Σ.

For any tree t, the *height* of t, denoted by $\pi(t)$, is defined by $\pi(t)=0$ if $t\in\Sigma_0$ or $t\in X_p$ and $\pi(t)=1+\max\{\pi(t_1),\ldots,\pi(t_n)\}$ if $t=\sigma(t_1,\ldots,t_n)$.

A tree $t_0(x_1,\ldots,x_n)$ is a *prefix* of a term t if there exist $t_1,\ldots,t_n$ such that $t=t_0(t_1,\ldots,t_n)$. For any $p\in\mathbb{N}$, $[p]$ denotes the set $\{1,\ldots,p\}$. A torsion θ from $[p]$ to $[q]$ is a mapping from $[p]$ to $[q]$. We denote it by $\langle q;\theta(1),\ldots,\theta(p)\rangle$. By $id_{[n]}$ we denote the identity mapping on $[n]$.

A *rewriting rule* over an alphabet σ is a couple (l,r) of terms of $T_\Sigma(X)$, usually denoted $l\to r$, such that either $\pi(l)\geqslant 1$ and $\mathcal{V}(r)\subseteq\mathcal{V}(l)$ or l and r are elements of T_Σ. A *rewriting system* $\mathcal{S}$ over an alphabet Σ is a finite set of rewriting rules over Σ.

A rewriting rule (l,r) is *non-deleting* if $\mathcal{V}(r)=\mathcal{V}(l)$. A rewriting system $\mathcal{S}$ is *non-deleting* if each rule of $\mathcal{S}$ is non-deleting.

A rewriting system $\mathcal{S}$ over an alphabet Σ is *letter-to-letter* if, for each rule of $\mathcal{S}$, trees which appear in the left-hand side and in the right-hand side either are of height equal to 1 or are included in Σ.

We write $t\to_{\mathcal{S}} t'$ if t is rewritten in t' by using one rule of $\mathcal{S}$. By $\overset{+}{\to}_{\mathcal{S}}$ (respectively $\overset{*}{\to}_{\mathcal{S}}$) we denote the transitive (respectively reflexive and transitive) closure of $\to_{\mathcal{S}}$.

A rewriting system $\mathcal{S}$ over an alphabet Σ is *noetherian* if there does not exist any infinite sequence $t_0\to_{\mathcal{S}} t_1\to_{\mathcal{S}}\cdots t_i$ being in $T_\Sigma(X)$ for any i.

A rewriting system $\mathcal{S}$ is *confluent* if $\forall x,\forall y,\forall z\in T_\Sigma(X)$, $(z\overset{*}{\to}_{\mathcal{S}} x$ and $z\overset{*}{\to}_{\mathcal{S}} y)\Rightarrow\exists t\in T_\Sigma(X)$ $(x\overset{*}{\to}_{\mathcal{S}} t$ and $y\overset{*}{\to}_{\mathcal{S}} t)$.

Let $\mathcal{S}$ be a noetherian and confluent rewriting system. The unique irreducible form of any term t is denoted by $\mathcal{S}(t)$ or $\widehat{t}$ if there is no ambiguity about the considered rewriting system.

A finite state *top-down tree transducer* is a 5-tuple $T=\langle\Sigma,\Delta,Q,I,R\rangle$ where Σ and Δ are ranked alphabets of, respectively, input and output symbols, Q is a finite set of unary symbols called states, I is the subset of Q of initial states and R is a finite set of rules of the form $q(\sigma(x_1,\ldots,x_n))\to t$ with $q\in Q$, $\sigma\in\Sigma_n$ and $t\in T_\Delta(Q(X_n))$ ($Q(X_n)$ is the set of trees $\{q(x)|q\in Q,x\in X_n\}$) or of the form $q(\sigma)\to t$ with $\sigma\in\Sigma_0$ and $t\in T_\Delta$.

These rules will be denoted by $q(\sigma(x_1,\ldots,x_n))\to\delta(q_1(x_{\theta(1)}),\ldots,q_p(x_{\theta(p)}))$ with $\sigma\in\Sigma_n$, $\delta\in T_\Delta(X_n)$, $q,q_1,\ldots,q_p$ states of Q, and θ torsion from $[p]$ to $[n]$ (if $n=0$ we write $q(\sigma)\to\delta$).

So the torsion θ can express permutations, duplications or deletions of subtrees.[2]

A transducer is *letter-to-letter* if, for every rule, δ belongs to Δ.

The rules define patterns for rewriting trees, so we write $t\to u$ if t is rewritten in u in one step. By $\overset{*}{\to}$ we denote the reflexive and transitive closure of $\to$. A sequence of rewriting steps $q(t)\overset{*}{\to}u$ is called a *computation*.

[2] For instance, by using the rule $q(\sigma(x_1,x_2))\to\delta(q_1(x_{\theta(1)}),q_2(x_{\theta(2)}))$ where $\theta=\langle 2;2,1\rangle$, a permutation of the two subtrees of σ is realized. For the rules $q(\sigma(x_1,x_2))\to\delta(q_1(x_{\theta(1)}),q_2(x_{\theta(1)}),q_3(x_{\theta(2)}))$ where $\theta=\langle 2;1,1,2\rangle$ and $q(\sigma(x_1,x_2))\to\delta(q_1(x_{\theta(2)}))$ where $\theta=\langle 2;2\rangle$, we have, respectively, a duplication and a deletion of the first subtree of σ.

For any state q of a top-down transducer, we denote by $\widehat{T}_q$ the transformation realized from q. Formally, $\widehat{T}_q = \{(t,u) \in T_\Sigma \times T_\Delta \mid q(t) \xrightarrow{*} u\}$.

Let $\mathscr{T}$ be a computation on a term $t = \sigma(t_1, \ldots, t_n)$ from a state q of a top-down transducer T:

$$\mathscr{T} : q(\sigma(t_1, \ldots, t_n)) \to \delta(q_1(t_{\theta(1)}), \ldots, q_p(t_{\theta(p)})) \xrightarrow{*} \delta(u_1, \ldots, u_p).$$

The *initial transformation* on t from state q is the triple (σ, δ, θ).

$\widehat{T}_{q_{(\sigma,\delta,\theta)}}$ denotes the transformation realized from state q by using the initial transformation (σ, δ, θ).

For any top-down transducer T, for any set of states E, for any initial transformation (σ, δ, θ), $\widehat{T}_{E_{(\sigma,\delta,\theta)}} = \bigcup_{q \in E} \widehat{T}_{q_{(\sigma,\delta,\theta)}}$ and $\widehat{T}_E = \bigcup_{q \in E} \widehat{T}_q$.

Finally, we denote by $\widehat{T}$ the tree transformation associated with T: $\widehat{T} = \widehat{T}_I$.

Two sets of states E and F are *globally equivalent* if $\widehat{T}_E = \widehat{T}_F$. Two transducers T and T' are *equivalent* if $\widehat{T} = \widehat{T}'$.

The *domain* of a tree transformation $\widehat{T}$, denoted by $dom(\widehat{T})$, is the set $\{t \in T_\Sigma \mid \exists u \in T_\Delta, (t,u) \in \widehat{T}\}$.

A state q of a transducer is *infinitary* if $\widehat{T}_q$ is infinite. A transducer all states of which are infinitary is said to be *infinitary*.

A top-down tree transducer is *deterministic* if the set of initial states is a singleton and there are no two rules with the same left-hand side.

A transducer is *linear* (respectively *non-deleting*) if for each rule the torsion θ is injective (respectively surjective).

Property. *For any letter-to-letter tree transducer T, for any couple of trees (t,u) of $\widehat{T}$, we have $\pi(t) = \pi(u)$ if T is non-deleting and $\pi(t) \geqslant \pi(u)$ in the other cases.*

Example. Let $\Sigma_0 = \{0\}$ and $\Sigma_1 = \{S\}$, $\Delta_0 = \{1\}$, $\Delta_1 = \{S\}$ and $\Delta_2 = \{*\}$, $Q = \{f, q\}$ and $I = \{f\}$.

Consider the following set of rules R:

$$\begin{cases} f(S(x)) \to *(q(x), f(x)), & f(S(x)) \to 1, \quad f(0) \to 1, \\ q(S(x)) \to S(q(x)), & q(0) \to 1. \end{cases}$$

This transducer, by consuming an input tree of the form $S^n(0)$ (which can be interpreted as the integer n) computes syntactical trees of $A_p^n = n!/(n-p)!$ for $p \in [n]$. This transducer is letter-to-letter and it is neither deterministic, nor linear, nor non-deleting, nor finite-valued (*finite-valued* transducers are investigated in [14]).

Notation. We denote by **NdT-LL** the class of all non-deleting top-down letter-to-letter tree transducers.

Remark. For easier exposition, we will restrict ourselves to letters of rank at most 2. It is the typical case from which constructions and results to be discussed below are

easily transferred to the general situation (with slight adaptations in the statement of some lemmas).

3. Tree automata with equivalence constraints

Non-linearity is an important phenomenon which appears very often in tree processing (in logic programming or in rewriting systems for instance). As the set of all ground instances of a non-linear term is not a recognizable forest, the classical notion of tree automaton has been extended in order to manipulate equalities or disequalities. In 1990, Bogaert and Tison introduced a class of bottom-up tree recognizers, for which equality and disequality tests between direct subterms are allowed [5]. So, they defined a class of tree languages, denoted by $\mathbf{REC}_{\neq}$, closed under boolean operators and for which emptiness is decidable. More recently, Lugiez and Moisset [12] extended the previous class by considering equality tests, modulo associativity and commutativity.

In this paper, we propose an other extension by introducing equivalence tests between direct subterms.

Informally, such automata have rules of the form $\sigma(q_1(x_1),\ldots,q_n(x_n))[d] \to q$ where d is an equivalence constraint. For instance, if $\sigma(q_1(x_1),q_2(x_2))[1,2] \to q$ is a rule of such an automaton, t_1 and t_2 are equivalent trees and $t_1 \xrightarrow{*} q_1$, $t_2 \xrightarrow{*} q_2$ then $\sigma(t_1,t_2) \xrightarrow{*} q$.

We show that, when the equivalence relation is induced by a noetherian, confluent and non-deleting letter-to-letter rewriting system τ (two trees being equivalent if and only if their irreducible forms are equal), for the so defined class of tree automata, denoted by $\mathbf{REC}_{\tau}$, equivalence is decidable.

3.1. The class REC_{τ}

A good survey on "classical" tree automata can be found in [10].

An *equivalence description* on n elements is a partition of $[n]$. Let $\equiv$ be an equivalence relation on T_{Σ} and let d be an equivalence description on $[n]$. A tuple of terms $(t_i)_{i\in[n]}$ *satisfies* the equivalence description d if and only if for any $X \in d$, for any i and $j \in X$, $t_i \equiv t_j$, and for any X and Y in d with $X \neq Y$, $i \in X$, $j \in Y$ implies $t_i \not\equiv t_j$.

A bottom-up *automaton with equivalence tests between direct subterms* is a 4-tuple $\langle \Sigma, Q, F, R\rangle$ where Σ is a ranked alphabet, Q is a finite set of states, F is the subset of Q of final states and R is a finite set of rules. Rules are usually denoted by $\sigma(q_1,\ldots,q_n)[d] \to q$ where d is an equivalence description on $[n]$.

Let A be such an automaton, we have $t \to_A t'$ with $t = t_0(a(q_1(t_1),\ldots,q_n(t_n)))$, $t' = t_0(q(a(t_1,\ldots,t_n)))$ if there exists in R a rule of the form $a(q_1(x_1),\ldots,q_n(x_n))[d] \to q$ such that $(t_i)_{i\in[n]}$ satisfies the equivalence description d.

By $\xrightarrow{*}_A$ we denote the reflexive and transitive closure of $\to_A$.

A tree t is recognized by A if there exists a final state q such that $t \xrightarrow{*}_A q(t)$. The set of all trees recognized by automaton A is denoted by $\mathscr{T}(A)$.

An automaton with equivalence constraints is *deterministic* (resp. *complete*) if and only if for any letter $\sigma \in \Sigma_n$, for any n-tuple of states $q_1,\ldots,q_n$ and for any equivalence description d there exists at most (resp. at least) one rule of the form $\sigma(q_1,\ldots,q_n)$ $[d] \to q$.

Using the method of Bogaert and Tison [5], we can compute a complete and deterministic automaton from any non-complete and non-deterministic one. Thus, we can show that, for any equivalence relation, the so-defined class of tree automata is effectively closed under union, intersection and complementation.

Let τ be a noetherian and confluent non-deleting rewriting system. We denote by $\mathbf{REC}_\tau$ the class of automata with equivalence tests between direct subterms where the equivalence relation is defined so that t and t' are said to be equivalent if and only if $\tau(t)=\tau(t')$ (by $\tau(t)$ we denote the irreducible form of t). For convenience, the class of tree languages which are recognizable by an automaton of $\mathbf{REC}_\tau$ will be also denoted by $\mathbf{REC}_\tau$. Furthermore, if τ is letter-to-letter, the set of the irreducible forms of the trees recognized by an automaton of $\mathbf{REC}_\tau$ is a forest of $\mathbf{REC}_{\neq}$. An automaton which recognizes this forest is called a *normal forms automaton*. To construct this automaton, we use a canonical form of τ called *1-reduced* form. We first define a partial order relation over the set of rules of any letter-to-letter rewriting system such that a rule r is said to be "more general" than a rule r' if everywhere the rule r' is applied, the rule r can be used with the same result.

Definition. Let $r : \sigma(x_{\mu(1)},\ldots,x_{\mu(n)}) \to \delta(x_{\theta(1)},\ldots,x_{\theta(m)})$ and $r' : \sigma(x_{\mu'(1)},\ldots,x_{\mu'(n)}) \to \delta(x_{\theta'(1)},\ldots,x_{\theta'(m)})$ be two rules of a letter-to-letter rewriting system. r is *more general* than r', or r' is *less general* than r, if there exists a mapping ϕ such that $\phi \circ \mu = \mu'$ and $\phi \circ \theta = \theta'$. We write $r \geqslant r'$ (or $r' \leqslant r$).

Example. Let $r_1 : \sigma(q(x_1),q(x_2),q(x_1),q(x_3)) \to q(\delta(x_1,x_1,x_2,x_3))$, $r_2 : \sigma(q(x_1),q(x_2),q(x_1),q(x_2)) \to q(\delta(x_1,x_1,x_2,x_2))$ and $r_3 : \sigma(q(x_1),q(x_2),q(x_3),q(x_1)) \to q(\delta(x_1,x_1,x_2,x_3))$.

We have $r_1 \geqslant r_2$ but neither $r_1 \geqslant r_3$ nor $r_3 \geqslant r_1$.

Definition. A rewriting system S over the alphabet Σ is *1-reduced* if it is letter-to letter, noetherian, confluent and the irreducible form $\hat{t} = \tilde{\sigma}(\hat{t}_{\theta(1)},\ldots,\hat{t}_{\theta(m)})$ in $T_\Sigma(X)$ of any term $t = \sigma(t_1,\ldots,t_n)$ of $T_\Sigma(X)$ is obtained as follows:

1. For any i in $[n]$, $t_i \xrightarrow{*} \hat{t}_i$.
2. (a) *First case*: $\sigma(\hat{t}_1,\ldots,\hat{t}_n) \to \tilde{\sigma}(\hat{t}_{\theta(1)},\ldots,\hat{t}_{\theta(m)})$ by using the less general rule of the form $\sigma(x_{\mu(1)},\ldots,x_{\mu(n)}) \to \tilde{\sigma}(x_{\theta(1)},\ldots,x_{\theta(m)})$ that can be applied.
   (b) *Second case*: No rule can be applied on $\sigma(\hat{t}_1,\ldots,\hat{t}_n)$.[3] Then $\tilde{\sigma} = \sigma$ and $\hat{t} = \sigma(\hat{t}_1,\ldots,\hat{t}_n)$.

That is to say, by using a bottom-up strategy of rewriting, we get the irreducible form and moreover each node is rewritten at most once.

[3] We have no rule of the form $\sigma(x,y) \to \sigma(y,x)$ because τ is a noetherian rewriting system.

From now, we will only consider non-deleting rewriting systems.

Property 3.1. For any noetherian, confluent and non-deleting letter-to-letter rewriting system τ over an alphabet Σ, there exists a 1-reduced rewriting system over Σ, denoted by τ_{1r}, such that, for any term t, $\tau(t) = \tau_{1r}(t)$.

Proof (*Hint*). Let τ be a noetherian, confluent and non-deleting letter-to-letter rewriting system. We construct τ_{1r} as follows:

- For any letter σ of rank 0, $\sigma \to \tilde{\sigma}$ is a rule of τ_{1r} if and only if $\sigma \xrightarrow{+}_{\tau} \tilde{\sigma}$ and $\tilde{\sigma}$ is irreducible.
- For any letter σ of rank 1, $\sigma(x) \to \tilde{\sigma}(x)$ (resp. $\tilde{\sigma}(x,x)$) is a rule of τ_{1r} if and only if $\sigma(x) \xrightarrow{+}_{\tau} \tilde{\sigma}(x)$ (resp. $\tilde{\sigma}(x,x)$) and $\tilde{\sigma}(x)$ (resp. $\tilde{\sigma}(x,x)$) is irreducible.
- For any letter σ of rank 2, $\sigma(x_{\theta(1)}, x_{\theta(2)}) \to \tilde{\sigma}(x_{\mu(1)}, x_{\mu(2)})$ (resp. $\tilde{\sigma}(x_{\mu(1)})$) is a rule of τ_{1r} if and only if $\sigma(x_{\theta(1)}, x_{\theta(2)}) \xrightarrow{+}_{\tau} \tilde{\sigma}(x_{\mu(1)}, x_{\mu(2)})$ (resp. $\tilde{\sigma}(x_{\mu(1)})$) and $\tilde{\sigma}(x_{\mu(1)}, x_{\mu(2)})$ (resp. $\tilde{\sigma}(x_{\mu(1)})$) is irreducible.

By induction on the height of the terms we prove that, for any term t, $t \to_{\tau_{1r}} \widehat{t}$ if and only if $t \xrightarrow{*}_{\tau} \widehat{t}$ and that τ_{1r} is 1-reduced. □

In the following, we will consider that, for any class $\mathbf{REC}_\tau$, the noetherian, confluent and non-deleting letter-to-letter rewriting system τ which induce the equivalence relation is 1-reduced.

3.2. Normal form automaton

In this section, we associate with any automaton A of $\mathbf{REC}_\tau$ the automaton A_τ of $\mathbf{REC}_{\neq}$ such that $IRR(\mathscr{F}(A)) = \mathscr{F}(A_\tau)$ (where $IRR(S)$ denotes the set of the irreducible forms of the terms of S).

Construction: Let τ be a 1-reduced non-deleting rewriting system defined on a finite alphabet Σ_τ. With any automaton $A = \langle \Sigma, Q, F, R \rangle$ of $\mathbf{REC}_\tau$, where $\Sigma \subset \Sigma_\tau$, we associate the automaton $A_\tau = \langle \Sigma_\tau, Q_\tau, F, R_\tau \rangle$ of $\mathbf{REC}_{\neq}$ constructed in two steps.

3.2.1. First step: construction of $Q_0 \subseteq Q_\tau$ and $R_0 \subseteq R_\tau$.

Initially, for any state q in Q, we add q in Q_0.

In the following arrays, which summarize the cases that σ is a nullary symbol and σ is a unary symbol, rows are indexed by rules of τ, columns are indexed by rules of R and the cells contain rules of R_0:

case $\rho(\sigma) = 0$	$\sigma \to q$
$\sigma \to \tilde{\sigma}$	$\tilde{\sigma} \to q$
no matching rule	$\sigma \to q$

case $\rho(\sigma) = 1$	$\sigma(q_i) \to q_j$
$\sigma(x) \to \tilde{\sigma}(x)$	$\tilde{\sigma}(q_i) \to q_j$
$\sigma(x) \to \tilde{\sigma}(x,x)$	$\tilde{\sigma}(q_i, q_i)[1,2] \to q_j$ [*]
no matching rule	$\sigma(q_i) \to q_j$

[*] A duplication turns into an equality constraint.

For the case that σ is a symbol of rank 2, we distinguish four disjoint subcases:

1. There is no rule in τ whose left-hand side contains the symbol σ, in this case, from $\sigma(q_i,q_j)[1],[2] \to q$ (resp. $\sigma(q_i,q_j)[1,2] \to q$) rule of R we construct in R_0 the rule $\sigma(q_i,q_j)[1],[2] \to q$ [**] (resp. $\sigma(q_i,q_j)[1,2] \to q$ [**]).
2. The only one rule of τ whose left-hand side contains σ has one of the following forms $\sigma(x,y) \to \tilde{\sigma}(x,y)$ or $\sigma(x,y) \to \tilde{\sigma}(y,x)$,

	$\sigma(q_i,q_j)[1],[2] \to q$	$\sigma(q_i,q_j)[1,2] \to q$
$\sigma(x,y) \to \tilde{\sigma}(x,y)$	$\tilde{\sigma}(q_i,q_j)[1],[2] \to q$	$\tilde{\sigma}(q_i,q_j)[1,2] \to q$ [**]
$\sigma(x,y) \to \tilde{\sigma}(y,x)$	$\tilde{\sigma}(q_j,q_i)[1],[2] \to q$	$\tilde{\sigma}(q_j,q_i)[1,2] \to q$ [**]

3. The left-hand side of the only rule of τ which transforms σ is of the form $\sigma(x,x)$,

	$\sigma(q_i,q_j)[1],[2] \to q$	$\sigma(q_i,q_j)[1,2] \to q$
$\sigma(x,x) \to \tilde{\sigma}(x,x)$	$\emptyset$	$\tilde{\sigma}(q_i,q_j)[1,2] \to q$ [**]
$\sigma(x,x) \to \tilde{\sigma}(x)$	$\emptyset$	$\tilde{\sigma}(\{q_i,q_j\}) \to q$, $\{q_i,q_j\} \in Q_0$[4]

3. The two rules of τ whose left-hand side contains the symbol σ are of the form $\sigma(x,y) \to \tilde{\sigma}'(x,y)$ or $\sigma(x,y) \to \tilde{\sigma}'(y,x)$ and $\sigma(x,x) \to \tilde{\sigma}''(x,x)$ or $\sigma(x,x) \to \tilde{\sigma}''(x)$:

	$\sigma(q_i,q_j)[1],[2] \to q$	$\sigma(q_i,q_j)[1,2] \to q$
$\sigma(x,y) \to \tilde{\sigma}'(x,y)$	$\tilde{\sigma}'(q_i,q_j)[1,],[2] \to q$[**]	$\emptyset$
or $\sigma(x,y) \to \tilde{\sigma}'(y,x)$	$\tilde{\sigma}'(q_i,q_j)[1,],[2] \to q$[**]	$\emptyset$
$\sigma(x,x) \to \tilde{\sigma}''(x,x)$	$\emptyset$	$\tilde{\sigma}''(q_i,q_j)[1,2] \to q$[**]
or $\sigma(x,x) \to \tilde{\sigma}''(x)$	$\emptyset$	$\tilde{\sigma}''(\{q_i,q_j\}) \to q$, $\{q_i,q_j\} \in Q_0$

[**] Equivalence between subterms turns into equality between their normal forms.

We obtained, in the left-hand side of some rules, new states which are in fact subsets of Q but these new states are not reachable. In the second step, we construct the smallest set of rules R_τ which contains R_0, with these new states in the right-hand side of the rules and such that the following property is satisfied:

Property. *For any state $\{q_1,\ldots,q_p\}$ of Q_τ, for any letter σ of Σ_τ:*

- *$\sigma \to \{q_1,\ldots,q_p\}$ is a rule of R_τ if for any i in $[p]$, $\sigma \to q_i$ is a rule of R_0*
- *$\sigma(E) \to F$ is a rule of R_τ if for any state q_i in F, there exists a subset E_i of E such that $\sigma(E_i) \to q_i$ is a rule of R_0*
- *$\sigma(E,E')[d] \to F$ is a rule of R_τ if for any state q_i in F, there exists a state k_i in E, a state k_i' in E' such that $\sigma(k_i,k_i')[d] \to q_i$ is a rule of R_0.*

[4] In the case $q_i = q_j$, we obtain as new state $\{q_i\}$ which is different from q_i.

3.2.2. Second step: Algorithm of "completion" of Q_0 and R_0.

```
begin
R_τ ← R_0; I ← 0;
repeat
  I ← I + 1; Q_I ← Q_{I−1};
  for any p-tuple of states {q_1,...,q_p} of Q_{I−1} do
    for any letter σ of Σ_τ do
      [i] case σ of rank 0:
      if ∀i ∈ [p], σ → q_i ∈ R_τ then R_τ ← R_τ ∪ {σ → {q_1,...,q_p}}.
      [ii] case σ of rank 1:
      For every E = ⋃_{i∈[p]} E_i where E_i ∈ Q_{I−1} and such that
      for any i ∈ [p], σ(E_i) → q_i is a rule of R_0 do
      R_τ ← R_τ ∪ {σ(E) → {q_1,...,q_p}}; Q_I ← Q_I ∪ {E}.
      [iii] case σ of rank 2:
      For every E' = ⋃_{i∈[p]}{k'_i} where k'_i ∈ Q, E'' = ⋃_{i∈[p]}{k''_i} where k''_i ∈ Q
      and such that for any i ∈ [p], σ(k'_i, k''_i)[d] → q_i is a rule of R_0 do
      R_τ ← R_τ ∪ {σ(E', E'')[d] → {q_1,...,q_p}};Q_I ← Q_I ∪ {E', E''}.
    end of for
  end of for
until Q_I = Q_{I−1}
Q_τ ← Q ∪ Q_I
end
```

Remark. Note that at each step of this algorithm, the previous property is satisfied. It can be used to establish the correctness of the completion algorithm.

So for any step of computation of a term $\widehat{t}$ in A_τ, from the applied rule we are able to determine the corresponding rule of A used in the computation of t.

We have for this automaton of $\mathbf{REC}_\tau$ the following fundamental property:

Lemma 3.1. *Let A be an automaton of $\mathbf{REC}_\tau$, with τ a non-deleting 1-reduced rewriting system, and let A_τ be the normal form automaton associated with A. Let $\{q_1,\ldots,q_p\}$ be a state of A_τ. Then, for any t in T_Σ, $t \xrightarrow{*}_{A_\tau} \{q_1,\ldots,q_p\}$ if and only if for any i in $[p]$, $t \xrightarrow{*}_{A_\tau} q_i$.*

Proof. It is by induction on the depth of the trees. The property is obviously true for trees of depth 0. Suppose property is true up to the height n and let t be a tree of height $n+1$.

Case 1: $t = \sigma(t')$. $t \xrightarrow{*}_{A_\tau} \{q_1,\ldots,q_p\} \Rightarrow \exists \sigma(\{k_1,\ldots,k_m\}) \to \{q_1,\ldots,q_p\} \in R_\tau$ and $t' \xrightarrow{*}_{A_\tau} \{k_1,\ldots,k_m\}$. By construction (ii), $\sigma(\{k_1,\ldots,k_m\}) \to \{q_1,\ldots,q_p\} \in R_\tau \Rightarrow \forall i \in [p]$ $\exists E_i \subset \{k_1,\ldots,k_m\}$ such that $\sigma(E_i) \to q_i$ is a rule of R_τ. Now by the induction hypothesis $t' \xrightarrow{*}_{A_\tau} \{k_1,\ldots,k_m\} \Leftrightarrow \forall j \in [m]\, t' \xrightarrow{*}_{A_\tau} k_j$ and so $t' \xrightarrow{*}_{A_\tau} \{k_1,\ldots,k_m\} \Rightarrow \forall E_i \subset \{k_1,\ldots,k_m\}$, $t' \xrightarrow{*}_{A_\tau} E_i$. Thus $\forall i \in [p], t \xrightarrow{*}_{A_\tau} q_i$.

Conversely, $\forall i \in [p] t \xrightarrow{*}_{A_\tau} q_i \Rightarrow \forall i \in [p] \exists \sigma(E_i) \to q_i \in R_\tau$ and $t' \xrightarrow{*}_{A_\tau} E_i$. Let $E = \bigcup_{i=1}^{p} E_i$, by construction (ii) we have $\sigma(E) \to_{A_\tau} \{q_1, \ldots, q_p\}$. Now by the induction hypothesis $\forall i \in [p] t' \xrightarrow{*}_{A_\tau} E_i \Rightarrow t' \xrightarrow{*}_{A_\tau} E$ and so $t \xrightarrow{*}_{A_\tau} \{q_1, \ldots, q_p\}$.

Case 2: $t = \sigma(t', t'')$. $t \xrightarrow{*}_{A_\tau} \{q_1, \ldots, q_p\} \Rightarrow \exists \sigma(E', E'')[d] \to \{q_1, \ldots, q_p\} \in R_\tau$, $t' \xrightarrow{*}_{A_\tau} E'$, $t'' \xrightarrow{*}_{A_\tau} E''$ and (t', t'') satisfies d. By construction (iii), $\sigma(E', E'')[d] \to \{q_1, \ldots, q_p\} \in R_\tau \Rightarrow \forall i \in [p], \exists k_i' \in E', \exists k_i'' \in E''$ such that $\sigma(k_i', k_i'')[d] \to q_i$ is a rule of R_τ. Now by the induction hypothesis, $t' \xrightarrow{*}_{A_\tau} E' \Leftrightarrow \forall k_j' \in E'\ t' \xrightarrow{*}_{A_\tau} k_j'$ and $t'' \xrightarrow{*}_{A_\tau} E'' \Leftrightarrow \forall k_j'' \in E''\ t'' \xrightarrow{*}_{A_\tau} k_j''$. Thus, $\forall i \in [p]$, $\sigma(t', t'') \xrightarrow{*}_{A_\tau} q_i$.

Conversely, $\forall i \in [p] \quad \sigma(t', t'') \xrightarrow{*}_{A_\tau} q_i \Rightarrow \forall i \in [p] \exists \sigma(k_i', k_i'')[d] \to q_i \in R_\tau$, $t' \xrightarrow{*}_{A_\tau} k_i'$, $t'' \xrightarrow{*}_{A_\tau} k_i''$ and (t', t'') satisfies d. Let $E' = \bigcup_{i=1}^{p} k_i'$ and $E'' = \bigcup_{i=1}^{p} k_i''$, by construction (iii) $\sigma(E', E'')[d] \to \{q_1, \ldots, q_p\} \in R_\tau$. By the induction hypothesis $t' \xrightarrow{*}_{A_\tau} E'$, $t'' \xrightarrow{*}_{A_\tau} E''$ and so $t \xrightarrow{*}_{A_\tau} \{q_1, \ldots, q_p\}$. □

Then, using the previous construction and Lemma 3.1, we state the following lemmas, on which is based Property 3.2.

Lemma 3.2. *Let A be an automaton of $\boldsymbol{REC}_\tau$, with τ a non-deleting, 1-reduced rewriting system, and let A_τ be the normal form automaton associated with A. For any term t in T_Σ, if $t \xrightarrow{*}_A q$ then $\widehat{t} \xrightarrow{*}_{A_\tau} q$.*

Proof. It is by induction on the height of the trees. The proof is obvious in the case $t = \sigma$, with σ a letter of rank 0.

Suppose property is true up to the height n and let $t = \sigma(t_1, t_2)$ be a tree of height $n+1$. $t \xrightarrow{*}_A q \Rightarrow \exists \sigma(q_1, q_2)[d] \to q$ in the set of rules of A, $t_1 \xrightarrow{*}_A q_1$, $t_2 \xrightarrow{*}_A q_2$ and (t_1, t_2) satisfies d. If $d = [1,2]$ then $\widehat{t}_1 = \widehat{t}_2$ else $(d = [1], [2])$ $\widehat{t}_1 \neq \widehat{t}_2$. By the induction hypothesis, $t_1 \xrightarrow{*}_A q_1$ and $t_2 \xrightarrow{*}_A q_2$ imply $\widehat{t}_1 \xrightarrow{*}_{A_\tau} q_1$ and $\widehat{t}_2 \xrightarrow{*}_{A_\tau} q_2$. Because τ is 1-reduced, $\widehat{t}$ is computed from $\sigma(\widehat{t}_1, \widehat{t}_2)$ by using at most one rule of τ.

According to the first step of the construction of A_τ, we distinguish the following cases:

(a) If there is no rule of τ with σ as a left-hand side then $\sigma(q_1, q_2)[d] \to q \in R_\tau$ (1. of construction 3.2.1). In this case $\widehat{t} = \sigma(\widehat{t}_1, \widehat{t}_2)$ and so $\widehat{t} \xrightarrow{*}_{A_\tau} q$.

(b) The rule of τ applied to obtain $\widehat{t}$ from $\sigma(\widehat{t}_1, \widehat{t}_2)$ is

  (a) $\sigma(x_1, x_2) \to \tilde{\sigma}(x_{\theta(1)}, x_{\theta(2)})$. In this case $\widehat{t} = \tilde{\sigma}(\widehat{t}_{\theta(1)}, \widehat{t}_{\theta(2)})$ and $\tilde{\sigma}(q_{\theta(1)}, q_{\theta(2)})[d] \to q \in R_\tau$ (2. of construction 3.2.1), thus $\widehat{t} \xrightarrow{*}_{A_\tau} q$.

  (b) $\sigma(x, x) \to \tilde{\sigma}(x, x)$: Then $\widehat{t}_1 = \widehat{t}_2$ $(= \widehat{t'})$. In this case, $\widehat{t} = \tilde{\sigma}(\widehat{t'}, \widehat{t'})$, $\tilde{\sigma}(q_1, q_2)[1,2] \to q \in R_\tau$ (3. of construction 3.2.1) and so $\widehat{t} \xrightarrow{*}_{A_\tau} q$.

  (c) $\sigma(x, x) \to \tilde{\sigma}(x)$: Then $\widehat{t}_1 = \widehat{t}_2$ $(= \widehat{t'})$. In this case $\widehat{t} = \tilde{\sigma}(\widehat{t'})$ and $\tilde{\sigma}(\{q_1, q_2\}(x)) \to q \in R_\tau$ (3. of construction 3.2.1). Now $\widehat{t}_1 \xrightarrow{*}_{A_\tau} q_1$, $\widehat{t}_2 \xrightarrow{*}_{A_\tau} q_2$ and $\widehat{t}_1 = \widehat{t}_2 = \widehat{t'}$ then $\widehat{t'} \xrightarrow{*}_{A_\tau} \{q_1, q_2\}$ (Lemma 3.1) and $\widehat{t} \xrightarrow{*}_{A_\tau} q$.

So, in all cases, $\widehat{t} \xrightarrow{*}_{A_\tau} q$. The proof is similar in the case $t = \sigma(t')$. □

Lemma 3.3. *Let A be an automaton of* $\mathbf{REC}_\tau$, *with τ a non-deleting 1-reduced rewriting system, and let A_τ be the normal form automaton associated with A. For any term $\widehat{t}$, if $\widehat{t} \xrightarrow{*}_{A_\tau} \{q_1,\ldots,q_p\}$ then for any q_i in $\{q_1,\ldots,q_p\}$, there exists at least one term t_i such that $t_i \xrightarrow{*}_\tau \widehat{t}$ and $t_i \xrightarrow{*}_A q_i$.*

Proof. Let $A=\langle \Sigma, Q, F, R\rangle$ and $A_\tau=\langle \Sigma_\tau, Q_\tau, F, R_\tau\rangle$. The proof is by induction on the height of the trees. The case $\widehat{t}=\tilde{\sigma}$, with $\tilde{\sigma}$ a term of depth 0 is obvious and so we will only consider the other cases:

Case 1: $\widehat{t}=\tilde{\sigma}(\widehat{t}_1,\widehat{t}_2)$. $\widehat{t} \xrightarrow{*}_{A_\tau} \{q_1,\ldots,q_p\} \Rightarrow \exists \tilde{\sigma}(E_1,E_2)[d] \to \{q_1,\ldots,q_p\} \in R_\tau$ and $\widehat{t}_1 \xrightarrow{*}_{A_\tau} E_1, \widehat{t}_2 \xrightarrow{*}_{A_\tau} E_2$. By the second step of construction of A_τ (iii): $\forall q_i \in \{q_1,\ldots,q_p\}$ $\exists k \in E_1$ and $l \in E_2$ such that $\tilde{\sigma}(k,l)[d] \to q_i \in R_0$.

By the induction hypothesis: $\widehat{t}_1 \xrightarrow{*}_{A_\tau} E_1$ and $k \in E_1 \Rightarrow \exists t_1$ such that $t_1 \xrightarrow{*}_\tau \widehat{t}_1$ and $t_1 \xrightarrow{*}_A k$ and $\widehat{t}_2 \xrightarrow{*}_{A_\tau} E_2$ and $l \in E_2 \Rightarrow \exists t_2$ such that $t_2 \xrightarrow{*}_\tau \widehat{t}_2$ and $t_2 \xrightarrow{*}_A l$.

According to the construction of A_τ, the rule $\tilde{\sigma}(k,l)[d] \to q_i$ of R_τ is associated with a rule of τ and a rule of A. Let us consider the most significant cases:

(a) There is no rule of τ with $\tilde{\sigma}(x,y)$ as a right-hand side. In this case, $\tilde{\sigma}(k,l)[d] \to q_i$ is a rule of R. Let $t_i=\tilde{\sigma}(t_1,t_2)$ then $t_i \xrightarrow{*}_A q_i$.

(b) The rule of A_τ is obtained from $\sigma(x,y) \to \tilde{\sigma}(x,y) \in \tau$ and $\sigma(k,l)[d] \to q_i \in R$.
In this case, let $t_i=\sigma(t_1,t_2)$ then $t_i \xrightarrow{*}_A q_i$.

(c) The rule of A_τ is obtained from $\sigma(x) \to \tilde{\sigma}(x,x) \in \tau$ then $k=l$ and $d=[1,2]$, and we have $\sigma(k) \to q_i \in R$.
In this case, let $t_i=\sigma(t_1)$ (or $t_i=\sigma(t_2)$) then $t_i \xrightarrow{*}_A q_i$.

Case 2: $\widehat{t}=\tilde{\sigma}(\widehat{t}')$. $\widehat{t} \xrightarrow{*}_{A_\tau} \{q_1,\ldots,q_p\} \Rightarrow \exists \tilde{\sigma}(E) \to \{q_1,\ldots,q_p\} \in R_\tau$ and $\widehat{t}' \xrightarrow{*}_{A_\tau} E$. By the induction assumption: $\forall k \in E$ $\exists t'$ such that $t' \xrightarrow{*}_\tau \widehat{t}'$ and $t' \xrightarrow{*}_A k$. By the second step of the construction of A_τ (Section 3.2.2), $\forall q_i \in \{q_1,\ldots,q_p\}, \exists E_i \subset E$ such that $\tilde{\sigma}(E_i) \to q_i \in R_0$.

(a) The rule of R_0 is obtained from $\sigma(x) \to \tilde{\sigma}(x)$ of τ and $\sigma(q) \to q_i$. Then $E_i=q$. Let $t_i=\sigma(t')$ then $t_i \xrightarrow{*}_A q_i$.

(b) The rule of R_τ is obtained from $\sigma(x,x) \to \tilde{\sigma}(x)$ of τ and $\sigma(q',q'')[1,2] \to q_i$ of R. Then $E_i=\{q',q''\}$. In this case, let $t_i=\sigma(t',t')$ then $t_i \xrightarrow{*}_A q_i$.

In all cases, $\forall q_i \in \{q_1,\ldots,q_p\} \exists t_i$ such that $t_i \xrightarrow{*}_\tau \widehat{t}$ and $t_i \xrightarrow{*}_A q_i$. □

So, from Lemmas 3.2 and 3.3, we get immediately:

Property 3.2. Let A be an automaton of $\mathbf{REC}_\tau$, with τ a non-deleting, noetherian and confluent letter-to-letter rewriting system. The set $IRR(\mathscr{F}(A))$, of the irreducible forms of the trees of $\mathscr{F}(A)$, is recognized by an automaton of $\mathbf{REC}_{\neq}$.

We now give two examples of tree automata with equivalence constraints with the construction of the normal form automata.

Examples

Let $\Sigma_0 = \{\bar{a}, \tilde{a}\}$, $\Sigma_1 = \{\alpha, \tilde{\sigma}\}$, $\Sigma_2 = \{\sigma\}$.
The equivalence relation on T_Σ is induced by the rewriting system τ.

Rewriting system τ

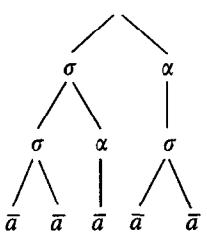
$$\begin{cases} \sigma(x,x) \to \tilde{\sigma}(x) \\ \alpha(x) \to \tilde{\sigma}(x) \\ \bar{a} \to \tilde{a} \end{cases}$$

Automaton of REC$_\tau$

$A = \langle \Sigma, Q, F, R \rangle$ is defined by:
- $Q = \{q, dead\}$, $F = \{q\}$
- and R composed of the rules:

$$\begin{cases} \sigma(q,q)[1,2] \to q \\ \sigma(q,q)[1],[2] \to dead \\ \alpha(q) \to q \\ \bar{a} \to q \end{cases}$$

This automaton recognize trees all branches of which have the same length.
For instance, σ is recognized by A



Normal form automaton A_τ

We obtain: $\begin{cases} \tilde{\sigma}(q) \to q \\ \tilde{a} \to q \end{cases}$

We get $\mathscr{F}(A_\tau) = \{\tilde{\sigma}^n(\tilde{a}) \mid n \in \mathbb{N}\}$.

Let $\Sigma_0 = \{1\}$, $\Sigma_1 = \{S\}$ and $\Sigma_2 = \{*\}$.

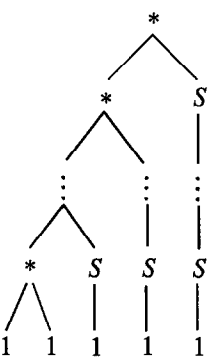
Rewriting system τ: $*(x,x) \to S(x)$

Automaton of REC$_\tau$

$B = \langle \Sigma, Q, F, R \rangle$ is defined by:
- $Q = \{q, q_f, dead\}$, $F = \{q\}$
- and R composed of the rules

$$\begin{cases} *(q,q_f)[1,2] \to q \\ *(q,q_f)[1],[2] \to dead \\ S(q_f) \to q_f \\ 1 \to q \\ 1 \to q_f. \end{cases}$$

The trees recognized by B are of the form:



These trees can be interpreted as factorial of integers.

Normal form automaton B_τ

We obtain: $\begin{cases} S(\{q,q_f\}) \to q \\ S(q_f) \to q_f \\ 1 \to q \\ 1 \to q_f \\ *(q,q_f)[1],[2] \to dead \end{cases}$

The state $\{q, q_f\}$ is not reachable.
By using the previous "completion" algorithm, we add the rules $S(\{q,q_f\}) \to \{q,q_f\}$ and $1 \to \{q,q_f\}$.
Finally, we get $\mathscr{F}(B_\tau) = \{S^n(1) | n \in \mathbb{N}\}$.

3.3. Decidability of equivalence in REC_τ

By means of the previous lemma and the decidability of emptiness in $\mathbf{REC}_{\neq}$ we prove the following lemma:

Lemma 3.2. *For any automaton A of $\mathbf{REC}_\tau$, the property "$\mathscr{F}(A)=\emptyset$" is decidable.*

Proof. $\mathscr{F}(A)\neq\emptyset$ implies that there exist at least a final state q and a tree t such that $t \xrightarrow{*}_A q$. Using property 3.2.2, we obtain $\widehat{t} \xrightarrow{*}_{A_\tau} q$ and so $\mathscr{F}(A_\tau)\neq\emptyset$. Conversely, we proceed in the same way. Thus, $\mathscr{F}(A_\tau)\neq\emptyset$ if and only if $\mathscr{F}(A)\neq\emptyset$. Emptiness is decidable in $\mathbf{REC}_{\neq}$ [5] and so it is in $\mathbf{REC}_\tau$. □

Finally we get:

Theorem 3.1. *For any non-deleting, confluent and noetherian letter-to-letter rewriting system τ, equivalence in $\mathbf{REC}_\tau$ is decidable.*

Proof. The class $\mathbf{REC}_\tau$ is effectively closed under union, intersection and complementation (part 3.1) and the property "$\mathscr{F}(A)=\emptyset$" is decidable (Lemma 3.4). □

This result will be used to establish the decidability of equivalence for non-deleting (and non-linear) top-down letter-to-letter tree transducers.

4. Equivalence of letter-to-letter non-deleting tree transducers

In this section, we show that the equivalence problem in **NdT-LL** can be reduced to the equivalence problem of tree automata. The main problem is that, even if T and T' are equivalent transducers, for some trees, computations may not be realized with the same torsions in T and T'.

We first prove that this phenomenon is of "bounded height".

4.1. Computations from equivalent states

Lemma 4.1. *Let T be a non-deleting infinitary top-down transducer. For any couple (E,F) of globally equivalent sets of states, for any (σ,δ,θ), the difference $\widehat{T}_{E_{(\sigma,\delta,\theta)}} - \widehat{T}_{F_{(\sigma,\delta,\theta)}}$ is finite.*

Proof. Let E and F be two globally equivalent sets of states of a non-deleting top-down transducer T. There are several cases, but we only consider two.

Case 1: σ is a letter of rank 1 and δ is a letter of rank 2. In this case, $\theta=\langle 1;1,1\rangle$ and $\widehat{T}_{E_{(\sigma,\delta,\theta)}}$ is the set of all pairs of ground trees $(\sigma(t),\delta(t_1,t_2))$ such that $q(\sigma(t)) \xrightarrow{*} \delta(t_1,t_2)$, for some $q\in E$. A similar fact is true for F. Thus, since E and F are globally equivalent, it follows that $\widehat{T}_{E_{(\sigma,\delta,\theta)}}=\widehat{T}_{F_{(\sigma,\delta,\theta)}}$.

Case 2: σ and δ are letters of rank 2. We consider the difference $\widehat{T}_{E_{(\sigma,\delta,\theta)}} - \widehat{T}_{F_{(\sigma,\delta,\theta)}}$. Let $\theta = id_{[2]}$ (the other case, $\theta = \langle 2;2,1\rangle$, is similar).

(i) For any couple of trees $(\sigma(t,t'),\delta(u,u')) \in \widehat{T}_E = \widehat{T}_F$, with $\pi(t) \neq \pi(t')$, because T is non-deleting, we have $\pi(u) = \pi(t)$, $\pi(u') = \pi(t')$ (Section 2) and so $\pi(u) \neq \pi(u')$. Consequently, $(\sigma(t,t'),\delta(u,u')) \notin \widehat{T}_{E_{(\sigma,\delta,\theta)}} - \widehat{T}_{F_{(\sigma,\delta,\theta)}}$.

(ii) Now, we only consider couples of trees $(\sigma(t,t'),\delta(u,u'))$, with $\pi(t) = \pi(t')$. With every couple $(\sigma(t,t'),\delta(u,u')) \in \widehat{T}_{E_{(\sigma,\delta,\theta)}} - \widehat{T}_{F_{(\sigma,\delta,\theta)}}$ we associate the sets $C_1 = \{(k_j,k'_j)$ such that $\exists k \in F, k(\sigma(x_1,x_2)) \to \delta(k_j(x_1),k'_j(x_2))$ is a rule of T and $(t,u) \notin \widehat{T}_{k_j}\}$ and $C_2 = \{(k_j,k'_j)$ such that $\exists k \in F, k(\sigma(x_1,x_2)) \to \delta(k_j(x_1),k'_j(x_2))$ is a rule of T and $(t', u') \notin \widehat{T}_{k'_j}\}$. So, for any couple of sets C_1 and C_2, for any rule $k(\sigma(x_1,x_2)) \to \delta(k_j(x_1),k'_j(x_2))$, we have either $(k_j,k'_j) \in C_1$ or $(k_j,k'_j) \in C_2$. Assume that $\widehat{T}_{E_{(\sigma,\delta,\theta)}} - \widehat{T}_{F_{(\sigma,\delta,\theta)}}$ is infinite. Then because T is a finite states transducer, there exist $(\sigma(t,t'),\delta(u,u'))$ and $(\sigma(v,v'),\delta(w,w'))$ in $\widehat{T}_{E_{(\sigma,\delta,\theta)}} - \widehat{T}_{F_{(\sigma,\delta,\theta)}}$

- computed from a state q of E with a same rule $q(\sigma(x_1,x_2)) \to \delta(q_i(x_1),q'_i(x_2))$,
- associated with the same sets C_1 and C_2,
- and such that $\pi(t) \neq \pi(v)$.

Let us consider the couple $(\sigma(t,v'),\delta(u,w'))$.

It is computed from E by using the rule $q(\sigma(x_1,x_2)) \to \delta(q_i(x_1),q'_i(x_2))$ and so it is in $\widehat{T}_{E_{(\sigma,\delta,\theta)}}$. But it cannot be computed from any state k of F by using the "initial transformation" (σ,δ,θ), because for any rule $k(\sigma(x_1,x_2)) \to \delta(k_j(x_1),k'_j(x_2))$, we have either $(t,u) \notin \widehat{T}_{k_j}$ or $(v',w') \notin \widehat{T}_{k_j}$. So $(\sigma(t,v'),\delta(u,w')) \in \widehat{T}_{E_{(\sigma,\delta,\theta)}} - \widehat{T}_{F_{(\sigma,\delta,\theta)}}$ which contradicts (i) above and so the difference $\widehat{T}_{E_{(\sigma,\delta,\theta)}} - \widehat{T}_{F_{(\sigma,\delta,\theta)}}$ is finite. □

Remark. Only letters of rank at most 2 are considered in this proof but there are no significant differences for the other cases. We obtain the same kind of lemma for letters of rank greater than 2 [1]. The following example illustrates the case σ of rank 2 and δ of rank 3.

Example. Let T and T' be two transducers of **NdT-LL** defined by:

T:
$q(\sigma(x,y)) \to \delta(q_1(x),q_2(y),q_1(x))$
$q_1(a(x)) \to a(q_1(x))$ $\quad$ $q_1(\bar{a}) \to \bar{a}$
$q_2(\alpha(x)) \to \alpha(q_2(x))$ $\quad$ $q_2(\bar{\alpha}) \to \bar{\alpha}$

T':
$k(\sigma(x,y)) \to \delta(k_1(x),k_2(y),k'_1(x))$ $\quad$ $k(\sigma(x,y)) \to \delta(k_1(x),k'_2(y),k_1(x))$
$k(\sigma(x,y)) \to \delta(k_1(x),k_3(y),k_4(y))$
$k_1(a(x)) \to a(k_1(x))$ $\quad$ $k_2(\alpha(x)) \to \alpha(k_2(x))$
$k_1(\bar{a}) \to \bar{a}$ $\quad$ $k_2(\bar{\alpha}) \to \bar{\alpha}$
$k'_1(a(x)) \to a(k'_{11}(x))$ $\quad$ $k'_2(\alpha(x)) \to \alpha(k'_{22}(x))$
$k'_{11}(a(x)) \to a(k'_{11}(x))$ $\quad$ $k'_{22}(\alpha(x)) \to \alpha(k'_{22}(x))$
$k'_{11}(\bar{a}) \to \bar{a}$ $\quad$ $k'_{22}(\bar{\alpha}) \to \bar{\alpha}$
$k_3(\bar{\alpha}) \to \bar{\alpha}$ $\quad$ $k_4(\bar{\alpha}) \to \bar{a}$

The difference $\widehat{T}_{q_{(\sigma,\delta,\theta)}} - \widehat{T}_{k_{(\sigma,\delta,\theta)}}$, with $\theta = \langle 2; 1, 2, 1\rangle$, is reduced to the couple $(\sigma(\bar{a}, \bar{\alpha}), \delta(\bar{a}, \bar{\alpha}, \bar{a}))$.

Corollary 4.1. *Let T be a non-deleting letter-to-letter top-down tree transducer. There exists a natural number, denoted by Λ, such that for any globally equivalent sets of states E and F, for any initial transformation (σ, δ, θ), as soon as $\pi(t) > \Lambda$, we have $(t,u) \in \widehat{T}_{E_{(\sigma,\delta,\theta)}} \Leftrightarrow (t,u) \in \widehat{T}_{F_{(\sigma,\delta,\theta)}}$.*

Proof. From Lemma 4.1 we conclude that for any globally equivalent sets of states E and F of an infinitary transducer T of **NdT-LL**, for any initial transformation (σ, δ, θ), there exists a natural number $\lambda_{(E,F),(\sigma,\delta,\theta)}$ such that, as soon as $\pi(t) > \lambda_{(E,F),(\sigma,\delta,\theta)}$, we have $(t,u) \in \widehat{T}_{E_{(\sigma,\delta,\theta)}} \Leftrightarrow (t,u) \in \widehat{T}_{F_{(\sigma,\delta,\theta)}}$.

Now let λ be an upper bound for these integers $\lambda_{(E,F),(\sigma,\delta,\theta)}$ obtained for all globally equivalent sets of states and for all final transformations (σ, δ, θ) and let N_F be the number of finitary states of T. We define Λ as the natural number $Sup(\lambda, N_F)$. □

In order to eliminate the differences $\widehat{T}_{E_{(\sigma,\delta,\theta)}} - \widehat{T}_{F_{(\sigma,\delta,\theta)}}$ (for E and F globally equivalent sets of states of a transducer T of **NdT-LL**) we will consider as atomic the rewritings of the trees of these finite sets. It is formalized by the following construction. We do not give an algorithm to compute this bound Λ but, even if the constructions are valid for any natural number Λ, the correctness of the results is connected to a large enough value of this integer.

4.2. *Λ-Semi-normalized forms*

For any natural number Λ, we associate with any non-deleting letter-to-letter top-down transducer $T = \langle \Sigma, \Delta, Q, F, R\rangle$ its Λ-semi-normalized form $T^\Lambda = \langle \Sigma^\Lambda, \Delta^\Lambda, Q^\Lambda, F^\Lambda, R^\Lambda\rangle$ and constructed as follows.

In order to control the height of the trees which are computed, for any state q of T, we will find in T^Λ the state $q^{<\Lambda}$, from which only trees of height less than Λ (identified with new letters) are computed and the state q^Λ for the other trees. So, $Q^\Lambda = \{q^{<\Lambda}, q^\Lambda \mid q \in Q\}$ and $F^\Lambda = \{q^{<\Lambda}, q^\Lambda \mid q \in F\}$.

For every couple of trees $(t,u) \in \widehat{T}_q$ such that $\pi(t) \leqslant \Lambda$, we construct a new rule of the form $q^{<\Lambda}(t) \to u$ if $\pi(t) < \Lambda$ or of the form $q^\Lambda(t) \to u$ if $\pi(t) = \Lambda$ where the ground trees t and u are identified with new nullary symbols. We also adapt the "non-ground" rules of T and we get as rules of T^Λ:

- $q^\Lambda(\sigma(x)) \to \delta(q_i^\Lambda(x))$ for any rule $q(\sigma(x)) \to \delta(q_i(x))$ of T,
- $q^\Lambda(\sigma(x)) \to \delta(q_i^\Lambda(x), q_j^\Lambda(x))$ for any rule $q(\sigma(x)) \to \delta(q_i(x), q_j(x))$ of T,

- $q^\Lambda(\sigma(x_1,x_2)) \to \delta(q_i^\Lambda(x_{\theta(1)}), q_j^\Lambda(x_{\theta(2)}))$
  $q^\Lambda(\sigma(x_1,x_2)) \to \delta(q_i^{<\Lambda}(x_{\theta(1)}), q_j^\Lambda(x_{\theta(2)}))$
  $q^\Lambda(\sigma(x_1,x_2)) \to \delta(q_i^\Lambda(x_{\theta(1)}), q_j^{<\Lambda}(x_{\theta(2)}))$

  } for any rule $q(\sigma(x_1,x_2)) \to \delta(q_i(x_{\theta(1)}), q_j(x_{\theta(2)}))$ of T

Computations in T and T^{Λ} are nearly identical with the only slight difference that when computations of trees and subtrees of height less than Λ are realized in several steps in T, they are realized in T^{Λ} in one step. So we get:

Proposition 4.1. *Let T be a non-deleting letter-to-letter top-down tree transducer. For any natural number Λ, for any sets of states E and F, $\widehat{T}_E = \widehat{T}_F \Leftrightarrow \widehat{T}_E^{\Lambda} = \widehat{T}_F^{\Lambda}$.*

Moreover we have:

Lemma 4.2. *Let T be a non-deleting letter-to-letter top-down transducer. There exists a natural number Λ such that for any couple (E,F) of globally equivalent sets of states, for initial transformation (σ,δ,θ), $\widehat{T}^{\Lambda}_{E_{(\sigma,\delta,\theta)}} = \widehat{T}^{\Lambda}_{F_{(\sigma,\delta,\theta)}}$.*

Proof (*Hint*). As a consequence of the construction of T^{Λ}, each term t over Σ of height less than or equal to Λ is now of height 0 over Σ^{Λ}. Then with Corollary 4.1 we immediately conclude. □

Remark. To avoid a multiplication of notations and because the context always allows to decide what transducer, among T and T^{Λ}, is concerned, for any set of states E of a non-deleting letter-to-letter top-down tree transducer T, the set of states $\{q^{\Lambda}, q^{<\Lambda} \mid q \in E\}$ of T^{Λ} is also denoted by E.

We now prove that, when Λ is large enough, from two globally equivalent sets of states E and F, for each $(t,u) \in \widehat{T}_E^{\Lambda} = \widehat{T}_F^{\Lambda}$, there exist computations of (t,u) in T_E^{Λ} and T_F^{Λ} such that each node of the input tree t is rewritten with the same torsion in both computations. In order to formalize this fact, we introduce a new form of the transducer for which the torsions applied in a computation are encoded in the node of the output tree.

Construction of $T^{\Lambda,d}$: From $T^{\Lambda} = \langle \Sigma^{\Lambda}, \Delta^{\Lambda}, Q^{\Lambda}, F^{\Lambda}, R^{\Lambda} \rangle$, non-deleting letter-to-letter top-down transducer, we construct $T^{\Lambda,d} = \langle \Sigma^{\Lambda}, \Delta^{\Lambda,d}, Q^{\Lambda}, F^{\Lambda}, R^{\Lambda,d} \rangle$ as follows:

- $q(t) \to u_{id}$ is a rule of $R^{\Lambda,d}$ and $u_{id} \in \Delta^{\Lambda,d}$ if and only if $q(t) \to u$ is a rule of R, with $t \in \Sigma^{\Lambda}$ and $u \in \Delta^{\Lambda}$
- $q(t(x_1,\ldots,x_n)) \to u_\theta(q_1(x_{\theta(1)}),\ldots,q_m(x_{\theta(m)}))$, $n \in \{1,2\}$, is a rule of $R^{\Lambda,d}$ and $u_\theta \in \Delta^{\Lambda,d}$ if and only if $q(t(x_1,\ldots,x_n)) \to u(q_1(x_{\theta(1)}),\ldots,q_m(x_{\theta(m)}))$ is a rule of R^{Λ}.

For any computation of (t,u) in $\widehat{T}^{\Lambda}$, we denote by (t,u^d) the corresponding couple of $T^{\Lambda,d}$. So we can now state the following lemmas:

Lemma 4.3. *Let T be a non-deleting letter-to-letter top-down transducer and let E and F be two globally equivalent sets of states of T. There exists a natural number Λ such that for any couple of trees (t,u) of $\widehat{T}_E^{\Lambda}$ $(= \widehat{T}_F^{\Lambda})$, for any prefix t_0 of t, for*

any $q \in E$, for any computation

$$q(t_0(t_1,\ldots,t_n)) \xrightarrow{*}_{T^{\Lambda,d}} u_0^d(q_1(t_{\theta(1)}),\ldots,q_m(t_{\theta(m)})) \xrightarrow{*}_{T^{\Lambda}} u_0^d(u_1,\ldots,u_m)$$

there exist a state $k \in F$ and states $k_1,\ldots,k_m$ such that

$$k(t_0(t_1,\ldots,t_n)) \xrightarrow{*}_{T^{\Lambda,d}} u_0^d(k_1(t_{\theta(1)}),\ldots,k_m(t_{\theta(m)})) \xrightarrow{*}_{T^{\Lambda}} u_0^d(u_1,\ldots,u_m).$$

Proof. It is by induction on the prefix t_0 of t. Let Λ be the natural number defined in Corollary 4.1.

- By lemma 4.2, it is obvious that for any (t,u), property is true when the prefix of t is a letter of Σ^{Λ}.
- Assume now that for trees of the form $t_0(t_1,\ldots,t_n)$ in $\widehat{T}_E^{\Lambda}$, property is true for the prefix t_0 and let us show that it is true again for a larger prefix.

For any $j \in [n]$, we consider the sets

$$Q_j = \{q_j \mid q(t_0(x_1,\ldots,x_n)) \xrightarrow{*}_{T^{\Lambda,d}} u_0^d(q_1(x_{\theta(1)}),\ldots,q_j(x_{\theta(j)}),\ldots,q_m(x_{\theta(m)}))$$

with $q \in E$ and $\forall i \in [m]$, $\widehat{T}_{q_i} \neq \emptyset\}$ and

$$K_j = \{k_j \mid k(t_0(x_1,\ldots,x_n)) \xrightarrow{*}_{T^{\Lambda,d}} u_0^d(k_1(x_{\theta(1)}),\ldots,k_j(x_{\theta(j)}),\ldots,k_m(x_{\theta(m)}))$$

with $k \in F$ and $\forall i \in [m], \widehat{T}_{k_i} \neq \emptyset\}$.

First, let us show that, for any $j \in [m]$, Q_j and K_j are globally equivalent sets of states. In fact, if Q_j and K_j were not globally equivalent, there would exist at least one couple $(\bar{\tau},\bar{\mu})$ in $\widehat{T}_{Q_j}^{\Lambda} - \widehat{T}_{K_j}^{\Lambda}$, that is to say a computation $q(t_0(\bar{t}_1,\ldots,\bar{\tau},\ldots,\bar{t}_n)) \xrightarrow{*}_{T^{\Lambda,d}} u_0^d(q_{i_1}(\bar{t}_{\theta(1)}),\ldots,q_j(\bar{\tau}),\ldots,q_{i_m}(\bar{t}_{\theta(m)}))$ and $\bar{t}_{\theta(i)} \xrightarrow{*}_{T^{\Lambda}} u_i$ with $q \in E$ and $q_j \in Q_j$, when for any computation $k(t_0(\bar{t}_1,\ldots,\bar{\tau},\ldots,\bar{t}_n)) \xrightarrow{*}_{T^{\Lambda,d}} u_0^d(k_{i_1}(\bar{t}_{\theta(1)}),\ldots,k_{i_j}(\bar{\tau}),\ldots,k_{i_n}(\bar{t}_{\theta(n)}))$ with $k \in F$ and $k_{i_j} \in K_j$, $(\bar{\tau},\bar{\mu})$ cannot be obtained from k_{i_j}. It contradicts the assumption that property holds for t_0 and so the sets of states Q_j and K_j are globally equivalent. Let $t_{\theta(i)} = \sigma(t'_1,\ldots,t'_l)$. So $q(t_0(t_1,\ldots,t_n)) \xrightarrow{*}_{T^{\Lambda,d}} u_0^d(\ldots,q_i(\sigma(t'_1\ldots,t'_l)),\ldots) \xrightarrow{*}_{T^{\Lambda}} u_0^d(\ldots,\delta(u'_1\ldots u'_p),\ldots)$ and $k(t_0(t_1,\ldots,t_n)) \xrightarrow{*}_{T^{\Lambda,d}} u_0^d(\ldots,k_i(\sigma(t'_1\ldots,t'_l)),\ldots) \xrightarrow{*}_{T^{\Lambda}} u_0^d(\ldots\delta(u'_1\ldots u'_p)\ldots)$. We have $q_i \in Q_i$ and $k_i \in K_i$ and $(\sigma(t'_1,\ldots,t'_l),\delta(u'_1,\ldots,u'_p)) \in \widehat{T}_{Q_i}^{\Lambda} = \widehat{T}_{K_i}^{\Lambda}$ as Q_i and K_i are globally equivalent. Consequently, by Lemma 4.2, for any initial torsion μ, $\widehat{T}_{Q_{i(\sigma,\delta,\mu)}}^{\Lambda} - \widehat{T}_{K_{i(\sigma,\delta,\mu)}}^{\Lambda} = \emptyset$. Finally, we obtain

$$\begin{aligned} &q(t_0(t_1,\ldots,t_n)) \xrightarrow{*}_{T^{\Lambda,d}} u_0^d(\ldots,q_i(\sigma(t'_1\ldots,t'_l)),\ldots) \\ &\quad \to_{T^{\Lambda}} u_0^d(\ldots,\delta(q'_1(t'_{\mu(1)}),\ldots,q'_p(t'_{\mu(p)})),\ldots) \xrightarrow{*}_{T^{\Lambda}} u_0^d(\ldots,\delta(u'_1\ldots u'_p),\ldots) \end{aligned}$$

and

$$\begin{aligned} &k(t_0(t_1,\ldots,t_n)) \xrightarrow{*}_{T^{\Lambda,d}} u_0^d(\ldots,k_i(\sigma(t'_1\ldots,t'_l)),\ldots) \\ &\quad \to_{T^{\Lambda}} u_0^d(\ldots,\delta(k'_1(t'_{\mu(1)}),\ldots,k'_p(t'_{\mu(p)})),\ldots) \xrightarrow{*}_{T^{\Lambda}} u_0^d(\ldots,\delta(u'_1\ldots u'_p),\ldots). \end{aligned}$$

Thus,

$$q(t_0(t_1,\ldots,t_n)) \xrightarrow{*}_{T^{\Lambda,d}} u_0^d(\ldots,q_i(\sigma(t'_1\ldots,t'_l)),\ldots)$$
$$\to_{T^{\Lambda,d}} u_0^d(\ldots,\delta_\mu(q'_1(t'_{\mu(1)}),\ldots,q'_p(t'_{\mu(p)})),\ldots) \xrightarrow{*}_{T^{\Lambda}} u_0^d(\ldots,\delta_\mu(u'_1\ldots u'_p),\ldots)$$

and

$$k(t_0(t_1,\ldots,t_n)) \xrightarrow{*}_{T^{\Lambda,d}} u_0^d(\ldots,k_i(\sigma(t'_1\ldots,t'_l)),\ldots)$$
$$\to_{T^{\Lambda,d}} u_0^d(\ldots,\delta_\mu(k'_1(t'_{\mu(1)}),\ldots,k'_p(t'_{\mu(p)})),\ldots) \xrightarrow{*}_{T^{\Lambda}} u_0^d(\ldots,\delta_\mu(u'_1\ldots u'_p),\ldots). \quad \square$$

Proposition 4.2. *Let T be a transducer of* **NdT-LL**. *There exists a natural number Λ such that, for any sets of states E and F, $\widehat{T}_E^{\Lambda,d} = \widehat{T}_F^{\Lambda,d} \Leftrightarrow \widehat{T}_E = \widehat{T}_F$.*

Proof (*Hint*) By Proposition 4.1, we have, for any natural number Λ, $\widehat{T}_E^{\Lambda} = \widehat{T}_F^{\Lambda} \Leftrightarrow \widehat{T}_E = \widehat{T}_F$. Now, we have obviously $\widehat{T}_E^{\Lambda,d} = \widehat{T}_F^{\Lambda,d} \Rightarrow \widehat{T}_E = \widehat{T}_F$. In the other direction, we apply the previous lemma (using as prefix of a tree the tree itself). $\square$

Let us note that, if some input subtrees are duplicated in a computation of a couple (t,u) of $T_E^{\Lambda,d}$, Proposition 4.2 allows us to claim that there exists a computation of (t,u) in $T_F^{\Lambda,d}$ which duplicates the same subtrees.

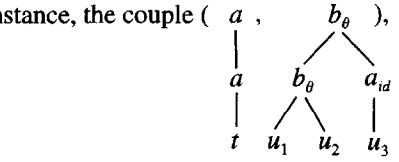
From these Λ-normalized forms, for which the torsions are expressed in the nodes of the output trees, we now construct automata with equivalence constraints.

4.3. Automaton associated with the Λ-normalized form of a transducer

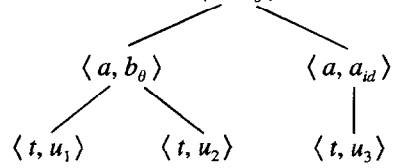
In this section, for any natural number Λ we associate with the Λ-normalized form $T^{\Lambda,d}$ of any non-deleting letter-to-letter top-down transducer T a bottom-up automaton with equivalence constraints, denoted by A^{Λ}, which recognizes encodings of the couples of trees of $\widehat{T}^{\Lambda,d}$.

As the torsions are expressed in the nodes of the output trees, it is no more useful to apply them. Because of the non-linearity, to build a "superposition" of a given input tree and of one of the output trees associated with it, whenever a non-linear rule is applied, each duplicated subtree will be "superposed" with its image.

For instance, the couple (a , b_θ), (with $\theta = \langle 1;1,1 \rangle$), of a tree transformation

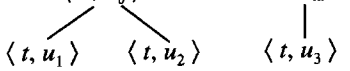


will be encoded in $\langle a, b_\theta \rangle$.



Note that, because a non-linear transducer can duplicate an input subtree and then process the copies differently, we obtain, from the same input tree t, different couples of trees (t,u_1), (t,u_2), (t,u_3). These couples are encoded into equivalent trees denoted by $< t,u_1\rangle$, $\langle t,u_2\rangle$, $\langle t,u_3\rangle$.

For instance, the superpositions $\langle a, b_\theta\rangle$ and $\langle a, a_{id}\rangle$ are "equivalent".



Using a noetherian and confluent non-deleting letter-to-letter rewriting system which associate with each superposition $\langle t,u\rangle$ the tree t, we can check whether or not two superpositions $\langle t,u\rangle$ and $\langle t',u'\rangle$ are "equivalent".

So any non-linear rule can be translated into a transition with equivalence constraints. We now successively define the superposition and the construction of the automaton.

Superposition: First of all, let us define the alphabet used in the superpositions.

Definition. The product $\Sigma\otimes\Delta$ of two ranked alphabets Σ and Δ is the subset of the cartesian product $\Sigma\times\Delta$ composed of couples of letters $\langle\sigma,\delta\rangle$ where $\sigma\in\Sigma$, $\delta\in\Delta$ and $\rho(\sigma)\leqslant\rho(\delta)$. The rank of a letter of $\Sigma\otimes\Delta$ is defined as follows:

Rank of $\sigma\in\Sigma$	Rank of $\delta\in\Delta$	Rank of $\langle\sigma,\delta\rangle$
0	0	0
1	1	1
1	2	2
2	2	2

In order to encode every couple of trees (t,u) of $T_{\Sigma^1}\times T_{\Delta^{1,d}}$ in a tree of $T_{\Sigma^1\otimes\Delta^{1,d}}$, denoted by $[t,u]$, we define the following partial function *superposition* such that $[t,u]=superposition(t,u)$

```
Function superposition(t,u)
begin
   if π(t) = 0 and u = δ_id
   then return ⟨t,u⟩
   else
      case: t = α(t1) and u = δ_id(u1)
      return ⟨α,δ_id⟩(superposition(t1,u1))
      case: t = α(t1) and u = δθ(u1,u2), θ = ⟨1;1,1⟩
      return ⟨α,δθ⟩(superposition(t1,u1),superposition(t1,u2))
      case: t = α(t1,t2) and u = δθ(u1,u2) where θ is a bijection on [2]
      return ⟨α,δθ⟩(superposition(t1,u_θ(1)),superposition(t2,u_θ(2)))
      otherwise: no output
   end if
end
```

Construction of the rewriting system τ: We now define the 1-reduced rewriting system τ:

Letter of $\Sigma^{\Lambda} \otimes \Delta^{\Lambda,d}$	$\rho(\sigma)$	$\rho(\delta)$	Rule of τ
$\langle \sigma, \delta_{id} \rangle$	0	0	$\langle \sigma, \delta_{id} \rangle \to \sigma$
$\langle \sigma, \delta_{id} \rangle$	1	1	$\langle \sigma, \delta_{id} \rangle(x) \to \sigma(x)$
$\langle \sigma, \delta_{\theta} \rangle$, θ bijection on [2]	2	2	$\langle \sigma, \delta_{\theta} \rangle(x, y) \to \sigma(x, y)$
$\langle \sigma, \delta_{\theta} \rangle$, $\theta = \langle 1; 1, 1 \rangle$	1	2	$\langle \sigma, \delta_{\theta} \rangle(x, x) \to \sigma(x)$

By induction on the height of the trees, we easily prove that for any $w \in T_{\Sigma^{\Lambda} \otimes \Delta^{\Lambda,d}}$ there exist $t \in T_{\Sigma^{\Lambda}}$ and $u \in T_{\Delta^{\Lambda,d}}$ such that $w = [t, u]$ and $w \xrightarrow{*}_{\tau} t$.

Construction of the automaton A^{Λ}: From $T^{\Lambda,d} = \langle \Sigma^{\Lambda}, \Delta^{\Lambda,d}, Q^{\Lambda}, F^{\Lambda}, R^{\Lambda,d} \rangle$, we construct the automaton $A^{\Lambda} = \langle \Sigma^{\Lambda} \otimes \Delta^{\Lambda,d}, Q^{\Lambda}, F^{\Lambda}, \mathscr{R}^{\Lambda} \rangle$ of $\mathbf{REC}_{\tau}$ as follows:

Rule of $R^{\Lambda,d}$	Rule of $\mathscr{R}^{\Lambda}$
$q(\sigma) \to \delta_{id}$	$\langle \sigma, \delta_{id} \rangle \to q$
$q_i(\sigma(x)) \to \delta_{id}(q_j(x))$	$\langle \sigma, \delta_{id} \rangle(q_j) \to q_i$
$q(\sigma(x_1, x_2)) \to \delta_{\theta}(q_1(x_{\theta(1)}), q_2(x_{\theta(2)}))$	$\langle \sigma, \delta_{\theta} \rangle(q_1, q_2) \to q$
$q(\sigma(x)) \to \delta_{\theta}(q_1(x), q_2(x))$	$\langle \sigma, \delta_{\theta} \rangle(q_1, q_2)[1, 2] \to q$

Lemma 4.4. *Let* A^{Λ} *be the automaton of* $\mathbf{REC}_{\tau}$ *associated with the* Λ*-normalized form* $T^{\Lambda,d}$. *We have* $q(t) \xrightarrow{*}_{T^{\Lambda,d}} u \Leftrightarrow [t, u] \xrightarrow{*}_{A^{\Lambda}} q$.

Proof. The proof is by induction on the terms. There are several cases, but we only consider two. Some obvious adaptations are necessary for the other ones.

Case 1: Let $t = \sigma(t_1, t_2)$. $q(\sigma(t_1, t_2)) \xrightarrow{*}_{T^{\Lambda,d}} \delta_{id}(q_1(t_1), q_2(t_2)) \xrightarrow{*}_{T^{\Lambda,d}} \delta_{id}(u_1, u_2)$ (we have a similar proof when the torsion θ is the permutation $\langle 2; 2, 1 \rangle$) implies that

(i) $q(\sigma(x_1, x_2)) \to \delta_{id}(q_1(x_1), q_2(x_2))$ is a rule of $T^{\Lambda,d}$,

(ii) $q_1(t_1) \xrightarrow{*}_{T^{\Lambda,d}} u_1$ and $q_2(t_2) \xrightarrow{*}_{T^{\Lambda,d}} u_2$.

By the induction hypothesis from (ii) we obtain that $[t_1, u_1] \xrightarrow{*}_{A^{\Lambda}} q_1$ and $[t_2, u_2] \xrightarrow{*}_{A^{\Lambda}} q_2$. Now, by construction, from the rule $q(\sigma(x_1, x_2)) \to \delta_{id}(q_1(x_1), q_2(x_2))$ of $T^{\Lambda,d}$, we obtain the rule $\langle \sigma, \delta_{id} \rangle(q_1, q_2) \to q$ in A^{Λ}. And so we conclude that $\langle \sigma, \delta \rangle([t_1, u_1], [t_2, u_2]) \xrightarrow{*}_{A^{\Lambda}} q$.

Conversely $[t, u] = \langle \sigma, \delta_{id} \rangle([t_1, u_1], [t_2, u_2]) \xrightarrow{*}_{A^{\Lambda}} q$ implies that

(i) there exists a rule of the form $\langle \sigma, \delta_{id} \rangle(q_1, q_2) \to q$ in A^{Λ},

(ii) and $[t_1, u_1] \xrightarrow{*}_{A^{\Lambda}} q_1$ and $[t_2, u_2] \xrightarrow{*}_{A^{\Lambda}} q_2$.

By the induction hypothesis from (ii) we obtain that $q_1(t_1) \xrightarrow{*}_{T^{\Lambda,d}} u_1$ and $q_2(t_2) \xrightarrow{*}_{T^{\Lambda,d}} u_2$. Now by the construction of A^{Λ}, we know that the rule $\langle \sigma, \delta_{id} \rangle(q_1, q_2) \to q$ in A^{Λ} comes from the rule $q(\sigma(x_1, x_2)) \to \delta_{id}(q_1(x_1), q_2(x_2))$ of $T^{\Lambda,d}$. So we conclude that $q(\sigma(t_1, t_2)) \xrightarrow{*}_{T^{\Lambda,d}} \delta_{id}(q_1(t_1), q_2(t_2)) \xrightarrow{*}_{T^{\Lambda,d}} \delta_{id}(u_1, u_2)$.

Case 2: Let $t = \sigma(t')$. $q(\sigma(t')) \xrightarrow{*}_{T^{\Lambda,d}} \delta_{\theta}(q_1(t'), q_2(t')) \xrightarrow{*}_{T^{\Lambda,d}} \delta_{\theta}(u_1, u_2) \Rightarrow$

(i) $q(\sigma(x)) \to \delta_{\theta}(q_1(x), q_2(x))$ is a rule of $T^{\Lambda,d}$,

(ii) $q_1(t') \xrightarrow{*}_{T^{\Lambda,d}} u_1$ and $q_2(t') \xrightarrow{*}_{T^{\Lambda,d}} u_2$.

By the induction hypothesis from (ii) we obtain that $[t',u_1] \xrightarrow{*}_{A^\Lambda} q_1$ and $[t',u_2] \xrightarrow{*}_{A^\Lambda} q_2$. Moreover the superpositions $[t',u_1]$ and $[t',u_2]$ are equivalent. Now by construction, from the rule $q(\sigma(x)) \to \delta_\theta(q_1(x),q_2(x))$ of $T^{\Lambda,d}$, we obtain the rule $\langle \sigma, \delta_\theta \rangle (q_1,q_2)[1,2] \to q$ in A^Λ. And so we conclude that $\langle \sigma, \delta_\theta \rangle([t',u_1],[t',u_2]) \xrightarrow{*}_{A^\Lambda} q$. Conversely $[t,u] = \langle \sigma, \delta_\theta \rangle([t',u_1],[t',u_2]) \xrightarrow{*}_{A^\Lambda} q$ implies that

(i) there exists a rule of the form $\langle \sigma, \delta_\theta \rangle (q_1,q_2)[1,2] \to q$ in A^Λ,

(ii) and $[t',u_1] \xrightarrow{*}_{A^\Lambda} q_1$ and $[t',u_2] \xrightarrow{*}_{A^\Lambda} q_2$.

By the induction hypothesis from (ii) we obtain that $q_1(t') \xrightarrow{*}_{T^{\Lambda,d}} u_1$ and $q_2(t') \xrightarrow{*}_{T^{\Lambda,d}} u_2$. Now by the construction of A^Λ, we know that the rule $\langle \sigma, \delta_\theta \rangle (q_1,q_2)[1,2] \to q$ in A^Λ comes from the rule $q(\sigma(x)) \to \delta_\theta(q_1(x),q_2(x))$ of $T^{\Lambda,d}$. So we conclude that $q(\sigma(t')) \xrightarrow{*}_{T^{\Lambda,d}} \delta_\theta(q_1(t_1),q_2(t_2)) \xrightarrow{*}_{T^{\Lambda,d}} \delta_\theta(u_1,u_2)$. □

We can now state the following property:

Proposition 4.3. *Let T be a transducer of* **NdT-LL** *and let E and F be two sets of states of T. There exists a natural number Λ such that $\widehat{T}_E = \widehat{T}_F \Leftrightarrow \{[t,u] \mid \exists q \in E \ [t,u] \xrightarrow{*}_{A^\Lambda} q\} = \{[t,u] \mid \exists k \in F [t,u] \xrightarrow{*}_{A^\Lambda} k\}$.*

Proof (*Hint*). By Proposition 4.2, there exists an integer Λ such that $\widehat{T}_E^{\Lambda,d} = \widehat{T}_F^{\Lambda,d} \Leftrightarrow \widehat{T}_E = \widehat{T}_F$. From Lemma 4.4, we have $q(t) \xrightarrow{*}_{T^{\Lambda,d}} u \Leftrightarrow [t,u] \xrightarrow{*}_{A^\Lambda} q$. So, $\widehat{T}_E^{\Lambda,d} = \widehat{T}_F^{\Lambda,d} \Leftrightarrow \{[t,u] \mid \exists q \in E [t,u] \xrightarrow{*}_{A^\Lambda} q\} = \{[t,u] \mid \exists k \in F [t,u] \xrightarrow{*}_{A^\Lambda} k\}$. □

From this result and the decidability of equivalence for automata with equivalence constraints, we get :

Theorem 4.1. *Equivalence of non-deleting top-down letter-to-letter transducers is decidable.*

Proof (*Hint*). From Proposition 4.3 and the decidability of equivalence in $\mathbf{REC}_\tau$, we deduce that equivalence of sets of states of a transducer T is semi-decidable; non-equivalence is obviously semi-decidable. So, equivalence is decidable. Now, two transducers $T = \langle \Sigma, \Delta, Q, I, R \rangle$ and $T' = \langle \Sigma, \Delta, Q', I', R' \rangle$ (with $Q \cap Q' = \emptyset$) are equivalent if and only if their sets of initial states I and I' are globally equivalent. To conclude, we consider the transducer $\mathscr{T} = \langle \Sigma, \Delta, Q \cup Q', I \cup I', R \cup R' \rangle$. □

4.4. Other classes of letter-to-letter transducers

The method used in the previous sections to establish the decidabi lity of equivalence for non-deleting top-down transducers is powerful enough to allow various extensions. In a previous work [2], we established the following result:

Theorem 4.2. *Equivalence of linear letter-to-letter top-down transducers is decidable.*

On the other hand, our method does not allow to investigate the general (non-linear and deleting) top-down case because, as it is illustrated in the following example, Lemma 4.1 fails.

Example. Let T and T' be non-linear and deleting letter-to-letter top-down transducers defined by

$$\begin{aligned} T\colon\ & q(\sigma(x,y)) \to \delta(q_1(x), q_1(y)) \\ & q(\sigma(x,y)) \to \delta(q_1(x), q_2(x)) \\ & q(\sigma(x,y)) \to \delta(q_2(y), q_1(y)) \end{aligned}$$

$$\begin{aligned} T'\colon\ & k(\sigma(x,y)) \to \delta(q_1(x), q_2(x)) \\ & k(\sigma(x,y)) \to \delta(q_2(y), q_1(y)) \end{aligned}$$

with

$$\begin{cases} q_1(a(x)) \to a(q_1(x)) & q_1(\bar{a}) \to \bar{a} \\ q_2(a(x)) \to a(q_2(x)) & q_2(\bar{a}) \to \bar{a} \\ q_2(a(x)) \to \bar{a} \end{cases}$$

T and T' are equivalent transducers:

$$\widehat{T} = \widehat{T'} = \{(\sigma(a^m(\bar{a}), a^n(\bar{a})), \delta(a^{m'}(\bar{a}), a^{n'}(\bar{a}))) \mid m' \text{ and } n' \leqslant \mathrm{Max}\{m,n\}\}.$$

Couples of trees $(\sigma(a^m(\bar{a}), a^n(\bar{a})), \delta(a^m(\bar{a}), a^n(\bar{a})))$ obtained by the first rule of T, which is torsion-free, will be produced in T' as follows:

$$\begin{aligned} k(\sigma(a^m(\bar{a}), a^n(\bar{a}))) &\xrightarrow{*} \delta(q_1(a^m(\bar{a})), q_2(a^m(\bar{a}))) \\ &\xrightarrow{*} \delta(a^m(\bar{a}), a^n(\bar{a})) \end{aligned}$$

in the case $n \leqslant m$ and

$$\begin{aligned} k(\sigma(a^m(\bar{a}), \alpha^n(\bar{\alpha}))) &\xrightarrow{*} \delta(q_2(\alpha^n(\bar{\alpha})), q_1(\alpha^n(\bar{\alpha}))) \\ &\xrightarrow{*} \delta(a^m(\bar{a}), \alpha^n(\bar{\alpha})) \end{aligned}$$

in the case $m \leqslant n$.

To end, let us note that the general bottom-up case can be completely solved by our techniques [3].

5. Conclusion

From Theorem 4.1, we can deduce a recursive procedure for testing equivalence of **NdT-LL**; this procedure is based on semi-decision algorithms. Thus, we cannot give an estimation of its complexity but surely it would be rather bad. So, some extensions of our work immediately arise:

- designing an efficient decision procedure,

- bounding the value of the integer A (the main difficulty comes from the fact that we consider differences between subsets of tree transformations).

To end, as obvious applications of our results, let us quote for instance

- checking the relevance of a rule of a letter-to-letter transducer,
- decision of equivalence for letter-to-letter transducers modulo commutativity of some operator symbols.

Acknowledgements

We wish to thank the referee who made valuable contributions which have helped ensure the accuracy of this paper.

References

[1] Y. André, Equivalence de transducteurs lettre à lettre d'arbres, Ph.D. Thesis. University of Lille 1, 1993.
[2] Y. André, F. Bossut, Decidability of equivalence for linear letter-to-letter top-down tree transducers, in: Proc. FCT'93, Lecture Notes in Computer Science, Vol. 710, Springer, Berlin, 1993, pp. 142–151.
[3] Y. André, M. Dauchet, Decidability of equivalence for a class of nondeterministic tree transducers, RAIRO Theoret. Inform. Appl. 28 (1994) 447–463.
[4] Y. André, F. Bossut, The equivalence problem for letter-to-letter bottom-up tree transducers is solvable, in: Proc. TAPSOFT'95, Lecture Notes in Computer Science, Vol. 915, Springer, Berlin, 1995, pp. 155–171.
[5] B. Bogaert, S. Tison, Equality and disequality constraints on direct subterms in tree automata, in: Proc. STACS 1992, Lecture Notes in Computer Science, Vol. 557, Springer, Berlin, 1992, pp. 161–171.
[6] A.C. Caron, J.L. Coquidé, M. Dauchet, Encompassment Properties and Automata with Constraints, in: Proc. RTA'93, Lecture Notes in Computer Science, Vol. 690, Springer, Berlin, 1993, pp. 328–341.
[7] J. Engelfriet, Bottom-up and top-down tree transformations: a comparison, Math. System Theory 9 (1975) 198–231.
[8] J. Engelfriet, Some open questions and recent results on tree transducers and tree languages, in: R.V. Book (Ed.), Formal Language Theory, Academic Press, New York, 1980, pp. 241–286.
[9] Z. Esik, Decidability results concerning tree transducers I, Acta Cybernet. 5 (1980), Szeged, 1–20.
[10] F. Gecseg, M. Steinby, Tree Automata, Akademiai Kiado, Budapest, 1984.
[11] G. Huet, Confluent reductions: abstract properties and applications to term rewriting system, J. ACM 27 (1980) 797–821.
[12] D. Lugiez, J.L. Moysset, Complement problems and tree automata in AC-like theories, in: Proc. STACS 93, Lecture Notes in Computer Science, Vol. 665, Springer, Berlin, 1993, pp. 515–524.
[13] W.C. Rounds, Trees, transducers and transformations, Ph.D. Thesis, Standford University, 1968.
[14] H. Seidl, Equivalence of finite-valued bottom-up finite state tree transducers is decidable, in: Proc. CAAP 90, Lecture Notes in Computer Science, Vol. 431, Springer, Berlin, 1990, pp. 269–284.
[15] H. Seidl, Single-valuedness of tree transducers is decidable in polynomial time, Theoret. Comput. Sci. 106 (1992) 135–181.
[16] J.W. Thatcher, Generalized sequential machine maps, J. Comput. System Sci. 4 (1970) 339–367.
[17] Z. Zachar, The solvability of the equivalence problem for deterministic frontier-to-root tree transducers, Acta Cybernet. 4 (1978) 167–177.