Word-into-Trees Transducers with Bounded Difference

Yves ANDRE* and Francis BOSSUT

L.I.F.L., U.R.A. 369 C.N.R.S.
University of Lille 1, 59655 Villeneuve d'Ascq Cedex. France.
e-mail:{andre, bossut} @ lifl.lifl.fr
* also at University of Lille 3, I.U.T. "B" Tourcoing.

Abstract. Non-deleting Word-into-Trees Transducers with bounded difference are investigated in this paper. Informally, these transducers which produce trees from words have the property that the difference of height of any couple of trees (the input tree being a word) is bounded. We establish the fact that the tree transformations induced by such transducers have some good closure properties.

1 Introduction

We extend here a result of Elgot and Mezei [4] about rational relations with the property that the difference of length of two words in relation is bounded. These relations can be seen as the sets obtained by means of computations of 2-tape-automata with bounded delay which are also equivalent with letter-to-letter 2-automata with terminal function [7]. Mezei and Elgot showed that such rational relations are closed under intersection and set difference.
We take an interest in a class of (non-deterministic) finite state transducers which transform words into trees and verify the property of bounded difference (between the heights of any input word and its output trees). We prove here that the class of transformations induced by such transducers is closed under intersection and set difference. We cannot hope a similar result for an other class of tree transformations when even in the letter-to-letter case (obviously with bounded difference) these closure properties are not satisfied[1]
In section 3.2 using syntactic technics, we first normalize our transducers with bounded difference, following in such a way works of Frougny and Sakarovitch who propose a resynchronization of automata with bounded delay. Let us note that the transducers we obtain are not letter-to-letter transducers but transducers for which states appear at depth one in the right-hand side of the rules. We call them *flat* transducers.
In section 3.4 we show that word-into-trees letter-to-letter transducers can be simulated by automata with equivalence constraints between direct subterms.

[1] For instance let us consider the letter-to-letter transducers T_1 and T_2 defined as follows : $T_1 : q(\sigma(x,y)) \to \delta(q'(x),q'(y))$, $q'(a(x)) \to a(q'(x))$ and $q'(\bar{a}) \to \bar{a}$ and $T_2 : q(\sigma(x,y)) \to \delta(q'(y),q'(x))$, $q'(a(x)) \to a(q'(x))$ and $q'(\bar{a}) \to \bar{a}$. The intersection of the tree transformations associated with T_1 and T_2 is the set $\{(\sigma(a^n(\bar{a}),a^n(\bar{a})),\delta(a^n(\bar{a}),a^n(\bar{a}))\}$ which is not realizable by a tree transducer.

These automata belong to the general class of automata with constraints introduced by A.C. Caron [2] and denoted by **AC**. So word-into-trees letter-to letter transducers inherit the good properties of **AC**.
In the next sections (3.5 and 3.6) we express the intersection and the set difference induced by flat word-into-trees transducers in terms of intersection and set difference of transformations induced by letter-to-letter word-into-trees transducers.

2 Preliminaries

In this section, we just recall definitions and properties used in the following. We refer the reader to [8] for tree rewriting systems and to [5] 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 $k \geq 1$, X_k denotes the set of variables $\{x_1, .., x_k\}$.
Given a ranked alphabet Σ, a denumerable set X of variables and a finite set Q of unary symbols, $T_\Sigma(X)$ denotes the set of all *terms* (*trees*) over Σ and indexed by X and $T_\Sigma(Q(X))$ denotes the set of all *terms* (*trees*) over Σ and indexed by $Q(X)$, i.e. terms of the form $t(q_1(x_{i_1}), \ldots, q_n(x_{i_n}))$ (t being a linear term). For short, we denote by $T_\Sigma(Q(x))$ the set of terms $T_\Sigma(Q(\{x\}))$ for any $x \in X$. In the particular case $T_\Sigma(\emptyset)$, we will write T_Σ. Let Σ be a ranked alphabet, t be in $T_\Sigma(X)$ and $t_1, \ldots, t_n$ be trees over Σ, the result of substituting t_i for x_i in t is denoted by $t(t_1, \ldots, t_n)$. For any tree t, the *height* (or *depth*) 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 $t = \sigma(t_1, \ldots, t_n)$.
For any term t, we denote by $\mathcal{V}(t)$ the set of variables which appear in t.
A *rewriting rule* over an alphabet σ is a couple (l, r) of terms of $T_\Sigma(X)$, usually denoted $l \rightarrow r$, such that either $\pi(l) \geq 1$ 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. We write $t \rightarrow_\mathcal{S} t'$ if t is rewritten in t' by using one rule of $\mathcal{S}$. By $\xrightarrow{*}_\mathcal{S}$ we denote the reflexive and transitive closure of $\rightarrow_\mathcal{S}$.
A rewriting system $\mathcal{S}$ over an alphabet Σ is *noetherian* if there does not exist any infinite sequence $t_0 \rightarrow_\mathcal{S} t_1 \rightarrow_\mathcal{S} \ldots t_i \rightarrow_\mathcal{S} \ldots$ A rewriting system $\mathcal{S}$ is *confluent* if $\forall x, \forall y, \forall z \in T_\Sigma(X)$, $(z \xrightarrow{*}_\mathcal{S} x$ and $z \xrightarrow{*}_\mathcal{S} y) \Rightarrow \exists t \in T_\Sigma(X)$ $(x \xrightarrow{*}_\mathcal{S} t$ and $y \xrightarrow{*}_\mathcal{S} t)$. Let $\mathcal{S}$ be noetherian and confluent; the unique irreducible form of any term t is denoted by $\mathcal{S}(t)$.

2.1 Transducers

A finite state *top-down tree transducer* is a 5-tuple $T =< \Sigma, \Delta, Q, I, R >$ 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 $q(\sigma(x_1, \ldots, x_n)) \rightarrow t$ with $q \in Q$, $\sigma \in \Sigma_n$

and $t \in T_\Delta(Q(X_n))$ or of the form $q(\sigma) \to t$ with $\sigma \in \Sigma_0$ and $t \in T_\Delta$ [1].
A transducer is *flat* if , for every rule, all states appear at depth 1 in the right-hand side of the rules. A transducer is *letter-to-letter* if, for every rule, its right-hand side is of the form $\delta(q_1(x_{i_1}), \ldots, q_m(x_{i_m}))$ with $\delta \in \Delta$ and for any $j \in [m]$ $x_{i_j} \in \{x_1, \ldots, x_n\}$.
The rules define a rewriting system over $\Sigma \cup \Delta \cup Q$, 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.*
For any state q, the transformation realized from q is the set $\widehat{T_q} = \{(t,u) \in T_\Sigma \times T_\Delta \mid q(t) \overset{*}{\to} u\}$. We denote by $\widehat{T}$ the tree transformation associated with T, i.e. the set $\{(t,u) \in T_\Sigma \times T_\Delta \mid q_0(t) \overset{*}{\to} u, q_0 \in I\}$. The *domain* of a tree transformation $\widehat{T}$ is the set $\{t/(t,u) \in \widehat{T}\}$. Two transducers T and T' are *equivalent* if $\widehat{T} = \widehat{T'}$.
A transducer is *non-deleting* (resp. *linear*) if, for each rule, variables of the left-hand side appear at least (resp. at most) once in the right-hand side.

We call *height difference*, or *difference* for short, of a pair of trees (t,u) the integer $|\pi(t) - \pi(u)|$. A transducer is said to be *with bounded difference* if there exists an integer k such that the height difference of every pair of trees (t,u) of the tree transformation $\widehat{T}$ is smaller or equal to k.
Note that in the case of a non-deleting transducer T the height difference of any couple of trees (t,u) of $\widehat{T}$ can be defined as $\pi(u) - \pi(t)$ (because in this case we have $\pi(u) \geq \pi(t)$).
A finite state *word-into-trees* transducer, denoted by *wtt* for short, is a finite state transducer the input alphabet of which is composed of letters of rank 0 and of rank 1 only. Input trees can be seen as words.
In the sequel, we will consider non-deleting Word-into-Trees Transducers with Bounded Difference (note that a deleting transducer is not generally a transducer with bounded difference). We denote by $\mathbf{WTT}_r$ the class of all non-deleting Word-into-Trees Transducers with Bounded Difference.

Example
Let us consider the transducer $T = < \Sigma, \Delta, Q, I, R >$ of $\mathbf{WTT}_r$ where $\Sigma_0 = \{\bar{a}\}$, $\Sigma_1 = \{a\}$, $\Delta_0 = \{\bar{a}\}$, $\Delta_1 = \{a\}$, $\Delta_2 = \{b\}$ and let τ be a ground tree over Δ. The sets of states are $Q = \{q, q'\}$ and $I = \{q\}$. R is composed of the rules

$$\begin{cases} q(a(x)) \to b(b(q'(x), \tau), q'(x))^2 & q'(a(x)) \to b(\tau, q'(x)) \\ q'(a(x)) \to a(q'(x)) & q'(\bar{a}) \to \bar{a} \end{cases}$$

2.2 Automata with Constraints

In order to handle non-linearity, the classical notion of tree automata has been extended by adding some tests in the rules ([1], [2], [3]).

[1] Rules of the form $q(\sigma(x)) \to q'(x)$ are not allowed
[2] In these rules x stands for x_1

Here, we will use tree automata with equivalence tests between direct subterms.

An *equivalence description* on n elements is a partition of $[n]$. Let Θ 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 $j \in X$, $t_i \Theta t_j$, and for any X and Y in d with $X \neq Y$, $i \in X$, $j \in Y$ implies $\neg(t_i \Theta t_j)$.
A bottom-up *automaton with equivalence tests between direct subterms* is a 4-tuple $< \Sigma, Q, F, R >$ 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 $\sigma(q_1, .., q_n) \xrightarrow{d} q$ where d is an equivalence description on $[n]$.
Let A be such an automaton, we have $t \rightarrow_A t'$ 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 $a(q_1, \ldots, q_n) \xrightarrow{d} q$ such that $(t_i)_{i \in [n]}$ satisfies the equivalence description d. By $\xrightarrow{*}_A$ we denote the reflexive and transitive closure of $\rightarrow_A$. 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 $\mathcal{L}(A)$.

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 $\sigma(q_1, \ldots, q_n) \xrightarrow{d} q$. Using classical methods [2] we can compute a complete and deterministic automaton from any non-complete and non-deterministic one. Moreover the class **AC** is effectively closed under boolean operations : complementation, union and intersection. Especially we can construct the product of two such automata : the composition of the two rules $a(q_1, \ldots, q_n) \xrightarrow{c} q$ and $a(q'_1, \ldots, q'_n) \xrightarrow{c'} q'$ being the rule $a((q_1, q'_1), \ldots, (q_n, q'_n)) \xrightarrow{c \wedge c'} (q, q')$.

For any equivalence relation Θ, we denote by $\mathbf{REC}_\Theta$ the class of automata with equivalence tests between direct subterms where the equivalence relation is Θ.

2.3 Automata with "Full" Constraints

Let Θ be an equivalence relation. We denote by $\mathbf{REC}^f_\Theta$ the subclass of $\mathbf{REC}_\Theta$ composed of the automata such that, along any successful run, all the rules which are applied carry a *full* constraint, i.e an equivalence constraint between all the successors of the node. These rules are of the form $\alpha(q_1, \ldots, q_n) \xrightarrow{[1, \ldots, n]} q$. The class $\mathbf{REC}^f_\Theta$ verifies the following property :

Property 1. *The union, intersection and difference of tree languages recognizable by automata of $\mathbf{REC}^f_\Theta$ are also recognizable by some automaton of $\mathbf{REC}^f_\Theta$.*
Hint of proof :
The completion of an automaton does not affect its successful runs. So we consider only "complete" automata.

Let M_1 and M_2 be complete automata of $\mathbf{REC}^f_{\Theta}$. We construct both the automaton of union and intersection from the "product" of automata. For each successful run of the automata of the union or intersection, at least its "projection" on the first or the second component coincides with a successful run of M_1 or M_2, what means that a full constraint is satisfied on each node.
Now, to obtain the automaton of the difference, we build the "product" of automaton M_1 with the automaton which recognizes the complement of the set recognized by M_2. For each successful run of this automaton, its "projection" on the first component coincides with a successful run of M_1, what means that a full constraint is satisfied on each node. □

3 Non-deleting Word-into-Trees Transducers with Bounded Difference

3.1 It is decidable whether a non-deleting wtt is with bounded difference

We show that the problem of the bounded difference in $\mathbf{WTT}_r$ can be reduced to the problem of bounded difference for finite state transducers of words.
From any transducer $T =< \Sigma, \Delta, Q, q_0, R_T >$ with bounded difference, we can construct the transducer $T' =< \Sigma, \Delta', Q, q_0, R_{T'} >$ where
$q(\alpha(x)) \rightarrow t(q_1(x), \ldots, q_n(x)) \in R_T \Rightarrow q(\alpha(x)) \rightarrow \sigma_n(\delta^{l_1} q_1(x), \ldots, \delta^{l_n} q_n(x)) \in R_{T'}$ with $\sigma_n \in \Delta'_n, \delta \in \Delta'_1$ so that the $q_i(x)$'s are at the same depth in both rules.
Obviously T is with bounded difference iff T' is with bounded difference. Note that now, for each $(u, v) \in \widehat{T'}$, all the branches of v are of depth greater than $\pi(u)$.

From T', we construct $L =< \Sigma, \{\delta, \#\}, 2^Q \times Q, (\{q_0\}, q_0), R_L >$ where the set R_L is the result of the algorithm :

Beginning with the state $(\{q_0\}, q_0)$, we iterate the following procedure while new states appear.
For each new state $(\{q_1, \ldots, q_n\}, q_i)$:
• we introduce the rules $(\{q_1, \ldots, q_n\}, q_i)a(x) \rightarrow \delta^l((\{q'_1, \ldots, q'_k\}, q_{i_j})(x))$ in R_L if and only if
$$\begin{cases} \forall \lambda \in [n] \;\; \exists\; q_\lambda a(x) \rightarrow t_\lambda(q_{\lambda_1}(x), q_{\lambda_2}(x), \ldots) \in R_{T'} \text{ and} \\ \{q'_1, q'_2, \ldots, q'_k\} = \bigcup_{\lambda=1}^{n} \{q_{\lambda_1}, q_{\lambda_2}, \ldots\} \text{ and} \\ \textit{l is the depth of } q_{i_j} \textit{ in } t_i \textit{ in the rule } q_i a(x) \rightarrow t_i(q_{i_1}(x), \ldots, q_{i_j}(x), \ldots) \in R_{T'} \end{cases}$$
• we introduce the rules $(\{q_1, q_2, \ldots, q_n\}, q_i)\overline{a} \rightarrow \delta^l \#$ in R_L if and only if
$$\begin{cases} \forall \lambda \in [n] \;\; \exists\; q_\lambda \overline{a} \rightarrow t_\lambda \in R_T \\ \text{and } \textit{l is the depth of some branch of } t_j \textit{ in the rule } q_i\, \overline{a} \rightarrow t_i \in R_{T'} \end{cases}$$

It is obvious that we get the property :
$\forall (w, t) \in T_\Sigma \times T_\Delta : \;\; q_0 w \rightarrow^*_T t \Rightarrow (\{q_0\}, q_0) w \rightarrow^*_L \delta^n \#$ for all n length of some branch of t. So, if L verifies the bounded difference property then T' verifies also

this property. Conversely if $(\{q_0\}, q_0)w \rightarrow_L^* \delta^n\#$ there exists $(w,t) \in \widehat{T'}$ so that t has one branch of length n. So if L does not verify the property of bounded difference, T' does not verify it.

Property 2. *It is decidable to determine whether a word-into-trees transducer T verifies or not the property of bounded difference.*

PROOF :
T induces a transformation with bounded difference if and only if L is also a finite state transducer of words with bounded difference which is decidable (see for example [7]). □

Note that, as it is the case for any top-down tree transducer, emptiness is decidable for the transformation induced by any word-into-trees transducer.

3.2 Normalization of a transducer of $\mathbf{WTT}_r$

In this section, we show that we can associate with any transducer T of $\mathbf{WTT}_r$ a flat transducer $\mathcal{T}$ which realizes the same transformation. The idea is to substitute a flat rule for every rule of T for which the states of the right-hand side appear at a depth greater than 1 ; the delay in the construction of the output tree is memorized in new states. To construct these new states we need to define $\bar{Q}$ as the set $\{\bar{q}_i$ of rank 0 $/q_i \in Q\}$.
For instance, from the rule $q(a(x)) \rightarrow b(b(q'(x),\tau),q'(x))$ of T (in example of section 2.1), we construct the rule $q(a(x)) \rightarrow b(\boxed{b(\bar{q}',\tau)}(x), q'(x))$; the new state $\boxed{b(\bar{q}',\tau)}$ memorizing the delay in the construction of the output tree. At the next step of the transformation, from this state $\boxed{b(\bar{q}',\tau)}$, the output tree $b(.,\tau)$ will be produced, a new delay being eventually memorized.
Let p be such a new state. We will have $p(\sigma(x))$ as the left-hand side of a rule of $\mathcal{T}$ if, for every state $\bar{q}$ appearing in p, $q(\sigma(x))$ is the left-hand side of a rule of T.

Construction of a flat transducer
With $T =< \Sigma, \Delta, Q, q_0, R >$ we associate the flat transducer $\mathcal{T} =< \Sigma, \Delta, \mathcal{Q}, q_0, \mathcal{R} >$ constructed by the following algorithm :

```
begin $\mathcal{Q}_0 = \mathcal{Q}_1 = \{q_0\}$; $\mathcal{R} = \emptyset$; $n = 1$;
For every letter $\sigma$ of $\Sigma$ which is transformed from $q_0$
  Case $\sigma$ of rank 0
  From every rule $q_0(\sigma) \rightarrow \delta$ of $R$ with $\delta \in T_\Delta$, we add in $\mathcal{R}$ the rule $q_0(\sigma) \rightarrow \delta$.
  Case $\sigma$ of rank 1
  From every rule $q_0(\sigma(x)) \rightarrow \delta(u_1,\ldots,u_n)$ of $R$ where $\delta \in \Delta_n$ and for any $j \in [n]$
  $u_j \in T_\Delta(Q(x))$ (for some $j$ we can have $u_j = \tau \in T_\Delta$ or $u_j = q'(x), q' \in Q$),
  we add in $\mathcal{R}$ the rule $q_0(\sigma(x)) \rightarrow \delta(\bar{u}_1,\ldots,\bar{u}_n)$ so that for any $j \in [n]$,
  either $\bar{u}_j = u_j$ if $u_j \in T_\Delta$ or $u_j \in Q(x)$ (if $u_j = q'(x)$ then $q'$ is added to $\mathcal{Q}_n$)
  or $\bar{u}_j = \bar{p}_j(x)$; $\bar{p}_j$, obtained from $u_j$ by substituting $\bar{q}$ for $q(x)$, is a tree of $T_{\Delta(\bar{Q})}$;
  in this case, $\bar{p}_j$ is added to $\mathcal{Q}_n$.
```

```
REPEAT
  $n = n + 1$; $\mathcal{Q}_n = \mathcal{Q}_{n-1}$
  For every state $q \in \mathcal{Q}_{n-1} - \mathcal{Q}_{n-2}$
  Case $q \in Q$
     For every letter of $\Sigma$ which is transformed from q, we proceed as in the initial
     step of this algorithm.
  Case $q \notin Q$
     In this case, $q = \gamma(r_1, \ldots, r_m)$ with $\gamma \in \Delta$ and $r_1, \ldots, r_m \in T_\Delta(\bar{Q})$ (for
     some $j \in [m]$ we can have $r_j = \tau \in T_\Delta$ or $r_j = \bar{p}/p \in Q$).
     Let $\bar{p}_1, \ldots, \bar{p}_n$ be the elements of $\bar{Q}$ that appear in $q$.
     For every letter $\sigma$ of $\Sigma$ which is transformed from $q$ (i.e. which can be
     transformed in $T$ from the states $p_1, \ldots, p_n$) :
     Case $\sigma$ of rank 0
     from every set of rules of $R$ $\{p_i(\sigma) \to \delta_i, \delta_i \in T_\Delta\}$
     we add in $\mathcal{R}$ $q(\sigma) \to \gamma(\bar{r}_1, \ldots, \bar{r}_m)$ where either $\bar{r}_j = r_j$ if $r_j = \tau \in T_\Delta$
     or $\bar{r}_j$ is obtained by substituting $\delta_i$ for any $\bar{p}_i$ in $r_j$.
     Case $\sigma$ of rank 1
     From every set of rules of $R$ $\{p_i(\sigma(x)) \to u_i, i \in [n], u_i \in T_\Delta(Q(x))\}$
     we add in $\mathcal{R}$ the rule $q(\sigma(x)) \to \gamma(\bar{r}_1, \ldots, \bar{r}_m)$ with $\bar{r}_j = r_j$ if $r_j \in T_\Delta$
     or $\bar{r}_j = s_j(x)$ where $s_j$, tree over $T_\Delta(\bar{Q})$, is obtained by substituting $\bar{u}_i$ to
     every $\bar{p}_i$ in $r_j$.
UNTIL $\mathcal{Q}_n = \mathcal{Q}_{n-1}$. end
```

It is easy to observe that this algorithm will end as the transducer T from which the flat transducer $\mathcal{T}$ is constructed is a transducer with bounded difference. In such a transducer, rules for which, in the right-hand side, states appear at a depth greater than 1 can be applied only a finite number of times.

Example

Let us consider the transducer defined in section 2.1. The flat transducer equivalent with it is defined as follows :

$q(a(x)) \to b(\boxed{b(\bar{q}', \tau)}(x), q'(x))$ (Trees written into boxes are new states)

$q'(a(x)) \to a(q'(x))$ $\qquad$ $q'(a(x)) \to b(\tau, q'(x))$

$q'(\bar{a}) \to \bar{a}$

$\boxed{b(\bar{q}', \tau)}(a(x)) \to b(\boxed{b(\tau, \bar{q}')}(x), \tau)$ $\qquad$ $\boxed{b(\bar{q}', \tau)}(a(x)) \to b(\boxed{a(\bar{q}')}(x), \tau)$

$\boxed{b(\tau, \bar{q}')}(a(x)) \to b(\tau, \boxed{b(\tau, \bar{q}')}(x))$ $\qquad$ $\boxed{b(\tau, \bar{q}')}(a(x)) \to b(\tau, \boxed{a(\bar{q}')}(x))$

$\boxed{a(\bar{q}')}(a(x)) \to a(\boxed{b(\tau, \bar{q}')}(x))$ $\qquad$ $\boxed{a(\bar{q}')}(a(x)) \to a(\boxed{a(\bar{q}')}(x))$

We have $q(a(a(\bar{a}))) \to_T b(b(q'(a(\bar{a})), \tau), q'(a(\bar{a}))) \overset{*}{\to}_T b(b(a(q'(\bar{a})), \tau), a(q'(\bar{a})))$ $\overset{*}{\to}_T b(b(a(\bar{a}), \tau), a(\bar{a}))$ when we will have in the flat form of T $q(a(a(\bar{a}))) \to_{\mathcal{T}}$ $b(\boxed{b(\bar{q}', \tau)}(a(\bar{a})), q'(a(\bar{a}))) \overset{*}{\to}_{\mathcal{T}} b(b(\boxed{a(\bar{q}')}(\bar{a}), \tau), a(q'(\bar{a}))) \overset{*}{\to}_{\mathcal{T}} b(b(a(\bar{a}), \tau), a(\bar{a}))$

With this correspondence between the computations in T and the computations in $\mathcal{T}$ we can establish that T and $\mathcal{T}$ are equivalent transducers. So, we conclude

Theorem 1. ***WTT_r= flat WTT_r.***

3.3 The μ-forms of a Flat Transducer.

For any natural number μ, we associate with any flat transducer $T =< \Sigma, \Delta, Q, q_0, R_T >$ of $\mathbf{WTT}_r$ a transducer $T^\mu =< \Sigma', \Delta', Q' = Q \cup \{q'_0\}, q'_0, R_{T^\mu} >$ of $\mathbf{WTT}_r$ where $R_{T^\mu}, \Sigma', \Delta'$ are constructed as follows :

- any rule of R_T on letters of non-null arity is also a rule of R_{T^μ}; so $\Sigma'_1 = \Sigma_1$ and $\Delta \subset \Delta'$;
- for any rule of the form $q_0 a(x) \to t$ in R_T, we have the rule $q'_0 a(x) \to t$ in R_{T^μ};
- for any (u, v) in $\widehat{T}$ with $\pi(u) \leq \mu$, we have the rule $q'_0 u \to v$ in R_{T^μ} where u, v are now considered as letters of respectively Σ' and Δ';
- for any state q in Q, for any (u, v) in $\widehat{T}_q$ with $\pi(u) = \mu$, we have the rule $qu \to v$ in R_{T^μ} where u, v are now considered as letters of respectively Σ' and Δ'.

Computations in $\widehat{T}$ and $\widehat{T^\mu}$ are nearly identical with only the slight difference that computations on the input word or a suffix of the input word of length less than or equal to μ are realized in $\widehat{T^\mu}$ in one step.

3.4 Correspondence between Letter-to-letter wtt and Automata of $\mathbf{REC}^f_\Theta$

We establish, in this part, the fact that the computations of a letter-to-letter wtt can be simulated by the runs of an automata with constraints, and therefore, the transformation realized by such a transducer can be encoded into an automaton-definable set of trees.

Let us consider the class of letter-to-letter wtt from T_Σ into T_Δ. We associate with the pair of alphabets Σ, Δ the alphabet denoted $\Sigma \otimes \Delta$ such that :

$$\forall i \quad [\Sigma \otimes \Delta]_i = \Sigma_1 \times \Delta_i \quad \textit{and} \quad [\Sigma \otimes \Delta]_0 = \Sigma_0 \times \Delta_0.$$

So, we define two noetherian and confluent rewriting systems γ and $\overline{\gamma}$ composed of the following rules :

$\forall(\alpha, \beta) \in [\Sigma \otimes \Delta]_i \quad (\alpha, \beta)\underbrace{(x, x, \ldots, x)}_{i\ times} \mapsto \alpha(x)$ is a rule of γ.

$\forall(\alpha, \beta) \in [\Sigma \otimes \Delta]_0 \quad (\alpha, \beta) \mapsto \alpha$ is a rule of γ.

$\forall(\alpha, \beta) \in [\Sigma \otimes \Delta]_i \quad (\alpha, \beta)(x_1, x_2, \ldots, x_i) \mapsto \beta(x_1, x_2, \ldots, x_i)$ is a rule of $\overline{\gamma}$.

$\forall(\alpha, \beta) \in [\Sigma \otimes \Delta]_0 \quad (\alpha, \beta) \mapsto \beta$ is a rule of $\overline{\gamma}$.

Any term ω of $T_{\Sigma\otimes\Delta}$ has a normal form for γ which be denoted by $\gamma(\omega)$, and a normal form for $\overline{\gamma}$ denoted by $\overline{\gamma}(\omega)$.

Let Θ be the equivalence relation on $T_{\Sigma\otimes\Delta}$ such that $t\Theta t' \Leftrightarrow \gamma(t) = \gamma(t')$. We associate with $T =< \Sigma, \Delta, Q, q_0, R_T >$ (letter-to-letter wtt) the bottom-up automaton of $\mathbf{REC}^f_\Theta$ $M_T =< \Sigma \otimes \Delta, Q, q_0, \mathcal{T} >$ where

$$\mathcal{T} = \{(\alpha,\beta)(q_1,\ldots,q_n) \overset{[1,\ldots,n]}{\rightarrow} q \;/\; q\;\alpha(x) \rightarrow \beta(q_1(x),\ldots,q_n(x)) \in R_T\}.$$

The elements of $\mathcal{L}(M_T)$ are roughly speaking a kind of "superposition" of the components of the couples (u,v) of $\widehat{T}$. But, as the transducer T can duplicate an input and then process its copies differently, we read along all the branches of an element ω of $\mathcal{L}(M_T)$ exactly the same sequence u of labels of Σ which corresponds to the input word. The constraints of equivalence between all the successors $([1,2,\ldots,n])$ upon all the rules ensures this property. Now, if we consider only the second component of the nodes (symbols of Δ), we get an ouput tree v for the input word u as the transitions of $\mathcal{T}$ are compatible with those of R_T (we get together the input and output symbols and the transformations of states are the same). So we obtain :

Property 3. *Every letter-to-letter wtt T can be simulated by an automaton M of $\boldsymbol{REC}^f_\Theta$. What means that :*

$$\begin{cases} \forall (u,v) \in \widehat{T} \;\; \exists\, \omega \in \mathcal{L}(M_T) \text{ such that } \gamma(\omega) = u \text{ and } \overline{\gamma}(\omega) = v \text{ and} \\ \forall \omega \in \mathcal{L}(M_T) \;\; (\gamma(\omega), \overline{\gamma}(\omega)) \in \widehat{T}. \end{cases}$$

Example

Let $T =< \{a,\overline{a}\}, \{a,b,\overline{a}\}, \{q\}, q, R_T >$ be a letter-to-letter wtt where R_T contains the rules : $qa(x) \rightarrow a(q(x))|b(q(x),q(x))$ and $q\overline{a} \rightarrow \overline{a}$.
Then $M_T =< \{(a,a),(a,b),(\overline{a},\overline{a})\}, \{q\}, q, \mathcal{T} >$ where $\mathcal{T}$ contains the transitions $(a,a)q \rightarrow q \quad (a,b)(q,q) \overset{[1,2]}{\rightarrow} q$ and $(\overline{a},\overline{a}) \rightarrow q$.

$$\begin{tikzcd}[column sep=small]
 & & (a,b) \arrow[dl, no head, dotted] \arrow[drr, no head] & & & \\
 & (a,a) \arrow[d, no head] & & & (a,b) \arrow[dl, no head, dotted] \arrow[dr, no head, dotted] & \\
 & (a,b) \arrow[dl, no head, dotted] \arrow[dr, no head, dotted] & & (a,a) \arrow[d, no head] & & (a,a) \arrow[d, no head] \\
(\overline{a},\overline{a}) & & (\overline{a},\overline{a}) & (\overline{a},\overline{a}) & & (\overline{a},\overline{a})
\end{tikzcd}$$

Left-mentionned is the tree ω of $\mathcal{L}(M_T)$ corresponding to $(aaa\overline{a}, b(ab(\overline{a},\overline{a}), b(a\overline{a},a\overline{a}))) \in \widehat{T}$.
Observe that along all its branches, we read the same sequence $aaa\overline{a}$ on the first components of the labels of the nodes.

Conversely, with any automaton $M =< \Sigma \otimes \Delta, Q, F, \mathcal{T} >$ of $\mathbf{REC}^f_\Theta$, we can associate the letter-to-letter wtt $T =< \Sigma, \Delta, Q \cup \{\rho\}, \rho, R_T >$ where

$$\begin{aligned} R_T = {} & \{q\;\alpha(x) \rightarrow \beta(q_1(x),\ldots,q_n(x)) \;/\; (\alpha,\beta)(q_1,\ldots,q_n) \overset{[1,\ldots,n]}{\rightarrow} q \in \mathcal{T}\} \\ \bigcup & \{\rho\;\alpha(x) \rightarrow \beta(q_1(x),\ldots,q_n(x)) \;/\; (\alpha,\beta)(q_1,\ldots,q_n) \overset{[1,\ldots,n]}{\rightarrow} q \in \mathcal{T} \text{ and } q \in F\} \\ \bigcup & \{q\;\alpha \rightarrow \beta \;/\; (\alpha,\beta) \rightarrow q \in \mathcal{T}\}. \end{aligned}$$

So, we have obviously

Property 4. *Every automaton of $\boldsymbol{REC}^f_\Theta$ can be interpreted as a letter-to-letter wtt.*

3.5 Intersection of Flat Transducers

Let $T_1 =< \Sigma, \Delta, Q_1, q_0, R_{T_1} >$ and $T_2 =< \Sigma, \Delta, S_2, s_0, R_{T_2} >$ be two flat transducers of $\mathbf{WTT}_r$.
In the following, we will only consider normal computations of a couple (u, v) where $u = x_1 x_2 \ldots . x_n (x_i \in \Sigma)$. A computation will be "*normal*" in our sense if it begins by the rewriting of all the occurrences of x_1, next those of x_2 and so on until those of x_n and one process the rewritings of the different occurrences of x_i from left to right.
Let (u, v) be an element of $\widehat{T_1} \cap \widehat{T_2}$, ψ_1 be a normal computation of (u, v) in T_1 and ψ_2 be a normal computation of (u, v) in T_2:

I. At each step of the computation, the obtained trees t_i (resp. t'_i) can be decomposed into a tree $\tilde{t}_i$ (resp. $\tilde{t}'_i$) of $T_\Delta(X)$ composed with a n-uple $\overrightarrow{t}_i$ of trees of $Q_1(T_\Sigma)$ (resp. of $Q_2(T_\Sigma)$). So ψ_1 and ψ_2 can be developped as:
$\psi_1 : q_0\ u \rightarrow \tilde{t}_1\ \overrightarrow{t_1} \rightarrow \tilde{t}_2\ \overrightarrow{t_2} \rightarrow \ldots \rightarrow \tilde{t}_j\ \overrightarrow{t_j} \rightarrow \ldots \rightarrow v$
$\psi_2 : s_0\ u \rightarrow \tilde{t}'_1\ \overrightarrow{t'_1} \rightarrow \tilde{t}'_2\ \overrightarrow{t'_2} \rightarrow \ldots \rightarrow \tilde{t}'_j\ \overrightarrow{t'_j} \rightarrow \ldots \rightarrow v$
Suppose that there exists j such that $\tilde{t}_j \neq \tilde{t}'_j$ and for $i \in [1, j-1]$ $\tilde{t}_j = \tilde{t}'_j$. That means that, during the $j-1$ first steps, the rules applied in both computations rewrite the same symbol of Σ into the same tree of $T_\Delta(X)$, and at the j^{th} step :
- the rule used in ψ_1 is of the form :
    $q\ a(x) \rightarrow b(\tau_1, \tau_2, \ldots, \tau_n)$ with $\tau_i = q_i(x)$ or $\tau_i = r_i \in T_\Delta$
- the rule used in ψ_2 is of the form :
    $s\ a(x) \rightarrow b(\tau'_1, \tau'_2, \ldots, \tau'_n)$ with $\tau'_i = s_i(x)$ or $\tau'_i = r'_i \in T_\Delta$
but at least one $k (\in [n])$ is such that $(\tau_k = q_k(x)\ and\ \tau'_k = r'_k)$ or $(\tau'_k = s_k(x)\ and\ \tau_k = r_k)$.
The consequence is that, at this point of the computations, the length of the suffix u' of u which has not been yet transformed is less than $\pi(r_k)$(or $\pi(r'_k)$) because $q_k\ u' \rightarrow^*_{T_1} r'_k$ $(or\ s_k\ u' \rightarrow^*_{T_2} r_k)$.

II. The "μ-forms" of the transducers allow us to erase these local differences between these two computations :
Let be $\theta = max(\theta_1, \theta_2) + 1$ where $\theta_1 = max\{\pi(d_i) / \exists\ l_i \rightarrow d_i \in R_{T_1}\}$ and $\theta_2 = max\{\pi(d'_i) / \exists\ l'_i \rightarrow d'_i \in R_{T_2}\}$. Let T_1^θ and T_2^θ be the θ-forms of T_1 and T_2.
So, from ψ_1 we deduce the computation $\psi_1^\theta : q'_0\ u' \rightarrow^*_{T_1^\theta} v'$,
and from ψ_2 we deduce the computation $\psi_2^\theta : s'_0\ u' \rightarrow^*_{T_2^\theta} v'$.
Then, we have immediatly $(u', v') \in \widehat{T_1^\theta} \cap \widehat{T_2^\theta}$ and at each step of the computations ψ_1^θ and ψ_2^θ the applied rules rewrite the same symbol of Σ' into **the same** tree of $T_{\Delta'}(X)$.

III. Now, from T_1^θ and T_2^θ, we construct the letter-to-letter wtt $L_1 =< \Sigma', \Delta'', Q', q'_0, R_{L_1} >$ and $L_2 =< \Sigma', \Delta'', S', s'_0, R_{L_2} >$ in which each term of $T_{\Delta'}(X)$ which appears in the right-hand side of the rules of T_1^θ (resp. T_2^θ) is now considered as a single symbol :

- $q\ a(x) \rightarrow t(q_1(x), \ldots, q_n(x)) \in R_{T_1^\theta} \Rightarrow q\ a(x) \rightarrow t(q_1(x), \ldots, q_n(x)) \in R_{L_1}$
- $s\ a(x) \rightarrow t(s_1(x), \ldots, s_n(x)) \in R_{T_2^\theta} \Rightarrow s\ a(x) \rightarrow t(s_1(x), \ldots, s_n(x)) \in R_{L_2}$

and $t \in \Delta''$.
So, from ψ_1^θ we deduce the computation $\psi_1^{\theta,L} : q_0'\ u' \rightarrow_{L_1}^* v''$, and from ψ_2^θ we deduce the computation $\psi_2^{\theta,L} : s_0'\ u' \rightarrow_{L_2}^* v''$. Then, we have $(u', v'') \in \widehat{L_1} \cap \widehat{L_2}$.

IV. Thus, as L_1, L_2 are letter-to-letter transducers of $\mathbf{WTT}_r$, they can be simulated by automata of $\mathbf{REC}_\Theta^f$: M_{L_1} and M_{L_2} (see section 3.4).
As $\mathbf{REC}_\Theta^f$is closed under intersection, $\mathcal{L}(M_{L_1}) \cap \mathcal{L}(M_{L_2})$ is also an automaton-definable set of trees. Let $M_{L_1 \cap L_2}$ be the automaton which recognizes $\mathcal{L}(M_{L_1}) \cap \mathcal{L}(M_{L_2})$.

V. Let $\mathcal{I}$ be the canonical injection from $\Sigma' \otimes \Delta''$ into $\Sigma \times \Delta$. From $M_{L_1 \cap L_2}$, it is easy to construct a flat transducer $I_{1,2}$ of $\mathbf{WTT}_r$ from T_Σ into T_Δ such that $w \in \mathcal{L}(M_{L_1 \cap L_2}) \Leftrightarrow \mathcal{I}(w) \in \widehat{I_{1,2}}$.

Lemma 1. *The class of transformations realized by transducers of* $\boldsymbol{WTT_r}$ *is closed under intersection.*

From the previous constructions, we get $\widehat{I_{1,2}} = \widehat{T_1} \cap \widehat{T_2}$.

3.6 Set Difference

Let $T_1 =< \Sigma, \Delta, Q_1, q_0, R_{T_1} >$ and $T_2 =< \Sigma, \Delta, S_2, s_0, R_{T_2} >$ be flat transducers of $\mathbf{WTT}_r$.
We want to state that $\widehat{T_1} - \widehat{T_2}$ is also a transformation realized by a transducer of $\mathbf{WTT}_r$. Whereas consider $\widehat{T_1} - \widehat{T_2}$, we take an interest in $\widehat{T_1} - (\widehat{T_2} \cap \widehat{T_1})$ or more exactly in the elements which belong to $\widehat{T_1}$ and not to $(\widehat{T_2} \cap \widehat{T_1})$. Let $I_{1,2}$ a transducer of $\mathbf{WTT}_r$ such $\widehat{I_{1,2}} = (\widehat{T_2} \cap \widehat{T_1})$.
In order to compare computations in T_1 and in $I_{1,2}$, as previously, we use the "θ-forms" of these transducers where $\theta = max(\theta_1, \theta_2)+1$ with $\theta_1 = max\{\pi(d_i) / \exists\ l_i \rightarrow d_i \in R_{T_1}\}$ and $\theta_2 = max\{\pi(d_i') / \exists\ l_i' \rightarrow d_i' \in R_{I_{1,2}}\}$.
Following the same arguments as in the previous section, we get the following property :
The computations of a same couple in T_1^θ *and in* $I_{1,2}^\theta$ *are, with the exception of the states, the same.*
Now, from T_1^θ and $I_{1,2}^\theta$, we construct the letter-to-letter transducers $T_1^{\theta,L}$ and $I_{1,2}^{\theta,L}$ of $\mathbf{WTT}_r$ in which each term of $T_{\Delta'}$ which appear in the right-hand side of rule of T_1^θ (resp. $I_{1,2}^\theta$) is considered as a single symbol of Δ''.
These transducers can be simulated by automata $M_{T_1^{\theta,L}}$, $M_{I_{1,2}^{\theta,L}}$ of $\mathbf{REC}_\Theta^f$ that we rename respectively M_1 and $M_{1,2}$.
From M_1 and $M_{1,2}$, it is possible to construct an automaton of $\mathbf{REC}_\Theta^f$ which recognizes the elements of $\mathcal{L}(M_1)$ which do not belong to $\mathcal{L}(M_{1,2})$. Let us call this automaton M_{1-2}. This automaton can be decoded into a transducer T_{1-2} of $\mathbf{WTT}_r$.

Lemma 2. *The class of transformations induced by transducers of* $\boldsymbol{WTT_r}$ *is closed under set difference.*

HINT OF PROOF :

We prove that, for T_1, T_2 transducers of $\mathbf{WTT}_r$, we have $\widehat{T}_{1-2} = \widehat{T}_1 - \widehat{T}_2$.

• Let (u, v) be an element of $\widehat{T}_{1-2}$. It is obvious that $(u, v) \in \widehat{T}_1$, but suppose that it belongs to $\widehat{T}_2$.

Let (u', v') be a couple of $\Sigma' \times \Delta'$ which corresponds to (u, v). So the computations of (u', v') in T_1^θ and T_2^θ will be the same and then (u', v') will be encoded into a word ω which should belong to $\mathcal{L}(M_{1,2})$. So $\omega \notin \mathcal{L}(M_{1-2})$ which contradicts the hypothesis that $(u, v) \in \widehat{T}_{1-2}$.

• Conversely, let $(u, v) \in \widehat{T}_1 - \widehat{T}_2$. So (u, v) has no corresponding $(u', v') \in \widehat{T}_1^\theta \cap \widehat{T}_2^\theta$, therefore no corresponding $(u', v'') \in \widehat{T}_1^{\theta,L} \cap \widehat{T}_2^{\theta,L}$, and finally no corresponding $\omega \in \mathcal{L}(M_{T_{1,2}})$. Any corresponding ω to (u, v) is in $\mathcal{L}(M_1)$. Thus any corresponding ω to (u, v) is in $\mathcal{L}(M_{1-2})$ what means that $(u, v) \in \widehat{T}_{1-2}$.□

As a straightforward consequence of the previous result, we get our main theorem and its immediate corollary :

Theorem 2. *The class of transformations induced by transducers of* $\boldsymbol{WTT_r}$ *is closed under union, intersection and set difference.*

Corollary 1. *The equivalence of two transducers* T_1, T_2 *of* $\boldsymbol{WTT_r}$ *is decidable.*

References

[1] B. Bogaert and S. Tison. Equality and disequality constraints on direct subterms in tree automata. *Proceedings of STACS 1992.* LNCS 577, pp 161-171.

[2] A.C. Caron. Structures et décision en réécriture. *Ph. D. thesis.* University of Lille 1. 1993.

[3] A.C. Caron, J.L. Coquidé and M. Dauchet. Encompassment Properties and Automata with Constraints. *Proceedings of RTA'93.* LNCS 690, pp 328-341.

[4] C.C. Elgot and J.E. Mezei. On relations defined by generalized finite automata. In *IBM J. Res. Develop.*. Nber 9, pp 47–68, 1965.

[5] J. Engelfriet. Bottom-up and top-down tree transformations: a comparison. *Mathematical system theory.* Vol 9. pp 198-231. 1975.

[6] J. Engelfriet. Some open questions and recent results on tree transducers and tree languages. In *Formal language theory* ed. by R. V. Book, Academic press 1980, pp 241-286.

[7] C. Frougny and J. Sakarovitch. Synchronised rational relations of finite and infinite words. In *Theoretical Computer Science.* Nber 108, pp 45–82. 1993.

[8] G. Huet. Confluent reductions: Abstract properties and applications to term rewriting system. *J.A.C.M. 27.* pp 797-821. 1980.