The Recognizability Problem for Tree Automata with Comparisons between Brothers

Bruno Bogaert, Franck Seynhaeve, and Sophie Tison

LIFL, Bât M3, Université Lille 1, F59655 Villeneuve d'Ascq cedex, France
email: bogaert,seynhaev,tison@lifl.fr

Abstract. Several extensions of tree automata have been defined, in order to take in account non-linearity in terms. Roughly, these automata allow equality or disequality constraints between subterms. They have been used to get decision results, e.g. in term rewriting. One natural question arises when we consider a language recognized by such an automaton: is this language recognizable, i.e. are the constraints necessary? Here we study this problem in the class $REC_{\neq}$ corresponding to comparisons between brothers and we prove its decidability. It gives e.g. a decision procedure for testing whether the image by a quasi-alphabetic homomorphism of a recognizable tree language is recognizable.

1 Introduction

Even if many concepts in tree languages can be viewed as extensions of the word case, some new difficulties and phenomena arise when we consider trees, in particular "non-linearity" (a term is non linear if it contains two occurrences of the same variable). For example, the family of recognizable sets is not closed under non-linear homomorphism. Actually tree automata can't deal with non linear terms: e.g. the set of terms containing an occurrence of $f(x,x)$ is not recognizable. As non linear terms occur very often, e.g. in logic or equational programming, several extensions of tree automata have been defined, in order to take in account non-linearity in terms.

The first one is the class of automata with equality tests (*Rateg* automata) [13]; unfortunately, the emptiness property is undecidable for this class. Several "decidable" classes have then been defined, dealing with restrictions to the tests in order to keep good decidability and closure properties.

First, Bogaert and Tison [3] introduced $REC_{\neq}$ automata (tree automata with comparisons between brothers) and denoted $REC_{\neq}$ the set of languages recognized by these automata. The rules use tests in order to impose either equalities, or differences between brother terms: rules like $f(q,q)[x_1 = x_2] \to q_1$ or $f(q,q)[x_1 \neq x_2] \to q_2$ are allowed. The emptiness problem in $REC_{\neq}$ has been proved decidable in [3] and the class has good closure properties.

One more general class with good decidability properties has then been introduced (Caron et al. [5,4,6]): the class of reduction automata, which roughly allow arbitrary disequality constraints but only finitely many equality constraints on

W. Thomas (Ed.): FOSSACS'99, LNCS 1578, pp. 150-164, 1999.

each run of the automaton. By using these classes interesting decision results have been got; for example, the encompassment theory [1] can be shown decidable by using reduction automata and decidability of ground reducibility is a direct consequence of this result ([7]).

One natural question arises when we consider a language recognized by an automaton with tests: is this language recognizable, and in this case can we compute the corresponding "classic" automaton? In other words, can we decide whether "constraints are really necessary to define the language"? Getting rid of constraints allows e.g. to use classical algorithms for recognizable sets. For the class of reduction automata, this problem contains strictly the decidability of recognizability for the set of normal forms of a rewrite system, problem solved but whose proofs are very technical [12,14].

Here we give a positive answer to this problem for $REC_{\neq}$ languages: we can decide whether such a language is recognizable (and compute a classic automaton when it exists). This partial result has some interesting corollaries; it gives e.g. a decision procedure for testing whether the image by a quasi-alphabetic homomorphism of a recognizable tree language is recognizable. (This result can be connected with the cross-section theorem; the cross-section theorem is false in general for trees; it is true when the morphism is linear [1], or when the morphism is quasi-alphabetic and the image is recognizable. It is conjectured true when the image is recognizable [8]). The result can also be used to decide properties of term rewrite systems. When a rewrite system R has "good" properties (same occurrences of a variable are "brothers": it includes the case of shallow systems [11]), it gives a procedure to test recognizability of the set of normal forms of R which is much easier than the general one and it allows testing whether the set of direct descendants $R(L)$ is recognizable for a recognizable language L: testing these properties can be useful e.g for computing normalizing terms, for computing reachable terms... ([15],[10]).

The spirit of the proof is natural: we define a kind of "minimization" very similar to the classical one (Myhill-Nerode theorem for tree languages [9,6]). The difficulty is to extend the notion of context by adding equality or disequality constraints. Then the point is that in the "minimized" automaton, it should appear "clearly" whether the constraints are necessary or not: e.g., when we get two rules $f(q,q)[x_1 = x_2] \to q_1$ and $f(q,q)[x_1 \neq x_2] \to q_2$, with q_1 and q_2 non equivalent, it should mean that we need the constraints and so that the language is not recognizable. Actually, the proof is a little more intricate and finite languages can disturb the "natural" minimization. E.g. the "minimized" automaton associated with the recognizable language $h^*(\{f(a,a), f(b,b)\})$ is $a \to q, b \to q, f(q,q)[x_1 = x_2] \to q_f, h(q_f) \to q_f$ and then uses constraints. So, a first step of the proof is devoted to eliminate these degenerate cases.

After basic definitions given in Section 2, $REC_{\neq}$ automata are introduced in Section 3. The Section 4 is devoted to the proof.

[1] The encompassment theory is the set of first order formula with predicates $red_t(x)$, t term. In the theory $red_t(x)$ holds if and only if x is a ground term encompassing t i.e. an instance of t is a subterm of x.

2 Preliminaries

The set of nonnegative integers is denoted $\mathbb{N}$ and $\mathbb{N}^*$ denotes the set of finite–length strings over $\mathbb{N}$. For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$, so that $[0]$ is another name for the empty set $\emptyset$.

An alphabet Σ is *ranked* if $\Sigma = \bigcup_p \Sigma_p$ where $\Sigma_p \neq \emptyset$ only for a finite number of p's and the non empty Σ_p are finite and pairwise disjoint. Elements of Σ_p are said to be of *arity* p. Elements of arity 0 are called *constants*. We suppose that Σ contains at least one constant.

Let $\mathcal{X}$ be a set of variables. A *term* over $\Sigma \cup \mathcal{X}$ is a partial function $t : \mathbb{N}^* \to \Sigma \cup \mathcal{X}$ with domain $\mathcal{P}os(t)$ satisfying the following properties:

- $\mathcal{P}os(t)$ is nonempty and prefix-closed;
- If $t(\alpha) \in \Sigma_n$, then $\{i \in \mathbb{N} \mid \alpha i \in \mathcal{P}os(t)\} = \{1, 2, \ldots, n\}$;
- If $t(\alpha) \in \mathcal{X}$, then $\{i \in \mathbb{N} \mid \alpha i \in \mathcal{P}os(t)\} = \emptyset$.

The set of all terms (or *trees*) is denoted by $T_\Sigma(\mathcal{X})$. If $\mathcal{X} = \emptyset$ then $T_\Sigma(\mathcal{X})$ is denoted by T_Σ. Each element of $\mathcal{P}os(t)$ is called a *position*.
Let $t \in T_\Sigma(\mathcal{X})$ and $p \in \mathcal{P}os(t)$. We denote by $t|_p$ the subterm of t rooted at position p and by $t(p)$ the label of t at position p. $\forall i \in [n]$ such that $pi \in \mathcal{P}os(t)$, $t|_{pi}$ is said to be a *son* of the label $t(p)$.
Let $\mathcal{X}_n$ be a set of n variables. A term $C \in T_\Sigma(\mathcal{X}_n)$ where each variable occurs at most once in C is called a context. The term $C[t_1, \ldots, t_n]$ for $t_1, \ldots, t_n \in T_\Sigma$ denotes the term in T_Σ obtained from C by replacing for each $i \in [n]$ x_i by t_i. We denote by $\mathcal{C}^n(\Sigma)$ the set of contexts over n variables $\{x_1, \ldots, x_n\}$ and $\mathcal{C}(\Sigma)$ the set of contexts containing a single variable.

3 Tree Automata with Comparisons between Brothers

Automata with comparisons between brothers ($REC_{\neq}$ automata) have been introduced by Bogaert and Tison [3]. They impose either equalities, or differences between brother terms. These equalities and differences are expressed by constraint expressions. Here we will restrict to define normalized-complete $REC_{\neq}$ automata (each $REC_{\neq}$ automaton is equivalent to a automaton called normalized-complete $REC_{\neq}$ automaton [3]).
Rules of normalized-complete $REC_{\neq}$ automata impose, for each pair (pi, pj) of positions of a term t where p is a position and $i \neq j \in \mathbb{N}$, that $t|_{pi} = t|_{pj}$ or $t|_{pi} \neq t|_{pj}$. These comparisons are expressed by full constraint expressions.
First, we define the notion of full constraint expressions. Then we give the definition of normalized-complete $REC_{\neq}$ automata.

Definition 1. A full constraint expression c over n variables $(x_i)_{i \in [n]}$, $n \in \mathbb{N}$, (in the following x_i will always denote the i^{th} son of a node) is a conjunction of equalities $x_i = x_j$ and of disequalities $x_i \neq x_j$ such that there exists a partition $(E_i)_{i \in [m]}$ of $[n]$, $m \leq n$ satisfying:

$$c = \bigwedge_{k \in [m]} \bigwedge_{l, l' \in E_k} x_l = x_{l'} \wedge \bigwedge_{k, k' \in [m], k \neq k'} \bigwedge_{l \in E_k, l' \in E_{k'}} x_l \neq x_{l'} \qquad (1)$$

We denote $c = (E_i)_{i \in [m]}$ in order to simplify the notation, $card(c) = m$ the cardinality of c and CE'_n the set of full constraint expressions over n variables.

For example, $CE'_3 = \{(\{1,2,3\}), (\{1,2\},\{3\}), (\{1,3\},\{2\}), (\{2,3\},\{1\}), (\{1\},\{2\},\{3\})\}$.
In the case $n = 0$, the full constraint expression over no variable is denoted by $\top$ (null constraint).

Definition 2. A tuple of terms $(t_i)_{i\in[n]}$ satisfies a full constraint expression c iff the evaluation of c for the valuation $(\forall i \leq n, x_i = t_i)$ is $true$, when "$=$" is interpreted as equality of terms, "$\neq$" as its negation, "$\top$" as $true$, $\wedge$ as the usual boolean function and. For example, the tuple of constants (a, b, a) satisfies the full constraint expression $x_1 \neq x_2 \wedge x_1 = x_3 \wedge x_2 \neq x_3$.

Let us remark that if c and c' are full constraint expressions over n variables then $c \wedge c'$ is unsatisfiable if $c \neq c'$.

Definition 3. Let c be a full constraint of CE'_n and $(q_i)_{i\in[n]}$ be a n-tuple of states. We say that $(q_i)_{i\in[n]}$ satisfies the equality constraints of c if for $\forall k, l \in [n]$, $(c \Rightarrow (x_k = x_l)) \Rightarrow (q_k = q_l)$.

Let us now define normalized-complete $REC_{\neq}$ automata.

Definition 4. A normalized-complete automaton $\mathcal{A}$ with comparisons between brothers (normalized-complete $REC_{\neq}$ automaton) is a tuple $(\Sigma, Q, F, \mathcal{R})$ where Σ is a ranked alphabet, Q a finite set of states, $F \subseteq Q$ a set of final states and $\mathcal{R} \subseteq \bigcup_i \Sigma_i \times CE'_i \times Q^{i+1}$ a set of rules (a rule $(f, c, q_1, \ldots, q_n, q)$ will be denoted $f(q_1, \ldots, q_n)[c] \to q$) with:

- $\mathcal{A}$ deterministic i.e. for all rules $f(q_1, \ldots, q_n)[c] \to q$ and $f(q_1, \ldots, q_n)[c] \to q'$, $q = q'$;
- For each letter $f \in \Sigma_n$, each n-tuple $(q_i)_{i\in[n]} \in Q$, each constraint c of CE'_n such that $(q_i)_{i\in[n]}$ satisfies the equality contraints of c, there exists at least one rule $f(q_1, \ldots, q_n)[c] \to q$;
- And for each letter $f \in \Sigma_n$, each n-tuple $(q_i)_{i\in[n]} \in Q$, each constraint c of CE'_n such that $(q_i)_{i\in[n]}$ doesn't satisfy the equality contraints of c, there exists no rule $f(q_1, \ldots, q_n)[c] \to q \in \mathcal{R}$.

Let $f \in \Sigma_n$ and $(t_i)_{i\in[n]}$ be terms of T_Σ. The relation $\xrightarrow{*}_{\mathcal{A}}$ is defined as follows:

$$f(t_1, \ldots t_n) \xrightarrow{*}_{\mathcal{A}} q \text{ if and only if}$$
$$\left(\begin{array}{r} \exists f(q_1, \ldots, q_n)[c] \to q \in \mathcal{R} \text{ such that } \forall i \in [n], t_i \xrightarrow{*}_{\mathcal{A}} q_i \\ \text{and } (t_i)_{i\in[n]} \text{ satisfies the constraint } c \end{array}\right)$$

Let q be a state of Q. We denote by $\mathcal{L}_{\mathcal{A}}(q)$ the set of terms t such that $t \xrightarrow{*}_{\mathcal{A}} q$. A tree $t \in T_\Sigma$ is accepted by $\mathcal{A}$ if there exists a final state q such that $t \in \mathcal{L}_{\mathcal{A}}(q)$. The language $\mathcal{L}(\mathcal{A})$ recognized by $\mathcal{A}$ is the set of accepted terms. We denote by $REC_{\neq}$ the set of tree languages recognized by the class of $REC_{\neq}$ automata.

Example 5. Let $\mathcal{A} = (\{a, h, f\}, \{q, q_f, q_p\}, \{q_f\}, \mathcal{R})$ with $\mathcal{R}$:

$$\begin{array}{ccccc} a \to q & & h(q) \to q & & h(q_f) \to q_p \\ h(q_p) \to q_p & & f(q,q,q)[c] \to q_f & & f(q,q,q)[c'] \to q_p \ \forall c' \in CE'_3 \setminus \{c\} \\ f(q_1,q_2,q_3)[c'] \to q_p & \forall (q_1,q_2,q_3) \in Q^3 \setminus \{(q,q,q)\}, \forall c' \in CE'_3 & & & \end{array}$$

where c is the full constraint expression $x_1 = x_2 \wedge x_3 \neq x_1 \wedge x_3 \neq x_2$. Then $\mathcal{A}$ recognizes the language $\{f(h^n(a), h^n(a), h^m(a)) \mid m, n \in \mathbb{N}, m \neq n\}$.

4 Recognizability Problem

We consider the recognizability problem in the class $REC_{\neq}$:

Input: A ranked alphabet Σ and a language $\mathcal{F} \in REC_{\neq}$.

Question: Is $\mathcal{F}$ recognizable?

We will prove that the recognizability problem is decidable; furthermore, when the input language is recognizable, our algorithm computes a corresponding tree automaton.

The idea of the algorithm is the following: we define a kind of minimization, close to the classic one (Myhill-Nerode theorem for tree languages [9,6]) but dealing with constraints: roughly, two states will be equivalent, when they have the same behaviour for the same context with constraints. This needs defining constrained terms which are terms labeled with equality and disequality constraints. Then, the point is that, when the reduction works well, it should be the case that non necessary constraints are dropped. For example, let us suppose that we have two rules $f(q,q)[x_1 = x_2] \to q_1$ and $f(q,q)[x_1 \neq x_2] \to q_2$; when q_1 and q_2 are equivalent, it means that the constraints are not necessary.

However, the reasoning fails when the language associated with a state is finite: $a \to q, b \to q, f(q,q)[x_1 = x_2] \to q_f$ use constraints to define the finite (thus recognizable) language $\{f(a,a), f(b,b)\}$. So in a first step, we eliminate states q s.t. $\mathcal{L}_{\mathcal{A}}(q)$ is finite (section 4.1). Then we extend the notion of context to take in account equality and disequality constraints (section 4.2) and then, we define and compute "the" reduced automaton (section 4.3). Finally, we prove that the language is recognizable iff the reduced automaton is not "constraint-sensitive" (section 4.4), i.e. two rules whose left-hand-side differ only by constraints have the same right-hand-side. We deduce decidability of the recognizability problem in the class $REC_{\neq}$ and obtain an effective construction of the corresponding automaton, when the language is recognizable.

4.1 How to reduce to the "infinite" case

Let $\mathcal{F} \in REC_{\neq}$ and $\mathcal{A} = (\Sigma, Q, F, \mathcal{R})$ be a normalized-complete $REC_{\neq}$ automaton recognizing $\mathcal{F}$. Let us suppose that there exists at least a state q of $\mathcal{A}$ such that $\mathcal{L}_{\mathcal{A}}(q)$ is finite. Let us denote:

$$\mathcal{F}_1 = \bigcup_{q \in F, \mathcal{L}_{\mathcal{A}}(q) \text{ finite}} \mathcal{L}_{\mathcal{A}}(q) \quad \text{and} \quad \mathcal{F}_2 = \bigcup_{q \in F, \mathcal{L}_{\mathcal{A}}(q) \text{ infinite}} \mathcal{L}_{\mathcal{A}}(q).$$

Since $\mathcal{L}(\mathcal{A}) = \mathcal{F}_1 \cup \mathcal{F}_2$ and $\mathcal{F}_1$ is finite, $\mathcal{L}(\mathcal{A})$ is recognizable iff $\mathcal{F}_2$ is recognizable. The language $\mathcal{F}_2$ is recognized by the $REC_{\neq}$ automaton $\mathcal{B} = (\Sigma, Q, F', \mathcal{R})$ where $F' = \{q | q \in F, \mathcal{L}_{\mathcal{A}}(q) \text{ infinite}\}$. We construct a new alphabet Γ by encoding, for each state q such that $\mathcal{L}_{\mathcal{B}}(q)$ is finite, the terms of $\mathcal{L}_{\mathcal{B}}(q)$ in the symbols of Γ. We define a $REC_{\neq}$ automaton $\mathcal{B}'$ on Γ and a linear morphism φ from $T_{\Gamma}(\mathcal{X})$ onto $T_{\Sigma}(\mathcal{X})$ such that for each state q of $\mathcal{B}'$, $\mathcal{L}_{\mathcal{B}'}(q)$ is infinite and such that $\varphi(\mathcal{L}(\mathcal{B}')) = \mathcal{L}(\mathcal{B})$ and $\varphi^{-1}(\mathcal{L}(\mathcal{B})) = \mathcal{L}(\mathcal{B}')$. We deduce that ($\mathcal{L}(\mathcal{B})$ is recognizable) $\Leftrightarrow$ ($\mathcal{L}(\mathcal{B}')$ is recognizable) since φ is linear (the entire proof can be found in [2]). We deduce that the general case can be reduced to the infinite case since for each state q of the automaton $\mathcal{B}'$, $\mathcal{L}_{\mathcal{B}'}(q)$ is infinite.

Before studying the "infinite" case, let us give an example of construction of $\mathcal{B}'$ and φ. Let $\Sigma = \{a/0, f/2\}$ and $\mathcal{B} = (\Sigma, Q, F', \mathcal{R})$ where $Q = \{q, q_p, q_f\}$, $F' = \{q_f\}$ and $\mathcal{R}$ is composed of the following rules:

$$\begin{array}{rr} a \to q & f(q,q)[x_1 = x_2] \to q_f \\ f(q_f, q_f)[x_1 = x_2] \to q_f & f(q_p, q_p)[x_1 = x_2] \to q_p \\ f(q_1, q_2)[x_1 \neq x_2] \to q_p \ \forall (q_1, q_2) \in (Q \times Q) & \end{array}$$

$\mathcal{B}$ is a normalized-complete $REC_{\neq}$ automaton. Obviously $\mathcal{L}_{\mathcal{B}}(q) = \{a\}$ and, $\mathcal{L}_{\mathcal{B}}(q_p)$ and $\mathcal{L}_{\mathcal{B}}(q_f)$ are infinite. Then we consider $\Box$ a symbol not in Σ and we define the alphabet $\Gamma = \{f_{(\Box,\Box)}, f_{(\Box,a)}, f_{(a,\Box)}, f_{(a,a)}\}$.
Then $\mathcal{B}' = (\Gamma, Q', F', \mathcal{R}')$ is the $REC_{\neq}$ automaton where $Q' = \{q_p, q_f\}$ and $\mathcal{R}'$ is composed of the following rules:

$$\begin{array}{ll} f_{(a,a)} \to q_f & f_{(\Box,\Box)}(q_1, q_2)[x_1 \neq x_2] \to q_p \ \forall (q_1, q_2) \in (Q' \times Q') \\ f_{(\Box,a)}(q_1) \to q_p \ \forall q_1 \in Q' & f_{(\Box,\Box)}(q_f, q_f)[x_1 = x_2] \to q_f \\ f_{(a,\Box)}(q_2) \to q_p \ \forall q_2 \in Q' & f_{(\Box,\Box)}(q_p, q_p)[x_1 = x_2] \to q_p \end{array}$$

And $\varphi : T_{\Gamma}(\mathcal{X}) \to T_{\Sigma}(\mathcal{X})$ is the linear morphism defined as follows:

$$\begin{array}{rr} \varphi(f_{(a,a)}) = f(a,a) & \varphi(f_{(\Box,a)})(x_1) = f(x_1, a) \\ \varphi(f_{(\Box,\Box)})(x_1, x_2) = f(x_1, x_2) & \varphi(f_{(a,\Box)})(x_1) = f(a, x_1) \end{array}$$

So we can suppose in the rest of the proof that for each state q of the normalized-complete automaton $\mathcal{A} = (\Sigma, Q, F, \mathcal{R})$ recognizing $\mathcal{F}$, $\mathcal{L}_{\mathcal{A}}(q)$ is infinite.

4.2 Constrained Terms

In the class of recognizable tree languages, an equivalence relation using contexts is used in order to minimize the automata (Myhill-Nerode theorem for tree languages [9,6]). We define a similar notion in the class of $REC_{\neq}$ automata. As the rules of $REC_{\neq}$ automata contain comparisons between brother terms, we introduce the notion of terms imposing equalities and disequalities between brother terms, these comparisons being expressed by full constraint expressions. Such terms are called constrained terms. The label of a constrained term at a position p is the combination of a symbol and of a full constraint expression c such that the equality constraints of c are satisfied by the sons of the label and such that there is no disequality constraint between equal ground sons of the label. Leaves of a constrained term may also be states or occurences of an unique variable.
More formally, let x be a variable and Σ' be the ranked alphabet defined by $\forall n \in \mathbb{N}$, $\Sigma'_n = \{f_c \mid f \in \Sigma_n, c \in CE'_n\}$. A *constrained term* C over $\Sigma \cup Q$ is a term of $T_{\Sigma'}(Q \cup \{x\})$ where the states of Q are constants and $\forall p$ non leaf position of C, $\exists n > 0$, such that $C(p) = f_c \in \Sigma'_n$ with:

- The n-tuple $(C|_{pi})_{i \in [n]}$ satisfies the equality constraints of c;
- c contains no disequality constraint between equal ground sons i.e. $\forall i, j \in [n], (C|_{pi} \in T_{\Sigma'}$ and $C|_{pi} = C|_{pj}) \Rightarrow (c \Rightarrow (x_i = x_j))$.

Example 6. Let $g, f \in \Sigma_2$ and $q_1, q_2 \in Q$. Then $f_c(g_{c'}(q_1, x), g_{c'}(q_2, x))$ with $c = [x_1 = x_2]$ and $c' = [x_1 \neq x_2]$ is not a constrained term since $g_{c'}(q_1, x) \neq g_{c'}(q_2, x)$. But $f_c(g_{c'}(q_1, x), g_{c'}(q_1, x))$ with $c = [x_1 = x_2]$ and $c' = [x_1 \neq x_2]$ is a constrained term.

Constrained terms are terms hence we use the usual notion of $height$ of a term on constrained terms with $height(c) = 0$ if $c \in Q \cup \{x\}$ and $height(c) = 1$ if $c \in \Sigma'_0$. Let C be a constrained term and $q \in Q$, we denote by $C[q]$ the constrained term obtained from C by replacing each occurence of x by q.

Run on constrained terms We extend the notion of run on terms to run on constrained terms. Let C be a constrained term and q, q' be states. We denote $C[q] \xrightarrow{*}_{\mathcal{A}} q'$ iff

- Either $C = q'$ or $(C = x$ and $q = q')$;
- Or $C = f_c(C_1, \ldots, C_n)$ with $f_c \in \Sigma'_n$, $(C_i)_{i \in [n]}$ constrained terms such that $\forall i \in [n]\, C_i[q] \xrightarrow{*}_{\mathcal{A}} q_i$ and $f(q_1, \ldots, q_n)[c] \to q' \in \mathcal{R}$.

Let us now extend the notion of run to run between constrained terms. Let C, C' be constrained terms and q be a state. We denote $C[q] \xrightarrow{*}_{\mathcal{A}} C'$ iff there exists a set P of positions of C such that:

- $\forall p \in P$, $C'|_p \in Q$ and $C[q]|_p \xrightarrow{*}_{\mathcal{A}} C'|_p$;
- $\forall p \in \mathcal{P}os(C)$ not prefixed by a position of P, $C'(p) = C[q](p)$.

4.3 Minimization

Definition 7. let $\equiv_{\mathcal{A}}$ be the relation on Q defined by for all $q, q' \in Q$, $q \equiv_{\mathcal{A}} q'$ if for each constrained term C, $(C[q] \xrightarrow{*}_{\mathcal{A}} q_1 \in F \Leftrightarrow C[q'] \xrightarrow{*}_{\mathcal{A}} q_2 \in F)$.

The relation $\equiv_{\mathcal{A}}$ is obviously an equivalence relation. In the following, we associate with the automaton $\mathcal{A}$ a normalized-complete $REC_{\neq}$ $\mathcal{A}_m$ said "minimized" whose states are the equivalence classes of the relation $\equiv_{\mathcal{A}}$ and such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_m)$.
First we prove that the equivalence classes of the relation $\equiv_{\mathcal{A}}$ are computable. Then we define the automaton $\mathcal{A}_m$.

Equivalence Classes Algorithm EQUIV
input: Normalized-complete $REC_{\neq}$ automaton $\mathcal{A} := (\Sigma, Q, F, \mathcal{R})$
begin
  Set P to $\{F, Q \setminus F\}$ /* P is the initial equivalence relation*/
  repeat
    $P' := P$
    /* Refine equivalence P' in P */
    qPq' if $qP'q'$ and $\forall C$ constrained term of height 1,
      $C[q] \xrightarrow{*}_{\mathcal{A}} q_1$ and $C[q'] \xrightarrow{*}_{\mathcal{A}} q_2$ with $q_1 P' q_2$
  until $P' = P$
output: P set of equivalence classes of $\equiv_{\mathcal{A}}$
end

We denote by $\tilde{q}$ the equivalence class of a state q w.r.t. P, the set computed by the algorithm EQUIV. Let us prove that the algorithm EQUIV is correct i.e. that P is the set of equivalence classes of $\equiv_{\mathcal{A}}$ (Lemma 10). First we consider two rules whose left hand sides differ only by replacing all occurences of one state bounded by equalities imposed by the constraint by a state of the same equivalence class w.r.t. P. Then we prove that the right hand side of the two

rules belong to the same equivalence class w.r.t. P (Lemma 8). We deduce that the equivalence classes w.r.t. P are compatible with the rules of the automaton $\mathcal{A}$ (Corollary 9).

Lemma 8. *Let $f(q_1,\ldots,q_n)[c] \to q \in \mathcal{R}$ and $f(q'_1,\ldots,q'_n)[c] \to q' \in \mathcal{R}$ such that there exists $j \in [n]$ such that $q_j \in \tilde{q'_j}$, $\forall i \in [n], ((c \Rightarrow x_i = x_j) \Rightarrow q'_i = q'_j)$ and $((c \Rightarrow x_i \neq x_j) \Rightarrow q'_i = q_i)$. Then $q \in \tilde{q'}$.*

Proof. First the rule $f(q'_1,\ldots,q'_n)[c] \to q'$ is well defined since we can prove that $(q')_{i\in[n]}$ satisfies the equality constraints of c. Let us now consider the constrained term C defined by $head(C) = f_c$ and for each $i \in [n]$ if $c \Rightarrow x_i = x_j$ then $C(i) = x$ else $C(i) = q_i$.
Obviously $\forall i \in [n], C[q_j]|_i = f(q_1,\ldots,q_n)|_i$ then $C[q_j] \to_{\mathcal{A}} q$. Let us now prove that $\forall i \in [n], C[q'_j]|_i = f(q'_1,\ldots,q'_n)|_i$. Let $i \in [n]$. If $c \Rightarrow x_i = x_j$ then $q'_i = q'_j$. Then $C[q'_j]|_i = q'_j = q'_i = f(q'_1,\ldots,q'_n)|_i$. If $c \Rightarrow x_i \neq x_j$ then $C[q'_j]|_i = q_i = q'_i = f(q'_1,\ldots,q'_n)|_i$. Hence $\forall i \in [n], C[q'_j]|_i = f(q'_1,\ldots,q'_n)|_i$ then $C[q'_j] \to_{\mathcal{A}} q'$.
Moreover C is a constrained term of height 1 hence according to the EQUIV algorithm, we have $q \in \tilde{q'}$ since $q_j \in \tilde{q'_j}$ which ends the proof of Lemma 8.

Corollary 9. *Let $f(q_1,\ldots,q_n)[c] \to q \in \mathcal{R}$ and $f(q'_1,\ldots,q'_n)[c] \to q' \in \mathcal{R}$ such that $\forall j \in [n]\ q_j \in \tilde{q'_j}$. Then $q \in \tilde{q'}$.*

Let us now prove that the algorithm EQUIV is correct.

Lemma 10. *P is the set of equivalence classes of $\equiv_{\mathcal{A}}$ i.e.:*

$$\forall q, q' \in Q\ (q \equiv_{\mathcal{A}} q') \Leftrightarrow (q \in \tilde{q'}.)$$

Proof. First, we can prove that $\forall q, q' \in Q\ (q \notin \tilde{q'}) \Rightarrow (q \not\equiv_{\mathcal{A}} q')$ by induction on the step of the algorithm EQUIV where $q \notin \tilde{q'}$ appears. We deduce that $\forall q, q' \in Q\ (q \equiv_{\mathcal{A}} q') \Rightarrow (q \in \tilde{q'})$.
In order to prove the implication $\Leftarrow$, we first prove that:

$$\forall q, q' \in Q,\ q \in \tilde{q'} \Rightarrow \left(\forall C \text{ constrained term } \begin{cases} C[q] \xrightarrow{*}_{\mathcal{A}} s \\ C[q'] \xrightarrow{*}_{\mathcal{A}} s' \end{cases} \Rightarrow (s \in \tilde{s'})\right)$$

by induction on the height of the constrained term. Let $q, q' \in Q$ such that $q \in \tilde{q'}$ and C a constrained term such that $C[q] \xrightarrow{*}_{\mathcal{A}} s$ and $C[q'] \xrightarrow{*}_{\mathcal{A}} s'$.

C **of height** 0: **Either** $C \in Q$: Hence $\exists q" \in Q$ such that $C = q"$. $C[q] = C[q'] = q"$ hence $s = s' = q"$. Finally $s \in \tilde{s'}$;
Or $C = x$: $C[q] = q$ and $C[q'] = q'$ hence $s = q$ and $s' = q'$. Finally $s \in \tilde{s'}$ since $q \in \tilde{q'}$.

Induction hypothesis: Let $k \in \mathbb{N}$. Let us suppose that the property is true for all constrained term C of height less than or equal to k. Let C be a constrained term of height k+1. There exists $f \in \Sigma_n, c \in CE'_n, (C_i)_{i\in[n]}$ constrained terms such that $C = f_c(C_1,\ldots,C_n)$. According to induction hypothesis, $\forall i \in [n], C_i[q] \xrightarrow{*}_{\mathcal{A}} q_i$ and $C_i[q'] \xrightarrow{*}_{\mathcal{A}} q'_i$ with $q_i \in \tilde{q'_i}$.

$(C_i)_{i\in[n]}$ satisfies the equality constraints of c. Moreover $\mathcal{A}$ is deterministic hence $\forall k,l \in [n]$ such that $c \Rightarrow (x_k = x_l)$, we have $q_k = q_l$ and $q'_k = q'_l$ since $C_k = C_l$. We deduce $(q_i)_{i\in[n]}$ and $(q'_i)_{i\in[n]}$ satisfies the equality constraints of c. Hence since $\mathcal{A}$ is normalized-complete, there exists $f(q_1,\ldots,q_n)[c] \rightarrow s \in \mathcal{R}$ and $f(q'_1,\ldots,q'_n)[c] \rightarrow s' \in \mathcal{R}$.
Moreover $\forall i \in [n], q_i \in \tilde{q'_i}$. We deduce from the Corollary 9 that $s \in \tilde{s'}$.

At the beginning of the execution of EQUIV, $P = \{F, Q \setminus F\}$, hence:

$$\forall q \in F, \forall q' \in Q, (q' \in \tilde{q}) \Rightarrow (q' \in F) \tag{2}$$

since at each step of the algorithm qPq'. We deduce that $\forall q, q' \in Q\ (q \not\equiv_{\mathcal{A}} q') \Rightarrow (q' \notin \tilde{q})$ which ends the proof of Lemma 10.

Let us now define the automaton $\mathcal{A}_m$. Let us denote $\tilde{q}$ the equivalence class of a state q w.r.t. $\equiv_{\mathcal{A}}$. Let $\mathcal{A}_m = (\Sigma, Q_m, F_m, \mathcal{R}_m)$ defined as follows:

- Q_m is the set of equivalence classes of $\equiv_{\mathcal{A}}$.
- $F_m = \{\tilde{q} \mid q \in F\}$.
- $\mathcal{R}_m = \{f(\tilde{q_1},\ldots,\tilde{q_n})[c] \rightarrow \tilde{q} \mid \forall i \in [n]\ \exists q'_i \in \tilde{q_i}, \exists q' \in \tilde{q}$ such that $f(q'_1,\ldots,q'_n)[c] \rightarrow q' \in \mathcal{R}\}$.

We prove now that $\mathcal{A}_m$ is a normalized-complete $REC_{\neq}$ automaton (Lemma 11) and that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_m)$ (Lemma 12).

Lemma 11. *$\mathcal{A}_m$ is a normalized-complete $REC_{\neq}$ automaton.*

Proof. First we prove that $\mathcal{A}_m$ is deterministic. Let $f(\tilde{q_1},\ldots,\tilde{q_n})[c] \rightarrow \tilde{q} \in \mathcal{R}_m$ and $f(\tilde{q_1},\ldots,\tilde{q_n})[c] \rightarrow \tilde{s} \in \mathcal{R}_m$. According to the definition of $\mathcal{R}_m$:

- $\forall i \in [n]$, $\exists q'_i \in \tilde{q_i}, \exists q' \in \tilde{q}$ such that $f(q'_1,\ldots,q'_n)[c] \rightarrow q' \in \mathcal{R}$.
- $\forall i \in [n]$, $\exists q''_i \in \tilde{q_i}, \exists s' \in \tilde{s}$ such that $f(q''_1,\ldots,q''_n)[c] \rightarrow s' \in \mathcal{R}$.

$\forall i \in [n]\ q''_i \in \tilde{q'_i}$ hence according to Lemma 8, $s' \in \tilde{q'}$. Then $\tilde{q} = \tilde{s}$ since $\tilde{q} = \tilde{q'}$, $\tilde{s} = \tilde{s'}$ and $q' = \tilde{s'}$. Finally $\mathcal{A}_m$ is deterministic. Let us now prove that $\mathcal{A}_m$ is normalized-complete. Let $f \in \Sigma_n$, $\tilde{q_1},\ldots,\tilde{q_n} \in Q_m$ and $c \in CE'_n$ such that $(\tilde{q_i})_{i\in[n]}$ satisfies the equality contraints of c.
Let $(q'_i)_{i\in[n]}$ such that $\forall i \in [n]\ q'_i \in \tilde{q_i}$ and $\forall k,l \in [n]\ (c \Rightarrow (x_k = x_l)) \Rightarrow (q'_k = q'_l)$. The last condition is possible since $\forall k,l \in [n]\ (c \Rightarrow (x_k = x_l)) \Rightarrow (\tilde{q_k} = \tilde{q_l})$ and $(\tilde{q_i})_{i\in[n]}$ satisfies the equality contraints of c. $(q'_i)_{i\in[n]}$ satisfies the equality contraints of c and $\mathcal{A}$ is complete hence $\exists f(q'_1,\ldots,q'_n)[c] \rightarrow q \in \mathcal{R}$. Hence $f(\tilde{q'_1},\ldots,\tilde{q'_n})[c] \rightarrow \tilde{q} \in \mathcal{R}_m$ according to the EQUIV algorithm. Moreover $\forall i \in [n]$, we have $\tilde{q_i} = \tilde{q'_i}$ hence $f(\tilde{q_1},\ldots,\tilde{q_n})[c] \rightarrow \tilde{q} \in \mathcal{R}_m$.
Finally, we deduce $\mathcal{A}_m$ is a normalized-complete $REC_{\neq}$ automaton which ends the proof of Lemma 11.

Lemma 12. *$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_m)$.*

Proof. First we can prove by induction on the height of t that $\forall t \in T_\Sigma, \forall q \in Q, (t \xrightarrow{*}_{\mathcal{A}} q) \Rightarrow (t \xrightarrow{*}_{\mathcal{A}_m} \tilde{q})$. We deduce that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}_m)$.
Then we deduce $\mathcal{L}(\mathcal{A}_m) \subseteq \mathcal{L}(\mathcal{A})$ from the property (2) and the following property:

$$\forall t \in T_\Sigma, \forall q \in Q, (t \xrightarrow{*}_{\mathcal{A}_m} \tilde{q}) \Rightarrow (\exists q' \in \tilde{q} \text{ such that } t \xrightarrow{*}_{\mathcal{A}} q').$$

We deduce that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_m)$ which ends the proof of Lemma 12.

Remark 13. We can prove easily that $\forall q \in Q_m, \mathcal{L}_{\mathcal{A}_m}(q)$ is infinite and that $\forall q, q' \in Q_m, (q \equiv_{\mathcal{A}_m} q') \Leftrightarrow (q = q')$.

4.4 Characterization

Let $\mathcal{A}$ be a normalized-complete $REC_{\neq}$ automaton. According to the Section 4.3, we can consider automata satisfying properties of Remark 13. We give now a necessary and sufficient condition for the language recognized by $\mathcal{A}$ to be recognizable.

Proposition 14. *Let $\mathcal{A} = (\Sigma, Q, F, \mathcal{R})$ be a normalized-complete $REC_{\neq}$ automaton such that for each state q of $\mathcal{A}$, $\mathcal{L}_{\mathcal{A}}(q)$ is infinite and such that $\forall q, q' \in Q$, $(q \equiv_{\mathcal{A}} q') \Leftrightarrow (q = q')$. Then $\mathcal{L}(\mathcal{A})$ is recognizable if and only if for all rules $f(q_1, \ldots, q_n)[c] \to q$, $f(q_1, \ldots, q_n)[c'] \to q'$ of $\mathcal{R}$, we have $q = q'$.*

In order to prove Proposition 14, we need some technical lemmas. First, since the language recognized by each state of $\mathcal{A}$ is infinite, we prove that we can "instantiate" each constrained term to a ground term. In fact we prove (Definition 15 and Lemma 16) that we can associate with each constrained term over $\Sigma \cup Q$ a constrained term over Σ without occurence of x by replacing each occurence of a state q by an element of $\mathcal{L}_{\mathcal{A}}(q)$ and each occurence of x by an element of an infinite set of ground terms.

Definition 15. Let C be a constrained term. We denote:

- $\mathcal{V}(C)$ the set of variable positions of C: $\mathcal{V}(C) = \{p \in \mathcal{P}os(C) \mid C(p) = x\}$.
- $\mathcal{S}(C)$ the set of state positions of C: $\mathcal{S}(C) = \{p \in \mathcal{P}os(C) \mid C(p) \in Q\}$.
- For each $q \in Q$, $\mathcal{S}(C)(q) = \{p \in \mathcal{S}(C) \mid C(p) = q\}$.

Lemma 16. *Let C be a constrained term over $\Sigma \cup Q$ and T be an infinite set of terms of T_Σ. There exists a constrained term C' over Σ without occurence of x such that:*

- $\forall p \in \mathcal{P}os(C) \setminus (\mathcal{V}(C) \cup \mathcal{S}(C))$, $C'(p) = C(p)$,
- *Each variable of C is replaced by a constrained term associated with an element of T i.e. $\forall p \in \mathcal{V}(C), \exists t \in T$, $C'|_p = \mathrm{lab}_t$,*
- *Each state of C is replaced by a constrained term associated with an element of the language recognized by the state i.e. $\forall q \in Q, \forall p \in \mathcal{S}(C)(q), \exists t \in \mathcal{L}_{\mathcal{A}}(q), C'|_p = \mathrm{lab}_t$,*

where lab_t denotes for each term t the constrained term over Σ obtained from t, i.e. $\forall p \in \mathcal{P}os(t)$, if $t(p) = f \in \Sigma_n$, $n > 0$, then $\mathrm{lab}_t(p) = f_c$ with c the full constraint satisfied by $(t|_{pi})_{i \in [n]}$, else $\mathrm{lab}_t(p) = t(p)$.

Proof. Let C be a constrained term over $\Sigma \cup Q$ and T be an infinite set of terms of T_Σ. First let us deduce from the full constraint expressions of each position of C, full constraint expressions between the positions of C where the variable x occurs and between the positions of C where the same state occurs.
In fact if we consider the positions where the variable x occurs (positions of $\mathcal{V}(C)$), we express all the equalities between these positions imposed by the constraints of C. When none equality is imposed between two positions we impose a disequality since :

- Constraints impose only equalities between brothers hence between terms whose positions have the same length.
- According to the definition of constrained terms, equalities are only imposed between equal terms in a constrained term.

We can do the same for positions of $\mathcal{S}(C)(q)$ for each q of Q.
More formally, for each position p of C such that $C(p) \in \Sigma'$, we denote $\mathrm{cont}_C(p)$ the constraint obtained by projection from Σ' onto CE'_n and we define $\forall p \in \mathcal{S}(C) \cup \mathcal{V}(C)$ a variable z_p. We denote $c_{\mathcal{V}(C)}$ the full constraint expression over $(z_p)_{p \in \mathcal{V}(C)}$ and $\forall q \in Q, c_{\mathcal{S}(C)(q)}$ the full constraint expression over $(z_p)_{p \in \mathcal{S}(C)(q)}$ defined as follows:

1. We express the equalities imposed by the constraints:
   $\forall p \in \mathcal{P}os(C)$, $\forall i,j$ $(\mathrm{cont}_C(p) \Rightarrow (x_i = x_j)) \Rightarrow \forall \gamma$ such that $z_{pi\gamma}$ defined
   $$\begin{pmatrix} z_{pi\gamma} \in \mathcal{V}(C) \Rightarrow (z_{pi\gamma} = z_{pj\gamma}) \in c_{\mathcal{V}(C)} \\ z_{pi\gamma} \in \mathcal{S}(C)(q), q \in Q \Rightarrow (z_{pi\gamma} = z_{pj\gamma}) \in c_{\mathcal{S}(C)(q)} \end{pmatrix}$$
2. We apply the transitive closure to express all equalities:
   $(z_{p_1} = z_{p_2} \wedge z_{p_2} = z_{p_3}) \in c_{\mathcal{V}(C)} \Rightarrow (z_{p_1} = z_{p_3}) \in c_{\mathcal{V}(C)}$.
   $\forall q \in Q$, $(z_{p_1} = z_{p_2} \wedge z_{p_2} = z_{p_3}) \in c_{\mathcal{S}(C)(q)} \Rightarrow (z_{p_1} = z_{p_3}) \in c_{\mathcal{S}(C)(q)}$.
3. $\forall p, p' \in \mathcal{V}(C), p \neq p', (z_p = z_{p'}) \notin c_{\mathcal{V}(C)} \Rightarrow (z_p \neq z_{p'}) \in c_{\mathcal{V}(C)}$;
4. $\forall q \in Q, \forall p, p' \in \mathcal{S}(C)(q), p \neq p', (z_p = z_{p'}) \notin c_{\mathcal{S}(C)(q)} \Rightarrow (z_p \neq z_{p'}) \in c_{\mathcal{S}(C)(q)}$.

Since T is infinite and $\forall q \in Q$, $\mathcal{L}_\mathcal{A}(q)$ is infinite, there exists $(t_p)_{p \in \mathcal{V}(C)} \in T$ and $\forall q \in Q$, $(t_p)_{p \in \mathcal{S}(C)(q)} \in \mathcal{L}_\mathcal{A}(q)$ such that:

1. $\forall p \in \mathcal{V}(C) \cup \mathcal{S}(C)$, t_p is of height strictly greater than height of C and strictly greater than height of terms of the set $\{t_{p'} \mid p' \in \mathcal{V}(C) \cup \mathcal{S}(C)$, length of p' strictly less than length of $p\}$.
2. $\forall p \in \mathcal{V}(C), \forall p' \in \mathcal{S}(C)$, $t_p \neq t_{p'}$.
3. $\forall q, q', q \neq q', \forall p \in \mathcal{S}(C)(q), \forall p' \in \mathcal{S}(C)(q')$, $t_p \neq t_{p'}$.
4. $(t_p)_{p \in \mathcal{V}(C)}$ satisfies $c_{\mathcal{V}(C)}$.
5. $\forall q \in Q, (t_p)_{p \in \mathcal{S}(C)(q)}$ satisfies $c_{\mathcal{S}(C)(q)}$.

Let us remark that point point3 is satisfied for all families of terms since $\mathcal{A}$ is deterministic. Let C' be the term of $T_{\Sigma'}$ defined as follows:

- $\forall p \in \mathcal{P}os(C) \setminus (\mathcal{V}(C) \cup \mathcal{S}(C))$ $C'(p) = C(p)$;
- $\forall p \in \mathcal{V}(C) \cup \mathcal{S}(C)$ $C'|_p = \mathrm{lab}_{t_p}$.

For each $p \in \mathcal{V}(C)$, $p' \in \mathcal{S}(C)$, constraints of C impose $z_p \neq z_{p'}$ since $C(p) \neq C(p')$. This constraint is satisfied by lab_{t_p} and $\text{lab}_{t_{p'}}$ according to previous points 1 and 2. Similarly for each $p \in \mathcal{S}(C)(q)$, $p' \in \mathcal{S}(C)(q')$, $q \neq q'$, constraints of C impose $z_p \neq z_{p'}$. This constraint is satisfied by lab_{t_p} and $\text{lab}_{t_{p'}}$ according to previous points 1 and 3. We deduce that C' is a constrained term over Σ without occurence of x which ends the proof of Lemma 16.

Let us now prove that we can "instantiate" each constrained term over $\Sigma \cup Q$ to a constrained term over Σ by replacing each occurence of a state q by an element of $\mathcal{L}_{\mathcal{A}}(q)$ (Definition 17 and Lemma 18); similarly, given an infinite set of ground term T, we can "instantiate" each constrained term over Σ by replacing each occurence of x by a constrained term associated with an element of T (Lemma 19).

Definition 17. Let C be a constrained term over $\Sigma \cup Q$. A state-instance of C is a constrained term obtained from C, replacing each state q by a constrained term $\text{lab}_t, t \in \mathcal{L}_{\mathcal{A}}(q)$.

Lemma 18. *There exists a state-instance of each constrained term.*

Proof. Let C be a constrained term and C' be a constrained term obtained from C according to Lemma 16. Let C'' be the constrained term defined by

- $\forall p \in \mathcal{P}os(C) \setminus \mathcal{V}(C)$, $C''(p) = C'(p)$;
- $\forall p \in \mathcal{V}(C)$, $C''|_p = x$.

C" is obviously a state-instance of C which ends the proof of Lemma 18.

Let us remark that when C' is a state-instance of a constrained term C, then $\forall q \in Q, (C[q] \xrightarrow{*}_{\mathcal{A}} s \Rightarrow C'[q] \xrightarrow{*}_{\mathcal{A}} s)$.

Lemma 19. *Let C be a constrained term over Σ and T be an infinite set of terms of T_Σ. There exists $(t_p)_{p \in \mathcal{V}(C)} \in T$ such that C' defined by*

- $\forall p \in \mathcal{P}os(C) \setminus \mathcal{V}(C), C'(p) = C(p)$;
- $\forall p \in \mathcal{V}(C), C'_{|p} = lab_{t_p}$,

is a constrained term.

This lemma is an immediate corrolary of Lemma 16. Let us now prove that the condition of Proposition 14 is necessary.

Lemma 20. *Let us suppose that there exists two rules of $\mathcal{R}$, $f(q_1, \ldots, q_n)[c] \to q$ and $f(q_1, \ldots, q_n)[c'] \to q'$ such that $c \neq c'$ and $q \neq q'$. Then $\mathcal{L}(\mathcal{A})$ is not recognizable.*

Proof. Let us suppose that $\mathcal{L}(\mathcal{A})$ is a regular tree language: there exists $\mathcal{B} = (\Sigma, Q, F, \Delta)$ a deterministic and complete bottom-up tree automaton recognizing it. For each $q \in Q$, we denote $\mathcal{L}_{\mathcal{B}}(q)$ the set of terms t of T_Σ such that $t \xrightarrow{*}_{\mathcal{B}} q(t)$. Let us recall the following basic property:

Property 21. $\forall C \in \mathcal{C}^n(\Sigma), \forall q \in Q, \forall (t_i)_{i\in[n]} \in \mathcal{L}_{\mathcal{B}}(q), \forall (t'_i)_{i\in[n]} \in \mathcal{L}_{\mathcal{B}}(q)$

$$(C[t_1,\ldots,t_n] \in \mathcal{L}(\mathcal{B}) \Leftrightarrow C[t'_1,\ldots,t'_n] \in \mathcal{L}(\mathcal{B})).$$

The sketch of proof is the following: we construct two terms, saying t_1 and t_2 such that t_1 belongs to $\mathcal{L}(\mathcal{A})$ and t_2 does not. Furthermore, t_1 and t_2 will differ only on some positions p where $t_1(p) = t_2(p) = f$ but subterms at these positions in t_1 satisfy the constraint c while in t_2, subterms at the same positions satisfy the constraint c'.
From t_1 and t_2 we deduce a general context C_g, intuitively the common prefix of t_1 and t_2, such that there exists $q_{\mathcal{B}}$ state of $\mathcal{B}$, $(u_i)_{i\in[n]}$ and $(u'_i)_{i\in[n]}$ terms of $\mathcal{L}_{\mathcal{B}}(q_{\mathcal{B}})$, such that $C_g[(u_p)] \in \mathcal{L}(\mathcal{A})$ and $C_g[(u'_p)] \notin \mathcal{L}(\mathcal{A})$. This will contradict Property 21 since we supposed that $\mathcal{L}(\mathcal{A})$ is recognizable.

Since $\mathcal{A}$ is complete, we can suppose without loss of generality that c and c' differ only by the splitting of a set, i.e. $\exists (E_k)_{k\in K}, I, J \subseteq [n]$ such that:

$$c = ((E_k)_{k\in K}, I \cup J),\ card(c) = k+1;$$
$$c' = ((E_k)_{k\in K}, I, J),\ card(c') = k+2.$$

$q \neq q'$ hence $q \not\equiv_{\mathcal{A}} q'$. We deduce that there exists a constrained term C over $\Sigma \cup Q$ such that $(C[q] \xrightarrow{*}_{\mathcal{A}} s \in F \Leftrightarrow C[q'] \xrightarrow{*}_{\mathcal{A}} s' \notin F)$. We stand that $s \in F$ and according to Lemma 18, there exists $\bar{C}$ a state-instance of C. $\bar{C}[q] \xrightarrow{*}_{\mathcal{A}} s$ since $C[q] \xrightarrow{*}_{\mathcal{A}} s$ and $\bar{C}[q'] \xrightarrow{*}_{\mathcal{A}} s'$ since $C[q'] \xrightarrow{*}_{\mathcal{A}} s'$.

Let us consider the constrained term $F_1 = f_c(s_1,\ldots,s_n)$ where $\forall k \in I \cup J, s_k = x$ and $\forall k \notin I \cup J, s_k = q_k$. Lemma 18 ensures the existence of a state-instance F'_1 of F_1. Then $F'_1[q_I] \xrightarrow{*}_{\mathcal{A}} q$ since $F_1[q_I] \xrightarrow{*}_{\mathcal{A}} q$.
Let C_1 be the constrained term $\bar{C}[F'_1]$ and q_I be the unique state present in the rule r at positions belonging to $I \cup J$. The run on $C_1[q_I]$ leads to the final state s since $C_1[q_I] = \bar{C}[F'_1[q_I]] \xrightarrow{*}_{\mathcal{A}} \bar{C}[q] \xrightarrow{*}_{\mathcal{A}} s$.

The constrained term F_2 is obtained from F_1 by replacing the root symbol f_c by $f_{c'}$. Hence, F_2 and F_1 have the same projection onto $T_{\Sigma}(\{x\})$. From Lemma 18 there exists F'_2 a state-instance of F_2. F'_2 is choosen in such a way that root subterms at the same position $k \notin I \cup J$ in F'_1 and F'_2 are identical (remember that c and c' only differ by the splitting of $I \cup J$ into I and J). Then $F'_2[q_I] \xrightarrow{*}_{\mathcal{A}} q'$ since $F_2[q_I] \xrightarrow{*}_{\mathcal{A}} q'$.
In the same way as previously, C_2 denotes $\bar{C}[F'_2]$. Let us notice that F'_1 (resp. C_1) and F'_2 (resp. C_2) have the same projection onto $T_{\Sigma}(\{x\})$. The run on $C_2[q_I]$ leads to the non final state s' since $C_2[q_I] = \bar{C}[F'_2[q_I]] \xrightarrow{*}_{\mathcal{A}} \bar{C}[q'] \xrightarrow{*}_{\mathcal{A}} s'$.

As we supposed that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$ and as $\mathcal{L}_{\mathcal{A}}(q_I)$ is infinite then there exists $q_{\mathcal{B}}$ state of $\mathcal{B}$ such that the set $T = \mathcal{L}_{\mathcal{A}}(q_I) \cap \mathcal{L}_{\mathcal{B}}(q_{\mathcal{B}})$ is infinite.
As T is infinite, and according to Lemma 19, there exist terms $(u_p)_{p\in\mathcal{V}(C_1)} \in T$ such that C'_1 defined by

- $\forall p \in \mathcal{P}os(C_1) \setminus \mathcal{V}(C_1), C'_1(p) = C_1(p)$;
- $\forall p \in \mathcal{V}(C_1), C'_1|_p = lab_{u_p}$,

is a constrained term. As $\forall p, lab_{u_p} \xrightarrow{*}_{\mathcal{A}} q_I$, the run of this constrained term is the final state s. The term t_1, projection of C'_1 onto T_Σ satisfies $t_1 \in \mathcal{L}(\mathcal{A})$.
In the same way, there exist terms $(u'_p)_{p \in \mathcal{V}(C_2)} \in T$ such that C'_2 defined by

- $\forall p \in \mathcal{P}os(C_2) \setminus \mathcal{V}(C_2), C'_2(p) = C_2(p)$;
- $\forall p \in \mathcal{V}(C_2), C'_2|_p = lab_{u'_p}$,

is a constrained term and the run of C'_2 is the non final state s'. As $\mathcal{A}$ is deterministic, t_2, the projection of C'_2 over T_Σ does not belong to $\mathcal{L}(\mathcal{A})$.

Let C_g be the projection of C_1 onto $T_\Sigma(\{x\})$ (which is the same as the projection of C_2). C_g is a context -without labels- over a single variable x.
We replace each occurence of x in C_g by distinct new variables: it results a context C'_g over distinct new variables $(x_p)_{p \in \mathcal{V}(C_g)}$ defined by

- $\forall p \in \mathcal{P}os(C_g) \setminus \mathcal{V}(C_g), C'_g(p) = C_g(p)$;
- $\forall p \in \mathcal{V}(C_g), C'_g(p) = x_p$.

We can prove that $t_1 \in \mathcal{L}(\mathcal{A}) = C_g[(u_p)]$ and $t_2 = C_g[(u'_p)]$. Moreover, $\forall p, u_p \xrightarrow{*}_{\mathcal{B}} q_{\mathcal{B}}$ and $u'_p \xrightarrow{*}_{\mathcal{B}} q_{\mathcal{B}}$, which contradicts the Property 21 since $t_1 \in \mathcal{A}$ and $t_2 \notin \mathcal{A}$. We deduce that $\mathcal{L}(\mathcal{A})$ is not recognizable, which ends the proof of Lemma 20.

Let us now prove that the condition of Proposition 14 is sufficient.

Lemma 22. *Let us suppose that for all rules of $\mathcal{R}$, $f(q_1, \ldots, q_n)[c] \to q$ and $f(q_1, \ldots, q_n)[c'] \to q'$, we have $q = q'$. Then $\mathcal{L}(\mathcal{A})$ is recognizable and we can compute a tree automaton recognizing $\mathcal{L}(\mathcal{A})$.*

Proof. Let $\mathcal{B} = (\Sigma, Q, F, \Delta)$ be the tree automaton whose set of rules Δ is defined by: $\forall f \in \Sigma_n, \forall (q_i)_{i \in [n]} \in Q$, $f(q_1, \ldots, q_n) \to q \in \Delta$ where q is defined by a rule $f(q_1, \ldots, q_n)[c] \to q$ of $\mathcal{R}$ (q is unique according to hypothesis of the lemma). We easily prove that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$. Hence $\mathcal{L}(\mathcal{A})$ is recognizable which ends the proofs of Lemma 22 and of Proposition 14.

Let $\mathcal{A} = (\Sigma, Q, F, \mathcal{R})$ be a normalized-complete $REC_{\neq}$ automaton such that for each state q of $\mathcal{A}$, $\mathcal{L}_{\mathcal{A}}(q)$ is infinite. According to Remark 13 and Proposition 14, we deduce that the recognizability problem of $\mathcal{L}(\mathcal{A})$ is decidable. Finally, according to Section 4.1, we deduce the following theorem:

Theorem 23. *The recognizability problem in the class $REC_{\neq}$ is decidable.*

5 Conclusion

We proved here that recognizability problem is decidable in the class $REC_{\neq}$. It implies e.g. the decidability of recognizability of $\Phi(\mathcal{L})$ where Φ is a quasi-algebraic tree homomorphism (i.e. variables occurr at depth one in a letter's image) and $\mathcal{L}$ a recognizable language.
It provides also a rather simple algorithm for testing recognizability of the set of normal forms (resp. of the set of direct descendants of a recognizable language) for some subclasses of rewrite systems (like shallow ones).

Furthermore, the notions we define here -like constrained terms- could perhaps be extended and help to answer the two following open problems:
Is recognizability decidable in the class of reduction automata?
Can we decide whether the homomorphic image of a recognizable tree language is recognizable?

References

1. A. Arnold. Le théorème de transversale rationnelle dans les arbres. *Mathematical System Theory*, 13:275–282, 1980.
2. B. Bogaert, F. Seynhaeve, and S. Tison. The recognizability problem for tree automata with comparisons between brothers. Technical Report IT. 317, Laboratoire d'Informatique Fondamentale de Lille, Nov. 1998.
3. B. Bogaert and S. Tison. Equality and disequality constraints on direct subterms in tree automata. In P. Enjalbert, A. Finkel, and K. W. Wagner, editors, *9th Annual Symposium on Theoretical Aspects of Computer Science*, volume 577 of *Lecture Notes in Computer Science*, pages 161–171, 1992.
4. A. Caron, H. Comon, J. Coquidé, M. Dauchet, and F. Jacquemard. Pumping, cleaning and symbolic constraints solving. In *Proceedings, International Colloquium Automata Languages and Programming*, volume 820 of *Lecture Notes in Computer Science*, pages 436–449, 1994.
5. A. Caron, J. Coquidé, and M. Dauchet. Encompassment properties and automata with constraints. In C. Kirchner, editor, *Proceedings. Fifth International Conference on Rewriting Techniques and Applications*, volume 690 of *Lecture Notes in Computer Science*, pages 328–342, 1993.
6. H. Comon, M. Dauchet, R. Gilleron, S. Tison, and M. Tommasi. Tree automata techniques and applications, 1998. available through WWW from http://l3ux02.univ-lille3.fr/tata.
7. M. Dauchet, A.-C. Caron, and J.-L. Coquidé. Reduction properties and automata with constraints. *Journal of Symbolic Computation*, 20:215–233, 1995.
8. M. Dauchet and S. Tison. Réduction de la non-linéarite des morphismes d'arbres. Technical Report IT-196, Laboratoire d'Informatique Fondamentale de Lille, Université des Sciences et Technologies de Lille, Villeneuve d'Ascq, France, 1990.
9. F. Gécseg and M. Steinby. *Tree Automata*. Akademiai Kiado, 1984.
10. T. Genet. Decidable approximations of sets of descendants and sets of normal forms. volume 1379 of *Lecture Notes in Computer Science*, pages 151–165, Tsukuba, 1998.
11. J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82(1):1–33, July 1989.
12. G. Kucherov and M. Tajine. Decidability of regularity and related properties of ground normal form languages. In *Proceedings of 3rd International Workshop on Conditional Rewriting Systems*, pages 150–156, 1992.
13. J. Mongy. *Transformation de noyaux reconnaissables d'arbres. Forêts RATEG*. PhD thesis, Laboratoire d'Informatique Fondamentale de Lille, Université des Sciences et Technologies de Lille, Villeneuve d'Ascq, France, 1981.
14. S. Vágvölgyi and R. Gilleron. For a rewrite system it is decidable whether the set of irreducible ground terms is recognizable. *Bulletin of the European Association of Theoretical Computer Science*, 48:197–209, Oct. 1992.
15. J. Waldmann. Normalization of s-terms is decidable. volume 1379 of *Lecture Notes in Computer Science*, pages 138–150, Tsukuba, 1998.