Ground Reducibility is EXPTIME-complete

Hubert Comon
CNRS and Laboratoire Spécification et Vérification
Ecole Normale Supérieure de Cachan
61 Avenue du président Wilson
94235 CACHAN cedex, France
Hubert.Comon@lsv.ens-cachan.fr

Florent Jacquemard
Max-Planck Institut für Informatik
Im Stadtwald, Gebäude 46.1
D-66123 Saarbrücken
Germany
florent@mpi-sb.mpg.de

Abstract

We prove that ground reducibility is EXPTIME-complete in the general case. EXPTIME-hardness is proved by encoding the computations of an alternating Turing machine whose space is polynomially bounded. It is more difficult to show that ground reducibility belongs to DEXPTIME. We associate first an automaton with disequality constraints $\mathcal{A}_{\mathcal{R},t}$ *to a rewrite system* $\mathcal{R}$ *and a term* t. *This automaton is deterministic and accepts a term* u *if and only if* t *is not ground reducible by* $\mathcal{R}$. *The number of states of* $\mathcal{A}_{\mathcal{R},t}$ *is* $O(2^{\|R\| \times \|t\|})$ *and the size of the constraints are polynomial in the size of* $\mathcal{R}, t$. *Then we prove some new pumping lemmas, using a total ordering on the computations of the automaton. Thanks to these lemmas, we can give an upper bound to the number of distinct subtrees of a minimal successful computation of an automaton with disequality constraints. It follows that emptiness of such an automaton can be decided in time polynomial in the number of its states and exponential in the size of its constraints. Altogether, we get a simply exponential deterministic algorithm for ground reducibility.*

1 Introduction

Ground reducibility of a term t w.r.t. a term rewriting system $\mathcal{R}$ expresses that all ground instances (instances without variables) of t are reducible by $\mathcal{R}$. This property is fundamental in automating inductive proofs in equational theories without constructors [10]. It is also related to *sufficient completeness* in algebraic specifications (see e.g. [12]). Roughly, it expresses that all cases have been covered by $\mathcal{R}$ and that t will be reducible for any inputs. Many papers have been devoted to decision of ground reducibility. Let us report a brief history of the milestones, starting only in 1985 with the general case.

Ground reducibility was first shown decidable by D. Plaisted [14]. The algorithm is however quite complex: a tower of 9 exponentials though there is no explicit complexity analysis in the paper. D. Kapur et al. gave another decidability proof [12] which is conceptually simpler, though still very complicated, and whose complexity is a tower of 7 exponentials in the size of $\mathcal{R}$, t. More precisely, they show that checking the reducibility of all ground instances of t can be reduced to checking the reducibility of all ground instances of t of depth smaller than $N(\mathcal{R})$ where $N(\mathcal{R})$ is a tower of 5 exponentials in the size of $\mathcal{R}$. A third proof was proposed by E. Kounalis in [13]. The result is generalized to co-ground reducibility and the expected complexity is 5 exponentials, though there is no explicit complexity analysis in the paper. These three algorithms use combinatorial arguments and some "pumping property": if there is a deep enough irreducible instance of t, then there is also a smaller instance which is also irreducible. This yielded the idea of making explicit the pumping argument as a pumping lemma in some tree language. In support of this idea, when both t and the left members of $\mathcal{R}$ are *linear*, i.e. each variable appears only once, then the set of reducible instances of t is accepted by a finite tree automaton [8]. Hence the set of irreducible ground instances is also accepted by a tree automaton, by complement. This easily gives a simply exponential algorithm in the linear case. (As we will see this algorithm is optimal).

H. Comon expressed first the problem of ground reducibility as an emptiness problem for some tree language [3]. He also gave a decision proof whose complexity is even worse than the former ones. A.-C Caron, J.-L. Coquidé and M. Dauchet proved a very beautiful result in 1993 [2, 5], enlighting the pumping properties and their difficulty. They actually show a more general result: the first-order theory of unary encompassment predicates is decidable. And it turns out that ground reducibility can be expressed as a simple formula in this logic. Their technique consists in associating an automaton with each formula, in the spirit of Büchi's and Rabin's method. The kind of automata which is appropriate here is what they call *reduction automata*, a particular case of *automata with constraints* introduced by M. Dauchet in 1981. Such tree automata have the ability to check for equality or disequality of some subtrees before applying a transition rule. In general, emptiness of languages recognized by such automata is undecidable. However, when we only allow a fixed number of equality tests on each computation branch, then emptiness becomes decidable. Unfortunately, their result does not give any information about possible efficient algorithms. The complexity which results from their proof is not better than Plaisted's bound. We tried to specialize the tree automata technique for ground reducibility and we got in this way a triple exponential bound [4]. This is better than previous methods, but still far from the lower bound.

The problem in all works about ground reducibility is that they give a bound on the depth of a minimal irreducible instance of t (or a minimal term accepted by the automaton). However, after establishing carefully such an upper bound, they use a brute-force algorithm, checking the reducibility of all terms of depth smaller than the bound, which increases the complexity by a double exponential.

We use here a different approach. We still rely on automata with disequality constraints. However, we do not try to give a bound on the depth of an accepted term. Rather, we show a stronger result: with an appropriate notion of minimality, a minimal term accepted by the automaton contains at most an exponential number of distinct subterms. To prove this, we use a generalization of pumping to arbitrary replacements for which the term is decreasing according to some well chosen well founded ordering. With a few more ingredients, this yields an algorithm for deciding the emptiness of an automaton with disequality constraints which runs in polynomial time w.r.t. the number of states and in exponential time w.r.t. the size of the constraints. On the other hand, we show that ground reducibility of t w.r.t. $\mathcal{R}$ can be reduced to the emptiness problem for an automaton $\mathcal{A}$ with disequality constraints whose number of states is an exponential in the size of $\mathcal{R}$ and t and whose constraints are polynomial in size. Altogether, we have a simply exponential algorithm for ground reducibility.

This result is optimal since ground reducibility is EXPTIME-hard, already for linear rewrite systems and linear t. A $O(2^{\frac{n}{\log n}})$ lower bound was proved by Kapur et al [11]. We give here a simple proof of EXPTIME-hardness: the computations of alternating Turing machines with polynomially bounded space can be encoded (in polynomial time) into ground reducibility. Hence, quite surprisingly, ground reducibility is not (at least in theory) harder in the general case than in the linear case.

In section 2 we recall the definition of automata with disequality constraints. In section 3, we show how to construct an automaton with disequality constraints whose emptiness is equivalent to the ground reducibility of t w.r.t. $\mathcal{R}$ and we analyse carefuly the complexity of such a construction. Section 4 is devoted to pumpings lemmas for automata with disequality constraints. These lemmas are applied in section 5 to derive an optimal algorithm which checks the emptiness of (the language recognized by) an automaton with disequality constraints. Finally, we study the lower bound of ground reducibility in section 6.

Some proofs are long and technical, hence not included in the present abstract. Except for a small combinatorial argument due to B. Reed, the complete proofs can be found in F. Jacquemard's thesis [9] (in French). A complete version of the paper is also available on the Web (`http://www.mpi-sb.de/~florent/art/`).

2 Automata with disequality constraints

$\mathcal{F}$ will always be a fixed finite set of function symbols (together with their arity). The set of (ground) terms built on $\mathcal{F}$ is written $\mathcal{T}$ (or $\mathcal{T}(\mathcal{F})$). A *position* is a string of positive integers. Λ is the empty string. Positions are ordered according to the prefix ordering: $p \leq q$ iff there is a string r such that $p \cdot r = q$.

As usual, a finite term t can be viewed as a mapping from its set of positions $Pos(t)$ into $\mathcal{F}$. For instance, if $t = f(g(a), b)$, $Pos(t) = \{\Lambda, 1, 11, 2\}$ and e.g. $t(1) = g$. If $p \in Pos(t)$, we write $t|_p$ for the subterm of t at position p and $t[u]_p$ for the term obtained by replacing $t|_p$ with u in t. A *context* is a term $C[\cdot]_p$ with a hole (formally, it is a pair of a term and a position p, the subterm at position p being irrelevant).

Definition 1 *An* automaton with disequality constraints *(or ADC for short) is a tuple (Q, Q_f, Δ) where Q is a finite set of states, Q_f is the subset of Q of* final states *and Δ is a finite set of transition rules of the form:*

$$f(q_1, \dots, q_n) \xrightarrow{c} q$$

where $f \in \mathcal{F}$ has arity n, $q_1, \dots, q_n \in Q$ and c is a conjunction of disjunctions of constraints $\pi_1 \neq \pi_2$ where π_1, π_2 are strings of natural numbers. The empty conjunction is written $\top$. q is called the target state *of the rule.*

Definition 2 *A ground term t* satisfies *a constraint $\pi_1 \neq \pi_2$ (which we write $t \models \pi_1 \neq \pi_2$) if both π_1 and π_2 are positions of t and $t|_{\pi_1} \neq t|_{\pi_2}$. This notion of satisfaction is extended to conjunctions and disjunctions as expected. (In particular $t \models \top$ for every t).*

Definition 3 *A* run *of the automaton $\mathcal{A}$ on a term t is a function ρ from $Pos(t)$ into Δ such that, for every $p \in Pos(t)$, if $t(p)$ has arity n then $\rho(p)$ is a rule $f(q_1, \ldots, q_n) \xrightarrow{c} q$ and*

- *for every $1 \leq i \leq p$, $\rho(p \cdot i)$ is a rule whose target is q_i*
- $t \models c$

If only the first condition is met by ρ, ρ will be called a weak run.

A term t is accepted by $\mathcal{A}$ *if there is a run ρ of $\mathcal{A}$ on t such that $\rho(\Lambda)$ is a rule whose target is a final state.*

Runs of $\mathcal{A}$ can also be seen as terms over the alphabet Δ.

Example 4 Let $\mathcal{F} = \{f, a, b\}$ and $Q = \{q\} = Q_f$.

$$\Delta = \{ \begin{array}{lcccllcl} r_1 : & a & \rightarrow & q & r_2 : & b & \rightarrow & q \\ r_3 : & f(q,q) & \xrightarrow{1 \neq 2} & q & \} & & & \end{array}$$

This defines an automaton (which accepts the terms irreducible by the rule $f(x, x) \rightarrow 0$).

$f(a, b)$ is accepted since $\rho = r_3(r_1, r_2)$ is a run on t such that r_3 yields a final state. $f(a, a)$ is not accepted by $\mathcal{A}$: there is a weak run $r_3(r_1, r_1)$ on $f(a, a)$ but the disequality of r_3 is not satisfied.

Note that in general ADC can be non-deterministic (more than one run on a term) or not completely specified (no run on some term). However, given a run ρ, there is a unique term $[\rho]$ associated to ρ.

Definition 5 *Let $\mathcal{A} = (Q, Q_f, \Delta)$ be an ADC and ρ a weak run of $\mathcal{A}$ on t. An* equality *of ρ is a triple of positions (p, π_1, π_2) such that $p, p \cdot \pi_1, p \cdot \pi_2 \in Pos(t)$, $\pi_1 \neq \pi_2$ is in the constraint of $\rho(p)$ and $t|_{p \cdot \pi_1} = t|_{p \cdot \pi_2}$.*

In particular, a weak run without any equality is a run. The equalities in a run are also classified according to a particular position p_0 of t:

- (p, π_1, π_2) is *close* to p_0 iff $p \leq p_0 < p \cdot \pi_1$ or $p \leq p_0 < p \cdot \pi_2$
- (p, π_1, π_2) is *far* from p_0 (or *remote*) iff $p \cdot \pi_1 \leq p_0$ or $p \cdot \pi_2 \leq p_0$.

These two possible situations are depicted on figures 1 and 2.

3 Reducing Ground reducibility to an emptiness problem for ADC

In this section, we show how to construct an ADC whose emptiness is equivalent to the ground reducibility problem

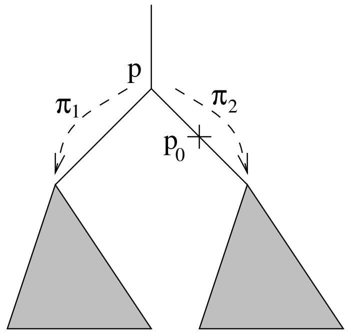

Figure 1. An equality close to p_0

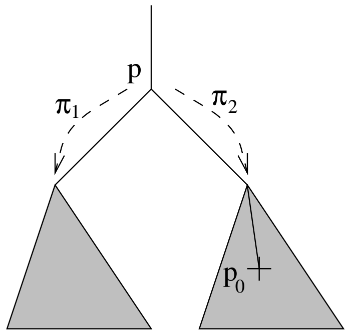

Figure 2. An equality far from p_0

and we show precisely the size of such an automaton. We start with an ADC accepting the set of irreducible ground terms.

We assume the reader familiar with term rewriting systems (see [6] for a survey). We use the subsumption quasi-ordering on terms : $s \leq t$ if there is a substitution σ such that $s\sigma = t$. Two terms s, t are *similar* if $s \leq t$ and $t \leq s$. The set of variables occurring in a term t is denoted $Var(t)$. Finally, the *size* of a term t, which is denoted $\|t\|$, is the cardinal of its positions and the *size* of a rewrite system $\mathcal{R}$, which is denoted $\|\mathcal{R}\|$, is the sum of the sizes of its left members.

3.1 Normal forms ADC

Let $\mathcal{L}$ be the set of left hand sides of a rewrite system $\mathcal{R}$. First, let $\mathcal{L}_1$ be the subset of linear terms in $\mathcal{L}$, $\mathcal{L}_2$ its complement in $\mathcal{L}$ and $\mathcal{L}_3$ the set of linearized versions of terms in $\mathcal{L}_2$ (i.e. terms obtained by replacing in some $t \in \mathcal{L}_2$ each occurrence of a variable by a new variable, yielding a linear term). The initial set of states Q_0 consists in all strict subterms of elements in $\mathcal{L}_1 \cup \mathcal{L}_3$ plus two special states: q_x which accepts all terms and q_r which accepts only reducible terms (hence is a failure state). We assume that all terms are considered up to renaming (in particular any two terms are assumed to share no variables in what follows). Also, states will be written q_t instead of t.

The set of states Q of the automaton consists in all unifiers of unifiable subsets of $Q_0 \setminus \{q_r\}$, plus the additional state q_r (in this way Q contains Q_0). The rules are defined by:

$$f(q_{t_1}, \ldots, q_{t_n}) \xrightarrow{c} q_t$$

with

- $t = r$ and $c = \top$ if one of the t_i's is r or if $f(t_1, \ldots, t_n)$ is an instance of some $u \in \mathcal{L}$.
- If $f(t_1, \ldots, t_n)$ is not an instance of any term in $\mathcal{L}$, then t is the unifier of all terms $u \in Q_0 \setminus \{q_r\}$ such that $f(t_1, \ldots, t_n)$ is an instance of u.
- When $q_t \neq q_r$, the constraint c is defined by

$$\bigwedge_{\substack{l \in \mathcal{L}_2 \\ l \text{ and } t \text{ unifiable}}} \quad \bigvee_{\substack{x \in Var(l) \\ l|_\pi = l|_{\pi'} = x \\ \pi \neq \pi'}} \quad \pi \neq \pi'$$

The final states are all states, except q_r.

Let $\mathcal{A}_{\mathrm{NF}(\mathcal{R})}$ be the automaton constructed in this way. $\mathcal{A}_{\mathrm{NF}(\mathcal{R})}$ is not necessary complete (the automaton may have no run on terms that are reducible by a non-left linear rule). It is however deterministic.

Proposition 6 *The above automaton $\mathcal{A}_{\mathrm{NF}(\mathcal{R})}$ accepts the set of terms that are irreducible by $\mathcal{R}$. Its number of states is an exponential in the size of $\mathcal{R}$. The constraints have always a size bounded by the size $\mathcal{R}$.*

3.2 Ground reducibility and ADC

If t is a linear term, then its ground reducibility is equivalent to the emptiness of the intersection of $L(\mathcal{A}_{\mathrm{NF}(\mathcal{R})})$ with the set of instances of t. Since the class ADC is closed by intersection with a regular language (it can be computed in time the product of the sizes of both automata), deciding ground reducibility amounts to decide emptiness of an ADC whose number of states is $O(2^{\|\mathcal{R}\|} \times \|t\|)$ and constraints have a size $O(\|\mathcal{R}\|)$. It is a bit more difficult when t is not linear since, in such a situation, the set of irreducible instances of t is not necessarily recognized by an ADC. For this reason, we have to compute directly an automaton having the desired property. Let $\mathcal{A}_{\mathrm{NF}(\mathcal{R})} = (Q_{\mathrm{NF}}, Q^f_{\mathrm{NF}}, \Delta_{\mathrm{NF}})$. We compute $\mathcal{A}_{\mathrm{NF,t}} \stackrel{\text{def}}{=} (Q_{\mathrm{NF,t}}, Q^f_{\mathrm{NF,t}}, \Delta_{\mathrm{NF,t}})$ as follows:

- $Q_{\mathrm{NF,t}} \stackrel{\text{def}}{=} \{t\sigma|_p \mid p \in Pos(t)\} \times Q_{\mathrm{NF}}$ where σ ranges over substitutions from $NLV(t)$ (the set of variables occurring at least twice in t) into Q^f_{NF}.
- For all $f(q_1, \ldots, q_n) \xrightarrow{c} q \in \Delta_{\mathrm{NF}}$, and all $u_1, \ldots, u_n \in \{t\sigma|_p \mid p \in Pos(t)\}$, $\Delta_{\mathrm{NF,t}}$ contains the following rules:
  - $f([q_{u_1}, q_1], \ldots, [q_{u_n}, q_n]) \xrightarrow{c \wedge c'} [q_{f(u_1,\ldots,u_n)}, q]$ if $f(u_1, \ldots, u_n) = t\sigma_0$ and c' is constructed as sketched below.
  - $f([q_{u_1}, q_1], \ldots, [q_{u_n}, q_n]) \xrightarrow{c} [q_{f(u_1,\ldots,u_n)}, q]$ if $[q_{f(u_1,\ldots,u_n)}, q] \in Q_{\mathrm{NF,t}}$ and we are not in the first case.
  - $f([q_{u_1}, q_1], \ldots, [q_{u_n}, q_n]) \xrightarrow{c} [q_q, q]$ in all other cases

c' is constructed as follows. From $f(u_1, \ldots, u_n)$ we can retrieve the rules applied at position p in t. Assume that the rule at p checks $\pi_1 \neq \pi_2$. This amounts to check $p\pi_1 \neq p\pi_2$ at the root position of t. Let $\mathcal{D}$ be all disequalities $p\pi_1 \neq p\pi_2$ obtained in this way. The non linearity of t implies some equalities: let $\mathcal{E}$ be the set of equalities $p_1 = p_2$, for all positions p_1, p_2 such that $t|_{p_1} = t|_{p_2}$ is a variable. Now, c' is the set of disequalities $\pi \neq \pi'$ which are not in $\mathcal{D}$ and that can be inferred from $\mathcal{D}, \mathcal{E}$ using the rules

$$pp_1 \neq p_2,\ p = p' \ \vdash\ p'p_1 \neq p_2$$
$$p \neq p',\ pp_1 = p_2 \ \vdash\ p'p_1 \neq p_2$$

For instance, let $t = f(x, f(x, y))$ and assume that the automaton $\mathcal{A}_{\mathrm{NF}}$ contains a rule $f(q, q) \xrightarrow{1 \neq 2} q$. Then the automaton $\mathcal{A}_{\mathrm{NF,t}}$ will contain the rule $f([q_q, q], [q_{f(q,q)}, q]) \xrightarrow{1 \neq 2 \wedge 1 \neq 22} q$.

The final states are $[q_u, q_f]$ where $q_f \in Q^f_{\mathrm{NF}}$ and u is an instance of t.

Proposition 7 *t is ground reducible by $\mathcal{R}$ iff the language accepted by $\mathcal{A}_{\mathrm{NF,t}}$ is empty. The number of states of this automaton is $O(2^{c \times \|t\| \times \|\mathcal{R}\|})$ where c is a constant. The size of the constraints of each rule is $O(\|t\|^4 \times \|\mathcal{R}\|^2)$.*

Moreover, the number of rules of the automaton is $O(2^{c \times \|t\| \times \|R\| \times a} \times \|\mathcal{F}\|)$ where a is the maximal arity of a function symbol and $\|F\|$ is the number of function symbols.

The automaton does not recognize only irreducible ground instances of t. However, if u is accepted by $\mathcal{A}_{\mathrm{NF,t}}$, then we can construct a term u' which is an irreducible instance of t and which is still accepted. u' is obtained by replacing $u|_{p_1}, \ldots, u_{p_n}$ with $u|_{p_0}$ in u whenever $p_0, p_1, \ldots, p_n$ are positions of the same variable in t.

4 Generalized pumping lemmas

This is the crux part of our proof. We assume here a well founded ordering $\gg$, total on runs and monotonic (i.e. $\rho \gg \rho'$ implies that for every context C, $C[\rho]_p \gg C[\rho']_p$).

Definition 8 *A* pumping *(w.r.t. $\gg$) is a replacement $\rho[\rho']_p$ where ρ, ρ' are runs such that the target state of ρ' is the same as the target state of $\rho|_p$ and $\rho \gg \rho[\rho']_p$*

This definition generalizes the usual pumping definition: a usual pumping is also a pumping according to the above definition, as soon as $\gg$ contains the $\gg$ the *embedding ordering*.

Lemma 9 *Every pumping $\rho[\rho']_p$ is a weak run and every equality of it is either far from p or close to p.*

Hence, given a large enough run ρ, we will successively show how to construct a weak run by pumping which does not contain any close equality (this uses combinatorial arguments only) then we show how to remove far equalities by further successive pumpings.

4.1 Pumping without creating close equalities

Given an ADC $\mathcal{A}$ and an integer k, we let:

$$g(\mathcal{A}, k) \stackrel{\text{def}}{=} (e \times k + 1) \times |Q| \times 2^{c(\mathcal{A})} \times c(\mathcal{A})!$$

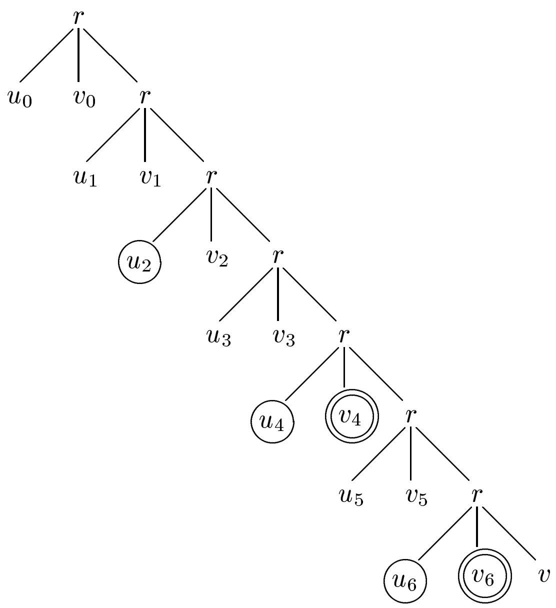

Figure 3. A run with a possible pumping

where $e \stackrel{\text{def}}{=} \sum_{n=0}^{+\infty} \frac{1}{n!}$ is the exponential basis and $c(\mathcal{A})$ is the maximal size of a constraint in $\mathcal{A}$. Then we have a pumping lemma which generalizes those of [5, 4]:

Lemma 10 *If ρ is a run of $\mathcal{A}$ and $p_1, \ldots, p_{g(\mathcal{A},k)}$ are positions of ρ such that $\rho|_{p_1} \gg \ldots \gg \rho|_{p_{g(\mathcal{A},k)}}$ and target states of $\rho|_{p_1}, \ldots, \rho|_{p_{g(\mathcal{A},k)}}$ are identical, then there are indices $i_0, \ldots, i_k$ such that the weak run $\rho[\rho_{p_{i_j}}]_{p_{i_0}}$ does not contain any close equality.*

Example 11 This example illustrates the principle of the proof of lemma 10. Let $\mathcal{F}$ contain a ternary symbol and constants and $\mathcal{A}$ be an ADC containing the following transition rule:

$$r: \quad f(q_1, q_2, q_3) \xrightarrow{1 \neq 31 \wedge 1 \neq 32} q$$

Consider moreover the run:

$$\begin{array}{lllllll}
\rho = & r(& u_0, & v_0, & & & \\
 & & r(& u_1, & v_1, & & \\
 & & & r(& u_2, & v_2, & \\
 & & & & r(& u_3, & v_3, \\
 & & & & & r(& u_4, \quad v_4 \\
 & & & & & & r(\quad u_5, v_5, \\
 & & & & & & \quad\quad r(u_6, v_6, v)))))))
\end{array}$$

which is also depicted on figure 3.

We are going to show that ρ is large enough so as to be able to find a pumping which does not create any close equality. Assume first that replacing each subtree rooted with r at position 3 creates a close equality. This means that, for all $i = 2, ..6$, $u_i = u_0$ or $v_i = v_0$. Then it is possible to extract a subsequence of three indices i_1, i_2, i_3 such that $(u_0 = u_{i_1} = u_{i_2} = u_{i_3}) \vee (v_0 = v_{i_1} = v_{i_2} = v_{i_3})$. Assume we are in the first case of the alternative and that, for instance $u_0 = u_2 = u_4 = u_6$. Now we replace the subterm $r(u_2, v_2, \ldots)$ with $r(u_4, v_4, \ldots)$ and $r(u_6, v_6, v)$ respectively. Since $u_2 = u_4 = u_6 \neq u_1$, if each of these replacements creates a close equality, we must have $v_1 = v_4 = v_6$. Finally, replacing $r(u_4, v_4, \ldots)$ with $r(u_6, v_6, v)$, we cannot create a close equality since $u_4 = u_6 \neq u_3$ and $v_4 = v_6 \neq v_3$.

In this example, the ordering $\gg$ was any simplification ordering on runs. A similar example could be built where the positions p_i are incomparable w.r.t. the prefix ordering. Note that we did not use yet its totality.

4.2 Pumping without creating equalities

In the following definition, $\mathcal{M}$ is supposed to express some "non minimality" of the run: if $\mathcal{M}$ holds true, then there are many possible replacements which yield smaller weak runs without creating any close equality. The goal is to show that, for large enough runs, it is possible to construct a run which is smaller w.r.t. $\gg$.

Definition 12 *$\mathcal{M}$ is the predicate (defined relatively to an ADC $\mathcal{A}$ and an ordering $\gg$) which holds true on ρ, p, k iff p is a position of ρ and k runs $\rho|_p \gg \rho_k \gg \ldots \gg \rho_1$ such that $\rho(p), \rho_1(\Lambda), \ldots, \rho_k(\Lambda)$ have the same target state and for all $1 \leq i \leq k$ the pumping $\rho[\rho_i]_p$ does not contain any close equality.*

Let $h(\mathcal{A}, k) = (1 + c(\mathcal{A})) \times g(\mathcal{A}, k + c(\mathcal{A})) + k + c(\mathcal{A})$. The following propagation lemma is the crux part of our proof. (It is also very technical to prove). It explains how to get rid of remote equalities, if we have enough pumpings which do not create close equalities.

Lemma 13 (Propagation lemma) *Let ρ be a run of $\mathcal{A}$, $p \in Pos(\rho)$ and k be an integer such that $k^2 \geq h(\mathcal{A}, k)$. Then, if $\mathcal{M}(\rho, p, h(\mathcal{A}, k))$ is true, one of the following holds:*

1. *there is a run ρ' such that $\rho|_p \gg \rho'$ and $\rho[\rho']_p$ is a run*
2. *There exists a $p' < p$ such that $\mathcal{M}(\rho, p', h(\mathcal{A}, k))$ is true.*

Sketch of the proof:

If we are not in the first case of the lemma, then for each run ρ_i such that $\rho|_p \gg \rho_i$ and $\rho[\rho_i]_p$ does not create a close equality (as in the definition of $\mathcal{M}$), $\rho[\rho_i]_p$ contains a far equality. This means that each ρ_i already occurs at some position π_i in ρ with π_i incomparable with p. More precisely, there are prefixes $p_1, \ldots, p_{h(\mathcal{A},k)}$ of p and disequalities $\pi'_i \neq \pi''_i$ checked at p_i such that $\rho[\rho_i]_p|_{p_i \pi''_i} = \rho[\rho_i]_p|_{p_i \pi'_i}$. And $p_i \pi'_i$ is a prefix of p. The situation is depicted on figure 4.

Let $k_1 = \frac{h(\mathcal{A},k) - k - c(\mathcal{A})}{n(\mathcal{A})}$ ($n(\mathcal{A})$ is the maximal number of constraints checked by a rule). $h(\mathcal{A}, k)$ is chosen in such a way that $\mathcal{M}(\rho, p, h(\mathcal{A}, k))$ implies $\mathcal{M}(\rho, p', k)$ for p' ranging in the set of the $k_1 + c(\mathcal{A})$ largest positions in $\{p_1, \ldots, p_{h(\mathcal{A},k)}\}$ (this is a simple extraction: among all far equalities created in $\rho[\rho_i]_p$ some have to be far enough). However, this is not quite satisfactory since we do not "propagate" the predicate $\mathcal{M}$ with the same arguments.

Now, consider the k_1 smallest positions in $\{p_1, \ldots, p_{h(\mathcal{A},k)}\}$. By extraction, we may assume that $p_1 \pi''_1 < \ldots < p_{k_1} \pi''_{k_1}$. Now the terms $\rho|_{p_i \pi'_i}$ are distinct and we apply to them lemma 10. This yields $k + c(\mathcal{A})$ possible pumpings at a position $p_{i_0} \pi'_{i_0}$, which do not create close equalities. One of these pumpings is indicated by the arrow on figure 4.

The final idea is to combine the two above constructions. Roughly, we have on one hand k replacements $\rho[\rho_i]_p$ such that $\mathcal{M}(\rho, p', k)$ and on the other hand we have k pumpings $\rho[\rho'_j]_{p_{i_0}}$ such that also $\mathcal{M}(\rho, p', k)$ where $p' < p$. Combining each pumping with each replacement, we get $\mathcal{M}(\rho, p', k^2)$ which yields the lemma according to the assumption on k. $\square$

Note that we used some other properties of $\gg$ in this proof, for instance its monotonicity. We also used its totality, in order to be able to apply lemma 10.

Then, we initiate the process with lemma 10 and use the propagation lemma to push the position under which no equality is created, up to the root of the tree. With simple sufficient conditions for the inequality $k^2 \geq h(\mathcal{A}, k)$, this yields:

Lemma 14 *Let $\mathcal{A}$ be an ADC . There is a bound $b(\mathcal{A}) = c \times |Q|^3 \times 2^{P(c(\mathcal{A}))}$ where P is a polynomial, such that, if $\mathcal{M}(\rho, p, b(\mathcal{A}))$ for some position p of ρ then there is a run $\rho \gg \rho'$ such that $\rho(\Lambda)$ and $\rho'(\Lambda)$ have the same target state.*

The lemma states that, if a run is large enough, so as to accommodate $b(\mathcal{A})$ pumpings which do not create close equalities, then there is a strictly smaller run. Now, an ADC accepts at least a tree iff it accepts a tree with a minimal run (w.r.t. $\gg$). Lemma 14 states that such a minimal run has to contradict the predicate $\mathcal{M}$, hence to be small enough. The algorithm of the next section exploits this property, searching a minimal run within a given amount of space.

5 Emptiness decision for ADC

In this section we present the following result:

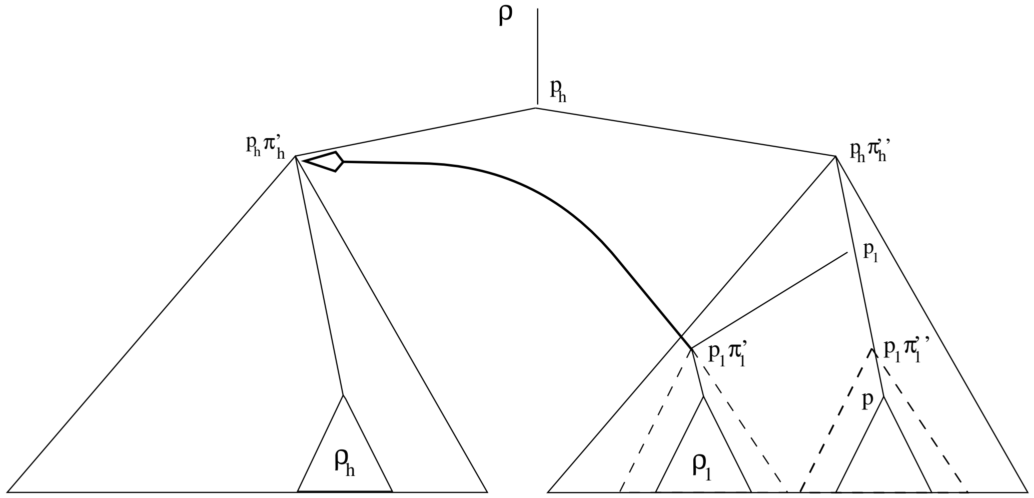

Figure 4. Equalities created by the replacements at position p

Theorem 15 *There is an algorithm which decides the emptiness of an ADC and which runs in time $O(P_1(|Q|) \times 2^{P_2(c(\mathcal{A}))})$ where P_1, P_2 are polynomials.*

We use a marking algorithm in which each state is marked with some successful runs yielding this state. This generalizes the usual marking algorithm for finite bottom-up tree automata: we do not keep only the information that a state is inhabited, but also keep witnesses of this fact. The witnesses are used to check the disequality constraints higher up in the run.

It can be simply stated as follows:

> start with a mapping f which associates each state q with an empty set E_q^0, then saturate the states E_q^0 using the rule:
>
> $$\{\rho_1, \ldots, \rho_n\} \in \bigcup_{q \in Q} E_q^n \vdash r(\rho_1, \ldots, \rho_n) \in E_{q_0}^{n+1}$$
>
> If $r(\rho_1, \ldots, \rho_n)$ is a run whose target state is q_0 and $\neg\mathcal{M}(r(\rho_1, \ldots, \rho_n), p, b(\mathcal{A}))$ for every position p which is a prefix of some position checked by r. (This expresses a minimality condition).

We have to prove on one hand that the saturated set

$$E^* \stackrel{\text{def}}{=} \bigcup_{q \in Q} \bigcup_{n \geq 0} E_q^n$$

contains an accepting run iff $\mathcal{A}$ accepts at least one tree (completeness) and on the other hand that E^* can be computed with the expected complexity (termination).

Lemma 16 *If E^* does not contain any accepting run then $\mathcal{A}$ does not accept any term.*

Sketch of the proof: Assume that $\mathcal{A}$ accepts t and ρ is a successful run on t. Assume moreover that ρ is minimal w.r.t. $\gg$. Let $\rho = r(\rho_1, \ldots, \rho_n)$. By lemma 14, $\mathcal{M}(\rho, p, b(\mathcal{A}))$ is true for every position p of ρ. By construction, this means that $\rho \in E^*$ whenever $\rho_1, \ldots, \rho_n \in E^*$. We conclude by an induction on the number of states. □

In order to give a complexity bound, we need an additional argument (a generalization of König's theorem for bipartite graphs to hypergraphs). Let us first define a notion of dependency in hypergraphs:

Definition 17 *Let S be a set and n, k be integers. The nuples $\overline{s_1}, \ldots, \overline{s_k}$ of elements in S are* independent *iff there is a set $I \subseteq \{1, \ldots, n\}$ such that*

- $\forall i \in I, s_{1,i} = \ldots = s_{k,i}$
- $\forall i \notin I, \forall j \neq j', s_{j,i} \neq s_{j',i}$

Let $\mathcal{A}$ be an ADC and $\{\pi_1, \ldots, \pi_{c(\mathcal{A})}\}$ be the set of suffixes of positions which are checked by some rule of $\mathcal{A}$. Let $\rho = r(\rho_1, \ldots, \rho_n)$ be a run of $\mathcal{A}$. Then Check(ρ) is the tuple $(\rho_1, \ldots, \rho_{c(\mathcal{A})}) \in (\mathcal{T}(\Delta) \cup \{\perp\})^{c(\mathcal{A})}$ such that $\rho_i = \perp$ if π_i is not checked by r and $\rho_i = \rho|_{\pi_i}$ otherwise.

Lemma 18 *Let ρ be a run of the ADC $\mathcal{A}$ and $p \in Pos(\rho)$, $k > b(\mathcal{A})$ and $\gg$ be a total ordering. If there are $k+1$ runs $\rho|_p = \rho_{k+1} \gg \ldots \gg \rho_1$ such that, Check$(\rho_1), \ldots$ Check(ρ_{k+1}) are independent then $\mathcal{M}(\rho, p, k - c(\mathcal{A}))$ is true.*

The proof of this lemma is quite simple: since the runs are independent, we can replace some run with another, without creating equalities.

Now, we have the extension of König's theorem from bipartite graphs to hypergraphs:

Theorem 19 (B. Reed, private communication) *Let S be a set and K, n be integers. Let $G \subseteq S^n$. If every subset $G_1 \subseteq G$ of independent elements has a cardinal $|G_1| \leq K$, then $|G| \leq K^n \times n!$.*

With this additional argument, we can give an upper bound to the cardinal of sets E_q^n:

$$|\bigcup_{q \in Q} E_q^n| \leq |Q| \times (b(\mathcal{A} + c(\mathcal{A}))^{c(\mathcal{A})} \times c(\mathcal{A})!$$

Which proves its termination with $n \leq O(|Q|^{3 \times c(\mathcal{A})} \times 2^{P_0(c(\mathcal{A}))})$ where P_0 is a polynomial.

Now, we have to estimate the cost of each inference step. Of course, we assume that identical subterms are shared. More precisely, each new run is a rule whose sons are already in the set. Hence checking that $r(\rho_1, \dots, \rho_n)$ is a run can be performed in time $c(\mathcal{A})$ since disequalities are checked in constant time. Finally, it remains to check the predicate $\mathcal{M}$. If $\gg$ satisfies the following additional property:

$$(\rho \gg \rho' \text{ and } \rho \in E_{q_0}^n) \text{ implies } \rho' \in \bigcup_{q \in Q} E_q^n$$

then only the runs which are already in E_q^n are possible candidates for $\rho_1, \dots, \rho_k$ in the definition of $\mathcal{M}$. There are at most $|E_q^n|$ possible runs to consider and $c(\mathcal{A})$ positions p. Finally, for each pumping, verifying whether or not it creates a close equality requires time $c(\mathcal{A})$. Hence minimality condition can be checked in time at most

$$|Q| \times (b(\mathcal{A} + c(\mathcal{A}))^{c(\mathcal{A})} \times c(\mathcal{A})! \times c(\mathcal{A})^2$$

Together with the bound on the number of steps, we get the desired property.

It still remains to exhibit an ordering $\gg$ which satisfies all our requirements:

Lemma 20 *There is an ordering $\gg$ which is monotonic, well-founded, total on $T(\Delta)$ and such that, if $t \gg u$, then the depth of t is larger or equal than the depth of u.*

sketch of the proof: consider the following interpretation of t: $I(t)$ is the triple $(d(t), M(t), t)$ where $d(t)$ is the depth of t, $M(t)$ is the multiset of strict subterms of t. Triples are ordered with the lexicographic composition of 1– the ordering on natural numbers 2– The multiset extension of $\gg$ 3– a lexicographic path ordering extending a total precedence (see [6] for complementary definitions on orderings). $\gg$ itself is defined as $u \gg t$ iff $I(u) > I(t)$. $\square$

As a consequence of theorem 15 and proposition 7, we get:

Theorem 21 *Ground reducibility of a term t w.r.t a rewrite system $\mathcal{R}$ can be decided in deterministic time $O(2^{P(\|t\|, \|\mathcal{R}\|)})$ where P is a polynomial.*

The coefficients of P may depend linearly from a (the maximal arity of a function symbol) and $\|\mathcal{F}\|$.

Let us emphasize that using an ordering which has the properties of lemma 20 instead of the usual pumping ordering was crucial in the proof; for instance the totality of the ordering allows much more pumpings that usually. Hence looking for minimal runs restricts the search space in a dramatic way. The construction of the propagation lemma relies on these properties.

6 Lower bound

Theorem 22 *Ground reducibility is EXPTIME-hard, for linear rewrite systems $\mathcal{R}$ and linear terms t.*

The proof is very much similar to H. Seidl's proof [15] that inclusion of tree languages is EXPTIME-hard. We encode computation trees of an alternating Turing machine $\mathcal{M}$ which works on a polynomially bounded space as a term in $\mathcal{T}(\mathcal{F})$. (This encoding is polynomial). Then, from an input w of the Turing machine, we build a term t and a rewrite system $\mathcal{R}$ such that there is a ground instance in $\mathcal{T}(\mathcal{F})$ of t which is irreducible iff $\mathcal{M}$ accepts w. $\mathcal{R}$ and t are built in polynomial time w.r.t. w and $|\mathcal{M}|$. The irreducible ground instances of t simply describes the accepting configurations: $t = \top(x)$. The rules in $\mathcal{R}$ essentially check that the initial configuration is w and discards trees which are not computation trees: this can be verified locally with a polynomial number of rules.

It is also possible to prove EXPTIME-hardness by reducing the emptiness problem for the intersection of n tree automata. The latter is EXPTIME-complete ([16, 7]).

7 Conclusion

We proved that ground reducibility is EXPTIME-complete for both the linear and the non-linear case. This closes a pending question. However, we do not claim that this result in itself gives any hint on how to implement a ground reducibility test. As we have seen, it is not tractable in general. A possible way to implement these techniques as efficiently as possible was suggested in [1]. On the average, some algorithms may behave well. In any case, we claim that tree automata help both in theory and in practice.

References

[1] A.-C. Caron, H. Comon, J.-L. Coquidé, M. Dauchet, and F. Jacquemard. Pumping, cleaning and symbolic constraints solving. In *Proc. Int. Conference on Algorithms, Languages and Programming*, Lecture Notes in Computer Science, vol. 820, Jerusalem, July 1994. Springer-Verlag.

[2] A.-C. Caron, J.-L. Coquidé, and M. Dauchet. Encompassment properties and automata with constraints. In C. Kirchner, editor, *5th International Conference on Rewriting Techniques and Applications*, volume 690 of *Lecture Notes in Computer Science*, Montreal, Canada, June 1993. Springer-Verlag.

[3] H. Comon. An effective method for handling initial algebras. In *Proc. 1st Workshop on Algebraic and Logic Programming, Gaussig, LNCS 343*. Springer-Verlag, Nov. 1988.

[4] H. Comon and F. Jacquemard. Ground reducibility and automata with disequality constraints. In P. Enjalbert, editor, *Proc. 11th Symp. on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, vol. 775, Caen, 1994. Springer-Verlag.

[5] Dauchet, Caron, and Coquidé. Automata for reduction properties solving. *JSCOMP: Journal of Symbolic Computation*, 20, 1995.

[6] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–309. North-Holland, 1990.

[7] T. Frühwirth, E. Shapiro, M. Vardi, and E. Yardeni. Logic programs as types for logic programs. In *Proc. 6th IEEE Symp. Logic in Computer Science, Amsterdam*, pages 300–309, 1991.

[8] J. Gallier and R. Book. Reductions in tree replacement systems. *Theorical Computer Science*, 37:123–150, 1985.

[9] F. Jacquemard. *Automates d'arbres et Réécriture de termes*. PhD thesis, Université Paris-Sud, 1996.

[10] J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82(1), July 1989.

[11] D. Kapur, P. Narendran, D. Rosenkrantz, and H. Zhang. Sufficient completeness, ground reducibility and their complexity. *Acta Inf.*, 28:311–350, 1991.

[12] D. Kapur, P. Narendran, and H. Zhang. On sufficient completeness and related properties of term rewriting systems. *Acta Inf.*, 24(4):395–415, 1987.

[13] E. Kounalis. Testing for the ground (co)-reducibility in term rewriting systems. *Theoretical Computer Science*, 106(1):87–117, 1992.

[14] D. Plaisted. Semantic confluence tests and completion methods. *Information and Control*, 65:182–215, 1985.

[15] H. Seidl. Deciding equivalence of finite tree automata. *Siam Journal of Computing*, 19(3):424–437, 1990.

[16] H. Seidl. Haskell overloading is DEXPTIME-complete. *Inf. Process. Lett.*, 52:57–60, 1994.