Reachability is Decidable for Ground AC Rewrite Systems

Richard Mayr
Institut für Informatik
Technische Universität München
Arcisstr. 21, D-80290 München
Germany
mayrri@informatik.tu-muenchen.de

Michaël Rusinowitch
LORIA and INRIA-Lorraine
615, rue du Jardin Botanique, BP 101
54602 Villers-Lès-Nancy CEDEX
France
rusi@loria.fr

Abstract

The reachability problem for ground associative-commutative (AC) rewrite systems is decidable. We show that ground AC rewrite systems are equivalent to *Process Rewrite Systems (PRS)* for which reachability is decidable [4]. However, the decidability proofs for PRS are cumbersome and thus we present a simpler and more readable proof in the framework of ground AC rewrite systems. Moreover, we show decidability of reachability of states with certain properties and decidability of the boundedness problem.

1 Introduction

Ground AC systems are obtained by combining ground rewrite systems (i.e. rewrite systems without substitution) with the associative and commutative laws for a symbol denoted by $+$. These laws are applied as *symmetric* rewrite rules, i.e. from right to left and vice-versa.

The reachability problem for ground AC systems has already been considered by A. Deruyver and R. Gilleron in [2] who proved that it is decidable under some restrictions: the $+$ symbols can occur only at the root of any side of a rule; if $+$ occurs at the root of the right-hand side then it has to occur at the root of the left-hand side too. This last condition forbids introducing new $+$ symbols in a term.

We show that ground AC rewrite systems are equivalent to *Process Rewrite Systems (PRS)* that were introduced by R. Mayr [4] as an extension of Petri nets with subroutines. This observation was surprising at first, since the two formalisms have different origins. Ground AC systems were defined as an extension of ground rewrite systems by associative and commutative operators like '$+$' or '$*$'. PRS were introduced as a process model that extends Petri nets by subroutines.

It has been shown in [4] that reachability for PRS is decidable. Here, by following a similar technique, we present a simpler and more readable proof of this result in the framework of ground AC rewrite systems. Previous known decidability results about ground AC systems (with several AC symbols) concern the word problem (symmetric reachability) [5] and the unifiability problem [6]. The latter was also obtained by splitting the rewrite systems in order to get *homogeneous* subsystems.

We also consider the problem if there is a reachable state that satisfies certain properties that are described by state formulae. We show that this "reachable property problem" is decidable for ground AC systems. Finally, we show that the problems of termination and boundedness are decidable, too.

2 Ground AC Systems vs. PRS

Ground AC rewrite systems are defined as ground rewrite systems (i.e. rewrite systems without substitution) that operate on the set of terms $T(F)$ over a signature F which contains an associative and commutative operator $+$. Note that F can contain function symbols with any arity, like $f(x_1, \ldots, x_k)$. For a detailed survey on term rewriting the reader may consult [1].

Process Rewrite Systems (PRS) [4] are defined as **prefix**-rewrite systems that operate on process terms. These process terms are built from atomic processes by the operators '.' for sequential- and '$\|$' for parallel composition. The operator for sequential composition is associative and the operator for parallel composition is associative and commutative.

Now we show that the two formalisms are equivalent. For both PRS and ground AC systems it is possible to construct equivalent systems where each rule contains only one operator. (These new systems can simulate the behavior of the original systems modulo a uniformly bounded number of silent actions.) We show the equivalence only for systems where rules contain only one operator. Monadic ground AC systems are ground AC systems where all function symbols have arity one. In a first step we show that PRS and monadic ground AC systems are equivalent.

From PRS to monadic ground AC: For each symbol X in the PRS we introduce a 1-ary function symbol $X(\ldots)$. We use a special symbol ϵ for the empty term. For every PRS-rule there is a corresponding ground AC rule:

PRS	**Ground AC**
$X.Y \rightarrow Z$	$Y(X(\epsilon)) \rightarrow Z(\epsilon)$
$X \rightarrow Y.Z$	$X(\epsilon) \rightarrow Z(Y(\epsilon))$
$X \rightarrow Y$	$X(\epsilon) \rightarrow Y(\epsilon)$
$X \rightarrow Y \Vert Z$	$X(\epsilon) \rightarrow Y(\epsilon) + Z(\epsilon)$
$X \Vert Y \rightarrow Z$	$X(\epsilon) + Y(\epsilon) \rightarrow Z(\epsilon)$

From ground AC to PRS: For every function or constant symbol f in the ground AC system we use a symbol f in the PRS. For every ground AC-rule there is a corresponding PRS rule:

Ground AC	**PRS**
$f(c) \rightarrow d$	$c.f \rightarrow d$
$c \rightarrow f(d)$	$c \rightarrow d.f$
$c \rightarrow d$	$c \rightarrow d$
$c \rightarrow d + e$	$c \rightarrow d \Vert e$
$c + d \rightarrow e$	$c \Vert d \rightarrow e$

This shows that PRS and monadic ground AC rewrite systems are equivalent. In a second step we show that monadic ground AC and general ground AC are equivalent. One direction is trivial. Now we show how to encode a general ground AC system in a monadic ground AC system. Let f be a k-ary function symbol. Then we introduce k new 1-ary function symbols $f_1, \ldots, f_k$. In all rewrite rules we replace every term of the form $f(c_1, \ldots, c_k)$ by

$$f(f_1(c_1) + \ldots + f_k(c_k))$$

This new system is monadic, by contains more than one operator in a rule. This can be transformed into an equivalent one (modulo silent actions) with only one operator in each rule.

Thus we have that Ground AC = monadic ground AC = PRS. In PRS the rewrite rules are labeled with atomic actions, while this is not customarily done in ground AC systems. However, it can done in ground AC systems just as well.

3 Reachability for Ground AC Systems

We consider finite sets of ground rewrite rules Δ. For terms t_1, t_2 we define that $t_1 \succ^{\Delta} t_2$ iff t_1 can be transformed into t_2 by applying the rules in Δ. The reachability problem is to decide if $t_1 \succ^{\Delta} t_2$ for given t_1, t_2 and Δ.

3.1 Splitting the Rules

Let Δ be a finite set of rules on F. A term or a rule is *homogeneous* if it contains at most one occurrence of a non-constant operator.

Lemma 1 *For every finite ground AC system Δ a finite set of homogeneous rules Δ' can be effectively constructed such that for all terms $t_1, t_2 \in T(F)$, $t_1 \succ^{\Delta} t_2$ iff $t_1 \succ^{\Delta'} t_2$*

Proof We derive Δ' by applying to Δ the following abstraction rules as much as possible:

$$\text{Abstract 1:} \quad a \to b \vdash a_{|p} \to g, a[g]_p \to b$$

where $a_{|p}$ is a maximal homogeneous non-constant strict subterm of the term a at a position p and g is a new constant.

$$\text{Abstract 2:} \quad a \to b \vdash a \to b[g]_p, g \to b_{|p}$$

where $b_{|p}$ is a maximal homogeneous non-constant strict subterm of b at a position p and g is a new constant.

$$\text{Abstract 3:} \quad a \to b \vdash a \to g,\ g \to b$$

where a, b are homogeneous, non-constants, with different root symbols and g is a new constant.

The application of these transformations terminates since: Abstract 3 decreases the number of rules with different non-constant operators at the root of each side; every application of Abstract 1 or Abstract 2 decrease the multiset of depths of the rule sides and do not create rules that can be transformed by Abstract 3. Every rewrite step by Δ can be simulated by rewrite steps with Δ'. The other direction follows from the fact that the symbols g are new constants. ■

3.2 Deciding Reachability by Completion

In this section rules of the form $c \to d$ where c and d are constants will be called *simple rules*. The following completion algorithm adds simple rules $c \to d$ to a set of rules R, whenever $c \succ^{R} d$.

Lemma 2 *Let R_1 and R_2 be two homogeneous rewrite systems s.t. the sets of operators used in the rules of R_1 and R_2 are disjoint and R_1 and R_2 contain the same simple rules. Let $R = R_1 \cup R_2$. If there is a pair of constants (u, v) s.t. $u \succ^R v$ and $(u \to v) \notin R$ then there is a pair of constants (u', v') s.t. $(u' \to v') \notin R$, but $u' \succ^{R_1} v'$ or $u' \succ^{R_2} v'$.*

Proof Choose a pair of constants u', v' s.t. $(u' \to v') \notin R$, but $u' \xrightarrow{\sigma} v'$ where σ is a sequence of rule applications of minimal length. More precisely the length of σ is minimal over the choice of u', v' and σ. By our preconditions this must exist.

We will assume that σ must contain non-simple rules from both R_1 and R_2 and derive a contradiction. W.r. we assume that the first non-simple rule in σ is from R_1. By our assumption there must be a non-simple rule from R_2 in σ. The first such rule must have the form $u'' \to t$, where u'' is a constant and t is not a constant, because the sets of operators in R_1 and R_2 are disjoint. The operator in t must disappear later, because the sequence σ ends with the constant v. This can only be done if the term t is rewritten to a constant later in the sequence σ. Thus there must a constant v'' and a subsequence σ' of σ s.t. $u'' \xrightarrow{\sigma'} v''$. This is a contradiction to the minimality of the length of σ.

Thus we have that the non-simple rules in σ are either all from R_1 or all from R_2. Since R_1 and R_2 contain the same simple rules we can assume that σ contains only rules from R_1 or only rules from R_2. Thus we have $u' \succ^{R_1} v'$ or $u' \succ^{R_2} v'$. ∎

Lemma 3 *Let R_1 and R_2 be two homogeneous rewrite systems s.t.*

1. *The sets of operators used in the rules of R_1 and R_2 are disjoint.*
2. *The relations $\succ^{R_1}$ and $\succ^{R_2}$ are decidable.*
3. *If one adds rules of the form $u \to v$ to R_1 or R_2, where u and v are constants, then the new relations $\succ^{R_1'}$ and $\succ^{R_2'}$ are also decidable.*

Let $R = R_1 \cup R_2$. Given two constants x, y, it is decidable whether $x \succ^R y$.

Proof To prove the lemma it is sufficient to give an algorithm for generating all couples of constants u, v such that $u \succ^R v$, and check if (x, y) is generated.

First we add all rules in R_1 of the form $c \to d$, where c and d are constants to R_2 and vice versa. By the third precondition this does not change the decidability. By precondition 1 we can apply Lemma 2. Thus if there is a pair of constants (u, v) which satisfies $u \succ^R v$, but is not in R_1 or R_2, then there is another pair (u', v') that is also not in R_1 or R_2, but satisfies $u' \succ^{R_1} v'$ or $u' \succ^{R_2} v'$. By precondition 2 this pair can be found. Then we add this pair $u' \to v'$ to R_1 and R_2 and get R'_1 and R'_2. By precondition 3 this does not change the decidability. We repeat this procedure with the new R'_1 and R'_2 until no new pair can be added. It terminates, because there are only finitely many different constants in R. By Lemma 2 we have then added all pairs (u, v) s.t. $u \succ^R v$. Let $\tilde{R}_1$ and $\tilde{R}_2$ be the final results of this process. It then holds that $x \succ^R y \iff (x \to y) \in \tilde{R}_1 \cup \tilde{R}_2$. ■

Theorem 1 *The reachability problem is decidable for ground rewrite systems with an associative-commutative operator.*

Proof Consider an instance of the reachability problem for a set of rules Δ and terms t_0, t. The question is if $t_0 \succ^\Delta t$. We introduce two new constants x, y and define $\Delta' := \Delta \cup \{x \to t_0, t \to y\}$. Thus we have $t_0 \succ^\Delta t \iff x \succ^{\Delta'} y$. By Lemma 1 a system of homogeneous rules Δ'' can be constructed s.t. $x \succ^{\Delta'} y \iff x \succ^{\Delta''} y$. Δ'' can be partitioned into R_1 and R_2 s.t. R_1 are all the rules that contain the commutative operator $+$ and R_2 is the rest. Thus the first precondition of Lemma 3 is satisfied. The relation $\succ^{R_1}$ corresponds to Petri net reachability, which is decidable [3]. The relation $\succ^{R_2}$ is the reachability relation for a normal ground rewrite system and is also decidable [2]. Adding simple rules to R_1 and R_2 does not change this. Thus the other preconditions of Lemma 3 are satisfied. Thus it can be used to decide $x \succ^{\Delta''} y$, which is equivalent to $t_0 \succ^\Delta t$. ■

Remark: Note that the algorithm of Lemma 3 uses only polynomially many instances of Petri net reachability, each of which is smaller than the input. The same techniques can be applied to show the decidability of the reachability problem for ground rewrite systems with arbitrarily many different associative and commutative operators.

4 The Reachable Property Problem

Here we use the notations from [4], Section 6. The rewrite rules in Δ are assigned labels, which can be interpreted as atomic actions. These labels then form atomic propositions in state-formulae. For example let a_i be a label. The term t satisfies the state-formula a_i (denoted $t \models a_i$) if a rule with label a_i can be applied to t. A general state-formula is a boolean combination of atomic propositions. We give a method that decides the problem $t_0 \models \Diamond\Phi$, i.e. the problem if there is a reachable state that satisfies the state formula Φ. It suffices to consider the case where Φ is $a_1 \wedge \ldots \wedge a_k \wedge \neg b_1 \wedge \ldots \wedge \neg b_l$. We will abbreviate $\neg b_1 \wedge \ldots \wedge \neg b_l$ by B, $a_1 \wedge \ldots \wedge a_k$ by A and $\bigwedge_{i \in I} a_i$ by A_I. ($A_\emptyset = true$). We also define $K = \{1, \ldots, k\}$.

We solve the problem for systems Δ in transitive normal form. To simplify the problem we assume that rules of the form $f(c_1, \ldots, c_k) \to d$ do not carry any label. This is no restriction since every system can be transformed to satisfy this condition in the following way: For every term $f(c_1, \ldots, c_k)$ that occurs on the left side of a rule determine the set A of actions that are enabled by $f(c_1, \ldots, c_k)$. Then add a new constant and e and the rule $f(c_1, \ldots, c_k) \to e$. Then for every action $a \in A$ add a rule $e \xrightarrow{a} e$. Then transform the system into transitive normal form. This new system is equivalent to the old one as far as the reachable property problem is concerned.

For simplicity we also assume that there are only two non-constant symbols: $+$ and . where $+$ is AC. (The generalization to arbitrary symbols is straightforward.) We can assume w.r. that the initial state t_0 is a constant t_0, since otherwise we just add another rule $c_0 \to t_0$. Δ_{par} (resp. Δ_{seq}) contains all rules without '.' (resp. '+'). The rules in Δ_{par} are called *par-rules* and the rules in Δ_{seq} are called *seq-rules.* Let $C = \{c_1, \ldots, c_m\}$ be the set of constants in the system. A *monomial* is a sum of constants. We denote by f_Ψ the formula of L_N stating that a monomial satisfies a formula Ψ. For instance f_B is true for monomials that are not reducible by rules with a label in $\{b_1, \ldots, b_l\}$. Given a subset C' of C, we denote by $h_{C'}$ the formula of L_N stating that the set of constants in a monomial is C'. For instance $h_{\{c_1, c_1, c_3\}}(c_1 + c_1 + c_3 + c_3)$ evaluates to true. Terms will always be flattened using associativity of $+$. Hence $(a + b) + c = a + b + c$ and the depth of this term is 1.

The following result is a direct consequence of a result for Petri nets due to P. Jančar [8]:

Lemma 4 *It is decidable whether $c_0 \models \Diamond\Phi$, if Δ only contains par-rules.*

If σ is one of the shortest sequences such that $t_0 \xrightarrow{\sigma} t$ and $t \models \Phi$ there is no subderivation of $t_0 \succ^\Delta t$ such that $c \succ^\Delta t' \succ^\Delta c'$. Otherwise by replacing this subderivation with $c \to c'$ (since Δ is in transitive normal form) we can obtain a shorter string σ' such that $t_0 \xrightarrow{\sigma'} t$. Hence we can assume that by commuting rules applications we can build another string η with the same properties as σ which additionally has a special structure: $\eta = \eta_0\eta_1 \ldots \eta_d$ where all rules applications corresponding to η_i are applied at the same level i of terms.

4.1 A special case

In this section we solve the problem for the special case of $k = 0$. We compute the subset C_N of all $c \in C$ such that $c \models \Diamond B$. If no rule in B applies to c then obviously $c \in C_N$. Otherwise let μ be a shortest derivation from c to a t such that $t \models B$.

Using the same decomposition as above $\mu_0\mu_1$ is the maximal prefix of μ such that $c \overset{\mu_0\mu_1}{\rightarrow} v$ and all intermediate terms between c and v (including v) are of level ≤ 1.

i. Let us assume first that these terms are (non trivial) sums of constants. Then $v = U + U'$ where U, U' are possibly empty sums and:

1. $U \models B$
2. for all $c' \in U'$ we have $c' \succ^{\Delta} t_{c'}$ and $t_{c'} \models B$

By minimality of the derivation we can assume that $c \notin U'$.

ii. Now if the root symbol of v is '.' then either v is not reducible by a rule from B or one of its leaves is in C_N and is different from c by minimality. (recall that all rules have depth 1).

More generally by minimality of μ it will never contain a subderivation $c' \succ^{\Delta} s[c'] \succ^{\Delta} s[t']$. Otherwise pumping allows to find a shorter one. This shows also that the depth of t is bounded by $m = |C|$.

Hence a simple recursive algorithm solves the problem. Let $Nreach(c, n)$ be true iff c rewrites to a term of depth n not reducible by rules from B. $Nreach(c, 0)$ is easy to check, by Lemma 4, since Δ is in transitive normal form. We also introduce the auxiliary procedure $Ncheck(c, C')$. Given $c \in C$ and $C' \subseteq C$, $Ncheck$ tests whether there exists a monomial v such that $c \succ^{\Delta_{par}} v$, $v = U + U'$ and $f_B(U) \wedge h_{C'}(U')$ is true. Note that this test is effective thanks to the result of Jančar [8].

$Nreach(c, n)$
  for $C' \subseteq C$ do
    if $Ncheck(c, C')$ then
      if $\bigwedge_{c' \in C'} Nreach(c', n-1)$ then true; exit
  for $a, b \in C$ do
    if $c \rightarrow a.b \ \wedge (Nreach(a, n-1) \wedge Nreach(b, n-1))$ then true; exit
  $Nreach(c, n) =$ false

By computing $\{c \in C; Nreach(c, m) = \mathsf{true}\}$ we get the set C_N.

4.2 The general case

Now we study the general case. Let us analyze the structure of the term t_1 such that $c_0 \stackrel{\eta_0\eta_1}{\rightarrow} t_1$. Assume that the root symbol of t_1 is $+$ (the other case is simple). Hence the derivation contains only rules applications from Δ_{par}. We shall sort the constants in t_1 according to the property $\Diamond(A_P \wedge B)$ they satisfy, P ranging over the subsets of $K = \{1, \ldots, k\}$. Then $t_1 = U + \Sigma_{P \in \mathcal{P}(K)} U'_P$ where U, U'_P are possibly empty sums of constants and:

1. $U \models A_I \wedge B$
2. for all $c \in U'_P$ we have $c \succ^{\Delta} t_c$ and $t_c \models A_P \wedge B$
3. $I \cup \{P \in \mathcal{P}(K); U'_P \text{ is not empty}\} = K$

The pumping argument is now slightly more complex than in the base case: a subderivation starting from a constant c'': $c'' \succ^{\Delta} s[c''] \succ^{\Delta} s[t'']$ (where $s[t'']$ is not reduced anymore in μ) can be shortened if t'' and $s[t'']$ satisfy the same A_I's. Hence the length of any branch of t is bounded by $(k+1)m$.

Let $Reach(c, n, J)$ be true iff c rewrites to a term of depth n satisfying $A_J \wedge B$. We have that $Reach(c, 0, J)$ is decidable by Lemma 4, since Δ is in transitive normal form. Note that $Reach(c, n, \emptyset) = Nreach(c, n)$.

For the algorithm that decides $Reach(c, n, J)$ we need an auxiliary function *Check*. Let $c \in C$ be a constant, $j : 2^K \longmapsto 2^C$ a mapping and $I \subseteq K$. Then $Check(c, j, I)$ is true iff there exists a v such that: $c \succ^{\Delta_{par}} v$ and

$$(v = U + \Sigma_{P \in 2^K} U'_P) \wedge f_{I \wedge B}(U) \wedge (\bigwedge_{P \in 2^K} h_{j(P)}(U'_P))$$

Check is effective by the result of Jančar [8]. The algorithm for *Reach* is as follows:

$Reach(c, n, K)$
    for every mapping $j : 2^K \longmapsto 2^C$ and every $I \subseteq K$
        if $I \cup \{P \in \mathcal{P}(K); j(P) \text{ is not empty}\} = K$ **then**
            if $Check(c, j, I)$ **then**
                if $\bigwedge_{P \in \mathcal{P}(K)} (\bigwedge_{c' \in j(P)} Reach(c', n-1, P))$ **then true; exit**
    for $a, b \in C$ **do**
        if $c \rightarrow a.b$ **then**

for everyK_1, K_2 s.t. $K_1 \cup K_2 = K$
    if $(Reach(a, n-1, K_1) \wedge Reach(b, n-1, K_2))$ then true; exit
$Reach(c, n, K) =$ false

Now the *Reachable property* problem can be tested by computing

$$Reach(c_0, (k+1)m, K)$$

5 Termination Problems

In this section we investigate termination properties of ground AC systems. We consider the case where the systems are in transitive normal form.

Lemma 5 *It is decidable whether there exists an infinite derivation from a constant c.*

Proof For every term t we define $depth(t)$ as the maximal nesting-depth of function symbols in t. For example $depth(a + f(b + c + g(a, b))) = 2$.
Assume that there is an infinite run that starts at c s.t. in this run a constant c' is reduced to a larger non-constant term t and t is later again contracted to another constant c''. Since we consider systems in transitive normal form we have the rule $c' \to c''$ and can do this in one step. Thus if there is any infinite run then there is also an infinite run in which the depth of the terms never decreases. In other words, no seq-rules of the form $f(c_1, \ldots, c_k) \to d$ are used, but only seq-rules of the form $c \to f(d_1, \ldots, d_k)$.
Hence only the following situations can occur:

1. In the infinite sequence the depth of the terms is bounded. Then there must be a constant c' that occurs in some term in this sequence s.t. there is an infinite derivation from c' using only $\succ^{\Delta_{par}}$.

2. If the depth of the terms is not bounded then after some steps by the pigeon hole principle there is a long internal path in some term such that two identical symbols on it have been generated by the same rule $c' \to t$ where c' is a constant.

Hence there is an infinite derivation from c iff c rewrites to a term containing a constant c' such that

1. either there is an infinite derivation from c' using par-rules only.

2. c' rewrites to a term containing c'.

Case 1. amounts to check the existence of an infinite path for a Petri net, which can be done by constructing the coverability tree.
Case 2. can be decided using the results of the previous section. For any constant c' consider the system $\Delta_d = \Delta \cup \{c' \xrightarrow{\tau} c'\}$. Let $t_1, \ldots, t_k$ be the terms that can be reached from c' in one step. Checking 2. amounts to checking whether $t_k \models \Diamond\tau$ for some t_k. Once we have collected the set of constants c' for which the test is positive, we can test whether c can reach a term containing one of them using the same technique.

Thus the existence of an infinite run from some constant c is decidable. To decide this for a general term t we use a new constant c, add a rule $c \rightarrow t$ to the system, transform this new system into transitive normal form and check the existence of an infinite run from c.

A system terminates if there is no infinite run. We get the following theorem.

Theorem 2 *The termination property is decidable for ground AC systems.*

In a similar way we can decide the boundedness problem. There is a run from a constant c where the sizes of the terms are unbounded if from c one can reach a term containing c' s.t.

1. either there is an unbounded derivation from c' using par-rules only. This can be decided by the coverability graph.

2. c' rewrites to a term t containing c', but $t \neq c'$. This can be checked in the following way. Every rule whose right side contains c', but is not equal to c' is labeled with the action τ. Then it suffices to check whether $c' \models \Diamond\tau$.

Theorem 3 *Boundedness is decidable for ground AC systems.*

References

[1] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In Van Leeuwen, editor, *Handbook of Theoretical Computer Science*. North Holland, 1990.

[2] A. Deruyver and R. Gilleron. The Reachability Problem for Ground TRS and some Extensions, In Proceedings of the International Joint Conference on Theory and Practice of Software Development : Vol. 1. Volume 351 of LNCS, Springer Verlag, pp 227–243, 1989.

[3] E. W. Mayr. An algorithm for the general Petri net reachability problem. SIAM Journal of Computing, 13:441-460,1984.

[4] R. Mayr. Process Rewrite Systems. Electronic Notes in Theoretical Computer Science, volume 7. Proceedings of the Workshop "Expressiveness in Concurrency" (EXPRESS'97), 1997.

[5] P. Narendran and M. Rusinowitch. Any ground associative-commutative theory has a finite canonical system. Proceedings of RTA, LNCS 488, Springer-Verlag, 1991.

[6] P. Narendran and M. Rusinowitch. The unifiability problem in ground AC Theories. IEEE Symposium on Logic in Computer Science. Montreal (Canada), 1993.

[7] M. Oyamaguchi The Church-Rosser Property for Ground Term Rewriting Systems is Decidable", 1987, Theoretical Computer Science, Volume 49(1), pp 43–79.

[8] P. Jančar. Decidability of a temporal logic problem for Petri nets. *Theoretical Comuter Science*, 74:71–93, 1990.