Linear bounded automata and rewrite systems : Influence of initial configurations on decision properties[1]

A-C Caron
Université de Lille-Flandres-Artois
LIFL (URA 369-CNRS) UFR IEEA, Bâtiment M3
59655 Villeneuve d'Ascq Cedex France
mail: caronc@lifl.lifl.fr

Abstract

We prove that termination is undecidable for non-length-increasing string rewriting systems, using linear-bounded automata. On the other hand, we prove the undecidability of confluence for terminating rewriting systems when terms begin by a fixed symbol. These two results illustrate that sometimes restriction of problem to recognizable domains modify decidability properties, sometimes it does not. (We only consider finite terms).

Introduction

With two problems, we prove the influence of initial configurations on rewriting decision properties. The first problem concerns termination, and the second, confluence.

Termination problems are fundamental in rewriting because they correspond to program termination for all data [Dershowitz & Jouannaud]. Many termination criteria have been studied [Dershowitz] but, generally, termination is undecidable, even for one left-linear rule [Dauchet] or for a semi-Thue system [Huet & Lankford]. Termination problems for one linear-rule or one rule on words remain open. But in this last case, if the rule is non-length-increasing, termination is clearly decidable.

Here, we prove undecidability of termination of non-length-increasing string rewriting systems (i.e. non-length-increasing semi-Thue systems). This problem is similar to linear-bounded automata termination [Book] and has been stated in the case of graphs by Litovsky and Metivier [Litovsky & Metivier]. Therefore we revisit a paper of Hooper [Hooper], in which he studied termination of Turing machines and proved that termination is undecidable for linear-bounded automata, and more generally, for Turing machines. Using technics suggested by Hooper, we prove directly undecidability of termination for non-length-increasing string rewriting systems.

In a first part, we construct a class of linear bounded automata whose termination is reduced to the Post correspondance problem. This result is well-known but we use our construction in the second part, and observe that undecidability subsists if we suppress the constraint of beginning from an initial configuration. In the third part, we want to bring a fact out: the link between decidability and recognizable restrictions on terms. Recognizable restriction means that terms belong to a recognizable language. Therefore, in opposition to this

1. This work was supported by PRC "Mathématiques et informatique" and ESPRIT2 Working Group ASMICS

first result, we prove that confluence for terminating rewriting systems becomes undecidable if we restrict terms configuration to some recognizable set. (It is well-known that confluence is decidable for noetherian rewriting systems [Newman]). Confluence on recognizable tree languages is interesting because these languages are sorts (the finite automaton being the signature). Note that Otto proves that confluence on some congruence class is undecidable [Otto] but congruence classes are generally not recognizable.

I - Termination of linear-bounded automata

Linear-bounded automata have been created by Myhill [Myhill] and very studied since [Kuroda]. In particular, Hooper studied the undecidability of termination of Turing machine and linear-bounded automata [Hooper]. He called this problem immortality problem. Moreover, Hopcroft and Ullman showed that to every linear-bounded automaton, we can associate an equivalent terminating linear-bounded automaton [Hopcroft & Ullman].
In this part, we prove directly the undecidability of termination for a class of linear-bounded automata which restore their initial configuration when they do not stop, using a suitable construction for the more general result of the second part.

Definition 1-1 : A machine *terminates* if and only if it stops for all data.

Definition 1-2 : A *linear-bounded automaton* (LBA) can be seen as a particular Turing machine. Its tape is an input/output tape whose length is linearly dependent of data length. A LBA is a sextuple $(\Sigma,\Gamma,Q,Q_0,Q_f,\Delta)$, where Σ is the data alphabet, Γ the work alphabet, Q is the states set, Q_0 the initial states set, Q_f the final states set, Δ is the next-move function. We suppose that the tape has the form #<d>#, where #,<,> are never modified and d is the data.

Vocabulary : We use Turing machines notions : *instantaneous description*, *initial configuration*, *computation step*, *computation*. More precisely :
- an instantaneous description (denoted ID) is a writing $\#<m_1qam_2>\#$. It means that the head is reading the letter a, the word m_1 is on the left and m_2 is on the right, and q is the machine state.
- an initial configuration is an instantaneous description $\#<qm>\#$ where q is an initial state.
- a step computation $ID_1 \rightarrow ID_2$ means we can go from ID_1 to ID_2 with a transition of Δ.
- a computation is a succession of computation steps from an initial configuration ID_1 to a final configuration ID_f.

We are not interested in the result but in the computation stop. Therefore, final configuration is not important. But we need two notions : *beginning of computation* and *sub-computation* :
- a beginning of computation is a succession of computation steps, from an initial configuration ID_1
- $ID_i \rightarrow \dots \rightarrow ID_n$ is a sub-computation if there exists $ID_1 \rightarrow \dots \rightarrow ID_i \rightarrow \dots \rightarrow ID_n$ a beginning of computation which contains it.

Proposition 1-1 : [Hooper] *Termination of linear-bounded automata is undecidable.*

Post correspondance problem : [Post]

The Post correspondance problem $P(\varphi,\psi)$ over an alphabet X is given by two morphisms φ and ψ from I^* to X^*. $P(\varphi,\psi)$ is solvable if and only if there exists m_I in I^+ such that $\varphi(m_I) = \psi(m_I)$. The Post correspondance problem is well-known undecidable.

We are working on a specific class of deterministic LBA, denoted A_{Post}, associated to the Post problem. A machine $A\varphi\psi$ of this class is associated to two morphisms φ and ψ . If the tape of $A\varphi\psi$ contains two words m_I and m_X, such that $m_X = \varphi(m_I) = \psi(m_I)$ then the machine loops passing by its initial configuration again.

Definition 1-3 : A_{Post} is a set of LBA associated to the Post problem.

$A_{Post} = \{ A\varphi\psi = (\Sigma,\Gamma,Q\varphi\psi,\{q_0\},\emptyset,\Delta) \mid \varphi$ and ψ two morphisms from I^* to $X^*\}$, with

. $\Sigma = I \cup X \cup \{<,>\}$, I and X two disjoint finite alphabets.

. $\Gamma = \Sigma \cup \bar{I} \cup \bar{X}$, $\bar{X}$ and $\bar{I}$ constructed from X and I: for all x in X, $\bar{x}$ is in $\bar{X}$ and for all i in I, $\bar{i}$ is in $\bar{I}$.

. $Q\varphi\psi$ is the set of the states of the automaton $A\varphi\psi$.

Appendix I contains in details the behaviour fonction Δ of $A\varphi\psi$. The following little program and the two examples explain the behaviour of a machine $A\varphi\psi$

```
Program Aφψ ;
  * The data is a word with letter of I and X, The head is on the first letter *
  stop := false ;
  while not stop do
    * search for the letter of I the most on the right *
    * I(Head) means that the head reads a letter of I *
    while I(Head) do move := right ; od
    * we verify that m = in...i2i1x1x2...xm with φ(i1i2...in) = x1x2...xm *
    while not (alloverlined(tape) or stop) do
      i := Head ; overline(Head) ;
      * we search for the first non overlined letter of X, on the right *
      while Ioverlined(Head) do move := right ; od
      while Xoverlined(Head) do move := right ; od
      * now the tape has the form in...ij īj-1...ī1 x̄1...x̄k-1 xk...xm *
      * we verify that xk...xk+p = φ(ij) and if it is true, we overline xk...xk+p *
      researchφ(i) ;
      * stop = True if φ(i) is not on the right hand-side *
      * we now search for the first non overlined letter of I, on the left *
      while Xoverlined(Head) do move := left ; od
      while Ioverlined(Head) do move := left ; od
    od
    * we do exactly the same thing with ψ *
    while not (alloverlined(tape) or stop) do
      ...
    od
  od
end
```

Example 1:
$I = \{a,b\}$; $X = \{1,2,3\}$;
$\varphi(a) = 123$; $\varphi(b) = 32$; $\psi(a) = 23$; $\psi(b) = 321$.
- The tape contains the data ab32123. So the initial configuration is $\#<q_0ab32123>\#$
- The machine searches for the last non overlined letter of I. It is b.
The configuration is now $\#<aq_2b32123>\#$.
- The machine overlines b. The configuration is $\#<a\bar{b}q_{\varphi(b)1}32123>\#$
- The machine verifies that 3 is the first letter of $\varphi(b)$. Since it is true, it overlines the letter 1.
The configuration is $\#<a\overline{b3}q_{\varphi(b)2}2123>\#$.
- It verifies that 2 is the second letter of $\varphi(b)$ and overlines it. Since $\varphi(b)$ contains only two letters, the machine searches for the non overlined letter of I the most on the right. It is a.
The configuration is now $\#<q_{ret}a\overline{b32}123>\#$.
- The machine overlines a. The configuration is $\#<\bar{a}q_{\varphi(a)1}\overline{b32}123>\#$.
- It searches for the first non overlined letter of X. It is 1.
The configuration is $\#<\overline{ab32}q_{\varphi(a)1}123>\#$.
- It verifies that 1 is the first letter of $\varphi(a)$. Since it is true, it overlines it.
The configuration is $\#<\overline{ab321}q_{\varphi(a)2}23>\#$.
- In the same way, the automaton overlines 2 and 3, the second and the third letters of $\varphi(a)$.
The configuration is $\#q_{ret}<\overline{ab32123}>\#$
- There is no more non overlined letter of I. So the automaton replaces the overlined letters by the same non overlined letters, verifying that all the data is overlined.
The configuration is $\#<ab32123q_{reset}>\#$.
- The head goes to the letter of I the most on the right. It is b.
The configuration is $\#<aq_{again}b32123>\#$
- As before, the machine overlines b and searches for the image of b by ψ. It is 321.
The configuration is $\#<aq'_{ret}\ \overline{b321}23>\#$
- The machine overlines a and searches for the image of a by ψ. It is 23.
The configuration is now $\#\ q'_{ret}\ <\overline{ab32123}>\#$
- There is no more letter of I non overlined. So the automaton replaces the overlined letters by the same non overlined letters, verifying that all the tape is overlined.
The configuration is $\#<ab32123q'_{reset}>\#$
- Now, the machine has verified that the data had the form $\#<\tilde{m}m'>\#$ with $\varphi(m) = \psi(m) = m'$.
$\tilde{m}$ represents the mirror of the word m.
- The initial configuration is restored when the Post correspondance problem $P(\varphi,\psi)$ is satisfied.
The configuration is $\#<q_0ab32123>\#$.

Example 2:
The tape contains the word a132. The initial configuration is $\#<q_0a132>\#$
- The machine overlines a. The configuration is $\#<\bar{a}q_{\varphi(a)1}132>\#$
- It verifies that 1 is the first letter of $\varphi(a)$ and overlines it.
The configuration is now $\#<\overline{a1}q_{\varphi(a)2}32>\#$
- It verifies that 3 is the second letter of $\varphi(a)$. Since it is false, the machine stops.

Definition 1-4 : An initial configuration is *proper* if and only if it has the form $\#<q_0\tilde{m}_Im_X>\#$ (the mirror image of m_I is represented by $\tilde{m}_I$). with m_X word in X^*, m_I word in I^+, q_0 initial state of the machine.

Lemma 1-1 :
For all machines of the class A_{Post}, if the initial configuration is not proper then the machine stops.

Proof : Using the definition of the machine. ❑

Lemma 1-2 :
For all machines in A_{Post} starting from a proper initial configuration $\#<q_0\tilde{m}_I m_X>\#$, the machine loops passing by its initial configuration again if and only if $m_X = \varphi(m_I) = \psi(m_I)$.

Proof: From the definition of the machine. ❑

We get as corollary of these lemmas the next proposition.
Proposition 1-2:
(1) Termination in undecidable for the class A_{Post}
(2) If an automaton $A\varphi\psi$ loops for a data d then it passes by its initial configuration again.

Proof :
(1) According to lemmas 1-1 and 1-2, $A\varphi\psi$ loops if and only if $m_X = \varphi(m_I) = \psi(m_I)$. But it is not decidable wether $\varphi(m_I) = \psi(m_I)$ (Post correspondance problem). Therefore, termination of the class A_{Post} is undecidable.
(2) From lemma 1-2, if the machine loops then it passes by its initial configuration again. ❑

Lemma 1-3 :
If there exists a computation which loops from some configuration (not necessarily reachable) then there exists a beginning of computation starting from a proper initial configuration which loops.

Proof : See appendix II.

II - Termination of non-length-increasing string rewriting systems

We want to prove that termination of non-length-increasing string rewriting systems is undecidable.

Definition 2-1 : A *non-length-increasing string rewriting system* is a system where rules have the form $l \rightarrow r$ with $|l| \geq |r|$. l and r are words.

A particular class of non-length-increasing string rewriting systems is the class of length-preserving string rewriting systems.
Definition 2-2 : A *length-preserving string rewriting system* is a system where rules have the form $l \rightarrow r$ with $|l| = |r|$. l and r are words.

We construct a class R_{Post} of rewriting systems associated to the class A_{Post} of machines studied before: I and X are the two alphabets considered in paragraph I. $\Sigma = \{\#,<,>\} \cup I \cup X$. We construct I' and I" from I, X' and X" from X, and Σ', Σ'' from Σ : $\forall a \in \Sigma, a' \in \Sigma'$ et $a'' \in \Sigma''$. Q is a finite alphabet, disjoint from Σ.
From a machine $A\varphi\psi$ of the class A_{Post}, we construct the rewriting system $R\varphi\psi$.
For all transitions $(q_1,a_1) \rightarrow_A (a_2,q_2,Ri)$, for all a,b in Σ, we construct the rule:

$$a'q_1a_1''b'' \rightarrow_R a'a_2'q_2b''$$

and for all transitions $(q_1,a_1) \rightarrow_A (a_2,q_2,Le)$, for all a,b in Σ, we construct the rule:

$$a'b'q_1a_1'' \rightarrow_R a'q_2b''a_2''.$$

(intuitively, x' is a letter on the left side of the tape head, and x" a letter on the right side.)

R_{Post} is a class of length-preserving string rewriting systems.

Notation : to simplify, we write A for $A\varphi\psi$ and R for the associated system $R\varphi\psi$.

$\overset{*}{\rightarrow}_R$ is the transitive reflexive closure of $\rightarrow_R$

Lemma 2-1 :
for all m in $(\Sigma' \cup \Sigma'' \cup Q)^*$,
- either $m \in (\Sigma' \cup Q)^*.(\Sigma'' \cup Q)^*$
- or m can be written in one way $m_1a_1''u_1b_1'm_2a_2''u_2b_2'...m_na_n''u_nb_n'm_{n+1}$ *with* $n>0$, $m_1,m_2,...,m_{n+1} \in (\Sigma' \cup Q)^*.(\Sigma'' \cup Q)^*$, $u_1,u_2,...,u_n \in Q^*$, $a_1'',...,a_n'' \in \Sigma''$, $b_1',...,b_n' \in \Sigma'$

Proof : By induction on the length of m. ❑

Corollary 2-1 :
Every word m of $(\Sigma' \cup \Sigma'' \cup Q)^* - (\Sigma' \cup Q)^*.(\Sigma'' \cup Q)^*$ *can be written in one way* $w_1w_2...w_{n+1}$ *with* $w_i \in \Sigma'.(\Sigma' \cup Q)^*(\Sigma'' \cup Q)^*.\Sigma''$ *for* $1 < i < n+1$,
and $w_1, w_{n+1} \in (\Sigma' \cup Q)^*.(\Sigma'' \cup Q)^*$.

Proof :
In the decomposition given by lemma 2-1, we suppose $w_1 = m_1a_1''u_1$, $w_i = b_i'm_{i+1}a_{i+1}''u_{i+1}$ for $1 < i < n+1$ and $w_{n+1} = b_n'm_{n+1}$ ❑

Definition 2-3 : Let m be a word of $(\Sigma' \cup \Sigma'' \cup Q)^*$. We define the *signature of m* by
- If $m \in (\Sigma' \cup Q)^*.(\Sigma'' \cup Q)^*$ then its signature is the empty set.
- Else, m can be written in one way $m_1a_1''u_1b_1'...m_na_n''u_nb_n'm_{n+1}$ and its signature is the set of occurences of letters a_i''. An occurence is denoted by the length of the shorted prefix of m containing this occurence.

We remark that the signature underlines the occurences of letters of Σ'', the successor of which is in Σ', without consideration of the states of Q.

Examples :
$\Sigma = \{a,b,c,d,e,f,g\}$. m = acfgc. The occurence of the first letter c in m is 2, the occurence of the

second letter c in m is 5.
$\Sigma = \{0,1\}$, $Q=\{q\}$
$m = 0'1'0'q0'q0''1''$. $sign(m) = \varnothing$
$m = 0'1'0''q0''q0'1'q0'0''1'$. $sign(m) = \{5,11\}$.

Lemma 2-2 :
The size and the signature of a word m are unchanged when m is reduced with rules of R.

Proof :
Let m be a word of $(\Sigma' \cup \Sigma'' \cup Q)^*$.
1st case: $m \in (\Sigma' \cup Q)^*.(\Sigma'' \cup Q)^*$.
then $sign(m) = \varnothing$. We apply on m a rule of R.
a) the rule has the form $a'q_1a_1''b'' \to_R a'a_2'q_2b''$, a and b some letters of Σ. This rule can be applied only if m is written $m_1a'q_1a_1''b''m_2$ with $m_1 \in (\Sigma' \cup Q)^*$ and $m_2 \in (\Sigma'' \cup Q)^*$. Then, applying the rule, we get the word $m' = m_1a'a_2'q_2b''m_2$. Hence $|m| = |m'| = |m_1| + |m_2| + 4$. Moreover, $m' \in (\Sigma' \cup Q)^*.(\Sigma'' \cup Q)^*$ therefore $sign(m') = \varnothing = sign(m)$.
b) the rule has the form $a'b'q_1a_1'' \to_R a'q_2b''a_2''$, a and b some letters of Σ. This rule can be applied only if m is written $m_1a'b'q_1a_1''m_2$ with $m_1 \in (\Sigma' \cup Q)^*$ and $m_2 \in (\Sigma'' \cup Q)^*$. Applying this rule, we get the word $m' = m_1a'q_2b''a_2''m_2$. Hence $|m| = |m'| = |m_1| + |m_2| + 4$. Moreover, $m' \in (\Sigma' \cup Q)^*.(\Sigma'' \cup Q)^*$ therefore $sign(m') = \varnothing = sign(m)$.
2nd case: $m = m_1a_1''u_1b_1'...m_na_n''u_nb_n'm_{n+1}$
We apply one rule of R to m. A rule can be applied only on a subword of m in $(\Sigma' \cup Q)^*.(\Sigma'' \cup Q)^*$. But we have just shown that for $m \in (\Sigma' \cup Q)^*.(\Sigma'' \cup Q)^*$, rewriting preserves signature and size. Hence this result is true for each m in $(\Sigma' \cup \Sigma'' \cup Q)^*$ ❑

Lemma 2-3 :
For every word m in $(\Sigma' \cup \Sigma'' \cup Q)^*-(\Sigma' \cup Q)^*.(\Sigma'' \cup Q)^*$ *written* $w_1w_2...w_{n+1}$
$(m \overset{*}{\to}_R t) \Leftrightarrow (t = t_1t_2...t_{n+1}$ *such that* $w_i \overset{*}{\to}_R t_i$, $i=1,...,n+1)$

Proof :
$\Rightarrow$) Let $m = w_1w_2...w_{n+1}$. We have shown with lemma 2-2 that rules of R can be applied only on w_i. Hence $(m \overset{*}{\to}_R t) \Rightarrow (t = t_1t_2...t_{n+1}$ such that $w_i \overset{*}{\to}_R t_i$, $i = 1,...,n+1)$
$\Leftarrow$) if $t = t_1t_2...t_{n+1}$ with $w_i \overset{*}{\to}_R t_i$ $(i = 1,...,n+1)$ then $w_1w_2...w_{n+1} \overset{*}{\to}_R t_1t_2...t_{n+1}$.
Therefore $m \overset{*}{\to}_R t$. ❑

Definition 2-4 :
Let m a word of form $a'v'qw''b''$ with $q \in Q$, $v' \in \Sigma'^*$, $w'' \in \Sigma''^*$, $a' \in \Sigma'$ and $b'' \in \Sigma''$. We associate to m the configuration of the machine A: $C(m) = avqwb$.

Remark: a and b are never read by the machine and mark the end of the tape (instead of #).

Lemma 2-4 :
$m \in (\Sigma'^+.Q.\Sigma''^+)$, $m \to_R m' \Leftrightarrow C(m) \to_A C(m')$.

Proof :

$\Rightarrow$) To the word m, we associate the configuration C(m). The rules of R have the following two forms:

(r1) $a'q_1a_1''b'' \rightarrow_R a'a_2'q_2b''$, for some a and b in Σ.

(r2) $a'b'q_1a_1'' \rightarrow_R a'q_2b''a_2''$, for some a and b in Σ.

1st case : rule (r1).

To apply (r1) on m, m should be of form $v'a'q_1a_1''b''w''$, $v' \in \Sigma'^*$, $w'' \in \Sigma''^*$. Then $C(m) = vaq_1a_1bw$. We know that $m \text{-(r1)}\rightarrow_R m' \Leftrightarrow m' = v'a'a_2'q_2b''w''$. So $C(m') = vaa_2q_2bw$. But (r1) has been constructed from transition $(q_1,a) \rightarrow_A (a_2,q_2,D)$. Hence $C(m) \rightarrow_A C(m')$.

2nd case : rule (r2).

To apply (r2) on m, m should be of form $v'a'b'q_1a_1''w''$, $v' \in \Sigma'^*$ and $w'' \in \Sigma''^*$. Then $C(m) = vabq_1a_1w$. Since $m \text{-(r1)}\rightarrow_R m' \Leftrightarrow m' = v'a'q_2b''a_2''w''$, $C(m') = vaq_2ba_2w$. But (r2) has been constructed from transition $(q_1,a) \rightarrow_A (a_2,q_2,G)$. Hence $C(m) \rightarrow_A C(m')$.

$\Leftarrow$) Suppose that m and m' are such that $C(m) \rightarrow_A C(m')$. Transitions of A are of the following two forms:

(t1) $(q_1,a_1) \rightarrow_A (a_2,q_2,D)$.

(t2) $(q_1,a1) \rightarrow_A (a_2,q_2,G)$.

$C(m) = \alpha\sigma q_1a_1\omega\beta$ with $\alpha,\beta \in \Sigma$ and $\sigma,\omega \in \Sigma^*$. On C(m), we can apply (t1) or (t2).

1st case: transition (t1).

To every transition (t1) and for all a,b in Σ we associate the rule (r1) : $a'q_1a_1''b'' \rightarrow_R a'a_2'q_2b''$. The word m associated to configuration C(m) is $\alpha'\sigma'q_1a_1''\omega''\beta''$. The word m' such that $C(m) \text{-(t1)}\rightarrow_A C(m')$ is $\alpha'\sigma'a_2'q_2\omega''\beta''$. Therefore $m' = v'a'a_2'q_2b''w''$ supposing $v'a' = \alpha'\sigma'$ and $b''w'' = \omega''\beta''$. Then $m = v'a'q_1a_1''b''w''$. Hence, $m \rightarrow_R m'$.

2nd case : transition (t2).

To every transition (t2) and for all a,b in Σ we associate the rule (r2) : $a'b'q_1a_1'' \rightarrow_R a'q_2b''a_2''$. The word m associated to C(m) is $\alpha'\sigma'q_1a_1''\omega''\beta''$ (as in first case). m is written $v'a'q_1a_1''b''w''$ and the word m' such that $C(m)\text{-(t2)}\rightarrow_A C(m')$ is written $v'q_2a''a_2''b''w''$. Since C(m) is a configuration, v' contains at least one letter. Therefore we can apply a rule (r2) and $m \rightarrow_R m'$. ❑

Lemma 2-5 :

The machine A does not terminate $\Rightarrow$ The rewriting system R does not terminate

Proof : Suppose that the machine A does not terminate, then starting from an initial configuration C_0 there exists an infinite computation $C_0 \rightarrow_A C_1 \rightarrow_A \ldots$. But for every configuration C_i, there exists a word m_i in $\Sigma'^+.Q.\Sigma''^+$ such that $C_i = C(m_i)$. Hence there exists $m_0, m_1, \ldots$ in $\Sigma'^+.Q.\Sigma''^+$ such that $C(m_0) \rightarrow_A C(m_1) \rightarrow_A \ldots$. According to lemma 2-4, $C(m_0) \rightarrow_A C(m_1) \rightarrow_A \ldots \Leftrightarrow m_0 \rightarrow_R m_1 \rightarrow_R \ldots$. Therefore R does not terminate. ❑

Lemma 2-6 :

The rewriting system R does not terminate $\Rightarrow$ there exists an infinite subcomputation $C_1 \rightarrow_A C_2 \rightarrow_A \ldots$

Proof : We suppose that R does not terminate.

Hence there exists an infinite derivation $m \rightarrow_R \ldots$. From lemma 2-1, either $m \in (\Sigma' \cup Q)^*.(\Sigma''$

$\cup Q)^*$, or m is written $m_1a_1"u_1b_1'...m_{n+1}$.
1) if $m \in (\Sigma' \cup Q)^*.(\Sigma" \cup Q)^*$
On m, we can apply a rule of R. Therefore $m = v_1\sigma v_2$ with $v_1 \in (\Sigma' \cup Q)^*.Q^+$ or $v_1 \in Q^*$, $v_2 \in Q^+.(\Sigma" \cup Q)^*$ or $v_2 \in Q^*$, and $\sigma \in \Sigma'^+.Q.\Sigma"^+$. To σ we associate the A configuration : $C(\sigma)$. We cannot apply a rule on v_1 and v_2, so we apply it on σ. Hence $\sigma \rightarrow_R ...$. According to lemma 2-4, $C(\sigma) \rightarrow_A ...$
2) else $m = m_1a_1"u_1b_1'm_2a_2"u_2b_2'...m_{n+1} = w_1w_2...w_{n+1}$ (lemma 2-2).
From lemma 2-3, $m \overset{*}{\rightarrow}_R t \Leftrightarrow t = t_1...t_{n+1}$ with $w_i \overset{*}{\rightarrow}_R t_i$. Therefore, if the rewriting is infinite, there exists a w_i such that rewriting restricted to w_i is infinite. But we know that $w_i \in (\Sigma' \cup Q)^*.(\Sigma" \cup Q)^*$, so we can use the precedent case. ❑

Theorem 2-1 : *Termination of length-preserving string rewriting systems is undecidable.*

Proof : It suffices to verify that construction which associates a string rewriting system $R\varphi\psi$ to the linear-bounded automaton $A\varphi\psi$ reduces termination problem in class A_{Post} to termination problem for length-preserving string rewriting systems. For that, we can remark that proposition 1-2, lemma 1-3 and lemma 2-6 involve that if $R\varphi\psi$ does not terminate neither do $A\varphi\psi$. Indeed, if $R\varphi\psi$ does not terminate, $A\varphi\psi$ passes by a proper initial configuration again from which it loops.
From Lemma 2-5 we obtain the converse of the theorem. ❑

Corollary 2-2 :
Termination of non-length- increasing string rewriting system is undecidable.

Remark :
$A\varphi\psi$ could loop from a non-reachable configuration and stopped from every initial configurations. Therefore it could terminate in the machine sense, without assure termination for $R\varphi\psi$. Lemma 1-3 avoids this problem.

III - Undecidability of confluence of terminating rewriting systems on $q\Lambda^*$

In this section, we want to show that properties for linear-bounded automata cannot always be translated for non-length-increasing string rewriting systems. Indeed, for a linear-bounded automaton we start from an initial configuration, where the tape head is on the left side of the tape. That should mean, for a rewriting system, that we work on words starting by a special letter symbolizing initial state of linear-bounded automaton.

Definition 3-1 : [Huet]
A rewriting system is *confluent* if for all u that reduces to two terms t and t' there exists v such that t and t' reduce to v.

Theorem 3-1 : [Newman]
Confluence is decidable for terminating rewriting systems.

It is well known that confluence and ground confluence (confluence restricted to terms without variables) are not equivalent. In particular, ground confluence is undecidable for terminating term rewriting systems. Confluence implies ground confluence but the converse is false. Nevertheless, the following identification lemma shows that we can identify confluence of any semi-Thue system S with both confluence and ground confluence of the corresponding term rewriting system S'.

Identification lemma :
Let S be a semi-Thue system over a (non ranked) alphabet Σ. We associate to S the term rewriting system S' over the ranked alphabet $\Sigma' = \{a(x) \mid x \in \Sigma\} \cup \{\$\}$. ($ is a constant).
$S' = \{l(x) \to r(x) \mid l \to r \in S\}$
Then $t \overset{*}{\to}_S u \Leftrightarrow t(x) \overset{*}{\to}_{S'} u(x) \Leftrightarrow t(\$) \overset{*}{\to}_{S'} u(\$)$

proof : obvious ❑

As a corollary, confluence of S, confluence and ground confluence of S' coincide.
From now, we use this identification and work only in the word case. We describe in this paragraph a terminating rewriting system. This system is not confluent on words of Λ^* and its confluence is undecidable on $q\Lambda^*$, q a fixed symbol of Λ.

Let $A\varphi\psi$ be a machine of class A_{Post} studied in part I. We associate to this machine an alphabet: $\Lambda = Q\varphi\psi \cup \Sigma \cup \{q, q_{yes}, Y, N\}$. We modify the machine $A\varphi\psi$. We remote the last transitions which make the machine loop. Therefore, if the data has the form $\tilde{m}m'$ with $m' = \varphi(m) = \psi(m)$, the machine goes to the configuration $\#q_{end}<\tilde{m}m'>\#$ and stops.
To this new automaton, we associate a rewrite system R1. It contains all the rules simulating the transitions of the machine: A transition of form $(q,a) \to_{A\varphi\psi} (b,q',Le)$ is associated to the rule $hqa \to_{R1} q'hb$, a, b, h in Λ and q, q' in $Q\varphi\psi$; A transition of form $(q,a) \to_{A\varphi\psi} (b,q',Ri)$ is associated to the rule $qa \to_{R1} bq'$, a, b, in Λ and q, q' in $Q\varphi\psi$.
Moreover, R1 contains the rules:
$q_{end}< \to_{R1} q_{yes}$
$\forall a \in \Lambda\text{-}\{\#\}\ q_{yes}a \to_{R1} q_{yes}$
$q_{yes}\# \to_{R1} Y$
$q \to_{R1} <q_0$

We consider now this rewriting system R2:
$\forall f \in \Lambda\text{-}\{Y, N\},\ f \to_{R2} N$
$\forall a \in \Lambda,\ Ya \to_{R2} N$
$\forall a \in \Lambda,\ Na \to_{R2} N$

R is the system constituted by the rules of R1 and R2. R is terminating.

Lemma 3-1 :
$m \in I^+, m' \in X^*$, $q\tilde{m}m'>\# \overset{*}{\to}_R Y$ *if and only if* $\varphi(m) = \psi(m) = m'$.

(we denote $\tilde{m}$ the mirror image of m)

Proof :
$\Leftarrow$) if $\varphi(m) = \psi(m) = m'$ then according to part I, the machine $A\varphi\psi$, from an initial configuration $\#<q_0 i_n \ldots i_1 x_1 \ldots x_p>\#$ goes to the instantaneous description $\#q_{end}<i_n \ldots i_1 x_1 \ldots x_p>\#$ with $m = i_1 \ldots i_n$ and $m' = x_1 \ldots x_p$. Hence with R we can go from the word $qi_n \ldots i_1 x_1 \ldots x_p>\#$ to the word $<q_0 i_n \ldots i_1 x_1 \ldots x_p>\#$ and to $q_{end}<i_n \ldots i_1 x_1 \ldots x_p>\#$. Applying the rule $q_{end}< \rightarrow_{R1} q_{yes}$ we get $q_{yes} i_n \ldots i_1 x_1 \ldots x_p>\#$. Applying several times rules of form $q_{yes}a \rightarrow_{R1} q_{yes}$ we get the word $q_{yes}\#$. Finally with the rule $q_{yes}\# \rightarrow_{R1} Y$ we get the word Y.
$\Rightarrow$) we suppose that $q\tilde{m}m'>\# \rightarrow_R Y$ with $m = i_1 \ldots i_n$ and $m' = x_1 \ldots x_p$.
m and m' do not contain Y neither q_{yes} neither any state of $Q\varphi\psi$. There is only one manner to get Y: using the rule $q_{yes}\# \rightarrow_{R1} Y$. Moreover, the only manner to get q_{yes} is to apply the rule $q_{end}< \rightarrow_{R1} q_{yes}$. To generate q_{end}, we have applied rules simulating $A\varphi\psi$: $qi_n \ldots i_1 x_1 \ldots x_p>\# \rightarrow_R$ $<q_0 i_n \ldots i_1 x_1 \ldots x_p>\# \overset{*}{\rightarrow}_R q_{end}<i_n \ldots i_1 x_1 \ldots x_p>\#$.
But, we have seen in paragraph I that $\# <q_0 i_n \ldots i_1 x_1 \ldots x_p>\# \rightarrow_A \#q_{end}<i_n \ldots i_1 x_1 \ldots x_p>\#$ if and only if $\varphi(i_1 \ldots i_n) = \psi(i_1 \ldots i_n) = x_1 \ldots x_p$. ❑

Definition 3-2 :
A word w *is a normal form* if there exists no word v with $w \rightarrow_R v$.
A word w *has a normal form* if $w \overset{*}{\rightarrow}_R v$ for some normal form v.
The set of normal forms of a word m (called irreducible forms) is denoted by IRR(m).

Lemma 3-2 :
For all u in $\Lambda^+ - \{Y\}$, N is in IRR(u).

Proof:
- if u does not start by N neither by Y . Then applying the rule $f \rightarrow_{R2} N$ and possibly the rules of form $Na \rightarrow_{R2} N$ we obtain the reduction $u \overset{*}{\rightarrow}_R N$. So N is in IRR(u).
- if u starts by N : u = N.w
If $w = \varepsilon$ then u = N and so u is irreducible. Hence N is in IRR(u). Else we apply several times rules $Na \rightarrow_{R2} N$ and we get N in IRR(u).
- if u starts by Y : u = Y.w with $w \neq \varepsilon$. Hence we can apply a rule $Ya \rightarrow_{R2} N$ and possibly rules $Na \rightarrow_{R2} N$. Therefore N is in IRR(u). ❑

Lemma 3-3 :
$\forall m \in \Lambda^*, IRR(m) \subset \{Y,N,\varepsilon\}$. $\forall m \in \Lambda^+, IRR(m) \subset \{Y,N\}$

Proof:
Suppose that there exists $u \in IRR(m)$, $u \notin \{Y,N,\varepsilon\}$. Then, from lemma 3-2, u can be reduced to N. So u is not a normal form. Moreover, it is obvious that $\varepsilon \in IRR(m)$ if and only if $m = \varepsilon$. ❑

Lemma 3-4 :
Let qw be a word in $q\Lambda^*$.
If qw has the form $q\tilde{m}m'>\#$ then
i) $IRR(qw) = \{Y,N\}$ iff $\varphi(m) = \psi(m) = m'$

ii) IRR(qw) = {N} iff $\varphi(m) \neq m'$ *or* $\psi(m) \neq m'$
Else IRR(qw) = {N}.

Proof :
1) qw has the form $q\tilde{m}m'>\#$
i) According to lemma 3-1, $q\tilde{m}m'>\# \to_R Y$ if and only if $\varphi(m) = \psi(m) = m'$.
Moreover $q\tilde{m}m'>\# \to_R N$ from lemma 3-2. Hence $\{Y,N\} \subset IRR(qw)$ if and only if $\varphi(m) = \psi(m) = m'$. Lemma 3-3 shows that $IRR(qw) = \{Y,N\}$ iff $\varphi(m) = \psi(m) = m'$.
ii) $\varphi(m) \neq m'$ or $\psi(m) \neq m' \Leftrightarrow Y \notin IRR(qw)$. But $q\tilde{m}m'>\# \to_R N$. Hence $IRR(qw) = \{N\}$
2) qw has not the form $q\tilde{m}m'>\#$. Then qw does not correspond to a proper initial configuration of $A\varphi\psi$. Hence we apply rules of R without obtain the word $q_{end}<m$. Therefore $Y \notin IRR(qw)$. But $qw \to_R N$. Hence $IRR(qw) = \{N\}$. ❑

Theorem 3.2 : *q is a fixed symbol of a finite alphabet Λ. The following problem is undecidable.*
Instance : R semi-Thue, non-length-increasing, terminating, not confluent.
Question : is R confluent on $q\Lambda^*$ *?*

Proof : According to lemma 3-4, R is divergent on $q\Lambda^*$ if and only if there exists m such that $\varphi(m) = \psi(m)$. Hence R convergence on $q\Lambda^*$ is equivalent to Post problem. Consequently it's undecidable. ❑

Using the identification lemma, we get the corollary:
Corollary 3-1 : *The following problem is undecidable.*
Instance: R a terminating, rewriting system. S a sort.
Question : is R confluent on S ?

Remark: S can be choosen very simple. For example, S is the set of terms of root q.

References

[Book] Personnal communication.

[Dauchet] Max Dauchet
"Simulation of Turing Machines by a left-linear rewrite rule"
Rewriting Techniques and Applications. 3rd international conference, RTA-89
Chapel Hill, North Carolina, USA, April 1989 Proceedings in LNCS 355
N. Dershowitz (Ed.) p 109-120 (1987)

[Dershowitz] Nachum Dershowitz
"Termination of Rewriting" J.Symbolic Computation (1987) **3**, 69-116

[Dershowitz & Jouannaud] N. Dershowitz and J.P. Jouannaud
"Rewrite systems" Rapport de recherche 478. Unité associée au CNRS 410. (1989)

[Hooper] Philip K. Hooper
"The undecidability of the Turing machine immortality problem"
J.Symbolic Logic **31** (2) June 1966. (1966)

[Hopcroft & Ullman] J.E. Hopcroft and J.D. Ullman
"Some results on tape-bounded Turing machines"
J.A.C.M. Vol **16** (1), January 1967, pp 168-177.

[Huet] Gérard Huet
"Confluent reductions: abstact ptoperties and applications to term rewriting systems"
J.A.C.M. Vol **27**, (4), October 1980 pp 797-821. (1980)

[Huet & Lankford] G. Huet and D.S. Lankford
" On the uniform halting problem for term rewriting systems", Rapport laboria 283, Institut de Recherche en Informatique et en automatique, Le Chesnay, France, Mars 1978. (1978)

[Kuroda] S.-Y. Kuroda
"Classes of languages and linear-bounded automata"
Information and Controle **7**, 207-223 (1964).

[Litovsky & Metivier] Igor Litovsky and Yves Metivier
"Computing with graph rewriting systems with priorities" Rapport interne LaBRI 90-87

[Myhill] J.Myhill
"Linear bounded automata"
WADD Tech. Note No. 60-165, Wright-Patterson Air Force Base, Ohio. (1960)

[Newman] M.H.A. Newman
"On theories with a combinatorial definition of equivalence"
Annals of Mathematics **43** (2), p. 223-243. (1942)

[Otto] Friedrich Otto
"On deciding the confluence of a finite string-rewriting system on a given congruence class"
J. Comput. System Sciences **35**, 285-310 (1987)

[Post] Emil L. Post
"A variant of a recursively unsolvable problem".
Bulletin of the American Mathematical Society **52** p 264-268. (1946)

Appendix I : Definition of a machine $A\varphi\psi$

In the transitions, for i in I, $\varphi(i)_j$ represents j^{rd} letter of $\varphi(i)$,
$\varphi(i) = \varphi(i)_1 \ldots \varphi(i)_{k_i}$. We use the same notation for ψ: $\psi(i)_j$ represents j^{rd} letter of $\psi(i)$, $\psi(i) =$

$\psi(i)_1...\psi(i)_{ri}$.
$Q\varphi\psi = \{q_0, q_1, q_2, q_{ret}, q_{reset}, q_{again}, q'_{ret}, q'_{reset}, q_{end}\} \cup \{q_{\varphi(i)j} / i \in I$ and $\varphi(i) = \varphi(i)_1...\varphi(i)_{ki}$, $j \in [1,k_i]$ if $\varphi(i) \neq \varepsilon\} \cup \{q_{\varphi(i)0}$ if $\varphi(i) = \varepsilon\} \cup \{q_{\psi(i)j} / i \in I$ and $\psi(i) = \psi(i)_1...\psi(i)_{r_i}$, $j \in [1,r_i]$ if $\psi(i) \neq \varepsilon\} \cup \{q_{\psi(i)0}$ if $\psi(i) = \varepsilon\}$.
. The tape is of form #<m>#, m is a word in $(I \cup X)^*$. initially, tape head is on the first letter of m, and machine state is q_0.
. Δ is the next-move function:
A transition $(q,a) \to (b,q',Dir)$ means that the head reads the letter a, the state of the automaton is q, and after applying the transition, the letter replacing a is b, the new state is q' and the head goes to the direction Dir (Ri for rigth, Le for left).
$\forall$ i,i' $\in$ I and $\forall$ x $\in$ X,
a) rules of equality verification: $m_X = \varphi(m_I)$.
$(q_0, i) \to (i, q_1, Ri)$ % research of i on the rigth side
$(q_1, i) \to (i, q_1, Ri)$
$(q_1, x) \to (x, q_2, Le)$
$(q_2, i) \to (\bar{i}, q_{\varphi(i)1}, Ri)$ % research of its image by φ if it is not ε.
$(q_{\varphi(i)1}, \bar{i}) \to (\bar{i}, q_{\varphi(i)1}, Ri)$
$(q_{\varphi(i)1}, \bar{x}) \to (\bar{x}, \underline{q_{\varphi(i)1}}, Ri)$
$(q_{\varphi(i)j}, \varphi(i)_j) \to (\overline{\varphi(i)_j}, q_{\varphi(i)j+1}, Ri)$
$(q_{\varphi(i)k_i}, \varphi(i)_{ki}) \to (\overline{\varphi(i)_{ki}}, q_{ret}, Le)$ % we do it again for next i.
$(q_{ret}, \bar{x}) \to (\bar{x}, q_{ret}, Le)$
$(q_{ret}, \bar{i}) \to (\bar{i}, q_{ret}, Le)$
$(q_{ret}, i) \to (\bar{i}, q_{\varphi(i)1}, Ri)$ % if the image of i by φ is not ε
$(q_{ret}, i) \to (\bar{i}, q_{\varphi(i)0}, Le)$ % if the image of i by φ is ε
$(q_{ret}, <) \to (<, q_{reset}, Ri)$
$(q_2, i) \to (\bar{i}, q_{\varphi(i)0}, Le)$ % if the image of i by φ is ε
$(q_{\varphi(i)0}, i') \to (\bar{i}', q_{\varphi(i')1}, Ri)$ % if the image of i' by φ is not ε
$(q_{\varphi(i)0}, i') \to (\bar{i}', q_{\varphi(i')0}, Le)$ % if the image of i' by φ is ε
$(q_{\varphi(i)0}, <) \to (<, q_{reset}, Ri)$

b) rules of tape reset.
$(q_{reset}, \bar{i}) \to (i, q_{reset}, Ri)$
$(q_{reset}, \bar{x}) \to (x, q_{reset}, Ri)$
$(q_{reset}, >) \to (>, q_{recom}, Le)$
$(q_{again}, x) \to (x, q_{again}, Le)$

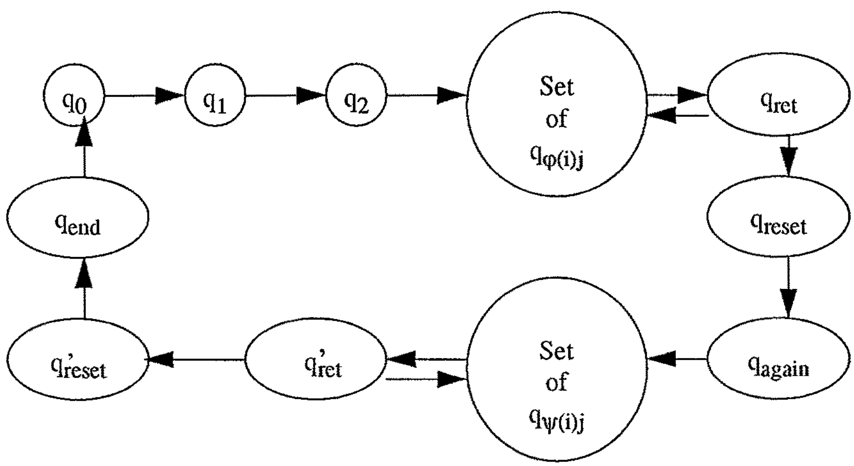
c) rules of equality verification : $m_X = \psi(m_I)$.
$(q_{again}, i) \to (\bar{i}, q_{\psi(i)1}, Ri)$ % if the image of i by ψ is not ε
$(q_{\psi(i)1}, \bar{i}) \to (\bar{i}, q_{\psi(i)1}, Ri)$
$(q_{\psi(i)1}, \bar{x}) \to (\bar{x}, \underline{q_{\psi(i)1}}, Ri)$
$(q_{\psi(i)j}, \psi(i)_j) \to (\overline{\psi(i)_j}, q_{\psi(i)j+1}, Ri)$
$(q_{\psi(i)ri}, \psi(i)_{ri}) \to (\overline{\psi(i)_{ri}}, q'_{ret}, Le)$
$(q'_{ret}, \bar{x}) \to (\bar{x}, q'_{ret}, Le)$
$(q'_{ret}, \bar{i}) \to (\bar{i}, q'_{ret}, Le)$

$(q'_{ret}, i) \rightarrow (\bar{i}, q_{\psi(i)1}, Ri)$
$(q'_{ret}, i) \rightarrow (\bar{i}, q_{\psi(i)0}, Le)$
$(q'_{ret}, <) \rightarrow (<, q'_{reset}, Ri)$
$(q_{again}, i) \rightarrow (\bar{i}, q_{\psi(i)0}, Le)$ % if the image of i by ψ is ε
$(q_{\psi(i)0}, i') \rightarrow (\bar{i}', q_{\psi(i')1}, Ri)$
$(q_{\psi(i)0}, i') \rightarrow (\bar{i}', q_{\psi(i')0}, Le)$
$(q_{\psi(i)0}, <) \rightarrow (<, q'_{reset}, Ri)$

d) rules of restoration of initial configuration
$(q'_{reset}, \bar{i}) \rightarrow (i, q'_{reset}, Ri)$
$(q'_{reset}, \bar{x}) \rightarrow (x, q'_{reset}, Ri)$
$(q'_{reset}, >) \rightarrow (>, q_{end}, Le)$
$(q_{end}, x) \rightarrow (x, q_{end}, Le)$
$(q_{end}, i) \rightarrow (i, q_{end}, Le)$
$(q_{end}, <) \rightarrow (<, q_0, Ri)$

Appendix II : Proof of lemma 1-3

Let ID a configuration such that $A\varphi\psi$ loops from it.
First case: ID is reachable
If ID is reachable, there exists an initial configuration from which the machine has reached ID. Therefore, there exists a beginning of computation which loops. The machine $A\varphi\psi$ loops if andonly if $P(\varphi,\psi)$ is verified. Moreover, when Post property is verified, the machine loops passing by a proper initial configuration again. Hence there exists a beginning of computation starting from a proper initial configuration which loops.
Second case: ID is not reachable.
Let's see the dependence graph between states of $A\varphi\psi$.



We write $Q\varphi$ the set of states $q_{\varphi(i)j}$ and $Q\psi$ the set of states $q_{\psi(i)j}$.
1) machine does not loop in one state, neither in a set $Q\varphi$ or $Q\psi$.
- For all states of $A\varphi\psi$, when the machine applies a transition staying in one state, then the head

always goes in the same direction. But the tape is finite, on the right and on the left. Hence the machine cannot stay in the same state doing an infinite number of computation steps.
- We consider the set $Q\varphi$
We stay in a state $q_{\varphi(i)j}$ if and only if j=1.
* if j > 1 then we go to state $q_{\varphi(i)j+1}$. Since j has a finite number of possible values, the machine does a finite number of computation steps staying in $Q\varphi$.
* if j = 0 then we replace a letter of I by a letter of $\bar{I}$, staying in a state of $Q\varphi$. But no transition in a state of $Q\varphi$ transforms a letter of $\bar{I}$ to a letter of I.Therefore the number of letter of I on the tape decreases. Moreover, if there are no letter of I on the tape, we cannot apply transition being in $q_{\varphi(i)0}$ and staying in a state of $Q\varphi$.
Hence, we do a finite number of computation steps, staying in $Q\varphi$.
- We can do the same proof for $Q\psi$.

2) The machine does not loop on $Q\varphi$ and q_{ret}
When it passes from $Q\varphi$ to q_{ret} , at least one letter of the tape has been overlined. Moreover, it passes from q_{ret} to $Q\varphi$ overlining one letter of the tape. Since there's a finite number of letters on the tape, there's a finite number of computation steps. We can have the same argument for $Q\psi$ and q'_{ret} .

3) From 1) and 2) we deduce the machine loops passing by q_0, q_1, q_2, q_{ret}, q_{reset}, q_{again}, q'_{ret} , q'_{reset} , q_{end} and by the sets $Q\varphi$ and $Q\psi$. But there is only one way to pass by q_0: using the transition $(q_{end} , <) \rightarrow (< , q_0 , D)$. Therefore, the machine is in an instantaneous description of form $\#<q_0m>\#$ which is, by definition, an initial configuration. Finally, according to precedent lemmas, if the machine loops from an initial configuration, then it's a proper one. ❑