**A GAP BETWEEN LINEAR
AND NON LINEAR TERM-REWRITING SYSTEMS (1)**

M. DAUCHET & DE COMITE

**UNIVERSITE DES SCIENCES ET TECHNIQUES
DE LILLE FLANDRES-ARTOIS
UFR D'IEEA - Informatique - Bât. M3
LIFL - UA 369 - CNRS
59655 VILLENEUVE D'ASCQ CEDEX**

ABSTRACT

We prove that every recursively enumerable set of terms is generated by a non linear t.r.s in 2 passes. Conversely, every set of terms which is generated by a linear t.r.s in a finite number of passes is recognizable.

INTRODUCTION

We recall the usual notion of pass for a t.r.s (see Fig. 1). In the non linear case, we must consider OI or IO passes [5, 6] (see example in part IA). A pass $t \models u$ is a sequence of rewritings, where all left-hand sides of rewriting rules occur in t without overlapping. The pass is OI if rewritings are computed from the outermost (top of tree) to the innermost of a term (bottom in tree notation). Recognizable tree languages are used in the usual sense [9]. It is well known that it is a "very small" class. All standard problems are decidable in this class [9]. We prove in part II that every recognizable tree language can be generated in two passes by a linear t.r.s and that **every tree language generated by a linear t.r.s in a finite number of passes is recognizable**. In part III, we prove that **every recursively enumerable tree language can be generated by a non linear t.r.s in two passes**. So, we quantify the well known [7] gap between the powerfullness of linear t.r.s and the powerfullness of general t.r.s

(1) Supported by P.R.C. "Mathematiques et Informatique".

PRELIMINARIES

A term rewriting system (t.r.s) is a finite set of rewriting rules [4,7]. Let $\rho : l \rightarrow r$ be a rule. l is the left-hand side and r is the right-hand side of ρ . l and r are trees with substitution variables $x_1, x_2, \dots, x_n$. A tree is **linear** iff no x_i occurs more than one time. A rule is linear iff its left-hand side and its right-hand side are linear. A t.r.s is linear iff every rule is linear $t \xrightarrow{*} u$ iff t can be rewritten in u by R . The rank of a rule is the number of different substitution variables.

Example :

$r_1 : b(b(x_1, x_2), x_1) \rightarrow b(x_1, x_2)$ is not linear,

$r_2 : b(x_1, x_2) \rightarrow a(x_1)$ is linear.

A letter a of rank n is sometimes denoted by $a(x_1, \dots, x_n)$.

A rule is **torsion free** iff its rank is some n , if it is linear and if $x_1, \dots, x_n$ occur in this order in both sides.

Example :

r_1 and r_2 are not torsion free.

$b(b(x_1, x_2), x_3) \rightarrow C(x_1, a, b(x_1, x_3))$ is torsion free.

T_Σ denotes the set of closed terms on the ranked alphabet Σ (i.e. the set of trees on Σ without substitution variables). A tree language is a set of terms. $[p]$ denotes the set $\{1, \dots, p\}$ for each integer p . We use **tree automata** in the usual point of view [4,6].

In a t.r.s point of view, a tree automaton can be seen as a 5-uple composed of :

- a ranked alphabet Σ
- a finite and distinguished set of **states** Q (rank 0) denoted $q_1, q_2, \dots, q, q', \dots$
- a final state q_f
- a set of ground rewriting rules of the form
- $a(q_1, \dots, q_n) \rightarrow q'$ ($a \in \Sigma$) or $q \rightarrow q'$.

A language L is recognizable iff there is some tree automaton M such as :

$$L = \{t \in T_\Sigma \mid t \xrightarrow[M]{*} q_f\}.$$

I. PASSES OF A TERM REWRITING SYSTEM

A. Definition of a pass

The linear case :

Let R be a linear t.r.s and t, u two terms. t is rewritten in u **in one pass** iff all the occurrences of rules used in the rewriting can be applied in parallel (i.e. all occurrences of left-hand sides occur in t without overlapping) (see Fig. 1). We suppose that at least one rule is used and note $t \xrightarrow[R]{1} u$

Figure 1.

The general case :

We must precise if subtrees are rewritten before or after generating or testing equalities. This is a well known problem [5,6]. There are two usual strategies.

OI pass : rewriting from the outermost (of a bracketed expression, i.e. from the top of a tree) to the innermost (i.e. the bottom of a tree).

IO pass : rewriting from the bottom to the top.

Example

Let us consider the t.r.s $b \rightarrow b' \quad a \rightarrow a'$.

In one pass OI we obtain

by one pass which can be split up into

but not $b(a, a') \models b'(a, a)$

Similarly in one pass IO we obtain $b(a, a') \models b(a', a')$ but not $b(a, a) \models b'(a, a')$

Notation

$\models$ denotes a pass for a linear t.r.s R. If R is non linear, we precise $\models_{\frac{OI}{R}}$ or $\models_{\frac{IO}{R}}$

Sometimes, we omit R and write $\models$.

B. Some notations

1) t is rewritten in u in n+1 passes (denoted by $t \models_{n+1} u$) iff for some v, $t \models_n v \models u$. In the general case, we note $\models_{\frac{nOI}{R}}$ or $\models_{\frac{nIO}{R}}$.

Nota Bene : $t \models_n u$ means t can be rewritten in u in n passes but it is also possible that $t \models_p u$ for some $p \neq n$.

2) Let F_1 and F_2 be two tree-languages

$$F_1 \models_{\frac{n}{R}} F_2 \text{ iff } F_2 = \{t_2 \mid \exists t_1 \in F_1, t_1 \models_{\frac{n}{R}} t_2\}$$

We say that "R rewrites F_1 in F_2 in n passes".

3) Let R a t.r.s defined on a ranked alphabet Δ and Γ_1, Γ_2 be two restricted alphabets ($\Gamma_1 \subset \Delta$ and $\Gamma_2 \subset \Delta$).

R_{Γ_1, Γ_2} denotes R **restricted** to Γ_1, Γ_2 :

$$F_1 \models_{\frac{n}{R_{\Gamma_1, \Gamma_2}}} F_2 \text{ iff } F_1 \models_{\frac{n}{R}} F_2, F_1 \subset T_{\Gamma_1} \text{ and } F_2 \subset T_{\Gamma_2}.$$

4) If $F_1 = T_{\Gamma_1}$, we omit it and note $\models_{\frac{n}{R_{\Gamma_1, \Gamma_2}}} F_2$. We say " R_{Γ_1, Γ_2} generates F_2

in n passes".

II. THE LINEAR CASE

Proposition 1

For every recognizable tree language F included in T_{Γ} there is some linear t.r.s R and some alphabet Γ' such as $R_{\Gamma', \Gamma}$ generated F in 2 passes. Furthermore, F is the set of normal forms of $R_{\Gamma', \Gamma}$.

ProofConstruction of R and Γ' :

Let M be a tree automaton which recognizes F. Q denotes the set of states $q_1, q_2, \dots$ of M. Q is distinct from Γ' and is identified with a set of letters of rank 1 $q_1(x_1), q_2(x_1), \dots$. We can suppose, there is only one final state q_f which does not occur on the left-hand side of rules of M. We define Γ' as being a copy of Γ by priming labels. t' and F' denote t and F 's corresponding copies.

For every rule $a(q_1, \dots, q_n) \rightarrow q$ of M, we construct the following rules of R.

Rules i) if $q \neq q_f$, $a'(x_1 \dots x_n) \rightarrow q(a(q_1(x_1) \dots q_n(x_n)))$

Rules ii) if $q = q_f$ and $n > 0$, $a'(x_1 \dots x_n) \rightarrow a(q_1(x_1) \dots q_n(x_n))$

Rules iii) if $q = q_f$ and $n=0$, $a' \rightarrow a''$ & $a'' \rightarrow a$. (We add the corresponding new letters a'' to Γ').

Nota Bene :

Rules iii) are only introduced to obtain, in exactly 2 passes, terms of F which are reduced to a single letter ; Rules iv) for every $q \neq q_f$ $q(q(x_1)) \rightarrow x_1$.

Sketch of the proof : (See Fig. 2)

The reader can prove that if $u' \in T_{\Gamma'}$, $v \in T_{\Gamma}$ and $u' \downarrow^2 v$, then the computation is necessarily as in Fig. 2 because

- letters of Γ' must be erased, and then two letters of Q separate two letters of Γ on each branch,

- letters of Q must be erased by rules iv). That means, a move of the automaton M is simulated.

- q_f appears necessarily at the top.

Figure 2

Proposition 2 : Let R be a linear t.r.s. For every integer n, the tree language generated by R in n passes is recognizable.

Proof : It is, by induction on n, a corollary of the following lemma :

Lemma : Let R be a linear t.r.s., F a recognizable tree language and F' such as

$F \xrightarrow[R]{1} F'$. Then F' is recognizable.

Proof of the lemma : Let M be an automaton which recognizes F. Let us construct M' which recognizes F'.

i) all rules of M are in M'.

ii) for every rule $\ell(x_1, \dots, x_p) \rightarrow r(x_{\sigma(1)}, \dots, x_{\sigma(q)})$ of R (σ is a one to one mapping of $[q]$ into $[p]$) :

if $\ell(q_1 \dots q_p) \xrightarrow[M]{*} q$ then we add a new set of rules with new intermediate states,

in order to get : $r(q_{\sigma(1)}, \dots, q_{\sigma(p)}) \xrightarrow[M]{*} q$.

Example :

$b(x_1, x_2) \rightarrow x_1$ and $c(x_1, x_2, x_3) \rightarrow b(b(x_1, x_3), x_2)$ are two rules of R and

$b(q_1, q_2) \xrightarrow[M]{*} q$, $c(q_1, q_2, q_1) \xrightarrow[M]{*} q_1$

We add the following rules to M' :

$q_1 \rightarrow q$ and $b(q_1, q_1) \rightarrow \bar{q}$ ($\bar{q}$ new state) $b(\bar{q}, q_2) \rightarrow q_1$.

End of example

It is easy to prove by induction on the size of u that for every state q of M :

$u \xrightarrow[M]{*} q$ iff $\left\{ \begin{array}{l} \text{(there is some } t \text{ such as } t \xrightarrow[M]{*} q \text{ and } t \xrightarrow[1]{*} u) \\ \text{or } t = u \end{array} \right.$

We suppress the case $t = u$ by a modification of M' which controls that at least one occurrence of right hand of rule of R is simulated by M' (usual tree automata technique). It is easy to conclude.

III. THE GENERAL CASE

Definition

- A tree grammar $G = (A, V, \Sigma, \rho)$ is given by a finite ranked alphabet V of non terminal letters, a finite ranked alphabet Σ of terminal letters, an axiom A of rank 0 ($A \in V$), a t.r.s ρ .

- G is torsion free if ρ is torsion free.

- The tree language $L(G)$ generated by G is defined by

$$L(G) = \{t \in T_{\Sigma} \mid A \stackrel{*}{\underset{\rho}{\vdash}} t\}$$

Proposition 3

A tree language is recursively enumerable iff there exists a torsion-free tree grammar which generates it.

The proof can be found in [1]. In order to prove the proposition 4, let us introduce the technical notion of derivation scheme. Let G be a tree grammar and $\rho = \{\rho_i \mid i \in I\}$ its set of rewriting rules (I is a finite set of indexes). l_i denotes the left-hand side of ρ_i and r_i the right-hand side.

For every $i \in I$, we define a new letter $*_i$. The rank of $*_i$ is the rank of ρ_i . Let us denote $*$ = $\{*_i \mid i \in I\}$.

Definitions

Let G be a torsion-free tree grammar.

i) Let $v \in T_{\Sigma \cup V \cup *}$. $v(l / *)$ denotes the term deduced from v by substituting l_i to every occurrence of $*_i$ (and for every $i \in I$).

ii) A finite sequence $(v_1, \dots, v_n)$ of $T_{\Sigma \cup V \cup *}$ is the scheme of a derivation δ in G iff :

- for every $j < n$, $v_j(r / *) = v_{j+1}(l / *)$.

- $v_1(l / *) = A$ and $v_n(r / *) \in T_{\Sigma}$.

Then the derivation δ can be split up into n passes (as ρ is torsion free, we are in the linear case).

$$\delta : A \vdash v_1(r / *) = v_2(l / *) \vdash v_2(r / *) \vdash v_n(r / *) \in L(G).$$

Proposition 4

For every recursively enumerable tree language F of T_{Σ} , we can construct a t.r.s

$R_{\Gamma, \Sigma}$ which generates F in 3 OI passes. Furthermore, F is the set of normal forms of $R_{\Gamma, \Sigma}$.

Proof

Let G be a torsion free grammar which generates F .

a) **Construction of the alphabets**

Let us define the alphabet Γ and a t.r.s R on a new alphabet Δ .

For each alphabet X , $\bar{X} = \{\bar{\alpha}_\lambda \mid \alpha_\lambda \in X\}$

$\Gamma = \bar{\Sigma} \cup \bar{V} \cup \bar{*} \cup \{\bar{\alpha}(x_1, x_2)\} \cup \{\bar{\alpha}(x_1)\}$

Where Σ , V and $*$ are alphabets associated to G as before and $\bar{\alpha}$ is a new letter.

$\Delta = \Gamma \cup \Sigma \cup V \cup \{\alpha, \alpha', \hat{\alpha}, k, k'\} \cup * \cup \#'$

Where α , α' and $\hat{\alpha}$ are new letters of rank 2, k and k' are new letters of rank 1.

b) **Notation**

For every t , $(t \mid k)$ is defined as follows

- if t is a letter of rank n , $(t \mid k) = k(t \mid k(x_1), \dots, k(x_n))$

- if $t = t'(t_1, \dots, t_n)$ and t' is a letter,

$(t \mid k) = k(t(k(t_1 \mid k), \dots, k(t_n \mid k)))$

c) **Construction of the t.r.s R**

" $\bar{\alpha}$ rules" : for every $A \rightarrow r \in \rho$ (A is the axiom of G).

$\bar{\alpha}(x_1) \rightarrow \hat{\alpha}(A, \alpha'(k'((r/k')), x_1))$

$\bar{\alpha}(x_1, x_2) \rightarrow \alpha(k(x_1), \alpha'(k'(x_1), x_2))$

$\bar{\alpha}(x_1) \rightarrow \alpha(k(x_1), k'(x_1))$

"erasrules" : $k(k(x_1)) \rightarrow x_1$ $k'(k'(x_1)) \rightarrow x_1$

" k k' rules" for every $a \in \Sigma$: $\bar{a} \rightarrow (a \mid k)$ $\bar{a} \rightarrow (a \mid k')$

" $*$ rules" for every $\#_i \in *$: $\bar{\#}_i \rightarrow (\#_i \mid k)$ and $\bar{\#}_i \rightarrow (\#_i' \mid k')$

" ℓ_r rules" for every $\#_i \in *$: $\#_i \rightarrow \ell_i$ and $\#_i' \rightarrow r_i$

" α rules" $\hat{\alpha}(x_1, x_2) \rightarrow x_2$ $\alpha'(x_1, \alpha(x_1, x_2)) \rightarrow x_2$.

d) **If $t \in F$ then $R_{\Gamma, \Sigma}$ generates t in 3 OI passes**

Let $(v_1, \dots, v_n)$ be the scheme of a derivation of t . Let $A \rightarrow r$ be the first pass. Figure 3 illustrates the three passes. (α -rules check v_j ($r / *$) = $v_{j+1}(\ell / *)$ for $i = 2, \dots, n-1$)

e) **If $R_{\Gamma, \Sigma}$ generates t , then $R_{\Gamma, \Sigma}$ generates t in 3 OI passes which simulate a G derivation as in Figure 3.**

The proof is tedious. Let us sketch it.

i) If $u \mid^* t$ then $\bar{u}$ is a subtree of $\bar{w}$ ($\bar{w} = \bar{w}' \cdot \bar{t}$).
 $R_{\Gamma, \Sigma}$

ii) If $\bar{w}' \cdot \bar{t} \mid^* t$, then $\bar{w}' \cdot \bar{t} \mid^* w'' \cdot (t \mid k) \mid^* t$
 or $\bar{w}' \cdot \bar{t} \mid^* w'' \cdot (t \mid k') \mid^* t$

(If occurrences of k and k' are mixed, they can't be erased).

iii) Let $w \mid^* \theta \mid^* t$. If α occurs in θ , then the first son of α can be
 $R_{\Gamma, \Sigma}$ $R_{\Gamma, \Sigma}$

neither k' , nor α , nor α' . The second son can be neither k nor α .

If α' occurs in θ , the first son of α' can be neither α , nor α' , nor k . The second son can be neither α' , nor k , nor k' .

If $\hat{\alpha}$ occurs in θ , $\hat{\alpha}$ is the top of θ .

iv) and so on.... if $\bar{w} \mid^* t$, then the rewriting is as in Fig. 3 !
 $R_{\Gamma, \Sigma}$

Figure 3

Nota bene : The initial restriction to Γ and the final one to Σ play an important part.

Other results

1) The 2nd and 3rd OI passes of Fig. 3 can be computed in a single IO pass.

2) The 1st OI pass on Fig. 2 can be computed in 2 IO passes.

The proof is obvious.

CONCLUSION We deduce from proposition 1, 2, 4 and other results the following theorem.

Gap theorem : Let $R_{\Gamma, \Sigma}$ be a t.r.s.

i) If R is linear, then every set of terms which is generated in a finite number of passes is recognizable. Every recognizable set of terms can be generated in 2 passes.

ii) If R is non linear, then every recursively enumerable set of terms can be generated in 2 passes (the first one is OI, the second one IO). We have the same result for 3 IO passes or for 3 OI passes.

REFERENCE

- [1] A. ARNOLD et M. DAUCHET. "Caractérisation algébrique des ensembles d'arbres récursivement énumérables". 1er Colloque AFCET-SMF, Paris, (1978), T2, pp. 1-14.
- [2] M. DAUCHET et J. MONGY. "Transformation de noyaux reconnaissables, capacité générative des bimorphismes de forêts". FCT'79, Akademie-Verlag, Berlin, pp.99-104.
- [3] F. DE COMITE. "Simulation linéaire de systèmes de réécriture". Thèse de 3ème cycle, Université de Lille I, (1984).
- [4] N. DERSHOWITZ. "Termination, rewriting techniques and applications". Lecture Notes in Computer Science 202, (1985), pp. 180-224.
- [5] J. ENGELFRIET. "Bottom-up top-down tree transformations. A comparison". Math. Systems Theory 9, (1975), pp. 198-231.
- [6] J. ENGELFRIET et M. SCHMIDT. "IO and OI". Journal of Computer and Systems Science 15, (1977), pp. 328-353, 16, (1978), pp. 67-99
- [7] J.V. GUTTAG, D. KAPUR et D.R. MUSSER. "Derived pairs, overlap closure and rewrite rules. New tools for analysing term rewriting systems". 9th Automata, Languages and Programming, Lecture Notes in Computer Science 140, (1982), pp. 300-312.
- [8] G. HUET et D.C. OPPEN. "Equations and rewrite rules : a survey". In Formal Language Theory : Perspectives and Open Problems. Academic Press, (1980).
- [9] J.W. THATCHER. "Tree automata : an informal survey". In "Currents in the theory of computing". (A.V. Aho, Ed.), Prentice Hall, (1973), pp. 143-178.