ON PUSHDOWN TREE AUTOMATA

Irène GUESSARIAN
LITP CNRS - UER de Mathématiques
Université Paris 7 - T 55 56 1er Et.
2, Pl. Jussieu - 75251 PARIS Cédex05

INTRODUCTION

Many structures in computer science can be represented by trees: derivation trees, syntax directed translations, search in files, etc... . There was thus developed, at first, a theory of recognizable tree languages, tree grammars and tree transducers /AD,E1,RA,T/. Tree language theory has since been helpful in a broad range of domains, e.g. decision problems in logics /RA/, formal language theory /A,T/ and program scheme theory /B, C, G, GU, N/. In order to be applicable to such a wide range of problems, the tree language theory had to be extended to allow context free tree languages; it thus became possible to model program scheme theory (which is our motivation) and also more general language theory (e.g. Aho's indexed languages /A, DO, E2, F, R/ - since an indexed language is the yield of a context free tree language). Now, any language theory usually presents three complementary aspects: grammatical, set-theoretical or algebraic and automaton theoretical. For recognizable tree languages, all three aspects have been well studied, even for infinitary languages /P, RA/, and comprehensive accounts can be found in the above given references /AD, E1, T,.../. For context free languages, the grammatical point of view was first studied in /DO, F, R/, the algebraic aspects were considered in /B, ES, GU, N/, and comprehensive and exhaustive surveys can be found in /B, E2, S/. But the automaton theoretical aspect has not been studied at all; it is merely hinted at in /B, R/, but we can say that the prevailing point of view is grammatical and no attempt has been made to find a more operational way of looking at context free tree languages. This paper is a first step in that direction. It is organized as follows: we first introduce pushdown tree automata (in short PDTA's), i.e. top down tree automata with a pushdown store which is also a tree. We prove that the class of languages recognized by such automata (either by empty store or by final state) is exactly the class of context free tree languages, and that each such language can be recognized by a restricted

automaton (whose pushdown store consists only of strings). We also prove the equivalence of various kinds of PDTA's and give numerous examples and counter-examples.

NOTATIONS

Let F (resp. Φ) be a finite ranked alphabet of base function symbols (resp. of variable function symbols). The rank of a symbol s in $F \cup \Phi$ is denoted by r(s). Symbols in F are denoted f,g,h,... if they have rank ≥ 1, and a,b,... if they have rank 0; F_i denotes the symbols of rank i in F. Symbols in Φ are denoted G,H,... . Let V be a set of variables; the variables have rank 0 and are denoted by u,v,w,... possibly with indices.

The notions of tree, node in a tree, occurrence of a variable or a subtree in a tree, substitution, etc... are supposed to be known (see /GU/). The set of finite trees (resp. finite trees with variables in V) on F is denoted by A(F) (resp. A(F,V)); it is called the *free F-magma*, or *F-algebra* (resp. the free F-magma over V), and is denoted, in the ADJ terminology FT_F (resp. $FT_F(V)$).

$A_i(F,V)$ denotes the trees having i variables , and $A^i(F)$ (resp. $A^i(F,V)$) denotes the trees of depth i; clearly, $\forall i$: $A^i(F) \subset A^i(F,V)$ and $A^i(F) \subset A(F) \subset A(F,V)$, and similar relations hold for the A_i's.

PUSHDOWN TREE AUTOMATA

Definition 1: A *context free tree grammar* G, also called a rewriting system, is a system G of n equations:

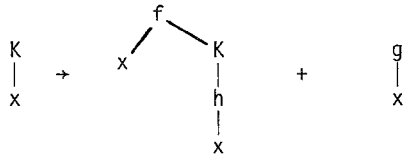
$$G: \quad G_i(v_1,\ldots,v_{r(G_i)}) = T_i$$

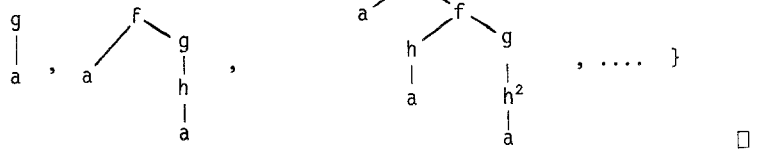
where for $i=1,2,\ldots,n$, $G_i \in \Phi$ and $T_i = \{t_i^j \ , \ j=1,\ldots,n_i\}$ is a finite subset of $A(F \cup \Phi, \{v_1,\ldots,v_{r(G_i)}\})$. □

Each pair $(G_i(\vec{v}),t_i^j)$ is called a production of G (notice the vector shorthand notation $G_i(\vec{v})$ for $G_i(v_1,\ldots,v_{r(G_i)})$).

Rewritings and derivations according to G are defined as usual. We denote by L(G,t) the language generated by G with axiom t (or L(G) when the axiom t is explicit by the context).

Example 1: Let G be defined by:

G : 

L(G, K | a) = {  , ... }

□

A pushdown tree automaton is a tree automaton /AD,E1/, together with a pushdown store consisting of a tree. We shall consider *top down* automata and see that, like top down tree recognizers, they have the essential ability of duplicating the pushdown store. The case of bottom up automata would be similar but is less suited to our purposes.

Definition 2: A *pushdown tree automaton* (PDTA) M is a seven-tuple (Q,F,X,q_0,Z_0,Q_f,R), where:

Q is a finite set of states

F is a finite ranked alphabet, called the input alphabet

X is a finite ranked alphabet, called the pushdown alphabet. We may suppose, w.l.g. that X and F are disjoint alphabets.

q_0 is the initial state

Z_0 is the starting tree, appearing initially on the pushdown store, and of the form:

$$Z_0 = \overset{E}{\underset{B_1 \ \cdots \ B_{r(E)}}{\diagup\,\vert\vert\,\diagdown}} \quad , \quad E \in X \text{ , and } B_1, \ \dots \ , B_{r(E)} \in X_0$$

Q_f is the set of final states ($Q_f \subset Q$)

R is a finite set of rules (or moves), such that each rule is a pair of one of the following forms:

(i) read move

$$\left(\overset{f:q}{\underset{v_1 \ \cdots \ v_{r(f)}}{\diagup\,\vert\vert\,\diagdown}} \ , \ \overset{E}{\underset{x_1 \ \cdots \ x_{r(E)}}{\diagup\,\vert\vert\,\diagdown}}\right) \ \vdash \ \overset{f}{\underset{v_1:q_1,t_1 \ \cdots \ v_{r(f)}:q_{r(f)},t_{r(f)}}{\diagup\,\vert\vert\,\diagdown}}$$

(ii) ε-move

$$(\varepsilon:q \ , \ \overset{E}{\underset{x_1 \ \cdots \ x_{r(E)}}{\diagup\,\vert\vert\,\diagdown}}) \ \vdash \ (\varepsilon:q' \ , \ t')$$

where $t', t_1, \dots, t_{r(f)}$ are elements of $A(X,\{x_1,\dots,x_{r(E)}\})$, and E is in X.

The computation relations $\vdash$ and $\vdash^*$ are defined as usual, as well as the notion of deterministic automaton (DPDTA): M is said to be *deterministic* iff for any pair (q,E), either there exists at most one ε-move in state q with pushdown root E and no read move, or there exists no ε-move and at most one read move for each f in F (in state q with pushdown root E).

The elements of X are denoted by B,C,D,... (symbols of rank 0), E,G,H,K,... (symbols of rank ≥1) , or $x_1,x_2,\dots$ (variables). The elements of F are denoted by the corresponding lower case letters and the variables representing elements of F are denoted by $v_1,v_2,\dots$.

The tree language accepted by final state by M is the set of trees T(M) defined as follows:

$$T(M) = \{ t \in A(F,V) \ / \ (t:q_0,Z_0) \vdash^* t' \in t(P(\vec{S}_0)/\vec{S}_0) \text{ , where } \vec{S}_0 = F_0 \cup V \text{ , and}$$

$$P(\vec{S}_0) = \{ \{\overset{s}{\underset{\varepsilon}{|}}\} \times Q_f \times A(X) \ / \ s \in \vec{S}_0 \} \ \}$$

where:

- $(t:q_0,Z_0)$ denotes the initial configuration where the root of t is scanned by M in state q_0 with pushdown store Z_0
- $t(P(\vec{S}_0)/\vec{S}_0)$ denotes the set of final configurations corresponding to the input tree t : in any such configuration, M should have processed the whole of t and arrived, after reading each leaf of t, in a state belonging to Q_f, with some pushdown store belonging to $A(X)$. So, $t(P(\vec{S}_0)/\vec{S}_0)$ is the set of trees deduced from t by substituting to every leaf of t labeled by an s in $\vec{S}_0$ an element of $\{\overset{s}{|}\}_{\varepsilon} \times Q_f \times A(X)$.

The tree language accepted by empty store by M is the set of trees $N(M)$ defined as follows:

$$N(M) = \{t \in A(F,V) \;/\; (t:q_0,Z_0) \vdash^* t' \in t(P'(\vec{S}_0)/\vec{S}_0) \text{ where } \vec{S}_0 = F_0 \cup V \text{ and}$$
$$P'(\vec{S}_0) = \{\, \{\overset{s}{|}\}_{\varepsilon} \times Q \times \varepsilon_{\tau} \;/\; s \in \vec{S}_0 \,\}$$

A tree t is in $N(M)$ if M processes the whole of t and arrives in a configuration where the pushdown store is empty after reading each leaf of t.

Variants of PDTA's which do not extend the class of defined languages (e.g. several initial states, a start symbol instead of a starting tree, a starting tree of depth greater than 2, etc...) may be introduced; these variants do not give additional power to the corresponding PDTA's.Let us check, for instance, that the ability to read in a single move input trees of depth greater than two does not extend the class of languages recognized by PDTA's.

Formally, let $F'=F-F_0$, $t(v_1,\ldots,v_n) \in A_n^i(F',V)$ where $i \geq 3$, $v_1,\ldots,v_n$ label the leaves of t, no two leaves have the same label and no leaf is labeled by a constant symbol; let us say that a PDTA is *extended* if it is allowed to make extended read moves of the following more general form:

$$t(v_1,\ldots,v_n):q,E(x_1,\ldots,x_{r(E)}) \quad \vdash \quad t(\,(v_1:q_1,t_1)/v_1,\ldots,(v_n:q_n,t_n)/v_n\,)$$

where E, the x_i's and t_j's are as in definition 2 and $t(w_1/v_1,\ldots,w_n/v_n)$ denotes the tree obtained by substituting the expression w_i to the leaf labeled by v_i, for $i=1,\ldots,n$.

Proposition 1: L is accepted by empty store (resp. final state) by some extended PDTA M iff it is accepted by empty store (resp. final state) by some PDTA M'.

Sketch of proof: The "if" part is trivial; for the "only if" part, we have to simulate each extended read move of M by a sequence of moves of M'. Formally, let:

$$m_i: \quad t_i(v_1,\ldots,v_{n_i}):q^i,E_i(x_1,\ldots,x_{r(E_i)}) \quad \vdash \quad t_i(\,(v_1:q_1^i,t_1^i)/v_1,\ldots,(v_{n_i}:q_{n_i}^i,t_{n_i}^i)/v_{n_i}\,)$$

for $i=1,\ldots,p$ and t_i of depth greater than 2 be the extended read moves of M. Then M' is defined as follows:

$Q' = Q \cup \{(m_i,o) \;/\; o$ is an internal occurrence in $t_i(\vec{v})$, for some $i=1,\ldots,p\}$

(Recall that an occurrence o is said to be internal if it is not a leaf, namely if its label $t_i(\vec{v})(o)$ has rank >0)

$F' = F$, $X' = X$, $q'_0 = q_0$, $Z'_0 = Z_0$, $Q'_f = Q_f$ and R' is defined as follows:

- ε-moves and non-extended read moves of M are in R'
- each extended read move m_i of M is replaced by the following set of moves:
  - let $f^i_\varepsilon = t_i(\vec{v})(\varepsilon) \in F'$ be the label of the root of $t_i(\vec{v})$; then R' contains the move :

    $f^i_\varepsilon(v_1,\ldots,v_{r(f^i_\varepsilon)}):q^i,E_i(\vec{x}) \vdash f^i_\varepsilon(w_1,\ldots,w_{r(f^i_\varepsilon)})$

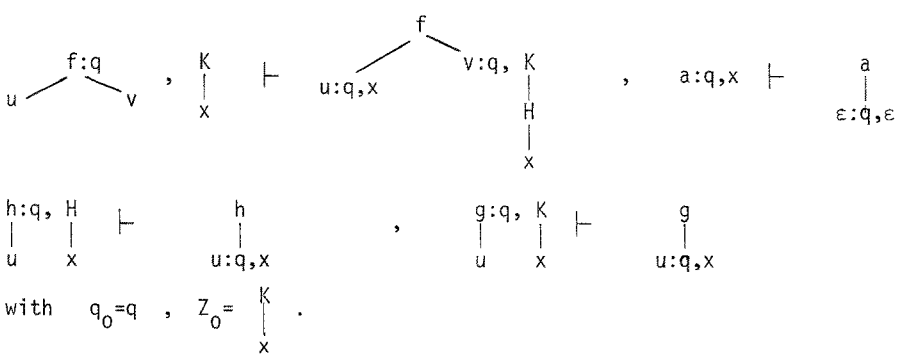
    where $w_j = (v_j:(m_i,j),E_i(\vec{x}))$ if j is internal in $t_i(\vec{v})$, and otherwise: $t_i(\vec{v})(j) = v_k \in V$, for some k , and $w_j = (v_j:q^i_k,t^j_k)$
  - for each internal occurrence $o\neq\varepsilon$ in $t_i(\vec{v})$, let $f^i_o = t_i(\vec{v})(o) \in F'$ be the label of o ; then R' contains the move:

    $f^i_o(v_1,\ldots,v_{r(f^i_o)}):(m_i,o),E_i(\vec{x}) \vdash f^i_o(w_1,\ldots,w_{r(f^i_o)})$

    where $w_j = (v_j:(m_i,oj),E_i(\vec{x}))$ if oj is internal in $t_i(\vec{v})$, and otherwise: $t_i(\vec{v})(oj) = v_k \in V$, for some k, and $w_j = (v_j:q^i_k,t^j_k)$.

For lack of space, we leave it to the reader to check that $N(M') = N(M)$ and $T(M') = T(M)$. □

Example 2: Consider the following automaton:



with $q_0=q$, $Z_0=$.

This automaton recognizes the language $L(G,K(a))$ of example 1. Moreover, this is an example of a real-time (without ε-moves), deterministic , and restricted automaton (i.e. its pushdown alphabet consists only of words, see definition 3 below). □

Theorem 1: L is $T(M)$ for some PDTA M iff L is $N(M')$ for some PDTA M'.

Sketch of proof: Show for instance that every $T(M)$ is an $N(M')$: M' simulates M until it reaches a final state, when it has the choice, either of going on simulating M , or of erasing the pushdown store with ε-moves. Formally, it suffices to add a state q_{er} to M together with the rules:

$\forall\, q_f \in Q_f$, $\forall\, E \in X$: $\varepsilon{:}q_f$,  $\vdash$ $\varepsilon{:}q_{er}$, x_1

$\forall\, E \in X$: $\varepsilon{:}q_{er}$,  $\vdash$ $\varepsilon{:}q_{er}$, x_1 □

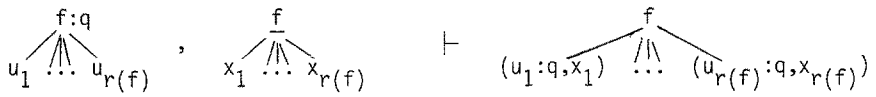
Theorem 2: Every context free tree language is an $N(M)$ for some PDTA M, and conversely, every $N(M)$ is context free.

Proof: The construction of M is quite straightforward for the first part of the theorem. Let G be a grammar with terminal alphabet F, non terminal alphabet Φ, and an axiom which we may suppose w.l.g. of the form $t_o = G(a_1,\ldots,a_{r(G)})$.

Let $\underline{F}$ be a copy of F, with say underlined letters having the same ranks as the corresponding letters of F. Then M has a single state q, input alphabet F, pushdown alphabet $X = \Phi \cup \underline{F}$, initial pushdown store $Z_o = \underline{t}_o = G(\underline{a}_1,\ldots,\underline{a}_{r(G)})$. For each t in $A(F\cup\Phi)$, let $\underline{t}$ denote the element of $A(\underline{F}\cup\Phi)=A(X)$ obtained by replacing each f in F by the corresponding $\underline{f}$ in $\underline{F}$. Then the productions of M are defined as follows:

- to each production  $\rightarrow$ t of G is associated an ε-move of M having the form: $\varepsilon{:}q$,  $\vdash$ $\varepsilon{:}q$, $\underline{t}$

- for each f in F, we have a read move of M, having the form:



Then the following facts are clear and give the desired result:

Fact 1: For each derivation $t \overset{*}{\rightarrow} t'$ of G, there is a sequence of ε-moves: $(\varepsilon{:}q,\underline{t}) \overset{*}{\vdash} (\varepsilon{:}q,\underline{t}')$ of M.

Fact 2: For each t in $A(F)$, there is a sequence of read moves of M which accepts t when beginning with pushdown store $\underline{t}$ (i.e. $(t{:}q,\underline{t}) \overset{*}{\vdash} t(P'(\vec{S}_o)/\vec{S}_o)$).

The construction, for the reverse part of the theorem, of some grammar G generating $N(M)$ is somewhat more complicated. We first give the idea of it: essentially, the changes of the pushdown store will be modeled by the productions of the grammar, but we will have also to model the additional ability of changing the state; hence, the non terminals of G will encode pairs (state, top pushdown symbol). Moreover, the arguments of each non terminal (q,K) should render all possible "next states" after K has been popped. First, we have to reduce M to a simpler form.

Notation: In the sequel, $\phi(t_1,\ldots,t_{r(\phi)})$ will be abbreviated in $\phi(\vec{t})$ for ϕ in $F \cup X$.

Lemma 1: If a tree language L is accepted by empty store by a PDTA M, then L is also accepted by empty store by a simplified PDTA M' whose only read moves on input symbols of rank 0 are of the following form: $(a:Q,B) \vdash \begin{array}{c} a \\ | \\ (\varepsilon:q,\varepsilon) \end{array}$.

Sketch of proof: Intuitively, a leaf is accepted whenever it is read, and this can only be done while popping a leaf of the store. M' has the same alphabets as M, and a set of states $Q_{M'} = Q_M \cup \bigcup_{a \in F_0} \{ Q_M^a \cup \bar{Q}_M^a \}$, where for each symbol a of rank 0 in the input alphabet F, Q_M^a and $\bar{Q}_M^a$ are two duplicate copies of the set Q_M of states of M. Then the moves of M' are obtained by adding to the moves of M the following set of moves:

- for each move m: $(f(\vec{u}):q,G(\vec{x})) \vdash f((u_1:q_1,t_1),\ldots,(u_{r(f)}:q_{r(f)},t_{r(f)}))$, add the set of moves m^a,
- for any a in F_0, m^a: $(f(\vec{u}):q,G(\vec{x})) \vdash f((u_1:q_1^a,t_1),\ldots,(u_{r(f)}:q_{r(f)}^a,t_{r(f)}))$
- for each ε-move: $(\varepsilon:q,t_0) \vdash (\varepsilon:q',t)$, with $t_0=G(\vec{x})$ or $t_0=B$, add the corresponding moves, for every a in F_0:
  $(\varepsilon:q,t_0) \vdash (\varepsilon:q'^a,t)$ and $(\varepsilon:\bar{q}^a,t_0) \vdash (\varepsilon:\bar{q}'^a,t)$
- for each read move on a leaf, $(a:q,G(\vec{x})) \vdash \begin{array}{c} a \\ | \\ (\varepsilon:q',t) \end{array}$ add the move:
  $(\varepsilon:q_a,G(\vec{x})) \vdash (\varepsilon:\bar{q}'^a,t)$
- finally, for each ε-move emptying the store: $(\varepsilon:q,B) \vdash (\varepsilon:q,\varepsilon)$, add the read moves: $(a:\bar{q}^a,B) \vdash \begin{array}{c} a \\ | \\ (\varepsilon:q,\varepsilon) \end{array}$, for every a in F_0.

M' simulates M, but delays reading the leaves by making a "guess" stored in the state which is checked when the store is emptied. □

Suppose now M is of the simplified form given in the previous lemma. Let $k = \mathrm{card}(Q)$, and $\Phi = \{G^q \ / \ q \in Q, \ G \in X\}$ where each G^q in Φ has rank $k \times r(G)$; similarly, if Y is the set of pushdown variable symbols, let $Y^Q = \{ x^q \ / \ x \in Y, \ q \in Q \}$ be a set of variables (x^q is intuitively intended to correspond to the selection of variable x and next state q). Define σ by: $\sigma: Q \times A(X,Y) \to A(\Phi,Y^Q)$ satisfies $\sigma(q,s) = s^q$ if s has rank 0 or is a variable, and, using a vector notation $\vec{\sigma}(t)$ instead of $(\sigma(q_1,t),\sigma(q_2,t),\ldots,\sigma(q_k,t))$, define by induction: $\sigma(q,G(t_1,\ldots,t_{r(G)})) = G^q(\vec{\sigma}(t_1),\ldots,\vec{\sigma}(t_{r(G)}))$ (in short $\sigma(q,G(\vec{t})) = G^q(\vec{\sigma}(\vec{t}))$).

The grammar G generating $N(M)$ is now defined as follows: G has terminal alphabet F and non terminal alphabet Φ, and:

- to each ε-move: $(\varepsilon:q,t_0) \vdash (\varepsilon:q',t)$ with $t_0=G(\vec{x})$ or $t_0=B$, is associated a production: $\sigma(q,t_0) \to \sigma(q',t)$ of G
- to each read move: $(f(\vec{u}):q,G(\vec{x})) \vdash f((u_1:q_1,t_1),\ldots,(u_{r(f)}:q_{r(f)},t_{r(f)}))$, is associated a production: $\sigma(q,G(\vec{x}))=G^q(\vec{\sigma}(\vec{x})) \to f(\sigma(q_1,t_1),\ldots,\sigma(q_{r(f)},t_{r(f)}))$ of G

- to each read move: $(a:q,B) \vdash \begin{array}{c} a \\ | \\ (\varepsilon:q',\varepsilon) \end{array}$, is associated a terminal production

$B^q=\sigma(q,B) \to a$ of G .

We then have:

Lemma 2: For every T in $A(X)$, t in $A(F)$, and q in Q : $\sigma(q,T) \overset{*}{\to} t$ iff there exists an accepting computation sequence $(t:q,T) \overset{*}{\vdash} t(P'(\vec{S}_0)/\vec{S}_0)$.

Proof: "only if" part: by induction on the length n of the derivation $\sigma(q,T) \overset{*}{\to} t$.

- If $n=1$ then: either $T=G(\vec{X})$ and $G^q(\vec{X}) \to \varepsilon=t$, which corresponds to the accepting move: $(\varepsilon:q,G(\vec{x})) \to (\varepsilon:q,\varepsilon)$.

or $T=B$ and $B^q \to a$, which corresponds to the accepting move:

$$(a:q,B) \to \begin{array}{c} a \\ | \\ (\varepsilon:q,\varepsilon) \end{array}$$

- Inductive step: if $\sigma(q,T) \xrightarrow{n+1} t$, then:

either $T=B$ and $\sigma(q,B)=B^q \to \sigma(q',T') \overset{n}{\to} t$, and by the induction $(t:q',T) \overset{*}{\vdash} t(P'(\vec{S}_0)/\vec{S}_0)$ is an accepting computation sequence , and by the construction of G there is a move $(\varepsilon:q,B) \vdash (\varepsilon:q',T')$, whence an accepting computation sequence starting from $(t:q,B)$

or $T=G(\vec{T}')$ and:

- either $\sigma(q,G(\vec{T}')) \to \sigma(q',T_1(\vec{T}')) \overset{n}{\to} t$, where we conclude as above since $(\varepsilon:q,G(\vec{x})) \vdash (\varepsilon:q',T_1)$ and $(t:q',T_1(\vec{T}')) \overset{*}{\vdash} t(P'(\vec{S}_0)/\vec{S}_0)$
- or $\sigma(q,G(\vec{T}')) \to f(\sigma(q_1,t_1(\vec{T}')),\ldots,\sigma(q_{r(f)},t_{r(f)}(\vec{T}'))) \xrightarrow{n} t$, whence we conclude that:
  1) $(f(\vec{u}):q,G(\vec{x})) \vdash f((u_1:q_1,t_1),\ldots,(u_{r(f)}:q_{r(f)},t_{r(f)}))$ is a move of M
  2) $t = f(t'_1, \ldots ,t'_{r(f)})$
  3) $\forall i=1,\ldots,r(f) \quad \sigma(q_i,t_i(\vec{T}')) \xrightarrow{p} t'_i$ for some $p \le n$

  whence by induction an accepting computation sequence of $(t:q,T)$ starting with the move $(f(\vec{t}'):q,G(\vec{T}')) \vdash f((t'_1:q_1,t_1(\vec{T}')), \ldots ,(t'_{r(f)}:q_{r(f)},t_{r(f)}(\vec{T}')))$

"if" part: similar proof using an induction on the maximum number of moves applied during the computation on each branch of t . □

Lemma 2 implies that G with axiom $\sigma(q_0,Z_0)$ generates $N(M)$. □

Example 3: Let M be defined by: $Q = \{q_0,q_1,q_2\}$, $F = \{b,c_1,c_2\}$, $X = \{G,C\}$ with $r(b)=2$, $r(G)=1$, $r(C)=r(c_1)=r(c_2)=0$. The moves are the following:

(i) $(\varepsilon:q_0,G(x)) \vdash (\varepsilon:q_0,G^2(x))$

(ii) $(b(u,v):q_0,G(x)) \vdash b((u:q_1,x),(v:q_2,x))$

(iii) for $i=1,2$ $(b(u,v):q_i,G(x)) \vdash b((u:q_i,x),(v:q_i,x))$

(iv) $(c_i:q_i,C) \vdash \begin{array}{c} c_i \\ | \\ (\varepsilon:q_i,\varepsilon) \end{array}$ for $i=1,2$

and $Z_0=G(C)$.

It is associated with G having non terminals G^0, G^1, G^2, of rank 3, and C^0, C^1, C^2, of rank 0, axiom $G^0(C^0,C^1,C^2)$, and productions:

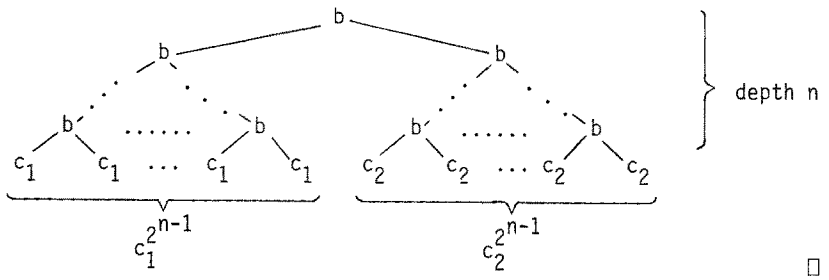
(i) $G^0(x^0,x^1,x^2) \to G^0(G^0(x^0,x^1,x^2),G^1(x^0,x^1,x^2),G^2(x^0,x^1,x^2))$

(ii) $G^0(x^0,x^1,x^2) \to b(x^1,x^2)$

(iii) for $i=1,2$ $G^i(x^0,x^1,x^2) \to b(x^i,x^i)$

(iv) for $i=1,2$ $C^i \to c_i$

and $L(G) = N(M)$ is the set of binary trees of the form:



□

The characterization given in theorem 2 is quite similar to the ones given in /B/ or /R/. The spirit of the proof is quite similar to the one in Rounds' paper. However, these papers stress the grammatical aspects of the problem, whereas we want to consider its operational and automaton theoretical aspects. This is why we introduce linear stacks and prove the subsequent simplification theorem, different from the ones in /B,R/, who reduce the number of states whereas we simplify the stack.

Definition 3: A restricted PDTA (in short RPDTA) is defined as a PDTA, except for:
- X which is an ordinary (non ranked) alphabet, the pushdown store consisting of words in the free monoid X^*.
- the set of moves which are now of the simpler forms:

(i) read move $(\overset{f:q}{\underset{v_1 \cdots v_{r(f)}}{/|\backslash}}, Ew) \vdash \overset{f}{\underset{v_1:q_1,w_1w \quad \cdots \quad v_{r(f)}:q_{r(f)},w_{r(f)}w}{/|\backslash}}$

(ii) ε-move $(\varepsilon:q,Ew) \vdash (\varepsilon:q',w'w)$

where $E \in X$, $w,w',w_1, \ldots ,w_{r(f)} \in X^*$.

□

Theorem 3: For every PDTA M, there exists a restricted PDTA M' such that $N(M)=N(M')$.

To prove this theorem, one would expect to simply use the fact that, if L is context-free, then the set of branches of L is a context-free language, hence is recognized by a pushdown automaton. Then one could construct the corresponding PDTA (by "sticking" together productions of the pushdown automaton). Unfortunately this PDTA will recognize a context-free tree language L' which is usually strictly greater than L (except when L is "branch closed", see A. Saoudi, mémoire de DEA Paris 7). We thus have to find a direct construction. It will be inspired from Gallier's paper /G/; for lack of space, we sketch this construction without actually proving it.

We first give the idea of the construction. We have to show that every context-free tree language is an $N(M)$ for some restricted PDTA M. We can no more have a single state PDTA where the whole tree which remains to be derived is stored in the pushdown store. Hence we shall use both the state and the pushdown store to code (or "remember") the derivations which remain to be done: i.e., the state remembers at which occurrence we are currently located in the right-hand side trees at the present moment of the derivation, and the pushdown store remembers which occurrences of variable function symbols still have to be derived. The state and the pushdown store are then used interactively to reconstruct the derivation of a tree.

Let G: $G_i(v_1,\ldots,v_{r(G_i)}) = T_i = \{ t_i^j \ , \ j=1,\ldots,n_i \ \} \quad i=1,\ldots,n$

be a context-free tree grammar with an axiom having the form $G_1(a_1,\ldots,a_{r(G_1)})$ (this may be supposed without loss of generality). Then M defined as follows is such that $L(G)=N(M)$:

$Q = \{q_0\} \cup \{(i,j,o) \ / \ i=1,\ldots,n \ , \ j=1,\ldots,n_i$ and o is an occurrence in $t_i^j \}$

The input alphabet F is the terminal alphabet of G

$X = \{(i,j,o) \ / \ i=1,\ldots,n \ , \ j=1,\ldots,n_i$ and o is an occurrence of a $G_k \in \Phi$ in $t_i^j\} \cup \{ \not\$ \}$

$q_0=q_0$, $Z_0=\not\$$ and the moves are defined by:

- initializations: $(\varepsilon:q_0,\not\$) \vdash (\varepsilon:(1,j,\varepsilon),\not\$) \quad \forall j=1,\ldots,n_1$
- to each occurrence o of a base function symbol f in a t_i^j corresponds a read move: $\forall E \in X$ $\quad f:(i,j,o)$ (with children $v_1 \ldots v_{r(f)}$) , $E \vdash$ f (with children $(v_1:(i,j,o1),E) \ldots (v_{r(f)}:(i,j,or(f)),E)$)

  Intuitively, reading f in a state corresponding to the position o in the tree t_i^j, we have to go down in the input tree without changing the store.
- to each occurrence o of a variable function symbol G_k in a t_i^j (i.e. $t_i^j(o)=G_k$), correspond push moves where we store G_k in the pushdown, thus remembering the recursive call which shall be done later, and reposition ourselves in a state which corresponds to beginning the derivation of G_k (i.e. at the roots of the right-hand sides $t_k^{j'}$ corresponding to G_k):

  $\forall E \in X \quad (\varepsilon:(i,j,o),E) \vdash (\varepsilon:(k,j',\varepsilon),(i,j,o)E) \quad j'=1,\ldots,n_k$
- to each occurrence of a variable v_m indicating that the current recursive call has been completed, corresponds a pop move, i.e. going to the next recursive call and repositioning (by means of the state) to the v_m argument of each occurrence of the popped symbol in the t_i^j's :

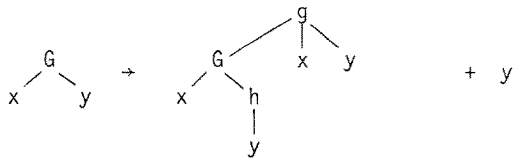
  $\forall w \in X^* \quad (\varepsilon:(i,j,o),(i',j',o')w) \vdash (\varepsilon:(i',j',o'm),w)$ for any (i,j,o), (i',j',o') such that $t_i^j(o)=v_m \in V$ and $t_{i'}^{j'}(o')= G_i \in \Phi$
- for each leaf of the axiom, a popping of the bottom marker $\not\$$ together with a checking of the leaf labels:

  $(a_m:(i,j,o),\not\$) \vdash (\varepsilon:(i,j,o),\varepsilon)$ for any (i,j,o) such that $t_i^j(o)=v_m$.

□

Let us apply this construction to the following simple example :

Example 4: Let G be the following grammar:

G : 

Since $n=1$ and $n_1=2$, we shall omit the components i,j in the states ($(1,1,\varepsilon)$ is denoted by ε and $(1,2,\varepsilon)$ is denoted by ε'); since there is but one occurrence of G in the t_i^j's we simplify the pushdown alphabet X into $\{G,\$\}$.

The moves of M are then defined as follows:

- $(\varepsilon:q_0,\$) \vdash (\varepsilon:\varepsilon,\$) + (\varepsilon:\varepsilon',\$)$
- $\left(\begin{array}{c} g:\varepsilon \\ x\ \ y\ \ z \end{array}, E\right) \vdash \begin{array}{c} g \\ (x:1,E)\ \ (y:2,E)\ \ (z:3,E) \end{array}$ $\quad \forall\, E \in X$

  $\begin{array}{c} (h:12,E) \\ | \\ x \end{array} \vdash \begin{array}{c} h \\ | \\ (x:121,E) \end{array}$ $\quad \forall\, E \in X$
- $(\varepsilon:1,E) \vdash (\varepsilon:\varepsilon,GE)$ $\quad \forall\, E \in X$

  $(\varepsilon:1,E) \vdash (\varepsilon:\varepsilon',GE)$
- $(\varepsilon:11,G) \vdash (\varepsilon:11,\varepsilon)$ and $(\varepsilon:2,G) \vdash (\varepsilon:11,\varepsilon)$

  $(\varepsilon:121,G) \vdash (\varepsilon:12,\varepsilon)$, $(\varepsilon:3,G) \vdash (\varepsilon:12,\varepsilon)$ and $(\varepsilon:\varepsilon',G) \vdash (\varepsilon:12,\varepsilon)$
- $(a:11,\$) \vdash (\varepsilon:11,\varepsilon)$ and $(a:2,\$) \vdash (\varepsilon:2,\varepsilon)$

  $(b:121,\$) \vdash (\varepsilon:121,\varepsilon)$, $(b:3,\$) \vdash (\varepsilon:3,\varepsilon)$ and $(b:\varepsilon',\$) \vdash (\varepsilon:\varepsilon',\varepsilon)$.

□

The previous results can be extended to higher-type tree languages /D/ , and, to some extent, to pushdown tree-transducers /BD/; this will be the subject of a forthcoming paper /DG/. However, this last extension is by no means trivial, as can be guessed from the following proposition and example.

Let us say that a tree language is

- *real-time* iff it is an $N(M)$ for some M without ε-moves
- *deterministic* iff it is an $N(M)$ for some deterministic M.

Proposition 2: The intersection of deterministic real-time languages may be non context-free.

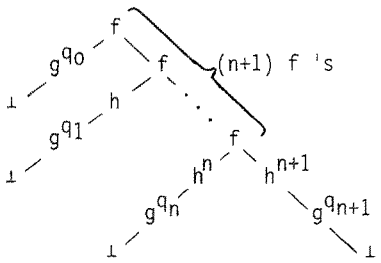
This is shown by the following example:

Let M_1 be defined by: $Q=\{q,\bar{q}\}$, $F=\{\perp,f,g,h\}$, $X=\{E,\$\}$, $Z_0=\$$, and:

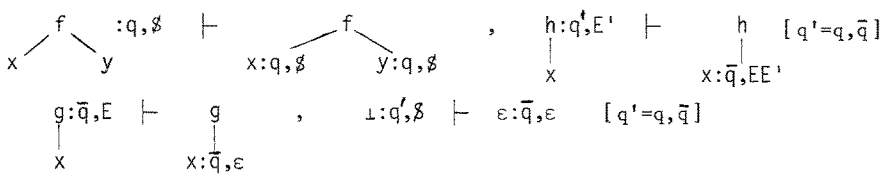
$\forall E' \in X$: $\left(\begin{array}{c} f \\ x\ \ y \end{array}:q,E'\right) \vdash \begin{array}{c} f \\ x:\bar{q},E' \quad y:q,EE' \end{array}$

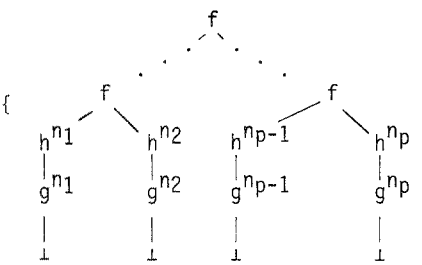
$\begin{array}{c} (h:q',E) \\ | \\ x \end{array} \vdash \begin{array}{c} h \\ | \\ x:\bar{q},\varepsilon \end{array}$ $[q'=q,\bar{q}]$, $\begin{array}{c} (g:\bar{q},\$) \\ | \\ x \end{array} \vdash \begin{array}{c} g \\ | \\ x:\bar{q},\$ \end{array}$

$\perp:\bar{q},\not{s} \vdash \varepsilon:\bar{q},\varepsilon$

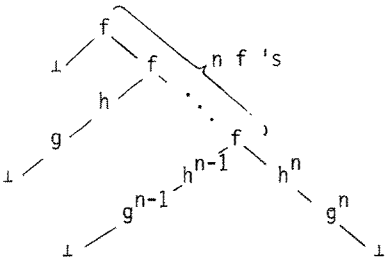
Then, $N(M_1) = \{$  $/ \quad n,q_0,\ldots,q_{n+1} \in \mathbb{N} \}$

Let M_2 be defined by: Q , F , X , Z_0 are as in M_1 and the moves are given by:

$\forall E' \in X$  $[q'=q,\bar{q}]$

Then, $N(M_2) = \{$  $/ \quad n_1,\ldots,n_p \in \mathbb{N} \}$

and, clearly:

$N(M_1) \cap N(M_2) = \{$  $/ \quad n \in \mathbb{N} \}$

which is non context-free (consider its branch language). □

Corollary: The intersection of deterministic, realtime, prefix-free context-free word languages may be non context-free. □

Acknowledgments: It is a pleasure to thank W. Damm for stimulating discussions, the (anonymous) referees, J. Engelfriet and H. Straubing for helpful comments.

REFERENCES

/A/ A.V. Aho, *Indexed grammars: an extension of the context free case*, JACM 15 (1968), 647-671.

/AD/ A. Arnold, M. Dauchet, *Transductions de forêts reconnaissables monadiques. Forêts corégulières*, RAIRO Inf. Théor. 10 (1976), 5-28.

/BD/ J. Bilstein, W. Damm, *Top-down tree transducers for infinite trees*, this Conference.

/B/ G. Boudol, *Langages algébriques d'arbres*, LITP Report, to appear.

/C/ B. Courcelle, *On jump deterministic pushdown automata*, MST 11 (1977), 87-109.

/D/ W. Damm, *The IO and OI hierarchies*, Thesis, Report n°41 (1980), RWTH Aachen.

/DG/ W. Damm, I. Guessarian, *Combining T and n*, submitted for publication.

/DO/ P. Downey, *Formal languages and recursion schemes*, Ph. D. Harvard (1974).

/E1/ J. Engelfriet, *Bottom-up and top-down tree transformations. A comparison*. MST 9 (1975), 198-231.

/E2/ J. Engelfriet, *Some open questions and recent results on tree transducers and tree languages*, Proc. Int. Symp. on For. Lang. Theor., Santa Barbara , Academic Press, to appear.

/ES/ J. Engelfriet, E.M. Schmidt, *IO and OI*, JCSS 15 (1977), 328-353 , and JCSS 16 (1978), 67-99.

/F/ M.J. Fischer, *Grammars with macro-like productions*, 9th SWAT (1968), 131-142.

/G/ J.H. Gallier, *Alternate proofs and new results about recursion schemes and DPDA's*, Report MS-CIS-80-7, Dpt of Comp. Sc., Univ. of Pennsylvania (1980), to appear.

/GU/ I. Guessarian, *Algebraic semantics*, Lect. Notes in Comp. Sc. n°99, Springer-Verlag, Berlin (1981).

/N/ M. Nivat, *On the interpretation of recursive polyadic program schemes*, Symp. Mat. 15, Rome (1975), 255-281.

/P/ N. Polian, *Langages infinis engendrés par les grammaires algébriques d'arbres*, Th. 3ème Cycle, in preparation, Poitiers (1981).

/RA/ M. Rabin, *Automata on infinite objects and Church's problem*, CBSM regional Conf. series in Math. n°13, AMS (1969).

/R/ W.C. Rounds, *Mappings and grammars on trees*, MST 4 (1970), 257-287.

/S/ J.M. Steyaert, *Lemmes d'itération pour les familles d'arbres*, Actes du Séminaire d'Inf. Théor. 1977-1978, LITP Report, Paris (1979).

/T/ J.W. Thatcher, *Tree automata: an informal survey*, in Currents in Theory of Comp., Aho (ed.), Prentice-Hall, London (1973).

****