

Bruno COURCELLE

IRIA-Domaine de Voluceau

78150 - Le Chesnay

France.

Abstract :

The equivalence problems for uninterpreted recursive program schemes and deterministic pushdown automata are reducible to each other. The equivalence class of a scheme is characterized by an infinite tree which is generated by the scheme as a language by a context-free grammar and which satisfies the equations of the system. Such trees are called algebraic. Roughly speaking, a tree is algebraic iff the set of its finite branches is a deterministic language. The interreducibility of the two equivalence problems for schemes and DPDA's follows.

Résumé :

Les problèmes de l'équivalence des schémas rékursifs non interprétés et des automates à pile déterministes sont réductibles l'un à l'autre. La classe d'équivalence d'un schéma est caractérisée par un arbre infini que le schéma engendre comme une grammaire algébrique (c'est-à-dire "context free") engendre un langage, et qui satisfait les équations du schéma.

Appelons algébriques de tels arbres. Grosso modo, un arbre est algébrique ssi l'ensemble de ses branches finies est un langage déterministe. L'interréductibilité des deux problèmes considérés en résulte.

Introduction :

We define recursive schemes and their semantics in §1 following Milner [6]. In §2 we consider a rewriting system associated with a scheme following Nivat [7] or Rosen [8] and state without proof that two schemes are equivalent iff they are tree-equivalent. In §3 we associate a strict deterministic grammar (Harrison and Havel [3]) with a scheme in order to represent by a language the tree it generates. The equivalence problem for schemes reduces to the DPDA problem. In §4 we prove the converse by introducing special strict deterministic grammars which are as powerful as the general ones and represent, up to an isomorphism, a proper subclass of the class of schemes. We conclude in §5 and relate this result to other, open or solved problems.

§1 Recursive schemes

Let F denote the set of base function symbols f, g, h, \dots . Each $f \in F$ is given together with an integer $\rho(f) \geq 1$ which is called the arity of f . Let $x_0, x_1, \dots, y, \dots$ denote individual variables and $V = \{x_0, x_1, \dots\}$. Let $M(F, V)$ be the set of finite terms written with symbols from $F \cup V$ and well-formed with respect to the arities. Example : $t = f(x, g(x, y), h(y, z))$ with $\rho(f) = 3$ and $\rho(g) = \rho(h) = 2$. Let now $\Phi = \{\phi_1, \phi_2, \dots, \phi_n\}$ be a finite set of (unknown) function symbols with arity $\rho(\phi_i) \geq 1$ for $1 \leq i \leq n$.

A recursive scheme (or scheme for short) on $F \cup \Phi$ is a system of n equations :

$$\Sigma \quad \left| \begin{array}{c} \vdots \\ \phi_i(x_1, \dots, x_{\rho(\phi_i)}) = t_i \\ \vdots \end{array} \right.$$

where for $1 \leq i \leq n$, $l_i = \rho(\phi_i)$, $t_i \in M(FU\Phi, V)$ and is of the form $f(t'_1, \dots, t'_k)$ for some $f \in F$. No other individual variables than x_1, \dots, x_{l_i} may appear in t_i .

We define now the semantics of our program schemes. Different notions of domains are available to us : discrete domains as in Nivat [7], complete lattices as in Scott [10], complete partial orders (for short c.p.o.) as in Milner [6]. We will choose this last possibility. For discussion about discrete domains see our conclusion. We prefer c.p.o. to complete lattices because they are much easier to work with. See also [11] for a discussion of this problem.

Definition : An interpretation for F is an object

$I = \langle D, \perp, \leq, \bar{f} \text{ for every } f \in F \rangle$ where

i) D is a set called the domain of I ,

ii) \leq is an order relation, \perp is the least element of D and \leq is complete i.e every countable chain

$A = \{a_1, a_2, \dots\}$ (i.e $a_1 \leq a_2 \leq \dots \leq a_n \dots$) of elements of D has a least upper bound denoted $\bigcup A$.

(Complete partial orders are such $\langle D, \perp, \leq \rangle$.)

iii) for $f \in F$, \bar{f} is a mapping $D^{\rho(f)} \rightarrow D$ which is continuous i.e $\bar{f}(\bigcup x_1, \dots, \bigcup x_n) = \bigcup \bar{f}(x_1, \dots, x_n)$. *

If D and D' are c.p.o's, we denote by $[D \rightarrow D']$ the set of continuous functions from D to D' . We order this set by $f \leq' f'$ iff $\forall d \in D, f(d) \leq' f'(d)$. We denote also by \perp' the function $\lambda d(\perp)$.

Lemma (Milner [6]) : If $\langle D, \leq, \perp \rangle$ and $\langle D', \leq', \perp' \rangle$ are c.p.o's, then $\langle [D \rightarrow D'], \leq', \perp' \rangle$ and $\langle D \times D', \leq \times \leq', (\perp, \perp') \rangle$ are c.p.o's.

Let I be an interpretation for a scheme Σ with unknown functions ϕ_1, \dots, ϕ_n . Let Δ be $[D^{\rho(\phi_1)} \rightarrow D] \times [D^{\rho(\phi_2)} \rightarrow D] \times \dots \times [D^{\rho(\phi_n)} \rightarrow D]$ where D is the domain of I and $k_i = \rho(\phi_i)$ for $1 \leq i \leq n$. Then Σ defines a mapping $\hat{\Sigma}$ from $\Delta \rightarrow \Delta$ which is continuous (see [6]) and has a least fix-point in Δ which is $\bigcup_{n \geq 0} \hat{\Sigma}^n(\perp_\Delta)$. For $1 \leq i \leq n$, we take $\phi_i^I = (\bigcup_{n \geq 0} \hat{\Sigma}^n(\perp_\Delta))_i \in [D^{k_i} \rightarrow D]$.

We sometimes call scheme a pair (Σ, ϕ) of a system Σ and a function ϕ defined by Σ .

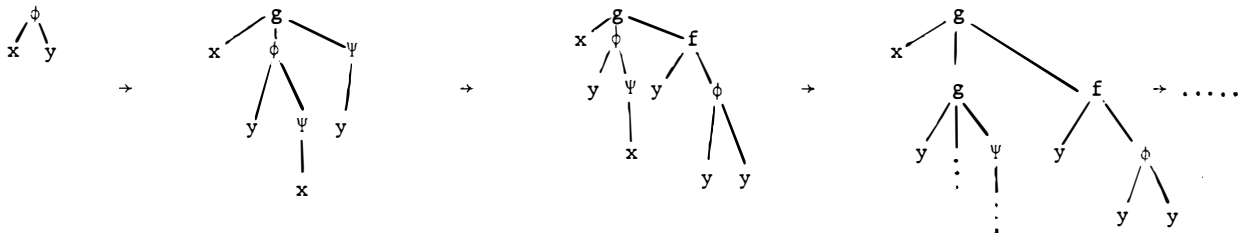
Two schemes (Σ, ϕ) and (Σ', ϕ') are equivalent if for every interpretation I , $\phi^I = \phi'^I$; we note this relation $\phi \equiv \phi'$, the reference to Σ and Σ' being understood.

\$2 Schemes as rewriting systems

We give now a characterization of the equivalence of schemes which will be more informative and manageable than the definition. We consider a scheme Σ as a rewriting system (this approach arises from [7], [2], and [8]). Take for instance (Σ, ϕ) where

$$\Sigma = \begin{cases} \phi(x, y) = g(x, \phi(y, \psi(x)), \psi(y)) \\ \psi(x) = f(x, \phi(x, x)) \end{cases}$$

Starting with $\phi(x, y)$ which we write as a tree : $\begin{array}{c} \phi \\ \swarrow \quad \searrow \\ x \quad y \end{array}$, and rewriting successively ϕ and ψ according to Σ , we get the following sequence of terms :



*) Note that such a function is monotone, i.e $d_1 \leq d'_1, \dots, d_k \leq d'_k$ implies $f(d_1, \dots, d_k) \leq f(d'_1, \dots, d'_k)$.

If we go on indefinitely, we get an infinite tree the nodes of which are labelled only by symbols of F . As substitutions can be performed in any order (see [7]), we get one and only one infinite tree. It can be formally defined as the least upper bound of the set of terms in $M(F \cup \Phi, V)$ which are generated from $\phi(x, y)$ by using Σ . We denote this tree by $T(\Sigma, \phi(x, y))$.

Theorem 1 : For schemes (Σ, ϕ) and (Σ', ϕ') , $\phi \equiv \phi'$ iff $T(\Sigma, \phi) = T(\Sigma', \phi')$.

A proof will appear in [2]. This theorem shows that schemes are equivalent iff they are tree-equivalent in the sense of Rosen ([8]).

We will always use trees informally. All what we will say about trees can be rigorously formalized and proved by using for instance Rosen's definitions of trees [9] (and this will be done in [2]).

We denote by T_n or $\overline{M}(F, \{x_1, \dots, x_n\})$ the set of infinite ranked trees of [9], i.e of trees such that i) each node is labelled by some $f \in F$ and has $\rho(f)$ sons and, ii) each leaf is labelled by some $x_i \in \{x_1, \dots, x_n\}$.

If t, t_1, \dots, t_n belong to $T_n, T_{n_1}, \dots, T_{n_m}$ respectively, we denote by $t(t_1, \dots, t_m)$ the element of T_M (where $M = \max \{n_1, \dots, n_m\}$) obtained by substituting t_i for each occurrence of x_i . It makes sense now to say that $\langle t_1, \dots, t_n \rangle$ belonging to $T_{k_1} \times T_{k_2} \times \dots \times T_{k_n}$ is a solution of a scheme $\Sigma = \langle \phi_i \mid 1 \leq i \leq n \rangle$ (with $k_i = \rho(\phi_i)$).

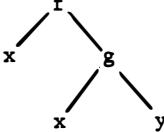
Theorem 2 : Σ has one and only one solution which is precisely $\langle T(\Sigma, \phi_1), \dots, T(\Sigma, \phi_n) \rangle$.

A tree which is a component of such a solution is called algebraic. Theorem 4 of [7] shows that $\langle T(\Sigma, \phi_1) \rangle$ is a solution of Σ . The unicity of the solution will be proved later.

§3 Algebraic trees and deterministic languages.

In order to decide whether two trees are equal, we will represent a tree by a language. (This coding has been used in [8]).

We begin by representing finite (ranked) trees (i.e elements of $M(F, V)$). We code each finite branch of a tree by a word which preserve information about the branching of nodes.

For instance, if $t =$  we take as set of branches of t , the set $B(t) = \{f_1 x, f_2 g_1 x, f_2 g_2 y\}$.

To any set F of function symbols we associate the alphabet $W(F) = \{f_i \mid f \in F \text{ and } 1 \leq i \leq \rho(f)\}$, and we define $B(t)$ inductively by :

- i) $B(x_j) = x_j$
- ii) $B(f(t_1, \dots, t_n)) = f_1 B(t_1) + f_2 B(t_2) + \dots + f_n B(t_n)$ (where $n = \rho(f)$ and $+$ denotes the union of languages).

Lemme 1 : For any t and t' , $t = t'$ iff $B(t) = B(t')$.

It is convenient to define a partition π^* of $W(F) \cup V$ by $a \equiv b \pmod{\pi}$ iff $a = b$ or $a = f_i$ and $b = f_j$ for some $f \in F$.

Lemma 2 : Let $L \subset (W(F) \cup V)^*$. There exists t such that $L = B(t)$ iff

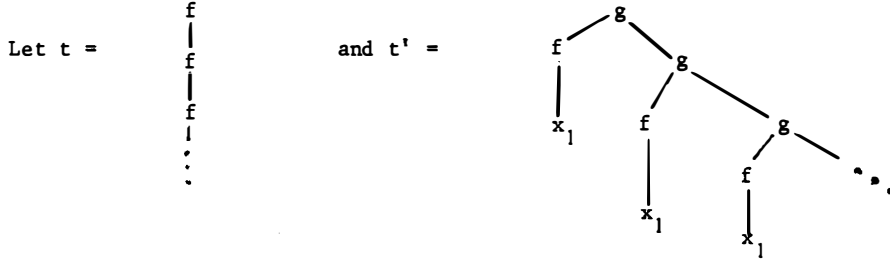
- i) $L \subset W(F)^* V$ and L is finite,
- ii) for any $\alpha\beta$ and $\alpha'\beta'$ in L , $\beta \neq 1$ implies $\stackrel{(1)}{\beta} \equiv \stackrel{(1)}{\beta'} \pmod{\pi}$ (and $\beta' \neq 1$),

Any partition of a set X can be considered as the equivalence relation: $a \equiv b \pmod{\pi}$ iff for some $\theta \in \pi$, a and $b \in \theta$. The symbol 1 will denote the empty word. For any $\beta \in X^*$ such that $\beta \neq 1$, we denote by $\stackrel{(1)}{\beta}$ the first symbol of β .

iii) for any $\alpha\beta$ in L and $b \equiv a \pmod{\pi}$, there exists some $\alpha\beta'$ in L .

Sketch of proof : Necessity clearly follows from our recursive definition of $B(t)$. One proves sufficiency by induction on $h(L)$ = the length of the longest word in L .

If t is an infinite but locally finite tree (i.e through any node passes at least one finite branch from root to a leaf) lemma 1 is still true. For arbitrary infinite trees we need something like the set of all prefixes of all branches. Let $Z(F) = \{f_0/f \in F\}$. For $t \in T_n$, let $A_0(t) \subset W(F)^*Z(F)$ be the set of αf_0 such that αf_1 is left factor of some (may be infinite) branch of t . Let $A(t) = A_0(t) \cup B(t)$ and $A_i(t) \subset W(F)$ for $1 \leq i \leq n$ such that $B(t) = \bigcup_{i=1}^n A_i(t)x_i$. Two examples will clarify these definitions :



belong to T_2 . Then $B(t) = \emptyset$ and $A(t) = A_0(t) = f_1^*f_0$. On the other hand, $A_1(t') = g_2^*g_1f_1$, $A_2(t') = \emptyset$, $B(t) = g_2^*g_1f_1x_1$ and $A_0(t) = g_2^*g_0 + g_2^*g_1f_0$. Hence $A(t) = g_2^*(g_1f_1x_1 + g_0 + g_1f_0)$.

Lemma 1' : For any t and t' , $t = t'$ iff $A(t) = A(t')$

By taking the partition π on $W(F) \cup Z(F) \cup V$ defined by $a \equiv b \pmod{\pi}$ iff $a = b$ or $a = f_i$ and $b = f_j$ and $Z(F) \cup V$ instead of V , the three conditions of lemma 2 characterize the languages $A(t)$. (We will refer to this result as lemma 2').

Lemma 3 : If $t \in T_n$ and $t_1, \dots, t_n \in T_m$, then $A(t(t_1, \dots, t_n)) = A_0(t) + A_1(t)A(t_1) + \dots + A_n(t)A(t_n)$.

This lemma allows us to construct a grammar which generates $A(\phi)$. We take first an example.

$$\text{Let } \Sigma \begin{cases} \phi(x_1, x_2) = f(x_1, \phi(x_2, \psi(x_1))) \\ \psi(x_1) = g(x_1, \phi(x_1, h(x_1))) \end{cases}$$

Let ϕ and ψ denote a solution of Σ . Let $\phi_i = A_i(\phi)$ for $i = 0, 1, 2$ and $\psi_i = A_i(\psi)$ for $i = 0, 1$. Then by using lemma 3 we get the following equations :

$$\begin{cases} \phi_0 + \phi_1 x_1 + \phi_2 x_2 = f_0 + f_1 x_1 + f_2 \phi_0 + f_2 \phi_1 x_2 + f_2 \phi_2 \psi_0 + f_2 \phi_2 \psi_1 x_1 \\ \psi_0 + \psi_1 x_1 = g_0 + g_1 x_1 + g_2 \phi_0 + g_2 \phi_1 x_1 + g_2 \phi_2 h_0 + g_2 \phi_2 h_1 x_1 \end{cases}$$

According to the definitions of $A_i(\phi)$ and $A_i(\psi)$, they split :

$$G(\Sigma) \begin{cases} \phi_0 = f_0 + f_2 \phi_0 + f_2 \phi_2 \phi_0 \\ \phi_1 = f_1 + f_2 \phi_2 \psi_1 \\ \phi_2 = f_2 \phi_1 \\ \psi_0 = g_0 + g_2 \phi_0 + g_2 \phi_2 h_0 \\ \psi_1 = g_1 + g_2 \phi_1 + g_2 \phi_2 h_1 \end{cases}$$

We know that such a system has one and only one solution in $(W(F) \cup Z(F))^*$, which consists of the languages generated by $\phi_0, \phi_1, \phi_2, \psi_0, \psi_1$ and production rules

$$\{\phi_0 \rightarrow f_0, \phi_0 \rightarrow f_2 \phi_0, \phi_0 \rightarrow f_2 \phi_2 \phi_0, \phi_1 \rightarrow f_1, \dots\}$$

From now on, a context-free grammar will be considered either as a set of production rules $\{S_1 \rightarrow m_{11}, S_1 \rightarrow m_{12}, S_2 \rightarrow m_{21}, \dots\}$ or as a system of equations $\{S_1 = m_{11} + m_{12}, S_2 = m_{21} + \dots, \dots\}$. We define now $G(\Sigma, \phi)$ in the general case.

Let $\Sigma = \langle \phi_i(x_1, \dots, x_{k_i}) = t_i, 1 \leq i \leq n \rangle$ be a scheme and $\phi = \{\phi_1, \dots, \phi_n\}$. Take $N = Z(\phi) \cup W(\phi)$ as set of non-terminal symbols, $Z(F) \cup W(F) \cup \{x_1, \dots, x_M\}$ (where $M = \max\{k_i\}$) as terminal alphabet. Define now $G(\Sigma)$ as the set of equations $\phi_{i,j} = \Sigma A_j(t_i)$ for $1 \leq i \leq n$ and $0 \leq j \leq k_i = \rho(\phi_i)$. Take as axiom S with equation $S = \Sigma A(\phi_{i_0}(x_1, \dots, x_{k_{i_0}}))$. Then $A(T(\Sigma, \phi_{i_0}(x_1, \dots, x_{k_{i_0}}))) = L(G(\Sigma), S)$.

From the unicity of the solution of a context-free grammar and Lemma 2', the second part of theorem 2 is proved. As we are interested in the equivalence problem, we will investigate now more closely the grammars which are associated to schemes. We will use the notion of strict deterministic grammar introduced by Harrison and Havel [3]. Consider the following properties of a context-free grammar $G(N, X, S)$ given with a partition π of $X \cup N$:

- A) If $\theta \in \pi$, then $\theta \subset X$ or $\theta \subset N$.
- B) $\{S\} \in \pi$
- C) G is reduced and ϵ -free,
- D) for any two rules $S \rightarrow \alpha\beta$ and $S' \rightarrow \alpha'\beta'$ such that $S \equiv S' \pmod{\pi}$, then
 - i) either $\beta = 1$, $\beta' = 1$ and $S = S'$,
 - ii) or $\beta \neq 1$, $\beta' \neq 1$ and $(1)_\beta \equiv (1)_{\beta'} \pmod{\pi}$.
- E) for any rule $S \rightarrow \alpha T \beta$ and $T' \in X \cup N$ such that $T \equiv T' \pmod{\pi}$, there exists a rule $S' \rightarrow \alpha T' \beta'$ such that $S \equiv S' \pmod{\pi}$.

Definition: G is strict deterministic for π (SD for short) iff it satisfies A, D and $X \in \pi$. G is super strict deterministic for π (SSD for short) iff it satisfies conditions A to E. Harrison and Havel only introduced SD grammars. From this definition, it is clear that a SSD grammar is SD. We will prove later that SSD grammars are as powerful as SD grammars.

Lemma 4 (Harrison [4]): A SD grammar is LR(o)

Theorem 3: Given (Σ, ϕ) and (Σ', ϕ') one can construct LR(o) grammars G and G' such that $\phi \equiv \phi'$ iff $L(G) = L(G')$.

Proof: From Lemma 2', $G(\Sigma)$ and $G'(\Sigma')$ are easily seen to be SD for the partition π of the set of nonterminals defined by $\phi_{i,j} \equiv \phi'_{i',j'} \pmod{\pi}$ iff $i = i'$. They are LR(o) from Lemma 4 and $\phi \equiv \phi'$ iff $L(G) = L(G')$ from Lemma 1'.

We introduce now a class of schemes for which the grammar $G(\Sigma)$ can be replaced by a much simpler one.

Definition: A **scheme** (Σ, ψ) is acceptable if for any $\phi \in \psi$, there exists $1 \leq i \leq \rho(\phi)$ such that $A_i(\phi) \neq \emptyset$.

Lemma 5: If (Σ, ϕ) and (Σ', ϕ') are acceptable, then $\phi \equiv \phi'$ iff $B(\phi) = B(\phi')$.

Sketch of proof: $\phi \equiv \phi'$ implies $B(\phi) = B(\phi')$. As Σ is acceptable, for any $\psi \in \phi$, $A(\psi) = \{\alpha f / \alpha \in W(F)^*, f \in F \text{ and } \alpha f_1 \in B(\psi)\}$. Hence $B(\phi) = B(\phi')$ implies $A(\phi) = A(\phi')$ and $\phi \equiv \phi'$ from Lemma 1. QED.

In fact, acceptable schemes generate locally finite trees for which lemma 1 is still true.

Given (Σ, ϕ_1) one constructs now a grammar $G'(\Sigma)$ which will generate $B(\phi_1)$. Let $\Sigma = \langle \phi_i(x_1, \dots, x_{k_i}) = t_i, 1 \leq i \leq n \rangle$. We take $W(\phi)$ as nonterminal alphabet and $W(F) \cup \{x_1, \dots, x_M\}$ as terminal one. $G'(\Sigma)$ consists of the set of equations $\phi_{i,j} = \Sigma A_j(t_i)$ for $1 \leq i \leq n$ and $1 \leq j \leq k_i$ and $S' = \sum_{1 \leq i \leq k_1} \phi_{1,i} x_i$. From Lemma 3, $L(G'(\Sigma), S') = B(\phi_1)$.

★

For any set of words $A = \{m_1, \dots, m_k\}$, ΣA means $m_1 + \dots + m_k$.

Take for example the same scheme Σ as before. Then

$$G'(\Sigma) = \begin{cases} \phi_1 = f_1 + f_2 \phi_2 \psi_1 \\ \phi_2 = f_2 \phi_1 \\ \psi_1 = g_1 + g_2 \phi_1 + g_2 \phi_2 h_1. \end{cases}$$

The language $B(\phi)$ is generated by S' with $S' = \phi_1 x + \phi_2 y$.

If Σ is acceptable and each ϕ is monadic, then $G'(\Sigma)$ is a simple deterministic grammar, and the equivalence problem is decidable. This case has been already studied in [1].

We have introduced SSD grammars in order to characterize grammars of the form $G'(\Sigma)$ up to an isomorphism. Let π be the partition on $W(\phi) \cup W(F) \cup \{x_1, \dots, x_M\}$ defined by $a \equiv b \pmod{\pi}$ iff $a = b$ or $a = f_i$ and $b = f_j$ for some $f \in F$ or $a = \phi_{k,i}$ and $b = \phi_{k,j}$ for some $\phi_k \in \phi$.

Conversely, if π is a partition of a finite set X , we define a set of base function symbols $F_{\pi, X} = \{\theta \in \pi \text{ with } \rho(\theta) = |\theta|\}$.

We can now state :

Lemma 6 : For any acceptable scheme (Σ, ϕ) , $G'(\Sigma)$ is SSD for partition π . Conversely, if $G(N, X, S)$ is SSD for π , one can construct (Σ, ϕ) on base function set $F_{\pi, X}$ with $\rho(\phi) = 1$ such that $L(G, S) = A_1(\phi)$.

Proof : The first part is clear from the definition of $G'(\Sigma)$, lemma 2 and the definition of SSD grammars. Conversely, given $G(N, X, S)$, one takes $\phi = \{\phi \in \pi / \phi \subset N\}$ and $\rho(\phi) = |\phi|$. From lemma 2 and the fact that $G(N, X, S)$ is SSD one easily constructs (Σ, ϕ_1) such that $G'(\Sigma) = G(N, X, S)$ with $\phi_1 = \{S\}$.

Example : Let $N = \{R, R', T, T', S\}$, $X = \{a, a', b, b', c\}$ and $G(N, X, S)$ be the grammar :

$$\begin{cases} S = aT + aT' + a'R + a'R'c \\ R = a + a'Ra'T' + a'R'c \\ R' = a'Ra + a'Ra'T \\ T = bR + bR' + b'a \\ T' = b'a'S. \end{cases}$$

and $\pi = \{\{R, R'\}, \{T, T'\}, \{S\}, \{a, a'\}, \{b, b'\}, \{c\}\}$.

One easily sees that $G(N, X, S)$ is SSD for π . In order to construct Σ , we take $\phi = \{\phi_S, \phi_{RR'}, \phi_{TT'}\}$ and $F = \{f_{aa}, f_{bb}, f_c\}$. Let now (Σ, ϕ_S) be the scheme :

$$\Sigma \begin{cases} \phi_S(x_1) = f_{aa}(\phi_{TT'}(x_1, x_1), \phi_{RR'}(x_1, f_c(x_1))) \\ \phi_{RR'}(x_1, x_2) = f_{aa}(x_1, \phi_{RR'}(f_{aa}(x_2, \phi_{TT'}(x_2, x_1)), f_c(x_1))) \\ \phi_{TT'}(x_1, x_2) = f_{bb}(\phi_{RR'}(x_1, x_1), f_{aa}(x_1, \phi_S(x_2))) \end{cases}$$

If we take $R = \phi_{RR',1}$, $R' = \phi_{RR',2}$, ... $a = f_{aa,1}$, ... etc then $G(N, X, S) = G'(\Sigma)$.

§4 DPDA problem reduces to the equivalence problem for schemes.

Let us first recall the following theorem (see [3] and [4] for the proof).

Theorem : Each problem of the following list reduces to the other ones :

- 1) Equivalence of SD grammars,
- 2) equivalence of LR(k) grammars for each k,
- 3) $N(A) = N(A')$ for any two DPDA's,
- 4) $T(A) = T(A')$ for any two DPDA's.

As these problems are equivalent (i.e. irreducible) we can call any one of them the DPDA problem. One does not know whether this problem is decidable or not, but it is interesting to know which problems are as difficult to solve. It may suggest new ways of attacking it. In the last section, we have proved that the equivalence for schemes reduces to the DPDA problem. We shall prove now the converse reduction which will follow from

Theorem 4 : For any two DPDA's A and A' , one can construct acceptable schemes (Σ, ϕ) and (Σ', ϕ') such that $N(A) = N(A')$ iff $\phi \equiv \phi'$.

We use a three-step construction :

Step 1 : given A, A' we construct $\underline{A}, \underline{A}'$ such that $N(A) = N(A')$ iff $N(\underline{A}) = N(\underline{A}')$

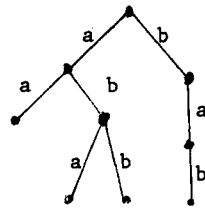
Step 2 : From \underline{A} we construct a SSD grammar G such that $L(G) = N(\underline{A})$.

Step 3 : From G we construct an acceptable scheme $(\Sigma, \phi(x_1))$ such that $A_1(\phi) = L(G)$ just by using lemma 6.

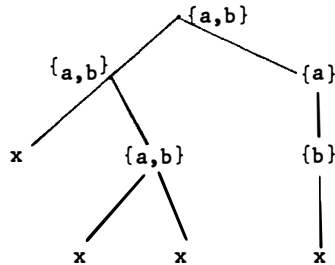
It follows from Lemma 5 that $\phi \equiv \phi'$ iff $N(A) = N(A')$ (if ϕ' is constructed in the same way from \underline{A}').

We describe now step one.

To each prefix-free language over some finite linearly ordered alphabet X we can associate a tree in a natural way : for example if $L = \{aa, aba, abb, bab\}$, then we get



Label each node with the set of labels of arcs going from it and leafs by x .



Take $F = \{\{a,b\}, \{a\}, \{b\}\}$ with $\rho(\{a,b\}) = 2$, $\rho(\{a\}) = \rho(\{b\}) = 1$. This tree is an element of $M(F, \{x\})$.

If L is infinite we get an element $T(L)$ of T_1 . It is locally finite and hence, characterized by $\underline{L} = A_1(T(L))$ which is not exactly L . If $L = N(\underline{A})$, we shall construct \underline{A} in such a way that $N(\underline{A}) = \underline{L}$. The second and third step will produce (Σ, ϕ) such that $T(\Sigma, \phi) = T(L)$.

We define now formally the language \underline{L} . Let X be a finite alphabet. An arbitrary but fixed linear ordering of X will be given. Let $\underline{X} = \{(\alpha, a) / a \in \alpha \subset X\}$.

If $L \subset X^*$, we define $\sigma_L : X^* \rightarrow 2^{\underline{X}}$ by $\sigma_L(w) = \{a \in \underline{X} / \exists w' \in X^*, waw' \in L\}$. To every word $w = a_1 \dots a_n$ in L , we associate $\underline{w} = b_1 \dots b_n$ in \underline{X}^* such that for $1 \leq j \leq n$, $b_j = (a_j, \sigma_L(a_1 \dots a_{j-1}))$. Going back to the example, $\underline{aba} = (a, \{a,b\})(b, \{a,b\})(a, \{a,b\})$ and $\underline{bab} = (b, \{a,b\})(a, \{a\})(b, \{b\})$.

Let $\underline{L} = \{\underline{w} \in \underline{X}^* / w \in L\}$.

Let now $F_X = \{\alpha \subset X / \alpha \neq \emptyset\}$ and $\rho(\alpha) = |\alpha|$. Let μ be the one-one mapping from $W(F_X)$ to \underline{X} such that $\mu(\alpha_i) = (a, \alpha)$ iff a is the i^{th} element of α according to the ordering of X .

It is now clear that $\underline{L} = \mu(A_1(T(L)))$. From now on we omit the mapping μ .

In order to do the first step of our proof, we recall some definitions and conventions about DPDA's.

A DPDA in an object $A = \langle X, K, \Gamma, \delta, q^0, Z^0, F \rangle$ (input alphabet, states, pushdown symbols, transition function, initial state, initial pushdown symbol, final states). The transition function δ is a partial function from $K \times (X \cup \{\epsilon\}) \times \Gamma^*$ to $K \times \Gamma^*$ such that for $a \in X$, $\delta(q, a, Z)$ and $\delta(q, \epsilon, Z)$ are not together defined. A configuration of A is an element of $K \times X^* \times \Gamma^*$; an initial configuration is any of the form (q^0, u, z^0) ; in a current configuration (q, u, Zm) we write the top symbol of the stack on the left and call (q, Z) the mode of the configuration; we call (q, Z) a reading mode (ϵ -mode) iff $\delta(q, a, Z)$ is defined for some $a \in X$ (for $a = \epsilon$). Between configurations, we define the relation $c \vdash c'$ by $(q, u, Zm) \vdash (q', u', m')$ iff $\delta(q, a, Z) = (q', m')$, $m' = m''m$, $a \in X$ implies $u = au'$ and $a = \epsilon$ implies $u = u'$. We denote by \vdash^* the transitive and reflexive closure of \vdash .

The deterministic prefix free languages (PFDet for short) are the sets $N(A) = \{u \in X^* / \exists q \in K (q^0, u, Z^0) \vdash_A^* (q, 1, 1)\}$.

The deterministic languages are the sets $T(A) = \{u \in X^* / \exists q \in F \text{ and } m \in \Gamma^* (q^0, u, Z^0) \vdash_A^* (q, 1, m)\}$

Lemma 7 : Let $L = N(A_0) \subset X^*$ a PFDet language. One can construct a DPDA $A = \langle X, K, \Gamma, \delta, q^0, Z^0, q^f \rangle$ such that :

- i) $L = N(A)$,
- ii) if $\delta(q, \epsilon, Z)$ is defined, then $\delta(q, \epsilon, Z) = (q', 1)$ for some $q' \in K$,
- iii) if $(q^0, u, Z^0) \vdash^* (q, 1, m)$ there exists $v \in X^*$ such that $(q, v, m) \vdash^* (q^f, 1, 1)$.

Notice that if $m = 1$ in condition iii), we have $q = q^f$.

Proof : By adding a bottom stack symbol, one can build from A_0 a DPDA A_1 such that $(q^0, u, Z^0) \vdash^* (q, v, 1)$ implies $q = q^f$ and $N(A_1) = N(A_0)$. We construct now A_2 from A_1 by modifying the transition function δ of A_1 in such a way that $N(A_2) = N(A_1)$ and A_2 satisfies ii). For any ϵ -mode (q, Z) such that $\delta(q, \epsilon, Z) = (q_1, m)$ and $m \neq 1$

- either i) for some $q' \in K$, $(q, 1, Z) \vdash^* (q', 1, 1)$, then take $\delta(q, \epsilon, Z) = (q', 1)$,
- either ii) for some $q' \in K$ and $Z'm' \in \Gamma^*$ such that (q', Z') is a reading mode $(q, 1, Z) \vdash^* (q', 1, Z'm')$; then take $\delta(q, \epsilon, Z) = \emptyset$ and, for $a \in X$, $\delta(q, a, Z) = (q'', m''m')$ whenever $\delta(q', a, Z') = (q'', m'')$,
- iii) neither of i) and ii) occur, then put $\delta(q, \epsilon, Z) = \emptyset$.

(Note that one can decide which of these cases occur. See [5]).

We construct now A from A_2 such that $N(A) = N(A_2)$, ii) is still true and iii) is satisfied; we do this by extending the pushdown alphabet in such a way that we know just by looking at the mode whether the stack can be emptied or not.

Formally, we take $2^K \times \Gamma \times 2^K$ as new pushdown alphabet and will define δ in such a way that any attainable configuration is of the form $(q_1, u, [Q_1, Z_1, Q_2][Q_2, Z_2, Q_3] \dots [Q_k, Z_k, \{q^f\}])$

where i) for $1 \leq i \leq k$, $Q_i \neq \emptyset$

- ii) $Q_1 = \{q \in K / \exists v \in X^* (q, v, Z_1 \dots Z_k) \vdash_{A_2}^* (q^f, 1, 1)\}$

For any $Z \in \Gamma$ and $Q \in 2^K$, define $F(Z, Q) = \{q \in K / \exists u \in X^* \exists q' \in Q, (q, u, Z) \vdash_{A_2}^* (q', 1, 1)\}$.

This function is computable. See [5]. We define the transition function δ' of A in terms of δ , the one of A_2 . For $a \in X \cup \{\epsilon\}$ $\delta'(q, a, [Q, Z, Q']) = (q', 1)$ iff $\delta(q, a, Z) = (q', 1)$ and $q' \in Q'$. For $a \in X$, then $\delta'(q, a, [Q, Z, Q']) = (p, [Q_1, Z_1, Q_2][Q_2, Z_2, Q_3] \dots [Q_1, Z_1, Q'])$ iff $\delta(q, a, Z) = (p, Z_1 \dots Z_1)$, $p \in Q_1$, $Q_1 = F(Z_1, Q') \neq \emptyset$, and for $1 \leq i \leq l-1$, $Q_i = F(Z_i, Q_{i+1}) \neq \emptyset$.

We take $[Q_0, Z^0, \{q^f\}]$ as a new initial pushdown symbol with $Q_0 = F(Z^0, \{q^f\})$. It is easy to check that A satisfies the required conditions. QED.

Proposition 1 : Given a PFDet language L , one can construct a DPDA A such that $N(A) = L$ and A satisfies the conditions of Lemma 7.

Proof : Let $L = N(A)$ for some DPDA A satisfying lemma 7. We define $\underline{A} = \langle X, K, \Gamma, \underline{\delta}, q^0, Z^0, q^f \rangle$ from A by taking

- i) $\underline{\delta}(q, \epsilon, Z) = \delta(q, \epsilon, Z)$
- ii) $\underline{\delta}(q, (a, \alpha), Z) = \delta(q, a, Z)$ iff $\alpha = \alpha_{q, Z} = \{b \in X / \delta(q, b, Z) \text{ is defined}\}$,
- iii) $\underline{\delta}$ is undefined in all other cases.

Observation 1 : If $(q^0, u, Z^0) \xrightarrow{*}_A (q, l, m)$ and (q, Z) is a reading mode, then $\alpha_{q, Z} = \sigma_L(u)$. This is true from the fact that A satisfies condition iii of lemma 7.

We can prove now that $\underline{L} = N(\underline{A})$. Let $w = a_1 \dots a_l$ belong to $N(A)$ and $(c_j)_{0 \leq j \leq l'}$ be the corresponding computation of A . Let $\underline{w} = b_1 \dots b_{l'}$. If $c_j = (q, a_1 \dots a_l, m)$ define $\underline{c}_j = (q, b_1 \dots b_{l'}, m)$. We prove by induction on j that for $0 \leq j \leq l'$, $\underline{c}_0 \xrightarrow{*}_{\underline{A}} \underline{c}_j$.

It is clear for $j = 0$. If $c_j \xrightarrow{\epsilon}_A c_{j+1}$ is an ϵ -transition then $\underline{c}_j \xrightarrow{\epsilon}_{\underline{A}} \underline{c}_{j+1}$ is an ϵ -transition of \underline{A} . If not, $c_j = (q, au, m)$ and $c_{j+1} = (q', u, m')$ with $w = vau$ for some $v \in X^*$. Then $\underline{c}_j = (q, (a, \alpha)u', m)$ with $\alpha = \sigma_L(v)$. From observation 1 and the definition of \underline{A} we get $\underline{c}_j \xrightarrow{\epsilon}_{\underline{A}} \underline{c}_{j+1} = (q', u', m')$. Hence $\underline{L} \subseteq N(\underline{A})$.

Now let $b_1 \dots b_{l'}$ be in $N(\underline{A})$ and $(c'_j)_{0 \leq j \leq l'}$ be the corresponding \underline{A} -computation. Let $c_j = (q, a_1 \dots a_l, m)$, if $c'_j = (q, b_1 \dots b_{l'}, m)$ and a_i is the first component of b_i , for $i \leq i' \leq l'$. By induction on j , one sees that $c_0 \xrightarrow{*}_A c_j$ and for $0 \leq i \leq l$, $a_i = (b_i, \sigma_L(a_1 \dots a_{i-1}))$ that is $b_1 \dots b_{l'} = \underline{a_1 \dots a_l}$ and $N(\underline{A}) \subseteq \underline{L}$. QED.

The same proof shows that if L is simple deterministic (regular) then \underline{L} is simple deterministic (regular). But if $L = \{w^c w \mid w \in \{a, b\}^*\}$ which is context-free, prefix-free nondeterministic, then \underline{L} is not context-free.

We begin now the second step of the proof of theorem 4.

We define on $Y = \underline{X}$ a partition π by the following equivalence relation : $(a, \alpha) \equiv (b, \beta)$ iff $\alpha = \beta$.

Proposition 2 : Let L be a PFDet language. One can construct a SSD grammar G for the partition π of X such that $L(G) = \underline{L}$.

Proof : One constructs G from the automaton $B = \underline{A} = \langle Y, K, \Gamma, \delta, q^0, Z^0, q^f \rangle$ obtained in Proposition 1. Note that B satisfies the three conditions of lemma 7 and the following one

iv) $\forall a, b \in Y, \delta(q, a, Z)$ and $\delta(q, b, Z)$ are both defined iff $a \equiv b \pmod{\pi}$.

Let G_0 be the canonical grammar of B (see [3]) which is intended to generate $N(B)$ by taking

- 1) Y as terminal alphabet,
- 2) $N_0 = K \times \Gamma \times K$ as non-terminal alphabet (we denote its elements by $\overline{qZq'}$),
- 3) $q^0 Z^0 q^f$ as axiom,
- 4) a set P_0 of production rules consisting of

$\overline{qZq'} \rightarrow a$ whenever $\delta(q, a, Z) = (q', l)$,

$\overline{qZq'} \rightarrow l$ whenever $\delta(q, \epsilon, Z) = (q', l)$,

$\overline{qZq'} \rightarrow a \overline{pZ_1 q_1} \overline{q_1 Z_2 q_2} \dots \overline{q_{k-1} Z_k q'}$ whenever $\delta(q, a, Z) = (p, Z_1 \dots Z_k)$ and $q_1, \dots, q_{k-1} \in K$.

Let $f : N_0^* \rightarrow \Gamma^*$ be the homomorphism defined by $f(\overline{qZq'}) = Z$. Let $H(p, q) = \{\overline{pZ_1 q_1} \overline{q_1 Z_2 q_2} \dots \overline{q_k Z_{k+1} q'} \mid k \geq 0, Z_i \in \Gamma \text{ and } q_i \in K\}$. The following lemma shows how G_0 simulates B .

Lemma 8 : For any $u \in X^*$, $q, q' \in K$, $Z \in \Gamma$ and $\alpha \in N_0^+$, then

- i) $\overline{qZq'} \xrightarrow{*}_L u \alpha$ iff $\exists p \in K$ such that $\alpha \in H(p, q')$ and $(q, u, Z) \xrightarrow{*} (p, l, f(\alpha))$ ($\xrightarrow{*}_L$ means left-most derivation)
- ii) $\overline{qZq'} \xrightarrow{*}_L u$ iff $(q, u, Z) \xrightarrow{*} (q', l, l)$.

The reader will find a proof of this lemma in Harrison and Havel [3], lemma 3.3. From this lemma we get $N(B) = \underline{L} = L(G_0)$.

Observation 2 : Given a configuration (q, u, m) , there exists at most one configuration $(q', l, Z'm')$ such that $(q, u, m) \xrightarrow{*} (q', l, Z'm')$ and (q', Z') is a reading mode.

Observation 3 : If $\overline{qZq'} \rightarrow l$, there is no other production of form $\overline{qZq'} \rightarrow m$ in G_0 .

We now transform G_0 into a reduced ϵ -free grammar G :

first step : for each $\overline{qZq'}$, if $L(G_0, \overline{qZq'}) = \emptyset$ delete from G_0 any production which contains $\overline{qZq'}$.

second step : for each $\overline{qZq'}$, if for no u and $v \in X^*$ $S_0 \xrightarrow{*}_{G_0} u \overline{qZq'} v$, delete from G_0 any production which contains $\overline{qZq'}$.

third step : for each $\overline{qZq'}$ such that $\overline{qZq'} \rightarrow l$, delete it from G_0 and replace $\overline{qZq'}$ by l in right hand sides of rules of G_0 .

Now, G is a reduced grammar in Greibach Normal Form which is equivalent to G_0 . Let $N \subset N_0$ be the set of its non terminals.

Let π the partition of N defined by the equivalence relation $\overline{pZp'} \equiv \overline{qRq'}$ iff $p = q$ and $Z = R$. We have already defined the partition π on Y and, we just have to prove that $G(N, Y, S_0)$ is SSD for π .

From observation 3 and the construction of G , G clearly satisfies conditions A and C of the definition of SSD grammars. From lemma 8 ii) and the fact that B satisfies condition iii) of lemma 7, $\overline{q^0 Z^0 q^f} \xrightarrow{G} u$ implies $q = q^f$. It means that $\{\overline{q^0 Z^0 q^f}\}$ is a class of π . Hence condition B is satisfied. We check now condition D : let $\overline{qZq'} \rightarrow \alpha\beta$ and $\overline{qZq''} \rightarrow \alpha\beta'$ belong to P . If $\alpha = 1$ and $\beta = 1$, then $q' = q''$ and $\beta' = 1$ from observation 3. If $\alpha = 1$ and $\beta \neq 1$, then $\beta' \neq 1$ (from the preceding case) and β and β' begin with conjugate terminal symbols (from condition iv) on B and the construction of G). Assume now that $\alpha \neq 1$. Let $u \in Y^*$ such that $\alpha_A^* u$ and $\beta = \overline{pZ'p'} \partial$. Then from lemma 8, $(q, u, Z) \vdash^* (p, 1, Z'm')$ for some $m' \in \Gamma^*$. As (p, Z') is not an ϵ -mode (since $\overline{pZ'p'} \in N$) β' cannot be 1, else one would have $(q, u, Z) \vdash^* (q'', 1, 1)$ contradicting observation 2. Then $\beta' = \overline{rZ''r'} \partial'$ but the same argument yields $(q, u, Z) \vdash^* (r, 1, Z''m'')$. As (r, Z'') is not an ϵ -mode and using observation 2 we get $r = p$ and $Z' = Z''$, which means $\binom{1}{1} \beta = \binom{1}{1} \beta'$. If $\beta = 1$, β' must be 1 from the same argument ; lemma 8 yields $(q, u, Z) \vdash^* (q', 1, 1)$ and $(q, u, Z) \vdash^* (q'', 1, 1)$ and observation 2 gives $q' = q''$. QED.

We check now condition E :

Let $\overline{qZq'} \rightarrow \alpha\beta$ be in P and $T' \equiv T \pmod{\pi}$ with $T' \neq T$. Assume first that $\alpha \neq 1$ and take $T = \overline{rSr'}$, $T' = \overline{rSr''}$. The given production rule comes from some rule in P_0 of the form $\overline{qZq'} \rightarrow \alpha' \overline{rSr'} \beta'$ (where α' and β' have been reduced through step 3 of the reduction of G_0). Let $u \in Y^*$ be derivable from $\alpha' \overline{rSr'}$. Then using lemma 8 and the fact that G is reduced, one can find $v \in Y^*$, $m \in \Gamma^*$ such that $(q^0, vu, Z^0) \vdash^* (q, u, Zm) \vdash^* (r'', 1, Z_1 \dots Z_k m)$ with $f(\beta') = Z_1 \dots Z_k$. From condition iii) on B , there exists w such that $(r'', w, Z_1 \dots Z_k m) \vdash^* (q^f, 1, 1)$. Clearly, there exists r_0, r_1, \dots, r_{k+1} in K , w_1, \dots, w_k factors of w such that for $0 \leq i \leq k$, $(r_i, w_{i+1}, Z_{i+1}) \vdash^* (r_{i+1}, 1, 1)$ and $r_0 = r''$. Consider now the production rule $\overline{qZr_{k+1}} \rightarrow \alpha' \overline{rSr''} \overline{r''Z_1r_1} \overline{r_1Z_2r_2} \dots \overline{r_kZ_kr_{k+1}}$ of P_0 .

During the first step of the reduction of G_0 , it cannot be deleted since each of its nonterminals generates a non empty language (clear for those of α' and for $\overline{rSr''}$ from hypothesis, it follows from lemma 8 for the others). It is not deleted during the second step because $(q^0, vuw, Z^0) \vdash^* (q, uw, Zm) \vdash^* (r_{k+1}, w', m) \vdash^* (q^f, 1, 1)$ for some right factor w' of w and lemma 8 gives $\overline{q^0 Z^0 q^f} \xrightarrow{G_0} \overline{vqZr_{k+1}} w'$.

This production rule can be simplified during the third step of the reduction and becomes $\overline{qZr_{k+1}} \rightarrow \alpha' \overline{rSr''} \beta''$ which was to be shown. Assume now that $\alpha = 1$. Then $T = a$ and $T' = b$ is some terminal symbol conjugated with a . From condition iv) on B (see the beginning of this proof), there exists some transition $\delta(q, b, Z) = (p, Z_1 \dots Z_k)$.

The same argument as before shows that for some $v, w \in Y^*$, $m \in \Gamma^*$, $(q^0, vbw, Z^0) \vdash^* (q, bw, Zm) \vdash^* (p, w, Z_1 \dots Z_k m) \vdash^* (q^f, 1, 1)$. Hence, some production $\overline{qZq''} \rightarrow b\beta'$ belonging to P_0 has not been deleted during the two first steps of the reduction. It may have been simplified giving $\overline{qZq''} \rightarrow b\beta''$ as desired. QED.

We can sum up our results by stating

Theorem 5 : The following problems are equivalent to the DPDA problem :

- i) equivalence of superstrict deterministic grammars,
- ii) equivalence of schemes (that is tree-equivalence of [8]),
- iii) equivalence of acceptable schemes.

Remark : By using the method of §3, one could study the slightly more general class of schemes, such that $\rho(f)$ and $\rho(\phi_i)$ may be 0, and a scheme is any system $\langle \phi_i(x_1, \dots, x_{l_i}) = t_i, 1 \leq i \leq n \rangle$ such that for $1 \leq i \leq n$, $t_i \in M(F, \{x_1, \dots, x_{l_i}\})$ and $l_i = \rho(\phi_i)$.

Theorem 5 is true for this class of schemes.

§5 Conclusion

As program schemes are intended to represent real programs, it is of interest to interpret base functions as much as possible. But if we interpret too much, decision problems become undecidable. For instance if one decides to interpret base functions as partial mappings from D^k to D (for some domain-set D) and certain 3-adic base functions as conditional operators (that is $g(x,y,z) = \text{if } p(x) \text{ then } y \text{ else } z$), then the equivalence problem becomes undecidable. If one restricts to monadic such schemes (with monadic base functions), the equivalence problem is equivalent to the DPDA problem (E. Friedman [12]).

It is also possible to interpret a little more than we did by using discrete interpretations (see Nivat [7]). A discrete interpretation is an interpretation in our sense such that $\langle D, \leq, \perp \rangle$ satisfies $\forall x \forall y, x \leq y$ iff $x = y$ or $x = \perp$.

Let \equiv_d be the equivalence relation between schemes corresponding to discrete interpretations. As before, for schemes (Σ, ϕ) and (Σ', ϕ') , $T(\Sigma, \phi) = T(\Sigma', \phi')$ implies $\phi \equiv_d \phi'$ but the converse is not true and we cannot reduce the equivalence of schemes to the equivalence of grammars as we did. Other methods must be introduced.

Aknowlegments :

I express my gratitude to Gilles Kahn and Ronald Rivest for reading the manuscript.

References :

- [1] B. Courcelle, J. Vuillemin : Semantics and Axiomatics of a Simple Recursive Language. Proceedings of 6th Annual ACM Symposium on Computing, Seattle (1974) pp 13-26.
- [2] B. Courcelle, J. Vuillemin, M. Nivat : In preparation.
- [3] M. Harrison, I. Havel : Simple Deterministic Grammars, in J.C.S.S. (vol 7) n°3 June 1973 pp 237-277.
- [4] M. Harrison, I. Havel : On the Parsing of Deterministic Languages, submitted for publication.
- [5] J. Hopcroft, J. Ullman : Formal languages and their Relation to Automata, Addison-Wesley, 1969.
- [6] R. Milner : Models of LCF. Memo AIM/CS332 Stanford (1973).
- [7] M. Nivat : On the Interpretation of Recursive Polyadic Program Schemes, to appear in Atti del Convegno di Informatica Teorica, Roma, 1973.
- [8] B. Rosen : Program Equivalence and Context-free Grammars, (revised) IBM Report RC 4822, April 1974.
- [9] B. Rosen : Tree Manipulating Systems and Church Rosser Theorems, J.A.C.M. 20 (1973) pp 160-187.
- [10] D. Scott : Outline of a Mathematical Theory of Computation, Memo Oxford PRG-2 (1970)
- [11] J. Vuillemin : Syntaxe, Sémantique et Axiomatique d'un Langage de Programmation Simple, Thesis to appear.
- [12] E. Friedman : Relationships between Monadic Recursion Schemes and Deterministic Context-free Languages. This Symposium.