Theoretical Computer Science 6 (1978) 255–279.

A REPRESENTATION OF TREES BY LANGUAGES I

Bruno COURCELLE

IRIA, Domaine de Voluceau, Rocquencourt, 78150 Le Chesnay, France

Communicated by Robin Milner
Received March 1976
Revised September 1977

Abstract. A tree can be represented by a language consisting of a suitable coding of its finite branches. We investigate this representation and derive a number of reductions between certain equivalence problems for context-free tree grammars and recursive program schemes and the (open) equivalence problem for DPDA's. This is the first part of this work: it is devoted to technical results on prefix-free languages and strict deterministic grammars. Application to context-free tree grammars will be published in the second part.

0. Introduction

A finite *tree* can be considered as a generalized word and the theory of *tree languages* (i.e. of sets of finite trees) is an extension of the theory of languages. To be precise, a tree means here a well formed term on a finite ranked alphabet. Context-free tree languages have been already defined and investigated in several papers among which we quote: Fischer [13] (with a different terminology), Rounds [28], Engelfriet and Schmidt [12], Nivat [23], Arnold and Dauchet [1, 2].

On the other hand, the theory of recursive program schemes introduces infinite trees which can be visualized as "infinite wellformed terms" on a finite ranked alphabet and are defined by certain tree grammars called here *schematic* [26, 23, 8]. Such trees are called *algebraic*. By a result of [24, 8], two schemes are equivalent if and only if the corresponding infinite trees are equal. Hence we are interested in the equivalence problem for schematic grammars. Since this problem seems to be difficult to solve directly, we reduce it here to a decision problem for context-free grammars.

The purpose of this paper is to investigate a representation of trees and tree languages by languages and its application to reductions between decision problems for context-free tree grammars, schematic grammars and context-free grammars. Let us describe informally this representation (which was first introduced in [26]). A finite tree can be represented, i.e. is completely defined by the language consisting of its finite branches (appropriately coded as words on a certain alphabet). The same holds for certain infinite trees called *locally finite*. If a locally finite tree is

generated by a schematic grammar, the language of its finite branches is generated by a context-free grammar that can be constructed from the schematic grammar and is *strict deterministic* [18].

We obtain in particular the two following results:

(1) A locally finite tree is algebraic if and only if the language of its branches is deterministic (i.e. is a deterministic context-free language).

(2) The equivalence problems for schematic grammars and DPDA's are interreducible.

In order to prove the "if" part of (1) we characterize those strict deterministic grammars which are associated with schematic grammars. We call them *complete deterministic*, a refinement of Harrison and Havel's definition. The concept of a *complete language* is introduced in order to characterize the languages which are (up to an isomorphism) the language of branches of some locally finite tree.

About context-free tree grammars, we also obtain that the equivalence problems for simple deterministic tree grammars and DPDA's are interreducible. This result is of special interest since the equivalence problem for simple deterministic grammars is solvable [21, 4] as the equivalence problem for DPDA's is open.

The paper is divided into two parts. The first one consists of the present introduction, and preliminary results of language theory. Definitions and notations about languages, context-free grammars and deterministic pushdown-automata (DPDA's) are fixed in Section 1. The concept of a complete language is defined and studied in Section 2. Two ways of completing a given language are considered, which become effective when applied to a deterministic language. Section 3 deals with strict deterministic grammars and several refinements of the definition of [18] which will be useful for applications to tree grammars. The second part, to be published in the next issue of this journal, will use the material of the first one to study tree grammars. Context-free tree languages and their sets of branches will be defined and studied in Section 4, infinite algebraic trees, (associated with recursive program schemes) in Section 5.

The numbering of a theorem, definition etc., indicates the section where it stands (e.g. Theorem 3.14 in Section 3, hence in the first part of the paper).

This paper is based on the author's doctoral dissertation [6]. Certain of the results have been presented at the 15th Annual Symposium on Switching and Automata Theory, New Orleans, 1974 [5] and at the 3rd G.I. Conference on Theoretical Computer Science at Darmstadt, 1977.

1. Preliminaries

We fix notations for sets, languages, context-free grammars and DPDA's. Our definitions differ on some technical points from the classical ones.

We let Card(X) denote the cardinality of a set X and 2^X its power set. For a partial mapping $A: X \to Y$, $A(x)\downarrow$ abbreviates "$A(x)$ is defined" and $A(x)\uparrow$ "$A(x)$ is undefined". If n is a positive integer, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. We denote by $X - Y$ the set of elements of X which do not belong to Y.

A *partition* of a set X is a set π of pairwise disjoint non-empty subsets of X the union of which is X. The elements of π are called the *blocks* of the partition π and π will be frequently enumerated as a family $\pi = \{X_i / i \in I\}$ indexed with some set I. An equivalence relation is canonically associated with a partition by:

$$a \equiv b (\mathrm{mod}\ \pi) \quad \text{iff } a, b \in X_i \quad \text{for some } i \in I.$$

In the context of a fixed partition π we use $a \equiv b$ instead of $a \equiv b \ (\mathrm{mod}\ \pi)$.

A partial order on the set of partitions of some set X is defined by:

$$\pi \leqslant \pi' \quad \text{iff every block of } \pi \text{ is contained in some block of } \pi'.$$

If π and π' are partitions of two disjoint sets X and X' then $\pi \cup \pi'$ is a partition of $X \cup X'$; if $Y \subset X$ we denote by $\pi \cap Y = \{\alpha \cap Y / \alpha \cap Y \neq \emptyset \text{ and } \alpha \in \pi\}$ the *restriction* of π to Y (it is a partition of Y).

We now recall some definitions and notations on words and languages. If X is a (possibly infinite) alphabet, X^* denotes the set of words on X and ε the empty word. The length of a word $u \in X^*$ is denoted $|u|$, its mirror image is denoted $\tilde{u}$ and, if $u \neq \varepsilon$, its leftmost symbol is denoted FIRST(u).

The set X^* is ordered by the *prefix order*: $u \prec v$ iff $v = uw$ for some $w \in X^*$ (u is a *prefix* of v). The relation $\prec$ extends to languages in the following way: for $u \in X^*$, $L, L' \subset X^*$, $u \prec L'$ iff $u \prec v$ for some $v \in L'$ and $L \prec L'$ iff $u \prec L'$ for all $u \in L$. A language $L \subset X^*$ is *prefix-free* (in French: *préfixe*) if $u, v \in L$ and $u \prec v$ imply $u = v$ and *prefix-closed* (in French: *clos par préfixe*) if $u \prec L$ implies $u \in L$.

For $u \in X^*$ and $L \subset X^*$, we define the *left-quotient* $u \backslash L = \{v \in X^* / uv \in L\}$.

If $L \subset X^*$, there exists a minimal set $Y \subset X$ such that $L \subset Y^*$. It is called the *minimal alphabet* of L and denoted $\alpha(L)$.

It will be useful to consider k-tuples of languages, especially in Sections 4 and 5 but also when considering strict deterministic grammars: this will give us some nice formulations.

For $k \in \mathbf{N}$, $k \geqslant 1$ a *k-language on X* is a k-tuple of languages $\vec{L} = (L_1, \ldots, L_k)$. Let $\mathrm{SUM}(\vec{L}) = \bigcup \{L_i / 1 \leqslant i \leqslant k\}$ and $\alpha(\vec{L}) = \alpha(\mathrm{SUM}(\vec{L}))$. A multilanguage is a k-language for some k. A k-language is *prefix-free* if $\mathrm{SUM}(\vec{L})$ is prefix-free and $L_i \cap L_j = \emptyset$ for $1 \leqslant i < j \leqslant k$.

We now give some background on *context-free grammars* (in French: *grammaires algébriques*).

A context-free grammar G with *terminal alphabet* X and *non-terminal alphabet* N (we write then $G(X, N)$ to indicate the alphabets) is a set of equations:

$$\begin{cases} \vdots \\ S_i = m_{i,1} + \cdots + m_{i,s_i} \\ \vdots \\ 1 \leqslant i \leqslant n, \qquad m_{i,j} \in V^* \end{cases}$$

where $N = \{S_1, \ldots, S_n\}$ and $V = X \cup N$.

We also define $R(G, S_i) = \{m_{i,j} / 1 \leqslant j \leqslant s_i\}$.

If $s_i = 0$ the i^{th} equation is $S_i = \emptyset$ and $R(G, S_i) = \emptyset$.

Such a system has a minimal solution in 2^{X^*}. By Schützenberger's theorem [29], it coincides with the n-language $(L(G, S_1), \ldots, L(G, S_n))$ where $L(G, S_i)$ is defined as usual. The derivation is denoted $u \to^* v$, the leftmost derivation is denoted $u \Rightarrow^* v$. If necessary we indicate the number of steps: $u \to^n v$ and/or the grammar involved: $u \Rightarrow^*_G v$, $u \to^n_G v$ etc. (0-step derivation is identity). For $m \in V^*$, let $L(G, m) = \{u \in X^* / m \to^* u\}$. Such a language is called *context-free* (in French: *algébrique*).

Context-free grammars are often defined with some *start symbol* or *axiom*. In order to generate k-languages we will use a k-tuple of start symbols and call it also an *axiom*. $\vec{A} = (A_1, \ldots, A_k) \in N^k$. The k-language generated is $\vec{L}(G) = (L(G, A_1), \ldots, L(G, A_k))$, also denoted by $L(G, \vec{A})$. We write $G(X, N, \vec{A})$ to specify the axiom. A grammar $G(X, N)$ is *$\emptyset$-reduced* if $L(G, S) \neq \emptyset$ for all $S \in N$. A grammar $G(X, N, \vec{A})$ is *A-reduced* if for all $T \in N$, $A_j \to^* uTv$ for some $1 \leqslant j \leqslant k$ and $u, v \in V^*$ (with $V = X \cup N$). If a grammar is both $\emptyset$-reduced and A-reduced, it is said *reduced*.

Any grammar can be effectively transformed into a $\emptyset$-reduced or A-reduced or reduced grammar. If it happens that $L(G, A_i) = \emptyset$ for some component A_i of $\vec{A}$, we delete A_i from N and replace it by the new symbol $\varnothing$ in $\vec{A}$. Hence, in general the axiom $\vec{A}$ of a grammar is a k-tuple of elements of $N \cup \{\varnothing\}$.

We will mainly deal with *prefix-free deterministic* languages [16], defined by *deterministic pushdown automatas* (*DPDA*'s) (in French: *automates déterministes à pile*).

A DPDA is a 6-tuple $\mathcal{A} = \langle Q, X, \Gamma, \delta, q_0, Z_0 \rangle$ with set of *states* Q, *input alphabet* X, *pushdown alphabet* Γ. ($\mathcal{A}$ will accept by empty store and needs no final states.) We assume that $q_0 \in Q$, $Z_0 \in \Gamma$ and that the *transition function* δ is a partial mapping: $Q \times (X \cup \{\varepsilon\}) \times \Gamma \to Q \times \Gamma^*$ such that $\delta(q, \varepsilon, Z)\downarrow$ implies $\delta(q, a, Z)\uparrow$ for all $q \in Q$, $Z \in \Gamma$ and $a \in X$. A *configuration* of $\mathcal{A}$ is a pair $(q, m) \in Q \times \Gamma^*$, the *mode* of (q, m) is $(q, \text{FIRST}(m))$ and $\text{FIRST}(m)$ is the *top-most symbol* of the *stack* m. The mode is undefined if $m = \varepsilon$. (Some authors write $(q, \tilde{m})$ the configuration that we write (q, m).) The behaviour of $\mathcal{A}$ is described by a binary relation, the *transition* on

the set $Q \times X^* \times \Gamma^*$ of *instantaneous descriptions*. The transition is denoted $\vdash$ and defined by $(q, u, Z) \vdash (q', v, m')$ iff $\delta(q, a, Z) = (q', m'')$, $u = av$ and $m' = m''m$. (Note that $a \in X \cup \{\varepsilon\}$). The language *accepted* by $\mathcal{A}$ is $N(\mathcal{A}) = \{u \in X^* / (q_0, u, Z_0) \vdash^* (q, \varepsilon, \varepsilon) \text{ for some } q \in Q\}$. It is prefix-free. Let PDet = $\{N(\mathcal{A})/\mathcal{A}$ is a DPDA$\}$ be the family of *prefix-free deterministic* languages. Two DPDA's $\mathcal{A}$ and $\mathcal{A}'$ are *equivalent* if $N(\mathcal{A}) = N(\mathcal{A}')$. The equivalence problem for DPDA's is still open (see Valiant [33] for decidable subcases).

A DPDA is *real-time* if $\delta(q, \varepsilon, Z)\uparrow$ for all q, Z. Following Valiant we denote by R_0 the class of languages $\{N(\mathcal{A})/\mathcal{A}$ is a real-time DPDA$\}$. Observe that $L = \{wc, w\, d\tilde{w}/w \in \{a, b\}^*\} \subset \{a, b, c, d\}^*$ is both prefix-free deterministic and $T(\mathcal{A})$ for some real-time DPDA $\mathcal{A}$ accepting by final state (see [16]) but does not belong to R_0 (easy proof left to the reader).

A configuration (q, m) of a DPDA $\mathcal{A}$ is *reachable* (in French: *accessible*) if $(q_0, u, Z_0) \vdash^* (q, \varepsilon, m)$ for some u. A mode (q, Z) is *reachable* if (q, Zm) is reachable for some $m \in \Gamma^*$. The reachability of a given mode or configuration is decidable (by methods basically derived of Lemma 2.5.3 of [16]). A configuration (q, m) is *live* if $(q, u, m) \vdash^* (q', \varepsilon, \varepsilon)$ for some $u \in X^*$ and $q' \in Q$. A DPDA $\mathcal{A}$ is *faithful* (in French: *fidèle*) if every reachable configuration of $\mathcal{A}$ is live.

Here is a technical lemma:

Lemma 1.1. *Given a DPDA $\mathcal{A}$, one can construct a faithful DPDA $\mathcal{A}'$ such that $N(\mathcal{A}) = N(\mathcal{A}')$.*

Proof. Let $\mathcal{A} = \langle Q, X, \Gamma, \delta, q_0, Z_0 \rangle$. It is not faithful if some transition can lead from a reachable configuration to a non-live one. We will prevent such a possibility in $\mathcal{A}'$. Note that the liveness of a configuration (q, Zm) of $\mathcal{A}$ does not depend only on the mode but also on the whole content of the pushdown store. But we will modify $\mathcal{A}$ in order to satisfy the former, and then, it will be possible to eliminate transitions leading to non-live configurations. To do so we first investigate the structure of live configurations.

For $m \in \Gamma^*$ and $k \subset Q$ let us define:

$$p(m, k) = \{q \in Q / \exists u \in X^*, \exists q' \in k, (q, u, m) \vdash^* (q', \varepsilon, \varepsilon)\}.$$

Intuitively, $p(m, k)$ is the set of states q such that the configuration (q, m) is transformed into (q', ε) for some $q' \in k$ by some input word $u \in X^*$. Clearly, (q, m) is live if and only if $q \in p(m, Q)$. And we have the following facts:

(1) $p(\varepsilon, k) = k$,

(2) $p(Zm, k) = p(Z, p(m, k))$,

(3) $p(Z, k)$ is computable (from Lemma 2.5.3 of [16] for instance).

We now construct a DPDA $\mathcal{A}'$ which simulates the transition sequences of $\mathcal{A}$. This DPDA puts on the pushdown store a certain information which allows it to know

before performing any transition, whether the next configuration is to be live or not. The transition is performed in the former case only. Hence every accessible configuration is live.

The configuration $(q, Z_1 \cdots Z_r)$ of $\mathcal{A}$ is simulated by $(q, [Z_1, k_1][Z_2, k_2] \cdots [Z_r, k_r])$ with $k_i = p(Z_{i+1} \cdots Z_r, Q)$ for $1 \leq i < r$ (hence $k_r = Q$). By fact 2 above this property is satisfied iff:

$$k_i = p(Z_{i+1}, k_{i+1}) \quad \text{for } 1 \leq i < r \text{ and } k_r = Q.$$

Assuming this, $\mathcal{A}'$ can make sure that $(q, Z_1 \cdots Z_r)$ is live by checking that $q \in p(Z_1, k_1)$. We give now a formal construction:

$$\mathcal{A}' = \langle Q, X, \Gamma', \delta', q_0, Z_0' \rangle,$$

$$\Gamma' = \Gamma \times 2^Q,$$

$$Z_0' = [Z_0, Q],$$

$$\delta'(q, a, [Z, k]) = (q', \varepsilon) \text{ if } \delta(q, a, Z) = (q', \varepsilon) \text{ and } q' \in k$$

$$= (q', [Z_1, k_1] \cdots [Z_m, k_m]) \text{ if}$$

$$\begin{cases} \delta(q, a, Z) = (q', Z_1 \cdots Z_m), \\ k_m = k, \\ k_i = p(Z_{i+1}, k_{i+1}) \quad \text{for } 1 \leq i < m, \\ q' \in p(Z_1, k_1). \end{cases}$$

undefined otherwise.

The reader will easily prove by induction on n that:

$(q_0, u, Z_0') \vdash\!\!\!\!-\!\!\!\!-\!\!\!\!-^n_{\mathcal{A}'} (q, \varepsilon, [Z_1, k_1] \cdots [Z_r, k_r])$ iff

$$\begin{cases} k_r = Q, \\ k_i = p(Z_{i+1}, k_{i+1}) \quad \text{for } 1 \leq i < r, \\ q \in p(Z_1, k_1), \\ (q_0, u, Z_0) \vdash^n_{\mathcal{A}} (q, \varepsilon, Z_1 \cdots Z_r), \\ (q, Z_1 \cdots Z_r) \text{ is live.} \end{cases}$$

It follows that $N(\mathcal{A}) = N(\mathcal{A}')$ and $\mathcal{A}'$ is faithful. □

Remarks 1.2. (1) Using Properties 1, 2, 3 of p one easily constructs a left-linear grammar generating $\{qm \in Q\Gamma^*/(q, m) \text{ is live}\}$. Hence this language is regular.

(2) The transformation of $\mathcal{A}$ into $\mathcal{A}'$ preserves a number of properties of $\mathcal{A}$: the real-time-ness, the number of states.

Finally, it will be useful to define *prefix-free deterministic multilanguages*. Let $a = \langle Q, X, \Gamma, \delta, q_0, Z_0\rangle$ be a DPDA. Let $\theta = (\theta_1, \ldots, \theta_k)$ be a sequence of pairwise disjoint subsets of Q. Let us define $\vec{N}_\theta(a) = (L_1, \ldots, L_k)$ where

$$L_i = \{u \in X^* / (q_0, u, Z_0) \overset{*}{\vdash} (q, \varepsilon, \varepsilon) \text{ for some } q \in \theta_i\};$$

this is clearly a prefix-free k-language.

Remark 1.3. With the notations of Lemma 1.1, let $a'' = \langle Q, X, \Gamma, \delta', q_0, Z_0''\rangle$ associated with a and θ as above with $Z_0'' = [Z_0, \bigcup_{1\leq i\leq k} \theta_i]$. Let also $\theta' = (\theta_1', \ldots, \theta_{k-1}', \theta_k')$ with $\theta_k' = Q - \bigcup_{1\leq i<k} \theta_i$ and $\theta_i' = \theta_i$ for $1 \leq i \leq k-1$. Then we have the following properties:

(i) a'' is faithful,

(ii) $\vec{N}_{\theta'}(a'') = \vec{N}_\theta(a)$,

(iii) $Q = \bigcup_{1\leq i\leq k} \theta_i'$.

This technical remark will be useful in proposition 2.19.

2. Complete languages

We introduce here π-strict and π-complete languages. These languages naturally appear when one considers the set of branches of a tree, as we will do in Sections 4 and 5. The notion of a complete language is an extension of that of complete prefix-free code defined in Nivat [22]. We examine the completeness of context-free and deterministic languages and define several ways of completing a given language which are effective for deterministic languages.

2.1. π-strict and π-complete languages

Definition 2.1. Let X be a finite alphabet and π a partition of X. A language $L \subset X^*$ is *π-strict* if

(1) L is prefix-free,

(2) for all $u, v, w \in X^*$, $a, b \in X$ if uav and $ubw \in L$, then $a \equiv b$ (which means $a \equiv b \pmod{\pi}$).

It is *π-complete* if it is π-strict and

(3) for all $u, v \in X^*$, $a, b \in X$ such that $uav \in L$ and $a \equiv b$ there exists $w \in X^*$ such that $ubw \in L$.

Examples. (1) Let $X = \{a, b, c, d, f\}$ and $\pi = \{\{a, b, c\}, \{d, f\}\}$. The language $L = \{a, bd, bfa, ca, cb, cc\}$ is π-strict but not π-complete; $L' = L \cup \{bfb, bfc\}$ is π-complete; $L'' = \{a, bab, bcd, bfc\}$ is prefix-free but not π-strict.

(2) By using an appropriate coding, the set of branches of a tree can be characterized as a π-complete language.

Let us define $\sigma(u, L)$ for $L \subset X^*$ and $u \in X^*$ to be $\{a \in X / uav \in L$ for some $v \in X^*\}$. Intuitively, $\sigma(u, L)$ is the set of letters a of X such that ua is a prefix of some word in L. In other words, when reading a word $w = uav$ from left to right, you are sure that w is not in L if a is not in $\sigma(u, L)$.

Condition 2 in Definition 2.1 means that $\sigma(u, L)$ is contained in some block of π for all prefix u of L. Condition 3 means that $\sigma(u, L)$ is empty or *is* a block of π.

The following lemma is given without proof since it is only a routine application of definitions. We will often do so in the sequel.

Lemma 2.2. *A language is π-complete iff it is maximal for inclusion among the family of π-strict languages.*

Remark 2.3. If $\pi = \{X\}$, a π-complete language is just a maximal prefix-free language, i.e. a *complete prefix-free code* (in French: *code préfixe complet*) as defined in [22]. We will use the abbreviation CPC in the sequel.

For an arbitrary language L, the set of non-empty $\sigma(u, L)$ for $u \in X^*$ is not necessarily a partition of X. But there exists a least partition ν such that $\sigma(u, L)$ is contained in some block of ν for each $u \in X^*$. Let us denote it by $\nu_X(L)$.

Proposition 2.4. *Let $L \subset X^*$ be prefix-free*

(1) *L is π-strict iff $\nu_X(L) \leqslant \pi$.*

(2) *L is π-complete iff it is $\nu_X(L)$-complete. Hence if $X = \alpha(L)$, the language L is complete for at most one partition of X.*

Hence the completeness of a language is an intrinsic property. In fact, a language is complete iff it is prefix-free and for all $u, v, w, u', v' \in X^*$, for all $a, b \in X$:

$$\left.\begin{array}{l} uav \in L \\ ubw \in L \\ u'av' \in L \end{array}\right\} \text{ imply } u'bw' \in L \quad \text{for some } w' \in X^*.$$

Proposition 2.5. (1) *If $L \subset X^*$ is π-complete and $u \in X^*$ then $u\backslash L$ is π-complete.*

(2) *If L is prefix-free, and $LL' \neq \emptyset$, then LL' is π-complete iff L and L' are π-complete.*

2.2. How to complete a language

We consider here how to transform a non complete language into a complete one. Later we will apply these constructions to context-free languages and obtain interesting differences between deterministic and non-deterministic languages.

Two different completions will be defined. The first one simply adds some words in order to obtain a complete language.

Let L be a π-strict language. Let $\bar{L}_\pi = L \cup K$ where $K = \{ua / u \in X^*, a \in X, a \notin \sigma(u, L)$ and $a \equiv b$ for some $b \in \sigma(u, L)\}$. Then:

Proposition 2.6. *If L is π-strict than $\bar{L}_\pi$ is π-complete. For any π-complete language L' such that $L \prec L'$ then $\bar{L}_\pi \prec L'$.*

This means that our completion $\bar{L}_\pi$ is minimal with respect to the order $\prec$ on languages.

We will write $\bar{L}$ instead of $\bar{L}_\pi$ if $\pi = \{X\}$. Hence $\bar{L}$ is the minimal CPC containing the prefix-free language L.

We now define the second completion. It applies to a prefix-free language L and produces a complete language $\check{L}$ on a different alphabet, such that L is an homomorphic image of $\check{L}$.

Let $L \subset X^*$ and $L' \subset Y^*$. We say that L is a *sharp image of L'* (in French: *image nette*) if there exists an homomorphism $\theta: Y^* \to X^*$ such that

(1) $\theta(Y) \subset X$,

(2) $\theta(L') = L$,

(3) the restriction of θ to L' is one-to-one.

Proposition 2.7. *Let $L \subset X^*$ be prefix-free. It is the sharp image of a complete language $\check{L}$. If L is a sharp image of a complete language L' then $\check{L}$ is a sharp image of L'.*

Proof. Let $\check{X} = \{[a, \alpha] / a \in \alpha \subset X\}$, let $\phi: \check{X} \to X$ be defined by $\phi([a, \alpha]) = a$.

Let $L \subset X^*$ be prefix-free and $\psi: X^* \to \check{X}^*$ be the following partial mapping (relative to L):

$$\begin{cases} \psi(\varepsilon) = \varepsilon, \\ \psi(ua) = \psi(u)[a, \sigma(u, L)] \quad \text{if } \sigma(u, L) \neq \emptyset, \\ \text{undefined otherwise.} \end{cases}$$

Let $\check{L} = \{\psi(u) / u \in L\}$ and $\check{\pi}$ the partition such that $[a, \alpha] \equiv [b, \beta] \pmod{\check{\pi}}$ iff $\alpha = \beta$.

Clearly, $L = \phi(\check{L})$ hence $\check{L}$ is prefix-free. Let $v \in \check{X}^*$, then $\sigma(v, \check{L}) = \{[a, \alpha] / a \in \alpha = \sigma(\phi(v), L)\}$ which is a block of $\check{\pi}$. Hence $\check{L}$ is $\check{\pi}$-complete. The mapping ϕ is one-to-one on $\check{L}$ since $\phi(\psi(u)) = u$ for $u \in L$. Hence L is a sharp image of the complete language $\check{L}$.

Let us give some intuition about this.

A prefix-free language L fails to be complete if it contains two words uav and $u'av'$ such that $\sigma(u, L) \neq \sigma(u', L)$. If we replace uav by $u[a, \sigma(u, L)]v$ and $u'av'$ by

$u'[a, \sigma(u', L)]v'$ i.e. if we "repaint" the letter a using different "colors" according to what precedes a in the two words (namely u or u') this failure disappears. We repaint every letter in every word similarly and obtain $\psi(u)$ from u. Then $\check{L}$ is the set of "repainted" words $\psi(u)$ for all $u \in L$.

We now prove that $\check{L}$ is final among complete languages having sharp image L.

Let $\theta: L' \to L$ such that L is a sharp image of a complete language $L' \subset Y^*$.

Claim. *Let uav and $u'av' \in L'$. Then $\sigma(\theta(u), L) = \sigma(\theta(u'), L)$.*

Proof. Assume the contrary and let $b \in \sigma(\theta(u), L)$, $b \notin \sigma(\theta(u'), L)$. Then for some $b' \in Y$, $w \in Y^*$, $\theta(b') = b$ and $ub'w \in L'$. Since uav and $u'av' \in L'$ and L' is complete, $u'b'w' \in L'$ for some $w' \in Y^*$. Hence $\theta(u')b\theta(w') \in L$ and $b \in \sigma(\theta(u'), L)$, contradiction. □

Let $\check{\theta}: Y^* \to \check{X}^*$ such that $\check{\theta}(a) = [\theta(a), \sigma(\theta(u), L)]$ for some $uav \in L'$ and anything, say $[\theta(a), \{\theta(a)\}]$, if a does not belong to the minimal alphabet of L'. By the claim, $\check{\theta}$ is well defined.

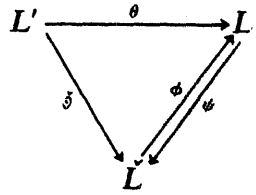
It is clear that $\theta = \phi \circ \check{\theta}$; hence $\check{\theta}$ restricted to L' is one-to-one. For any $u \prec L'$, $\check{\theta}(u) = \psi(\theta(u))$. By induction on u:

(1) $\check{\theta}(\varepsilon) = \varepsilon = \psi(\theta(\varepsilon))$

(2) $\check{\theta}(ua) = \check{\theta}(u)\check{\theta}(a)$
$= \psi(\theta(u))[\theta(a), \sigma(\theta(u), L)]$ (by inductive assumption and the claim),
$= \psi(\theta(ua))$.

Hence $\check{\theta}(L') = \psi(\theta(L')) = \psi(L) = \check{L}$. □

We have the following commutative diagram:



Corollary 2.8. *For prefix-free languages L and L':*

(1) *$L = L'$ iff $\check{L} = \check{L}'$,*

(2) *$\widehat{(LL')} = \check{L}\check{L}'$.*

2.3. The completeness of context-free languages

For context-free languages, we obtain some negative results:

Proposition 2.9. *It is not decidable whether a context-free language, known to be prefix-free, is complete.*

Proof. Let $(x_i, y_i)_{1\leq i\leq n}$ be a Post correspondence problem with $x_i, y_i \in \{a, b\}^+$. Let $X = \{a, b, c, d\} \cup [n]$. (To be correct, one should take an alphabet in one-to-one correspondence with $[n]$ instead of $[n]$, but no ambiguity will follow from this.)

Let $L = \{i_1 \cdots i_k cwd / k \geq 0, i_1, \ldots, i_k \in [n], w \in \{a, b\}^* \text{ and } (w \neq x_{i_k} \cdots x_{i_1} \text{ or } w \neq y_{i_k} \cdots y_{i_1})\}$. This linear language is contained in $L' = [n]^* c \{a, b\}^* d$ hence is prefix-free. Let $\pi = \{\{1, 2, \ldots, n, c\}, \{a, b, d\}\}$. It is clear that $\nu_X(L) = \pi$, hence L is complete iff it is π-complete. If the given problem has no solution, then $L = L'$ which is π-complete. If it has solutions, $L \not\subseteq L'$ and L cannot be π-complete since L' is. □

Remarks 2.10. (1) Similarly, to be a CPC is not decidable for a prefix-free context-free language.

(2) By a similar construction one can prove that $\nu_X(L(G))$ is not computable in general for a given context-free grammar G. See [6].

Proposition 2.11. *There exists a prefix-free context-free language L_0 which is not a sharp image of any complete context-free language; hence $\check{L}_0$ is not context-free.*

Proof. Let $L_0 = \{a^n b^n c^m d + a^n b^m c^m f / n, m \geq 0\}$. If $\check{L}_0$ is context-free, $L_0' = \check{L}_0 \cap \check{X}^*[d, \{d, f\}]$ and $L_0'' = \phi(L_0')$ (see Proposition 2.7) are so. But $L_0'' = \{a^n b^n c^n d / n \geq 0\}$. Contradiction. If L_0 is a sharp image of a complete language L', $\check{\theta}(L') = \check{L}_0$ and L' cannot be context-free. □

Example. Let $L' = \{\tilde{w} w c c / w \in \{a, b\}^*\}$ and $L = L' \cup \{a, b\}^* cd$. Then $\check{L}'$ is not context-free but $\check{L}$ *is* context-free (left to the reader). Note that L' and L are not deterministic and compare with Theorem 2.14 below.

Proposition 2.12. *There exists a prefix-free context-free language L_1 such that $\bar{L}_1$ is not context-free.*

Proof. Let $L_1 = \{a^m b^n c^p dd / m \neq n \text{ or } n \neq p\} \subset \{a, b, c, d\}^*$. Recall that $\bar{L}_1$ is the least CPC containing L_1 (see 2.4). Then $\bar{L}_1 \cap a^* b^* c^* d = \{a^n b^n c^n d / n \geq 0\}$ is not context-free. □

Remark 2.13. L_1 is included in the complete language $\{a, b, c\}^* d \{a, b, c\}^* d$. In general, if a context-free language L is contained in a regular prefix-free language K, then $L \subset \bar{K}$ and $\bar{K}$ is a regular CPC (by Theorem 2.14 below).

The following question seems to be open:

Is any context-free prefix-free language L contained in some context-free CPC L'?

2.4. The case of deterministic languages

The previous negative results show that the completeness condition does not fit to general context-free languages. On the other hand it fits very well to deterministic ones.

Theorem 2.14. *Given any prefix-free deterministic language $L \subset X^*$ one can effectively:*

(1) *compute $\nu_X(L)$,*

(2) *decide if L is complete, or if L is π-strict,*

(3) *construct a DPDA $\check{a}$ such that $N(\check{a}) = \check{L}$,*

(4) *construct a DPDA $\bar{a}_\pi$ such that $N(\bar{a}_\pi) = \bar{L}_\pi$ for a partition $\pi \geq \nu(L)$.*

Proof. By Lemma 1.1, one can assume that $L = N(a)$ for some faithful DPDA $a = \langle Q, X, \Gamma, \delta, q_0, Z_0 \rangle$. Since a is deterministic and faithful, for all $u \in X^*$ one and only one of the following three conditions is satisfied:

(1) $(q_0, u, Z_0) \vdash\!\!\!\!-\!\!\!\!-\!\!\!\!-^* (q, \varepsilon, m)$ for no $(q, m) \in Q \times \Gamma^*$ and $u \backslash L = \emptyset$.

(2) $(q_0, u, Z_0) \vdash\!\!\!\!-\!\!\!\!-\!\!\!\!-^* (q, \varepsilon, \varepsilon)$ for some $q \in Q$ and $u \in L$,

(3) $(q_0, u, Z_0) \vdash\!\!\!\!-\!\!\!\!-\!\!\!\!-^* (q, \varepsilon, Zm)$ for $q \in Q$, $Z \in \Gamma$ and $m \in \Gamma^*$ such that $\delta(q, \varepsilon, Z)\!\uparrow$. The configuration (q, Zm) is unique,

$$u \backslash L = \{v \in X^* / (q, v, Zm) \vdash\!\!\!\!-\!\!\!\!-^* (q', \varepsilon, \varepsilon) \text{ for some } q' \in Q\}$$

and $\sigma(u, L) = \{a \in X \backslash \delta(q, a, Z)\!\downarrow\}$ since a is faithful.

For $q \in Q$ and $Z \in \Gamma$ let $\Delta(q, Z) = \{a \in X / \delta(q, a, Z)\!\downarrow\}$. Hence $\{\sigma(u, L) / u \in X^*\} = \{\emptyset\} \cup \{\Delta(q, Z) / (q, Z)$ is a reachable mode of $a\}$. This set is computable: so is $\nu_X(L)$.

A DPDA a is *π-strict* (resp. *π-complete*) if it is faithful and for each reachable mode (q, Z), $\Delta(q, Z) \subset \alpha$ for some $\alpha \in \pi$ (resp. $\Delta(q, Z) \in \pi \cup \{\emptyset\}$). Hence for a faithful DPDA a, the language $N(a)$ is π-strict (resp. π-complete) iff a is π-strict (resp. π-complete); this property is decidable and the second assertion is proved.

Let $\check{a} = \langle Q, \check{X}, \Gamma, \check{\delta}, q_0, Z_0 \rangle$ such that:

$$\check{\delta}(q, \varepsilon, Z) = \delta(q, \varepsilon, Z) \quad \text{if } \delta(q, \varepsilon, Z)\!\downarrow,$$

$$\check{\delta}(q, [a, \alpha], Z) = \delta(q, a, Z) \quad \text{if } a \in \alpha = \Delta(q, Z).$$

By induction on n, one proves that

$$(q_0, u, Z_0) \vdash\!\!\!\!-\!\!\!\!-\!\!\!\!-^n_{\check{a}} (q, \varepsilon, m) \quad \text{iff } (q_0, \psi(u), Z_0) \vdash\!\!\!\!-\!\!\!\!-\!\!\!\!-^n_{a} (q, \varepsilon, m).$$

hence $N(\check{a}) = \check{L}$.

Assume now that for any reachable mode (q, Z) of $\mathcal{A}$, $\Delta(q, Z) \subset \alpha$ for some $\alpha \in \pi$. Let $\tilde{a}_\pi = \langle Q', X, \Gamma, \delta', q_0, Z_0 \rangle$ such that:

$$\begin{cases} Q' = Q \cup \{q_f\} \quad (\text{with } q_f \notin Q) \\ \delta'(q, \varepsilon, Z) = \delta(q, \varepsilon, Z) \quad \text{if } \delta(q, \varepsilon, Z)\downarrow \\ \delta'(q, a, Z) = \delta(q, a, Z) \quad \text{if } a \in \Delta(q, Z), \\ \qquad\qquad = (q_f, \varepsilon) \quad \text{if } a \notin \Delta(q, Z) \text{ but } a \equiv b \text{ for some } b \in \Delta(q, Z), \\ \delta'(q_f, \varepsilon, Z) = (q_f, \varepsilon) \quad \text{for all } Z \in \Gamma. \end{cases}$$

Then $N(\tilde{a}_\pi) = \tilde{L}_\pi$. □

Let SDet be the family of simple deterministic languages [21, 4], and PReg be the family of prefix-free regular languages. Hence PReg $\subset$ SDet $\subset R_0$.

Corollary 2.15. *The mapping* $L \rightsquigarrow \check{L}$ *preserves the families of languages* PReg, R_0 *and* SDet.

Remark 2.16. Since the number of states of DPDA is not modified by the construction of Lemma 1.1 and Theorem 2.14.3, $\deg(\check{L}) = \deg(L)$ in the sense of [18].

Corollary 2.17. *The mapping* $L \rightsquigarrow \tilde{L}_\pi$ *preserves the family* PReg *but neither* SDet *nor* R_0.

Proof. Let $L = \{a, b, c\}^*$ the simple deterministic language generated by $S = aSS + b$. Let $\pi = \{\{a, b, c\}\}$ and $L' = \tilde{L}_\pi$. Then $L' \notin R_0$: for any $u \prec L'$, either $u \in L'$ or $uc \in L'$. Hence $\min\{|v| / uv \in L'\} \leq 1$ and an R_0-language satisfying this property must be regular. But L' is clearly not. Since SDet $\subset R_0$, we are done. □

Proposition 2.18. *Let* $L \in R_0$ (*resp.* SDet) *be* π*-strict. There exists a* π*-complete* $L' \supset L$ *which belongs to* R_0 (*resp.* SDet).

Proof. Let $\mathcal{A} = \langle Q, X, \Gamma, \delta, q_0, Z_0 \rangle$ be faithful and real time such that for each reachable mode (q, Z), $\Delta(q, Z) \subset \alpha$ for some $\alpha \in \pi$. Let $\mathcal{A}' = \langle Q', X, \Gamma, \delta', q_0, Z_0 \rangle$ as in the proof of Theorem 2.14 except that

$$\delta'(q_f, a, Z) = (q_f, \varepsilon) \quad \text{for all } a \in \alpha \in \pi,$$

where α is an arbitrary but fixed block of π. Clearly $L \subset N(\mathcal{A}')$, $N(\mathcal{A}')$ is π-complete and belongs to R_0. The case of simple deterministic languages is left to the reader. □

Most properties of π-strict and π-complete languages easily extend to multilanguages. The following definitions and results will be used later: a multilanguage $\vec{L} = (L_1, \ldots, L_k)$ is *π-strict* (resp. *π-complete*) if it is prefix-free (see Section 1) and $\mathrm{SUM}(\vec{L})$ is π-strict (resp. π-complete). Hence $\vec{L}$ is an ordered partition of a π-strict (resp. π-complete) language some blocks of which may be empty.

With the notations of Remark 1.3, we obtain:

Proposition 2.19. *Let $\mathscr{A}$ be a faithful* DPDA *and θ a sequence of pairwise disjoint subsets of Q such that $Q = \bigcup_{1 \leq i \leq k} \theta_i$. For any partition π of X:*

(1) *$\vec{N}_\theta(\mathscr{A})$ is π-strict iff $\mathscr{A}$ is π-strict.*

(2) *$\vec{N}_\theta(\mathscr{A})$ is π-complete iff $\mathscr{A}$ is π-complete.*

3. Strict and complete deterministic grammars

Strict deterministic grammars, defined by Harrison and Havel [18] generate the prefix-free deterministic languages. We refine this notion into that of a *π-strict deterministic grammar* with respect to a partition π of the terminal alphabet. These grammars generate the deterministic languages which are π-strict, then called *π-strict deterministic* without ambiguity. We also define the *π-complete deterministic* grammars, which generate the *π-complete* (and) *deterministic* languages

The motivation of such definitions will appear in Sections 4 and 5. But we think they are natural generalizations of that of [18] which are interesting by themselves. Let us draw the reader's attention to Proposition 3.14 by which a property of a grammar is inferred from a property of the language it generates.

3.1. Strict and complete partitions of a context-free grammar

Definition 3.1. Let $G(X, N)$ be a context-free grammar (see Section 1 for notations) and $V = X \cup N$. A partition π of V is *strict* for G if:

P0: each block of π is contained in X or in N,

P1: for all $S, S' \in N$, for all $\alpha, \beta, \beta' \in V^*$, if $S \equiv S'$, $S \to \alpha\beta$ and $S' \to \alpha\beta'$ then:

$$\text{either } \beta = \beta' = \varepsilon \quad \text{and} \quad S = S'$$

$$\text{or} \quad \beta \neq \varepsilon, \beta' \neq \varepsilon \quad \text{and} \quad \mathrm{FIRST}(\beta) \equiv \mathrm{FIRST}(\beta')$$

A strict partition π is *complete* if:

P2: for all $S \in N$, for all $T, T' \in V$ and $\alpha, \beta \in V^*$, if $S \to \alpha T \beta$ and $T' \equiv T$, then $S' \to \alpha T' \beta'$ for some $S' \equiv S$ and $\beta' \in V^*$.

In the first case G is *π-strict deterministic* (denoted also π-SD), in the second one, G is *π-complete deterministic* (or π-CD). A grammar is *strict deterministic* (resp. *complete deterministic*) if it is π-SD (resp. π-CD) for some partition π.

In [18], Harrison and Havel do not introduce complete deterministic grammars. They only consider strict partitions π such that $X \in \pi$, but the two classes of strict deterministic grammars are the same. A lot of their results will be valid with slight changes only.

An *axiom* $\vec{A} = (A_1, \ldots, A_k) \in (N \cup \{\emptyset\})^k$ for a π-strict deterministic (resp. π-complete deterministic) grammar $G(X, N)$ must satisfy the following condition P3 (resp. P4):

P3: $N_0 = \{A_i / 1 \leq i \leq k \text{ and } A_i \neq \emptyset\}$ is included in some block of π.

P4: N_0 is a block of π.

Proposition 3.2. (1) *Let G be a* SD *grammar, (resp. with an axiom). The set of strict partitions for G has a least element denoted $\nu(G)$.*

(2) *One can decide if a given context-free grammar G is* SD. *If it is, one can compute $\nu(G)$.*

Proof. Theorem 2.4 and Algorithm 1 of [18]. □

Let $G(X, N, \vec{A})$ be A-reduced; let $X' \subset X$ be the set of terminal symbols which occur in the equations of G and $V' = X' \cup N$. Let $\nu'(G)$ be the least partition of V' which is strict for G. Then:

Proposition 3.3. *If G is complete deterministic then $\nu'(G)$ is the only partition of V complete for G. One can decide if G is complete deterministic.*

Proof. Let $\pi = \nu'(G)$ and assume that G is π'-complete deterministic; by Proposition 3.2, $\pi \leq \pi'$. We prove that $[T]_\pi = [T]_{\pi'}$ for all $T \in V'$.

Assume that $\vec{A} = (A_1, \ldots, A_k)$. Let us define the *distance* of any $T \in V'$ to $\vec{A}$ by $d(T) = \text{Min}\{n \in \mathbb{N} / A_i \to^n uTv$ for some $1 \leq i \leq k$ and $u, v \in V'^*\}$. Since G is A-reduced, $d(T) < \infty$ for all $T \in V'$ and we can prove that $[T]_\pi = [T]_{\pi'}$ for all $T \in V'$ by induction on $d(T)$.

Basis: If $d(T) = 0$ then $T \in N_0$. From the definitions, $N_0 = [T]_{\pi'}$ and $N_0 \subset [T]_\pi$; but $[T]_\pi = [T]_{\pi'}$ since $\pi \leq \pi'$.

Inductive step: $d(T) = n + 1$. There exists $S \in N$ such that $d(S) = n$ and $S \to \alpha T \beta$. Since π' is complete for G, $[T]_{\pi'} = \{T' \in V' / S' \to \alpha T' \beta'$ for some $S' \in [S]_{\pi'}$ and $\beta' \in V'^*\}$. Since $[S]_{\pi'} = [S]_\pi$ and π is a strict partition, it follows that $[T]_{\pi'} \subset [T]_\pi$. But $[T]_\pi \subset [T]_{\pi'}$ since $\pi \leq \pi'$. Hence $[T]_\pi = [T]_{\pi'}$. □

Remark 3.4. This result is similar to proposition 2.4.

Proposition 3.5. *If $G(X, N, \vec{A})$ is π-strict deterministic the associated ε-reduced (resp. A-reduced) grammar G' is π'-strict deterministic (where π' is the restriction of π to the alphabet of G').*

Proof. Theorem 2.1 of [18]. □

Remark 3.6. The A-reduction preserves the completeness of a strict deterministic grammar but the $\emptyset$-reduction does not.

Example 3.7. The following grammar $G(X, N, \vec{A})$ is π-complete deterministic:

$$\begin{cases} A_1 = aS_1A_1, \\ A_2 = aS_1A_2 + aS_2c + aS_2d + b, \\ S_1 = cS_1 + cS_2S_2, \\ S_2 = d + cS_2S_1R, \\ U = cA_1 + cA_2 + d, \\ R = aR + bRR. \end{cases}$$

$$\vec{A} = (A_1, A_2), \quad \pi = \{\{A_1, A_2\}, \{S_1, S_2\}, \{U\}, \{R\}, \{a, b\}, \{c, d\}\}.$$

By deleting U we get an A-reduced π-complete deterministic grammar.

It is clear that $L(G, A_1) = L(G, R) = \emptyset$. Hence, by $\emptyset$-reduction we get the grammar G' with axiom $(\emptyset, A_2)$:

$$\begin{cases} A_2 = aS_1A_2 + aS_2c + aS_2d + b, \\ S_1 = cS_1 + cS_2S_2, \\ S_2 = d. \end{cases}$$

It is π'-strict deterministic for $\pi' = \{\{A_2\}, \{S_1, S_2\}, \{a, b\}, \{c, d\}\}$ but not π'-complete deterministic since at least one element of the form cS_2S_1w for some $w \in V^*$ is missing in $R(G', S_1) \cup R(G', S_2)$.

3.2. Languages generated by SD *and* CD *grammars*

Definition 3.8. Let π be a partition of an alphabet X. Two words $u, u' \in X^*$ are *π-conjugated* iff $\{u, u'\}$ is a π-strict language. Hence a language $L \subset X^*$ is π-strict iff every two elements of L are π-conjugated.

Lemma 3.9. *Let $G(X, N)$ be a grammar and π a partition of $V = X \cup N$. Condition P1 is equivalent to the following:*

For all $S, S' \in N$, for all $m, m' \in V^*$ such that $S \equiv S'$, $S \to m$ and $S' \to m'$, m and m' are π-conjugated and if $m = m'$ then $S = S'$. □

This condition extends to derivations:

Proposition 3.10. *Let $G(X, N)$ be a π-strict deterministic grammar and $S, S' \in N$ such that $S \equiv S'$.*

If $u, u' \in V^*$, $S \Rightarrow^n u$ *and* $S' \Rightarrow^n u'$, *or if* $u, u' \in X^*$, $S \to^* u$ *and* $S' \to^* u'$ *then* u *and* u' *are* π*-conjugated. If* $u = u'$ *then* $S = S'$.

Proof. Slight modifications in the proofs of Lemma 2.2 and Theorem 2.2 of [18]. □

Proposition 3.11. *Let* $G(X, N)$ *be* π*-complete deterministic. For all* $S \in N$, $T, T' \in V$, $\alpha, \beta \in V^*$ *if* $S \Rightarrow^n \alpha T \beta$ *and* $T' \equiv T$ *then* $S' \Rightarrow^m \alpha T' \beta'$ *for some* $S' \equiv S$, $\beta' \in V^*$ *and* $m \leq n$.

Proof. By induction on n. Cases $n = 0$ and $n = 1$ are clear. We prove case $n+1$ assuming that the preceding ones are proved.

Let $S \to \gamma \Rightarrow^n \alpha T \beta$.

Subcase 1: $\gamma = \gamma_1 T \gamma_2$, $\gamma_1 \Rightarrow^{n_1} \alpha$, $\gamma_2 \Rightarrow^{n_2} \beta$ and $n = n_1 + n_2$. Since G is π-CD, $S' \to \gamma_1 T' \gamma_2'$ for some $S' \equiv S$ and $\gamma_2' \in V^*$. Hence $S' \Rightarrow^{n_1+1} \alpha T' \gamma_2'$ and $n_1 + 1 \leq n + 1$. Taking now $\beta = \gamma_2'$ and $m = n_1 + 1$, we are done.

Subcase 2: $\gamma = \gamma_1 R \gamma_2$, $\gamma_1 \Rightarrow^{n_1} u \in X^*$, $R \Rightarrow^{n_2} \alpha_1 T \beta_1$ with $u\alpha_1 = \alpha$ and $\beta_1 \gamma_2 \Rightarrow^{n_3} \beta$ with $n = n_1 + n_2 + n_3$. By the inductive assumption, $R' \Rightarrow^{m_2} \alpha_1 T' \beta_1'$ for some $R' \equiv R$, $m_2 \leq n_2$ and $\beta_1' \in V^*$. Since G is π-CD, $S' \to \gamma_1 R' \gamma_2'$ for some $S' \equiv S$ and $\gamma_2' \in V^*$. Hence

$$S' \to \gamma_1 R' \gamma_2' \overset{n_1}{\Rightarrow} u R' \gamma_2' \overset{m_2}{\Rightarrow} u\alpha_1 T' \beta_1' \gamma_2', \quad \text{i.e. } S' \overset{m}{\Rightarrow} \alpha T' \beta'$$

with $m = 1 + n_1 + m_2 \leq n + 1$ and $\beta' = \beta_1' \gamma_2'$. □

Proposition 3.12. (1) *Let* $G(X, N, \vec{A})$ *be* π*-strict deterministic. Then* $\vec{L}(G)$ *is* π*-strict and* $\nu_X(\vec{L}(G)) \leq \nu(G) \cap X$.

(2) *Let* $G(X, N, \vec{A})$ *be* π*-complete deterministic and* $\emptyset$*-reduced. Then* $\vec{L}(G)$ *is* π*-complete. If* G *is reduced, then* $\nu_X(L(\vec{G})) = \nu(G) \cap X$.

Proof. (1) From Lemma 3.9 and Proposition 3.10, it follows that $\vec{L}(G)$ is π-strict. Hence $\nu_X(\vec{L}(G)) \leq \nu(G) \cap X$. An example below shows that this inequality can be strict.

(2) Let $uav \in L(G, A_i)$ for some $a \in X$, $u, v \in X^*$. Hence $A_i \Rightarrow^n uav$ for some n. Let $b \equiv a$. By Proposition 3.11, $A_j \Rightarrow^m ub\beta'$ for some $1 \leq j \leq k$, $m \leq n$ and $\beta' \in V^*$. Since G is $\emptyset$-reduced, $\beta' \to^* w \in X^*$ and $ubw \in L(G, A_j)$. Hence $\vec{L}(G)$ is π-complete. If G is A-$\emptyset$-reduced, the set X' of terminal symbols occurring in G is the minimal alphabet of $\vec{L}(G)$ and $\pi \cap X' = \nu_X(\vec{L}(G)) \cap X'$ by Proposition 2.4. It follows that $\nu(G) \cap X = \nu_X(\vec{L}(G))$, since each element of $X - X'$ is alone in its block with respect to either $\nu(G)$ or $\nu_X(\vec{L}(G))$. □

Example 3.13. Let $G(X, N, S_0)$ be the grammar:

$$\begin{cases} S_0 = T_1 + T_2, \\ T_1 = aU_1 + aU_3, \\ T_2 = bU_2 + bU_3, \\ U_1 = da + db, \\ U_2 = dc + dd, \\ U_3 = c. \end{cases}$$

It is strict deterministic with minimal partition

$$\nu(G) = \{\{S_0\}, \{T_1, T_2\}, \{U_1, U_2, U_3\}, \{a, b, c, d\}\}$$

but not complete deterministic for any partition. It generates the π-complete language $L = \{ac, ada, adb, bc, bdc, bdd\}$ with $\pi = \{\{a, b\}, \{c, d\}\}$. But $\pi = \nu_X(L) < \nu(G) \cap X$.

The following proposition is a converse of Proposition 3.12(2). Example 3.13 shows that $\vec{L}(G)$ must be assumed π-complete and not only complete.

Proposition 3.14. *Let $G(X, N, \vec{A})$ be π-strict deterministic and reduced. If $\vec{L}(G)$ is π-complete, then G is π-complete deterministic.*

Notations. For $\sigma \subset V$, let $L(\sigma) = \bigcup \{L(G, T)/T \in \sigma\}$. For $\alpha \in V^*$, let $\gamma(S, \alpha) = \{T \in V/S \to \alpha T \beta \text{ for some } \beta \in V^*\}$.

Lemma 3.15. *Let $\sigma = \{S_1, \ldots, S_k\} \subset \sigma' \subset N$ such that σ' is a block of π. Let $\alpha \in V^*$ such that $\tau = \bigcup \{\gamma(S, \alpha)/S \in \sigma\} \neq \emptyset$. If $L(\sigma)$ is π-complete, then $\sigma = \sigma'$ and $L(\tau)$ is π-complete.*

Proof. The language $L(\sigma')$ is π-strict and $L(\sigma) \subset L(\sigma')$. Hence $L(\sigma) = L(\sigma')$. If $S \in \sigma'$ then $L(S) \neq \emptyset$ and $L(S) \subset L(\sigma)$. Hence $S \in \sigma$ by Proposition 3.10.

Let $\tau = \{T_1, \ldots, T_h\} = \bigcup \{\gamma(S, \alpha)/S \in \sigma\} \subset V$ and $u \in X^*$ such that $\alpha \to^* u$.

For $1 \leqslant i \leqslant h$, let $L_i = \bigcup \{L(G, \beta)/S \to \alpha T_i \beta \text{ for some } S \in \sigma \text{ and } \beta \in V^*\}$. Clearly, $L' = L(T_1)L_1 \cup \cdots \cup L(T_h)L_h \subset u \backslash L(\sigma)$. Let $v = uv' \in L(\sigma)$ and $S \to \gamma \Rightarrow^* uv'$ for some $S \in \sigma$. By the left part theorem [19] (but a direct proof is easy) $\gamma = \alpha\gamma'$ and $\gamma' \to^* v'$. Hence $\gamma' = T_i\beta$ for some $1 \leqslant i \leqslant h$ and $\beta \in V^*$ and $v' \in L(T_i)L(G, \beta)$. Hence $L' = u \backslash L(\sigma)$. The language L' is π-complete since $L(\sigma)$ is (Proposition 2.5(1)) and so is the h-language $(L(T_1), \ldots, L(T_h))$ by the following extension of Proposition 2.5(2):

Lemma 3.16. *If $L_1, \ldots, L_h, L'_1, \ldots, L'_h$ are nonempty languages such that $(L_1, \ldots, L_h)$ is prefix-free, then $L_1L'_1 \cup \cdots \cup L_hL'_h$ is π-complete iff $L'_1, \ldots, L'_h$ and $(L_1, \ldots, L_h)$ are π-complete.*

Hence $L(\tau) = \bigcup\{L(T_i)/1 \leq i \leq h\}$ is π-complete. □

Proof of Proposition 3.14. Without loss of generality we assume that each component of $\vec{A}$ is different from $\emptyset$.

It follows from Lemma 3.15 and the assumptions that N_0 is a block of π. We need only prove that for all $S \in N$, $\delta \in V^*$ and $T \in \gamma(S, \delta)$ we have $[T]_\pi = \bigcup\{\gamma(S', \delta)/S' \in [S]_\pi\}$. We will prove by induction on $d(S)$ (see the proof of proposition 3.3) that, for all $S \in N$, $\delta \in V^*$ such that $S \to \delta T\beta$:

$$\eta = \bigcup\{\gamma(S', \delta)/S' \in [S]_\pi\} \text{ is a block of } \pi \text{ and } L(\eta) \text{ is } \pi\text{-complete.}$$

Note first that η is always contained in some block η' of π. Hence by Lemma 3.15, $\eta = \eta'$ and η is a block of π if $L(\eta)$ is π-complete hence $\eta = [T]_\pi$).

The inductive argument proceeds as follows:

Basis: $d(S) = 0$. Lemma 3.15 with $\sigma = [S]_\pi = N_0$ and $\alpha = \delta$ shows that $L(\eta)$ is π-complete.

Inductive step: $d(S) = n + 1$. Let $R \in N$ such that $d(R) = n$ and $R \to \delta' S\beta'$. The inductive assertion can be applied to (R, δ') and shows that $L([S]_\pi)$ is π-complete. Then Lemma 3.15 with $\sigma = [S]_\pi$ and $\alpha = \delta$ shows that $L(\eta)$ is π-complete.

Since G is A-reduced, $d(S) < \infty$ for all $S \in N$ and we have proved that η defined above is a block of π for every $S \in N$ and $\delta \in V^*$, hence that G is π-complete deterministic. □

We now recall that strict deterministic grammars generate deterministic languages.

Proposition 3.17. *Let $G(X, N, \vec{A})$ be strict deterministic with $\vec{A} \in (N \cup \{\emptyset\})^k$. One can construct a* DPDA *a and $\theta \in (2^Q)^k$ (Q is the set of states of a) such that $\vec{L}(G) = \vec{N}_\theta(a)$.*

Proof. Let G be π-strict deterministic and $\emptyset$-reduced. Let $n = \text{Max}\{\text{Card}(\alpha)/\alpha \in \pi \cap N\}$. We can assume that $\vec{A} = (A_1, \ldots, A_k) \in N^k$ without loss of generality.

Harrison and Havel give in [18] the construction of a DPDA $a = \langle Q, X, \Gamma, \delta, q_1, Z_0\rangle$ with set of states $Q = \{q_1, \ldots, q_n\}$ such that for any $u \in X^*$ and $1 \leq i \leq k$, $A_i \to^* u$ iff $(q_1, u, Z_0) \vdash^* (q_n, \varepsilon, \varepsilon)$ [18, Lemma 3.5, Claims 2 and 3].

Hence $\vec{L}(G) = \vec{N}_\theta(a)$ with $\theta = (\{q_1\}, \{q_2\}, \ldots, \{q_k\})$. □

We introduce a new class of grammars which lies halfway between SD and CD grammars. These grammars arise from the converse of Proposition 3.17. Let π be a partition of $V = X \cup N$.

Definition 3.18. A grammar $G(X, N, \vec{A})$ is *π-semicomplete deterministic* if it satisfies conditions P0, P1 and P4 (hence is π-strict deterministic) and the following condition which is weaker than P2:

P'2: for all $S, T, T' \in N$, for all $\alpha, \beta \in V^*$ such that $T' \equiv T$ and $S \to \alpha T \beta$ there exists $S' \equiv S$ and $\beta' \in V^*$ such that $S' \to \alpha T' \beta'$.

The only difference with P2 is that T, T' range here over N (the non-terminal alphabet) and not over V (the full alphabet of the grammar). We use the abbreviation π-SCD for π-semicomplete deterministic.

Lemma 3.19. *Let X, Y and N be finite disjoint alphabets with partitions π, π' and ω respectively. Let $\phi: X \to Y$ be a mapping such that for each $\sigma \in \pi$, $\phi(\sigma) \subset \sigma'$ for some $\sigma' \in \pi'$ and the restriction of ϕ to σ is one-to-one.*

Let $G(X, N, \vec{A})$ be $(\pi \cup \omega)$-semicomplete deterministic. Then, its image $G'(Y, N, \vec{A})$ by ϕ is $(\pi' \cup \omega)$-semicomplete deterministic.

Intuitively, one obtains G' from G by replacing everywhere a terminal symbol $a \in X$ by $\phi(a)$ which belongs to Y. Non-terminals symbols are not modified. Since $a \equiv b \pmod{\pi}$ implies $(\phi(a) \equiv \phi(b) \pmod{\pi'}$ and $a = b$ iff $\phi(a) = \phi(b))$, G' is π'-SCD if G is π-SCD (but we do not give the tedious details of a complete proof).

We now recall the construction of a context-free grammar which generates the language accepted by a given PDA [16, 18].

Let $\mathcal{A} = \langle Q, X, \Gamma, \delta, q_0, Z_0 \rangle$ be a DPDA and $\theta = (\theta_1, \ldots, \theta_k)$ such that $\theta_i \subset Q$ and $\theta_i \cap \theta_j = \emptyset$ for $1 \leq i, j \leq k$ and $i \neq j$. Let $\bar{\theta} = \bigcup_{1 \leq i \leq k} \theta_i$.

Let $\mathrm{Alph}(Q \times \Gamma \times Q)$ be an alphabet in one-to-one correspondence with $Q \times \Gamma \times Q$. Its generic element is denoted $\overline{qZq'}$.

Let $N_1 = \{A_1, A_2, \ldots, A_k\} \cup \mathrm{Alph}(Q \times \Gamma \times Q)$. Let $G_1(X, N_1, \vec{A})$ be the grammar with axiom $\vec{A} = (A_1, \ldots, A_k)$ such that

$$R(G_1, A_i) = \{\overline{q_0 Z_0 q} / a \in \theta_i\} \quad \text{for } 1 \leq i \leq k,$$

$$R(G_1, \overline{qZq'}) = \{a / \delta(q, a, Z) = (q', \varepsilon) \text{ and } a \in X \cup \{\varepsilon\}\}$$
$$\cup \{a p \overline{Z_1 q_1} \overline{q_1 Z_2 q_2} \cdots \overline{q_k Z_{k+1} q'} / a \in X \cup \{\varepsilon\}, q_1, \ldots,$$
$$q_k \in Q \quad \text{and} \quad \delta(q, a, Z) = (p, Z_1, \ldots, Z_{k+1})\}.$$

Claim [16, Lemma 2.5.3]. *For* $u \in X^*$, $\overline{qZq'} \to^*_{G_1} u$ *iff* $(q, u, Z) \vdash^*_a (q', \varepsilon, \varepsilon)$.

Let $G(X, N, \vec{A})$ be the reduced grammar of G_1.

Let μ_1 be the partition of N_1 whose blocks are $\{A_1, \ldots, A_k\}$ and $\{\overline{qZq'}/q' \in Q\}$ for $q \in Q$ and $Z \in \Gamma$. Let $\mu = \mu_1 \cap N$. Let $\omega_1 = \{\{X\}\} \cup \mu_1$ and $\omega = \{\{X\}\} \cup \mu$. If a partition π of X is already given we define $\bar{\pi}_1 = \pi \cup \mu_1$ and $\bar{\pi} = \pi \cup \mu$.

Theorem 3.20. *With the above hypotheses and notations,*

(1) $L(\vec{G}_1) = L(\vec{G}) = \vec{N}_\theta(a)$

(2) G_1 *is* ω_1*-strict deterministic and* G *is* ω*-strict deterministic.*

Assume now that a *is faithful and* $Q = \theta = \bigcup \{\theta_i / 1 \leqslant i \leqslant k\}$

(3) G *is* ω*-semicomplete deterministic*

(4) *If* a *is* π*-complete then* G *is* $\bar{\pi}$*-complete deterministic.*

Proof. Part 1 follows from the claim. Part 2 is proved in [18. Lemma 3.2]. We first prove part 4:

Since a is π-strict, G_1 is $\bar{\pi}_1$-strict deterministic: it is clear from the definition of G_1. Hence G is $\bar{\pi}$-strict deterministic. Since a is π-complete and $Q = \bar{\theta}$, $\vec{N}_\theta(a)$ is $\bar{\pi}$-complete and $\vec{L}(G) = \vec{N}_\theta(a)$. Hence G is $\bar{\pi}$-complete deterministic by proposition 3.14.

We now prove part 3. Let $\check{a}$ associated with a by Theorem 2.14, $\check{G}_1$ and $\check{G}$ associated with $\check{a}$ by the above constructions Let $\phi : \check{X} \to X$ be the canonical mapping of Proposition 2.7. Clearly, $\phi(\check{G}_1) = G_1$ and $\phi(\check{G}) = G$. Since $\check{G}$ is complete deterministic, G is ω-semicomplete deterministic by Lemma 3.19. □

Example 3.21. Let $a = \langle \{p, q\}, \{a, b, c\}, \{S, T\}, \delta, p, S \rangle$ with $\theta = (\{p\}, \{q\})$ and a transition function δ such that:

$$\delta(p, a, S) = (p, SST) \qquad \delta(p, a, T) = (p, \varepsilon)$$

$$\delta(p, b, S) = (p, S) \qquad \delta(p, b, T) = (p, ST)$$

$$\delta(p, c, S) = (q, \varepsilon) \qquad \delta(q, a, T) = (q, T)$$

$$\delta(q, \varepsilon, S) = (q, \varepsilon) \qquad \delta(q, c, T) = (p, \varepsilon).$$

Grammar $G_1(X, N_1, \vec{A})$ is then:

$$\begin{cases} A_1 = \overline{pSp} \\ A_2 = \overline{pSq} \end{cases}$$

$$\begin{cases} \overline{pSp} = a\,\overline{pSp}\,\overline{pSp}\,\overline{pTp} + b\,\overline{pSp} \\ \qquad\quad + a\,\overline{pSp}\,\overline{pSq}\,\overline{qTp} \\ \qquad\quad + a\,\overline{pSq}\,\overline{qSp}\,\overline{pTp} \\ \qquad\quad + a\,\overline{pSq}\,\overline{qSq}\,\overline{qTp} \\ \overline{pSq} = a\,\overline{pSp}\,\overline{pSp}\,\overline{pTq} + b\,\overline{pSq} + c \\ \qquad\quad + a\,\overline{pSp}\,\overline{pSq}\,\overline{qTq} \\ \qquad\quad + a\,\overline{pSq}\,\overline{qSp}\,\overline{pTq} \\ \qquad\quad + a\,\overline{pSq}\,\overline{qSq}\,\overline{qTq} \end{cases}$$

$$\begin{cases} \overline{qSp} = \emptyset \\ \overline{qSq} = \varepsilon \end{cases}$$

$$\begin{cases} \overline{pTp} = a \quad + b\,\overline{pSp}\,\overline{pTp} \\ \qquad\qquad + b\,\overline{pSq}\,\overline{qTp} \\ \overline{pTq} = \quad b\,\overline{pSp}\,\overline{pTq} \\ \qquad\qquad + b\,\overline{pSq}\,\overline{qTq} \end{cases}$$

$$\begin{cases} \overline{qTp} = a\,\overline{qTp} \quad + c \\ \overline{qTq} = a\,\overline{qTq} \end{cases}$$

The different blocks of partition μ_1 are shown by braces. After reduction we get $G(X, N, \vec{A})$ which is clearly ω-SCD:

$$\begin{cases} A_1 = \overline{pSp} \\ A_2 = \overline{pSq} \end{cases}$$

$$\begin{cases} \overline{pSp} = a\,\overline{pSp}\,\overline{pSp}\,\overline{pTp} + b\,\overline{pSp} \\ \qquad\quad + a\,\overline{pSp}\,\overline{pSq}\,\overline{qTp} \\ \qquad\quad + a\,\overline{pSq}\,\overline{qSq}\,\overline{qTp} \\ \overline{pSq} = \quad b\,\overline{pSq} \quad + c \end{cases}$$

$$\begin{cases} \overline{qSq} = \varepsilon \end{cases}$$

$$\begin{cases} \overline{pTp} = a \quad + b\,\overline{pSp}\,\overline{pTp} \\ \qquad\qquad + b\,\overline{pSq}\,\overline{qTp} \end{cases}$$

$$\begin{cases} \overline{qTp} = a\,\overline{qTp} \quad + c. \end{cases}$$

Corollary 3.22. *Let a be a* DPDA *such that $\varepsilon \notin N(a)$. One can construct $G'(X, N', \vec{A})$ in Greibach normal form which satisfies properties* 1 *to* 4 *of G in Theorem* 3.20.

Proof. Let G be the reduced grammar obtained in Theorem 3.20. We show how to put it in Greibach normal form. Since G is associated with a *deterministic* push-down automaton, for all $S \in N$, one and only one of the following three cases occurs:

(i) $R(G, S) \subset XN^*$,

(ii) $R(G, S) = \{\varepsilon\}$ and S is alone in its block,

(iii) $R(G, S) \subset N^*$.

In order to put G in Greibach normal form, we first delete S from N and replace it by ε everywhere in right hand sides of equations when case (ii) occurs.

Secondly, Theorem 2.3 of [18] shows that $S \Rightarrow^n Tm$ with $n \geq 1$, $S, T \in N$ and $m \in (X \cup N)^*$ implies $S \neq T$; in particular, there is no left recursion in G. The transformation in Greibach normal form proceeds by remplacing a member Tm of some right-hand side of G (where $T \in N$) by $m_1 m + m_2 m + \cdots + m_k m$ if $m_1 + m_2 + \cdots + m_k$ is the right hand side of the equation of G associated with T. Since there is no left recursion in G (see [18]) the process can be repeated, giving at the end a grammar $G'(X, N, \vec{A})$ in Greibach normal form. Note that $N' \subset N$.

One can prove that properties 1 to 4 of Theorem 3.12 are preserved during the transformation. □

Example 3.23 (Continuation of 3.21.) By reduction to Greibach normal form, we get the following grammar $G'(X, N', \vec{A})$:

$$\begin{cases} \overline{pSp} = a\,\overline{pSp}\,\overline{pSp}\,\overline{pTp} + b\,\overline{pSp} \\ \qquad\quad + a\,\overline{pSp}\,\overline{pSq}\,\overline{qTp} \\ \qquad\quad + a\,\overline{pSq}\,\overline{qTp} \end{cases}$$

$$\overline{pSq} = \qquad\qquad b\,\overline{pSq} \qquad + c$$

$$\begin{cases} \overline{pTp} = a \qquad\qquad + b\,\overline{pSp}\,\overline{pTp} \\ \qquad\qquad\qquad\quad + b\,\overline{pSq}\,\overline{qTp} \end{cases}$$

$$\begin{cases} \overline{qTp} = a\,\overline{qTp} \qquad\qquad + c \end{cases}$$

with $R(G', A_1) = R(G', \overline{pSp})$ and $R(G', A_2) = R(G', \overline{pSq})$.

We conclude this section by summing up our results in the following theorems:

Theorem 3.24. *A (multi)-language is π-strict and deterministic (resp. π-complete and deterministic) iff it is generated by a π-strict deterministic (resp. π-complete deterministic $\emptyset$-reduced) grammar.*

Theorem 3.25. *The equivalence problems for strict deterministic grammars, complete deterministic grammars and* DPDA*'s are interreducible.*

In the second part of this work, these results will be applied to decision problems for context-free tree grammars.

References

[1] A. Arnold and M. Dauchet, Un théorème de duplication pour les forêts algébriques, *J. Comput. Systems Sci.* **13** (1976) 223–244.

[2] A. Arnold and M. Dauchet, Forêts algébriques et homomorphismes inverses, Rapport no. 55, Laboratoire de Calcul de l'Université de Lille, France (1975).

[3] G. Boudol, Langages polyadiques algébriques, Thèse de 3ème cycle, Université de Paris 7 (1976).

[4] B. Courcelle, Une forme canonique pour les grammaires simples déterministes, *Rev. Française d'Automat. Informat. Recherche Opérationnelle, Sér. Rouge* (1974) 19–36.

[5] B. Courcelle, Recursive schemes, algebraic trees and deterministic languages, *Proc. 15th Annual Symp. on Switching and Automata Theory*, New-Orleans (1974) 52–62.

[6] B. Courcelle, Sur les ensembles algébriques d'arbres et les langages déterministes, quelques applications à la théorie des schémas de programmes, Thèse, Université de Paris 7 (1976).

[7] B. Courcelle, On jump-deterministic pushdown automata, *Math. System Theory* (to appear).

[8] B. Courcelle and M. Nivat, Algebraic families of interpretations, *17th Ann. Symp. on F.O.C.S.*, Houston (1976).

[9] B. Courcelle and J. Vuillemin, Completeness results for the equivalence of recursive schemas, *J. Comput. System Sci.* **12** (1976) 179–197.

[10] J.E. Doner, Tree acceptors and some of their applications, *J. Comput. System Sci.* **4** (1970) 406–451.

[11] J. Engelfriet, Bottom-up and top-down tree transformations – A comparison, *Math. System Theory* **9** (1975) 199–231.

[12] J. Engelfriet and E.M. Schmidt, IO and OI, Report PB-47, Aarhus University, Denmark (1975).

[13] M. Fischer, Grammars with macro-like instructions, *Proc. 9th Ann. Symp. on Switching and Automata Theory* (1968) 131–142.

[14] E. Friedman, Equivalence problems for deterministic context-free languages and monadic recursion schemes, J. Comput. System Sci. **14** (1977) 344–359.

[15] E. Friedman, The inclusion problem for simple languages, *Theoret. Comput. Sci.* **1** (1976) 297–316.

[16] S. Ginsburg, *The Mathematical Theory of Context-Free Languages* (McGraw-Hill, New York, 1966).

[17] J. Goguen et al., Initial algebra semantics, *J. Assoc. Comput. Mach.* **24** (1977) 68–95.

[18] M.A. Harrison and I.M. Havel, Strict deterministic grammars, *J. Comput. System Sci.* **7** (1973) 237–277.

[19] M.A. Harrison and I.M. Havel, On the parsing of deterministic languages, *J. Assoc. Comput. Mach.* **21** (1974) 525–548.

[20] N. Honda and M. Oyamaguchi, The decidability of equivalence for deterministic stateless pushdown automata (submitted for publication).

[21] J. E. Hopcroft and A.J. Korenjak, Simple deterministic languages, *Proc. 7th Symp. on Switching and Automata Theory*, Berkeley (1966) 36–46.

[22] M. Nivat, Eléments de la théorie générale des codes, in: Caianiello ed., *Automata Theory* (Academic Press, London, 1966) 278–294.
[23] M. Nivat, On the interpretation of recursive programs schemes, *Symposia Mathematica* **15** (1975).
[24] M. Nivat, Interprétation universelle d'un schéma de programme recursif, *Proc. Int. School on Theory and Applications of Computers*, Erice (May 1976).
[25] B. Rosen, Tree-manipulating systems and church-rosser theorems, *J. Assoc. Comput. Mach.* **20** (1973) 160–187.
[26] B. Rosen, Program equivalence and context-free grammars, *J. Comput. System Sci.* **11** (1975) 358–374.
[27] D.J. Rosenkrantz and R.E. Stearns, Properties of deterministic top-down grammars, *Information and Control* **17** (1970) 226–256.
[28] W.C. Rounds, Tree-oriented proofs of some theorems on context-free and indexed languages, *Proc. 2nd ACM Symp. on Computing* (1970) 109–116.
[29] M. P. Schützenberger, On context-free languages and push-down automata, *Information and Control* **16** (1963) 246–264.
[30] R.E. Stearns, A regularity test for pushdown machines, *Information and Control* **11** (1967) 323–340.
[31] J. Thatcher, Tree automata: An informal survey, in: A. Aho, ed., *Currents in the Theory of Computing* (Prentice-Hall, Englewood Cliffs, NJ, 1973) 143–172.
[32] L.G. Valiant, The equivalence problem for deterministic finite turn pushdown automata, *Information and Control* **25** (1974) 123–133.
[33] L.G. Valiant, Decision procedures for families of deterministic pushdown automata, Report no. 7, Warwick University, England (1973).