THE EQUIVALENCE PROBLEM FOR LL- AND LR-REGULAR GRAMMARS[1]

(extended abstract)

Anton Nijholt
Koninginneweg 179
1075 CP Amsterdam The Netherlands

1. INTRODUCTION

Questions whether or not two grammars belonging to a family of grammars generate the same language have extensively been studied in the literature. These problems are called equivalence problems and if there exists an algorithm which for each pair of grammars of this family gives an answer to this question then the equivalence problem for this family of grammars is said to be decidable. Otherwise the problem is said to be undecidable. For example, the equivalence problem for the family of regular grammars is decidable. On the other hand, the equivalence problem for the family of context-free grammars is known to be undecidable.

The equivalence problem is open for various classes of grammars which generate deterministic languages. For simple deterministic and LL(k) grammars the problem has been solved. In this paper we study the equivalence problem for the class of LL-regular grammars and languages. The class of LL-regular grammars is obtained from the class of LL(k) grammars by allowing regular look-ahead instead of finite look-ahead, cf. Jarzabek and Krawczyk |8|, Nijholt |10,11,12| and Poplawski |16| for results on LL-regular grammars and languages. The class of LL(k) grammars is properly included in the class of LL-regular grammars and the class of LL(k) languages is properly included in the class of LL-regular languages. Contrary to the other families of languages which have been studied from the point of view of the equivalence problem, the class of LL-regular languages contains languages which are not deterministic.

It will be shown that the equivalence problem for LL-regular grammars is decidable. Apart from extending the known result for LL(k) grammar equivalence to LL-regular grammar equivalence, we obtain an alternative proof of the decidability of

---

(1) The preparation of this paper was partially supported by a Natural Sciences and Engineering Research Council of Canada Grant No. A-7700 during the author's stay at McMaster University, Hamilton, Ontario, Canada.

LL(k) equivalence. From |2i| we understand that the equivalence problem for LL-regular grammars has been studied before, but not solved. Our proof that this equivalence problem is decidable is simple. However, this is mainly because we can reduce the problem to the equivalence problem for real-time strict deterministic grammars, which is decidable, see Oyamaguchi, Honda and Inagaki |15| and Ukkonen |18|.

In this extended abstract the proofs of the theorems are not included. A complete paper will appear elsewhere.

Preliminaries

We assume that the reader is familiar with Aho and Ullman |1| or Harrison |3|. For notational reasons we review some concepts.

A context-free grammar (CFG for short) is denoted by the quadruple $G = (N,\Sigma,P,S)$, where N consists of the nonterminal symbols, Σ consists of the terminal symbols, $N \cap \Sigma = \emptyset$ (the empty set); $N \cup \Sigma$ is denoted by V (elements of V will be denoted by X, Y and Z; elements of $V^\times$ will be denoted by α, β, γ, δ and ω). We use ε to denote the empty word. The elements of $\Sigma^\times$ will be denoted by x, y, z and w. The set P of productions is a subset of $N \times V^\times$ (notation $A \to \alpha$ if (A,α) is in P) and $S \in N$ is called the start symbol of the grammar.

We have the usual notation $\Rightarrow$, $\underset{L}{\Rightarrow}$ and $\underset{R}{\Rightarrow}$ for derivations, leftmost derivations and rightmost derivations, respectively. The superscripts + and $\times$ will be used to denote the transitive and the reflexive-transitive closures of these relations.

For any string $\alpha \in V^\times$ define

$$L(\alpha) = \{w \in \Sigma^\times \mid \alpha \overset{\times}{\Rightarrow} w\}.$$

The language L(G) of a CFG G is the set L(S). Two grammars G_1 and G_2 are said to be equivalent if $L(G_1) = L(G_2)$.

For any string $\alpha \in V^\times$ we use α^R to denote the reverse of α. If L is a set of strings, then $L^R = \{w^R \mid w \in L\}$. If $\alpha \in V^\times$ then $|\alpha|$ denotes the length of α. For any $\alpha \in V^\times$ and non-negative integer k we use $k : \alpha$ to denote the prefix of α with length k if $|\alpha| \geq k$ and otherwise $k : \alpha$ denotes α. A production $A \to \varepsilon$ is called an ε-production; a CFG without ε-productions is called an ε-free grammar.

A CFG $G = (N,\Sigma,P,S)$ is said to be right linear if each rule is of the form $A \to uB$ or $A \to u$, with $A,B \in N$ and $u \in \Sigma^\times$. A subset L of $\Sigma^\times$ is said to be regular if there exists a right linear grammar G such that $L(G) = L$.

For any set Q, a partition π of Q is a finite set of mutually disjoint subsets of Q such that each element of Q is in one of these subsets. The elements of a partition are called blocks or equivalence classes. If two elements x and y belong to the same block $B \in \pi$ then we write $x \equiv y \pmod{\pi}$.

DEFINITION 1.1. Let $\pi = \{B_1, B_2, \ldots, B_n\}$ denote a partition of $\Sigma^\times$, where Σ is a finite set, into n blocks. Partition π is said to be a regular partition of $\Sigma^\times$ if all the sets B_i are regular. Partition π is a left congruence (right congruence) if for any strings x, y and z in $\Sigma^\times$, $x \equiv y \pmod{\pi}$ implies $zx \equiv zy \pmod{\pi}$ ($xz \equiv yz \pmod{\pi}$).

A partition $\pi' = \{B_1', B_2', \ldots, B_m'\}$ is a refinement of a regular partition $\pi = \{B_1, B_2, \ldots, B_n\}$ of $\Sigma^\times$ if each B_i of π is the union of some of the blocks of π'. It is well-known that every regular partition of a set $\Sigma^\times$ has a refinement of finite index which is both a left and a right congruence (which we call a congruence for short) (see Hopcroft and Ullman |7|).

In the forthcoming sections it is assumed that the grammars under consideration are reduced. We recall the definitions of strict deterministic and real-time strict deterministic grammars (cf. Harrison and Havel |4,5|).

DEFINITION 1.2. Let $G = (N,\Sigma,P,S)$ be a CFG and let ψ be a partition of V. Partition ψ is called strict if

(i) $\Sigma \in \psi$, and

(ii) For any $A, A' \in N$ and $\alpha, \beta, \beta' \in V^\times$, if
$A \to \alpha\beta$ and $A' \to \alpha\beta'$ are in P and $A \equiv A' \pmod{\psi}$, then either:
(a) both $\beta, \beta' \neq \varepsilon$ and $1 : \beta \equiv 1 : \beta' \pmod{\psi}$,
or
(b) $\beta = \beta' = \varepsilon$ and $A = A'$.

Now a grammar $G = (N,\Sigma,P,S)$ is called strict deterministic if there exists a strict partition of V.

In general, a strict deterministic grammar can have more than one strict partition of V. Let ψ_1 and ψ_2 be two partitions of V with induced equivalence relations $\equiv_1$ and $\equiv_2$, respectively, then $\psi_1 \leq \psi_2$ if and only if $\equiv_1 \subseteq \equiv_2$. The partitions form a semi-lattice with this ordering and under the meet-operation. In Harrison and Havel |4| an algorithm is given which computes the minimal strict partition of a strict deterministic grammar.

A strict deterministic grammar $G = (N,\Sigma,P,S)$ with minimal strict partition ψ is called a real-time strict deterministic grammar if it is ε-free and for all $A, A', B, B' \in N$; $\alpha, \beta \in V^\times$, if $A \to \alpha B$ and $A' \to \alpha B'\beta$ are in P, then $A \equiv A' \pmod{\psi}$ implies $\beta = \varepsilon$.

2. THE EQUIVALENCE PROBLEM FOR GRAMMARS WITH LOOK-AHEAD

One way to generalize definitions of classes of deterministically parsable

grammars is to let the decisions in the parsing process of these grammars be determined by look-ahead of the input string. This look-ahead may be finite or regular. Finite look-ahead is for instance used in the definition of LL(k) and LR(k) grammars. Regular look-ahead is used in the definitions of LL-regular and LR-regular grammars. In this section we will introduce regular look-ahead for strict deterministic and real-time strict deterministic grammars. Then it will be shown how the equivalence problems for these grammars with look-ahead can be reduced to the equivalence problems for strict deterministic and real-time strict deterministic grammars. In the following section we will study LL-regular grammars as a special case of the (real-time) strict deterministic grammars with regular look-ahead.

The generalization which we give here for (real-time) strict deterministic grammars conforms the generalizations in |13| for finite look-ahead. We use the following notation. Let $G = (N,\Sigma,P,S)$ be a CFG and let $\pi = \{B_1, B_2, \ldots, B_n\}$ be a regular partition of Σ^*. For any $\alpha \in V^*$,

$$\mathrm{BLOCK}(\alpha) = \{B_k \in \pi \mid L(\alpha) \cap B_k \neq \emptyset\}.$$

DEFINITION 2.1. A CFG $G = (N,\Sigma,P,S)$ is strong SD(π), where π is a regular partition of Σ^*, if there exists a partition ψ of $V = N \cup \Sigma$ such that

(i) $\Sigma \in \psi$

(ii) For any $w_1, w_2 \in \Sigma^*$; $A, A' \in N$; $\alpha, \beta, \beta', \omega_1, \omega_2 \in V^*$ with $A \equiv A' \pmod{\psi}$ and derivations

(a) $S \overset{*}{\underset{L}{\Rightarrow}} w_1 A \omega_1 \underset{L}{\Rightarrow} w_1 \alpha\beta\omega_1$

(b) $S \overset{*}{\underset{L}{\Rightarrow}} w_2 A' \omega_2 \underset{L}{\Rightarrow} w_2 \alpha\beta'\omega_2$

the condition

$$\mathrm{BLOCK}(\beta\omega_1) \cap \mathrm{BLOCK}(\beta'\omega_2) \neq \emptyset$$

always implies that either

(1) both $\beta, \beta' \neq \varepsilon$ and $1 : \beta \equiv 1 : \beta' \pmod{\psi}$, or

(2) $\beta = \beta' = \varepsilon$ and $A = A'$.

A strong SD(π) grammar $G = (N,\Sigma,P,S)$ with a minimal partition ψ is now called strong real-time SD(π) if G is ε-free and the following condition is satisfied:

For all $A, B, A', B' \in N$ and $\alpha, \beta \in V^*$, if $A \to \alpha B$ and $A' \to \alpha B'\beta$ are in P with $A \equiv A' \pmod{\psi}$ then if

$$S \overset{\times}{\underset{L}{\Rightarrow}} w_1 A \omega_1 \underset{L}{\Rightarrow} w_1 \alpha B \omega_1$$

$$S \overset{\times}{\underset{L}{\Rightarrow}} w_2 A' \omega_2 \underset{L}{\Rightarrow} w_2 \alpha B' \beta \omega_2$$

and

$$\text{BLOCK}(B\omega_1) \cap \text{BLOCK}(B'\beta\omega_2) \neq \emptyset \qquad (\times)$$

then $\beta = \varepsilon$.

Clearly, the real-time strict deterministic grammars are a special case (no look-ahead) of this definition. Notice that because of $(\times)$ $B \equiv B' \pmod{\psi}$.

We now show that the equivalence problem for strong real-time SD(π) grammars is decidable. We start with a strong real-time SD(π) grammar and convert it into a real-time strict deterministic grammar. The conversion will be done in such a way that two strong real-time SD(π) grammars are equivalent if and only if their associated real-time strict deterministic grammars are equivalent.

Let $G = (N,\Sigma,P,S)$ be any CFG without ε-productions and let $\pi = \{B_0,B_1,\ldots,B_n\}$ be a regular partition of $\Sigma^\times$. Without loss of generality we may assume that π is a left congruence and that $B_0 = \{\varepsilon\}$. It follows that $\pi^R = \{B_0^R, B_1^R,\ldots,B_n^R\}$ is a right congruence. Then π^R defines the states and the transitions of a (deterministic) finite automaton $M_\pi = (Q, \Sigma,\delta,q_0)$, where

Q is the set of states, $Q = \{q_0,q_1,\ldots,q_n\}$,
$q_0 \in Q$ is the initial state,
Σ is the input alphabet
$\delta : Q \times \Sigma \to Q$ is the transition function

and δ satisfies

$$B_i^R = \{w \mid \delta(q_0,w) = q_i\}$$

for $0 \leq i \leq n$.

Now let p_0 be a symbol not in Q and let $\perp$ be a special symbol not in Σ. Define a grammar $G_\pi = (N',\Sigma',P',S')$ as follows:

$$N' = \{S'\} \cup (Q \times N \times Q)$$

$$\Sigma' = (Q \cup \{p_0\}) \times (\Sigma \cup \{\perp\}) \times Q$$

and P' contains productions

(i) $S' \to \langle p_0 \perp p \rangle \langle p S q_0 \rangle$ for all $p \in Q$

(ii) If $A \to X_1 X_2 \ldots X_r$ is in P then $\langle pAq \rangle \to \langle pX_1p_1 \rangle \langle p_1X_2p_2 \rangle \ldots \langle p_{r-1}X_rq \rangle$ is in P', for any $p, q, p_1, \ldots, p_{r-1}$ in Q such that if $X_j \in \Sigma$, then $\delta(p_j, X_j) = p_{j-1}$, for $1 < j < r$; if $X_1 \in \Sigma$ then $\delta(p_1, X_1) = p$ and if $X_r \in \Sigma$, then $\delta(q, X_r) = p_{r-1}$.

We can reduce grammar G_π. Throughout this paper, whenever we use the subscript π then we refer to the grammar which is obtained with this construction.

Let G and G_π be as above. Define a homomorphism $\rho: V'^{\times} \to V^{\times}$ by

$\rho(\langle p_0 \perp p \rangle) = \varepsilon$ for every $p \in Q$

$\rho(\langle pXq \rangle) = X$ for each $p, q \in Q$ and $X \in V$

The proofs of the following claims are straightforward and therefore omitted.

CLAIM 2.1. For any $\langle rXs \rangle \in V'$ and $y \in \Sigma'^{\times}$, if $\langle rXs \rangle \overset{\times}{\Rightarrow} y$, then $\delta(s, \rho(y^R)) = r$.

Clearly, this claim can easily be extended to an arbitrary string $\alpha = \langle rX_1s_1 \rangle \langle s_1X_2s_2 \rangle \ldots \langle s_{n-1}X_ns_n \rangle$ in $V'^{\times}$. If $\alpha \overset{\times}{\Rightarrow} y$, where $y \in \Sigma'^{\times}$, then $\delta(s, \rho(y^R)) = r$.

CLAIM 2.2. For any $\langle pXq \rangle \in V'$, if $\langle pXq \rangle \overset{\times}{\Rightarrow} \alpha \langle rYs_1 \rangle \langle s_2Zt \rangle$ for some string $\alpha \langle rYs_1 \rangle \langle s_2Zt \rangle \beta$ in $V'^{\times}$, then $s_1 = s_2$.

CLAIM 2.3. For any $\langle pXq \rangle \in V'$ and $\omega' \in V'^{\times}$, if $\langle pXq \rangle \overset{\times}{\underset{L}{\Rightarrow}} \omega'$ in G_π then $X \overset{\times}{\underset{L}{\Rightarrow}} \rho(\omega')$ in G.

From Claim 2.3 it is immediately clear that $L(G) = \rho(L(G_\pi))$, where we have extended the definition of ρ to sets of strings.

CLAIM 2.4. For any $w, x \in \Sigma'^{\times}$, $\langle pXq \rangle \in V'$ and $\omega \in V'^{\times}$, if $S' \overset{\times}{\underset{L}{\Rightarrow}} w \langle pXq \rangle \omega \overset{\times}{\underset{L}{\Rightarrow}} wx$ in G_π, then $\rho(x) \in B_p$, where B_p is a block of partition $\pi = \{B_0, B_1, \ldots, B_n\}$.

With the help of these claims it is now straightforward to prove the follow-

ing lemmas.

LEMMA 2.1. If G is an ε-free strong SD(π) grammar then G_π is an ε-free strict deterministic grammar.

LEMMA 2.2. If G is a strong real-time SD(π) grammar then G_π is a real-time strict deterministic grammar.

Now consider two ε-free grammars G_1 and G_2 which are strong (real-time) $SD(\pi_1)$ and strong (real-time) $SD(\pi_2)$, respectively. Here π_1 and π_2 are regular partitions of the same set $\Sigma^\times$. Then G_1 and G_2 are both strong (real-time) SD(π) with respect to the regular partition

$$\pi = \{B \mid B_i \cap B_j = B, \quad B \neq \emptyset, B_i \in \pi_1, B_j \in \pi_2\}$$

For π we can construct the sequential machine M_π and the (real-time) strict deterministic grammars G^1_π and G^2_π. Clearly, if $L(G_1) = L(G_2)$ then $L(G^1_\pi) = L(G^2_\pi)$ and if $L(G_1) \neq L(G_2)$ then $L(G^1_\pi) \neq L(G^2_\pi)$. It follows that we have reduced the equivalence problem for strong (real-time) SD-regular grammars to the problem for (real-time) strict deterministic grammars.

Any real-time strict deterministic grammar can be converted into an equivalent real-time deterministic pushdown automaton (cf. Harrison |3|) which accepts with empty stack. In Oyamaguchi, Honda and Inagaki|15| the decidability of the equivalence problem for these automata has been shown.

COROLLARY 2.1. The equivalence problem for strong real-time SD(π) grammars is decidable.

In the following section it will be shown that each strong LL-regular grammar is a strong real-time SD-regular grammar. It is wellknown that strong LL-regular grammars can generate non-deterministic languages. The language

$$L = \{a^n b^k a^n, a^k b^n c^n \mid n \geq 1, k \geq 1\}$$

is an example of a language which is not real-time strict deterministic but it is deterministic. Moreover, L is a strong real-time SD-regular language.

Culik and Cohen |2| use a slightly different method than is presented here to convert an LR-regular grammar into an LR(0) grammar. Clearly, the argument which we gave above holds for LR-regular grammars as well. That is, we have the following proposition:

PROPOSITION 2.1. The equivalence problem for LR-regular grammars is decidable if and only if the equivalence problem for LR(0) grammars is decidable.

3. THE EQUIVALENCE PROBLEM FOR LL-REGULAR GRAMMARS

We start this section with the definition of LL-regular grammars (Nijholt|11|, Poplawski |16|).

DEFINITION 3.1. Let $G = (N,\Sigma,P,S)$ be a CFG and let $\pi = \{B_0,B_1,\ldots,B_n\}$ be a regular partition of Σ^*. Grammar G is an LL(π) grammar if, for each $w, x, y \in \Sigma^*$; $\alpha, \gamma, \delta \in V^*$ and $A \in N$, the conditions

(i) $S \overset{*}{\underset{L}{\Rightarrow}} wA\alpha \underset{L}{\Rightarrow} w\gamma\alpha \overset{*}{\underset{L}{\Rightarrow}} wx$

(ii) $S \overset{*}{\underset{L}{\Rightarrow}} wA\alpha \underset{L}{\Rightarrow} w\delta\alpha \overset{*}{\underset{L}{\Rightarrow}} wy$

(iii) $\mathrm{BLOCK}(\gamma\alpha) \cap \mathrm{BLOCK}(\delta\alpha) \neq \emptyset$

always imply that $\gamma = \delta$.

Notice that if $\mathrm{BLOCK}(\gamma\alpha) \cap \mathrm{BLOCK}(\delta\alpha) \neq \emptyset$ then there exist strings $x \in L(\gamma\alpha)$ and $y \in L(\delta\alpha)$ such that $x \equiv y \pmod{\pi}$. A CFG G is called LL-regular if it is LL(π) for some regular partition π of Σ^*. Notice that a grammar G is LL(k) if G is LL(π_k) for the regular partition

$$\pi_k = \{\{u\} \mid u \in \Sigma^* \text{ and } |u| < k\} \cup \{\{uw \mid w \in \Sigma^*\} \mid u \in \Sigma^k\}$$

where Σ^k is the set of all words over Σ with length k.

As in the case of LL(k) grammars it is possible to define strong LL-regular grammars.

DEFINITION 3.2. Let $G = (N,\Sigma,P,S)$ be a CFG and let $\pi = \{B_1,B_2,\ldots,B_n\}$ be a regular partition of Σ^*. Grammar G is a strong LL(π) grammar if, for each $w_1, w_2, x, y \in \Sigma^*$; $\alpha_1, \alpha_2, \gamma, \delta \in V^*$ and $A \in N$, the conditions

(i) $S \overset{*}{\underset{L}{\Rightarrow}} w_1A\alpha_1 \underset{L}{\Rightarrow} w_1\gamma\alpha_1 \overset{*}{\underset{L}{\Rightarrow}} w_1x$

(ii) $S \overset{*}{\underset{L}{\Rightarrow}} w_2A\alpha_2 \underset{L}{\Rightarrow} w_2\delta\alpha_2 \overset{*}{\underset{L}{\Rightarrow}} w_2y$

(iii) $x \equiv y \pmod{\pi}$

always imply that $\gamma = \delta$.

The class of LL-regular grammars properly includes the class of strong LL-regular grammars. However, the language families coincide. In Poplawski |16| a transformation can be found which converts any LL-regular grammar into an equivalent strong LL-regular grammar. Hence, without loss of generality we may assume that the LL-regular grammars which are considered are strong.

The language

$$L = \{aa^n ba^{2n} b,\ ba^n ba^{2n},\ aa^n ba^n a,\ ba^n ba^n b \mid n \geq 0\}$$

is an example of a non-deterministic language which is LL-regular (cf. |11|). Language

$$L = \{a^n b^k a^n,\ a^k b^n c^n \mid n \geq 1,\ k \geq 1\}$$

is an example of an LL-regular language which is not real-time strict deterministic.

Let G be an LL-regular grammar. The method which is given in |1| for eliminating ε-productions from an LL(k) grammar can easily be modified in order to obtain the result that for every LL-regular grammar we can find an equivalent ε-free LL-regular grammar. As mentioned above, we may assume that the LL-regular grammars under consideration are strong. The proof of the following theorem is again in the complete paper.

THEOREM 3.1. If G is an ε-free strong LL-regular grammar, then G is a strong real-time SD-regular grammar.

From Corollary 2.1 and Theorem 3.1 we may now conclude:

COROLLARY 3.1. The equivalence problem for LL-regular grammars is decidable.

It is natural to ask whether it is possible to convert LL-regular grammar G to an LL(1) grammar G_π. The method which is given in Culik and Cohen |2| yields for each LR-regular grammar G an LR(0) grammar G_π. Therefore it is not necessary to develop a parsing method for LR-regular grammars since the method for LR(0) grammars can be used. Unfortunately, the conversion which we use here does not necessarily yield an LL(1) grammar. In |12| a method has been given which converts an LL(π) grammar G into an LL(1) grammar G' such that $L(G_\pi) \subseteq L(G')$. Here G_π is the grammar which is obtained from LL(π) grammar G with the method described above. If we were able to obtain from LL(π) grammar G an LL(1) grammar G', with $L(G') = L(G_\pi)$ then we should have reduced the equivalence problem for LL-regular grammars to the equivalence problem for LL(1) grammars.

REFERENCES

1. Aho, A. V. and Ullman, J. D. The Theory of Parsing, Translation, and Compiling, Vols. 1 and 2. Prentice Hall, Inc., Englewood Cliffs, N.J., 1972 and 1973.

2. Culik, K. and Cohen, R. LR-regular grammars -- an extension of LR(k) grammars. J. Comput. System Sci. 7 (1973), 66-96.

3. Harrison, M. A. Introduction to Formal Language Theory. Addison-Wesley, Reading, Mass., 1978.

4. Harrison, M. A. and Havel, I. M. Strict deterministic grammars. J. Comput. System Sci. 7 (1973), 237-277.

5. Harrison, M. A. and Havel, I. M. Real-time strict deterministic grammars. SIAM J. Comput. 1 (1972), 333-349.

6. Harrison, M. A. Havel, I. M. and Yehudai, A. An equivalence of grammars through transformation trees. Theoret. Comput. Sci. 9 (1979), 173-206.

7. Hopcroft, J. E. and Ullman, J. D. Formal Languages and their Relation to Automata. Addison-Wesley, Reading, Mass., 1969.

8. Jarzabek, S. and Krawczyk, T. LL-regular grammars. Information Processing Letters 4 (1975), 31-37.

9. Korenjak, A. J. and Hopcroft, J. E. Simple deterministic languages. Conf. Record of 7th Annual Symp. on Switching and Automata Theory 1966, 36-46.

10. Nijholt, A. On the parsing of LL-regular grammars. In: Math. Foundations of Computer Sci., A. Mazurkiewicz (ed.), LNCS 45, Springer, Berlin, 1976, 446-452.

11. Nijholt, A. LL-regular grammars. Int. J. of Computer Math. 8 (1980), 303-318.

12. Nijholt, A. From LL-regular to LL(1) grammars. Report IR-61, Amsterdam, May 1980.

13. Nijholt, A. A framework for classes of grammars between the LL(k) and LR(k) grammars. TR No. 80-CS-25, McMaster University, Hamilton, 1980.

14. Olshansky, T. and Pnueli, A. A direct algorithm for checking equivalence of LL(k) grammars. Theoret. Comput. Sci. 4 (1977), 321-349.

15. Oyamaguchi, M., Honda, N. and Inagaki, Y. The equivalence problem for real-time strict deterministic languages. Information and Control 45 (1980), 90-115.

16. Poplawski, D. A. On LL-regular grammars. J. Comput. System Sci.18 (1979), 218-227.

17. Rosenkrantz D. J. and Stearns, R. E. Properties of deterministic top-down grammars. Information and Control 17 (1970), 226-255.

18. Ukkonen, E. A decision method for the equivalence of some non-real-time deterministic pushdown automata. 12th Ann. S. on Theory of Computing, 1980, 29-38.

19. Wood, D. Some remarks on the KH algorithm for s-grammars. BIT 13 (1973), 476-489.

20. Wood, D. Lecture notes on top-down syntax analysis. J. of the Computer Society of India 8 (1978), 1-22.

21. Zubenko, V. V. Simple pushdown storage automata and the equivalence problem in certain classes of LL(π) grammars (in Russian). Theory and Practice of systems programming, Inst. Kibernet., Akad. Nauk Ukrain. SSR, Kiev, 1976.