JOURNAL OF COMPUTER AND SYSTEM SCIENCES 24, 149–161 (1982)

The Equivalence Problem for *LL*- and *LR*-Regular Grammars*

ANTON NIJHOLT

Informatics Department, Nijmegen University, The Netherlands

Received March 17, 1981; revised November 1, 1981

The equivalence problem for context-free grammars is "given two arbitrary grammars, do they generate the same language?" Since this is undecidable in general attention has been restricted to decidable subclasses of the context-free grammars. For example, the classes of $LL(k)$ grammars and real-time strict deterministic grammars. In this paper it is shown that the equivalence problem for LL-regular grammars is decidable by reducing it to the equivalence problem for real-time strict deterministic grammars. Moreover, we show that the LL-regular equivalence problem is a special case of a more general equivalence problem which is also decidable. Our techniques can also be used to show that the equivalence problem for LR-regular grammars is decidable if and only if the equivalence problem for $LR(0)$ grammars is decidable.

1. INTRODUCTION

Questions of whether or not two grammars belonging to a family of grammars generate the same language have been extensively studied in the literature. These problems are called equivalence problems, and if there exists an algorithm which gives an answer to this question for each pair of grammars of this family 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 [9], Nijholt [11–13] and Poplawski [17] 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 $LL(k)$

* The preparation of this paper was partially supported by a Natural Sciences and Engineering Research Council of Canada Grant A-7700 during the author's stay at McMaster University. An abstract of this paper was published in the proceedings of the Third International Conference on Fundamentals of Computation Theory (Springer-Verlag, Berlin/New York, 1981).

0022-0000/82/020149-13$02.00/0

languages is properly included in the class of LL-regular languages. 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 $LL(k)$ equivalence. From [22] 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 *et al.* [16] and Ukkonen [19].

The method which we use can also be used for more general classes of grammars. In this way we are able to show that the equivalence problem for real-time strict deterministic grammars with regular look-ahead is also decidable. Moreover, we obtain the result that the equivalence problem for LR-regular grammars (cf. Culik and Cohen [2]) is decidable if and only if the equivalence problem for $LR(0)$ grammars is decidable.

Preliminaries

We assume that the reader is familiar with Aho and Ullman [1] or Harrison [4]. 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, $N \cap \Sigma = \phi$ (the empty set); $N \cup \Sigma$ is denoted by V (elements of V will be denoted by X, Y and Z; elements of V^* will be denoted by α, β, γ, δ and ω). We use ε to denote the *empty word*. The elements of Σ^* will be denoted by x, y, z and w. The set P of *productions* is a subset of $N \times V^*$ (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 $\Rightarrow$, $\Rightarrow_L$ and $\Rightarrow_R$ for *derivations*, *leftmost* derivations and *rightmost* derivations, respectively. The superscripts + and * will be used to denote the *transitive* and the *reflexive-transitive* closures of these relations

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

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

The *language* $L(G)$ of a CFG 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 $\alpha \in V^*$ 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 $\alpha \in V^*$ then $|\alpha|$ denotes the *length* of α. For any $\alpha \in V^*$ and nonnegative integer k we use $k{:}\ \alpha$ to denote the prefix of α with length k if $|\alpha| \geqslant k$ 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 $u \in \Sigma^*$. A subset L of Σ^* 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 $\pi = \{B_1, B_2, ..., B_n\}$ denote a partition of Σ^*, where Σ is a finite set, into n blocks. Partition π is said to be a *regular partition* of Σ^* 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 Σ^*, $x \equiv y \pmod{\pi}$ implies $zx \equiv zy \pmod{\pi}$ ($xz \equiv yz \pmod{\pi}$).

A partition $\pi' = \{B'_1, B'_2, ..., B'_m\}$ is a *refinement* of a regular partition $\pi = \{B_1, B_2, ..., B_n\}$ of Σ^* 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 Σ^* 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 [8]).

In the forthcoming sections it is assumed that the grammars under consideration are *reduced*, that is, for each $X \in V$ there exists a derivation

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$$

for some $\alpha, \beta \in V^*$ and $w \in \Sigma^*$. There exists an algorithm (cf. Aho and Ullman [1] or Harrison [4]) which produces for each context-free grammar an equivalent context-free grammar which is reduced.

We recall the definitions of strict deterministic and real-time strict deterministic grammars (cf. Harrison and Havel [5, 6]).

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, $A' \in N$ α, β, $\beta' \in V^*$, 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 $\psi_1 \leqslant \psi_2$ 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 [5] 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, $B' \in N$; $\alpha, \beta \in V^*$, 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. This look-ahead may be finite or regular. Finite look-ahead is for instance used in the definitions of $LL(k)$ and $LR(k)$ grammars. Moreover, in Friede [3] finite look-ahead has been used in connection with strict deterministic grammars. Regular look-ahead is used in the definitions of LL-regular and LR-regular grammars. In Culik and Cohen [2] it has been shown how to convert an LR-regular grammar into an $LR(0)$ grammar. In this section we will introduce regular look-ahead for strict deterministic and real-time 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 [14] for finite look-ahead. We use the following notation. Let $G = (N, \Sigma, P, S)$ be a CFG and let $\pi = \{B_1, B_2, ..., B_n\}$ be a regular partition of Σ^*. For any $\alpha \in V^*$,

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

Definition 2.1. A CFG $G = (N, \Sigma, P, S)$ is *strong* $SD(\pi)$, 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, $A' \in N$; α, β, β', ω_1, $\omega_2 \in V^*$ with $A \equiv A' \pmod{\psi}$ and derivations

(a) $S \Rightarrow_L^* w_1 A \omega_1 \Rightarrow_L w_1 \alpha\beta\omega_1$,

(b) $S \Rightarrow_L^* w_2 A' \omega_2 \Rightarrow_L w_2 \alpha\beta'\omega_2$,

the condition

$$\text{BLOCK}(\beta\omega_1) \cap \text{BLOCK}(\beta'\omega_2) \neq \phi$$

always implies that either

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

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

A strong $SD(\pi)$ grammar $G = (N, \Sigma, P, S)$ with a minimal partition ψ is now called *strong real-time* $SD(\pi)$ 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

$$S \underset{L}{\overset{*}{\Longrightarrow}} w_1 A \omega_1 \underset{L}{\Longrightarrow} w_1 \alpha B \omega_1,$$
$$S \underset{L}{\overset{*}{\Longrightarrow}} w_2 A' \omega_2 \underset{L}{\Longrightarrow} w_2 \alpha B' \beta \omega_2,$$

and

$$\text{BLOCK}(B\omega_1) \cap \text{BLOCK}(B'\beta\omega_2) \neq \phi, \qquad (*)$$

then $\beta = \varepsilon$.

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

We now show that the equivalence problem for strong real-time $SD(\pi)$ grammars is decidable. We start with a strong real-time $SD(\pi)$ 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 $SD(\pi)$ grammars are equivalent if and only if their associated real-time strict deterministic grammars are equivalent. It is known that this latter problem is decidable (cf. [16, 19]).

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 Σ^*. 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 $0 \leqslant i \leqslant n$.

Now let p_0 be a symbol not in Q and let $\perp$ be a special symbol. 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) $S' \to [p_0 \perp p][pSq_0]$ for all $p \in Q$.

(ii) If $A \to X_1 X_2 \cdots X_r$ is in P then $[pAq] \to [pX_1p_1][p_1X_2p_2] \cdots [p_{r-1}X_rq]$ 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 $\rho: V'^* \to V^*$ by

$$\rho([p_0 \perp p]) = \varepsilon \qquad \text{for every } p \in Q,$$

$$\rho([pXq]) = X \qquad \text{for each } p, q \in Q \text{ and } X \in V.$$

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

Claim 2.1. For any $[rXs] \in V'$ and $y \in \Sigma'^*$, if $[rXs] \Rightarrow^* y$, then $\delta(s, \rho(y^R)) = r$.

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

Claim 2.2. For any $[pXq] \in V'$, if

$$[pXq] \overset{*}{\Longrightarrow} \alpha[rYs_1][s_2Zt]\beta$$

for some string $\alpha[rYs_1][s_2Zt]\beta$ in V'^*, then $s_1 = s_2$.

Claim 2.3. For any $[pXq] \in V'$ and $\omega' \in V'^*$, if $[pXq] \Rightarrow_L^* \omega'$ in G_π then $X \Rightarrow_L^* \rho(\omega')$ 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 $w, x \in \Sigma'^*$, $[pXq] \in V'$ and $\omega \in V'^*$, if

$$S' \underset{L}{\overset{*}{\Longrightarrow}} w[pXq]\,\omega \underset{L}{\overset{*}{\Longrightarrow}} wx$$

in G_π, then $\rho(x) \in B_p$, where B_p is a block of partition $\pi = \{B_0, B_1, \ldots, B_n\}$.

Proof. From Claim 2.1 it follows that $\delta(q_0, \rho(x^R)) = p$. Hence, $\rho(x) \in B_p$. ∎

LEMMA 2.1. *If G is an ε-free strong $SD(\pi)$ grammar then G_π is an ε-free strict deterministic grammar.*

Proof. We show that if grammar G is (ε-free) strong $SD(\pi)$ for a strict partition ψ, then G_π is strict deterministic for a partition ψ' of V' which is defined as follows:

(i) $\Sigma' \in \psi'$, $\{S'\} \in \psi'$.

(ii) For any p, p', q, r in Q and $A, A' \in N$, $[pAq] \equiv [p'A'r] \pmod{\psi'}$ if and only if $p = p'$ and $A \equiv A' \pmod{\psi}$.

By definition, $\Sigma' \in \psi'$. Notice that for every pair of productions with left-hand side S' condition (ii) of Definition 1.4 is satisfied. Now consider nonterminals $[pAq]$ and $[pA'r]$ in N' with $A \equiv A' \pmod{\psi}$. We may assume that G_π is reduced. Therefore there exist $\alpha, \beta, \beta' \in V'^*$; $w_1, w_2, x, y \in \Sigma'^*$ and $\omega_1, \omega_2 \in V'^*$ such that

$$S' \underset{L}{\overset{*}{\Longrightarrow}} w_1[pAq]\,\omega_1 \underset{L}{\Longrightarrow} w_1\alpha\beta\omega_1,$$

$$S' \underset{L}{\overset{*}{\Longrightarrow}} w_2[pA'r]\,\omega_2 \underset{L}{\Longrightarrow} w_2\alpha\beta'\omega_2,$$

are derivations in G_π.

From Claim 2.3 it follows that we have derivations

$$S \underset{L}{\overset{*}{\Longrightarrow}} \rho(w_1)A\rho(\omega_1) \underset{L}{\Longrightarrow} \rho(w_1)\rho(\alpha)\rho(\beta\omega_1),$$

$$S \underset{L}{\overset{*}{\Longrightarrow}} \rho(w_2)A'\rho(\omega_2) \underset{L}{\Longrightarrow} \rho(w_2)\rho(\alpha)\rho(\beta'\omega_2),$$

in G. From Claim 2.2 and Claim 2.4 we may conclude that $\mathrm{BLOCK}(\rho(\beta\omega_1)) \cap \mathrm{BLOCK}(\rho(\beta'\omega_2)) \neq \phi$. Since G is $SD(\pi)$ it follows that either

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

(ii) $\rho(\beta) = \rho(\beta') = \varepsilon$ and $A = A'$.

It follows that either

(i)′ $1{:}\beta, 1{:}\beta' \neq \varepsilon$ and $1{:}\beta \equiv 1{:}\beta' \pmod{\psi'}$, since we can write $1{:}\beta = [sX_1t_1]$ and $1{:}\beta' = [sX_2t_2]$ for some s, t_1 and t_2 in Q and $X_1 = 1{:}\rho(\beta)$ and $X_2 = 1{:}\rho(\beta')$; or

(ii)′ $\beta = \beta' = \varepsilon$ and since we have $[pAq] \to \alpha$ and $[pAr] \to \alpha$ it follows that $q = r$.

This concludes the proof that ψ' is a strict partition. ∎

LEMMA 2.2. *If G is a strong real-time $SD(\pi)$ grammar then G_π is a real-time deterministic grammar.*

Proof. From Lemma 2.1 it follows that G_π is a strict deterministic grammar for partition ψ'. Obviously, G_π is ε-free. Now consider productions $[pAq] \to \alpha[rBq]$ and $[pA'q'] \to \alpha[rB's]\beta$ in P' with $A \equiv A' \pmod{\psi}$. We have to show that $\beta = \varepsilon$. Consider derivations

$$S' \underset{L}{\overset{*}{\Longrightarrow}} w_1[pAq]\,\omega_1 \underset{L}{\Longrightarrow} w_1\alpha[rBq]\,\omega_1,$$

$$S' \underset{L}{\overset{*}{\Longrightarrow}} w_2[pA'q']\,\omega_2 \underset{L}{\Longrightarrow} w_2\alpha[rB's]\beta\omega_2$$

in G_π. It follows that for CFG G we have derivations

$$S \overset{*}{\underset{L}{\Longrightarrow}} \rho(w_1) A \rho(\omega_1) \underset{L}{\Longrightarrow} \rho(w_1 \alpha) B \rho(\omega_1),$$

$$S \overset{*}{\underset{L}{\Longrightarrow}} \rho(w_2) A' \rho(\omega_2) \underset{L}{\Longrightarrow} \rho(w_2 \alpha) B' \rho(\beta\omega_2).$$

Since G is strong real-time $SD(\pi)$ and since $\mathrm{BLOCK}(B\rho(\omega_1)) \cap \mathrm{BLOCK}(B'\rho(\beta\omega_2)) \neq \phi$ (notice that block B_r is in the intersection), we must conclude that $B \equiv B' \pmod{\psi}$ and $\rho(\beta) = \varepsilon$. It follows that $\beta = \varepsilon$, which had to be proved. ∎

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 Σ^*. Then G_1 and G_2 are both strong (real-time) $SD(\pi)$ with respect to the regular partition

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

For π we can construct the sequential machine M_π and the (real-time) strict deterministic grammars G_π^1 and G_π^2. Clearly, if $L(G_1) = L(G_2)$ then $L(G_\pi^1) = L(G_\pi^2)$ and if $L(G_1) \neq L(G_2)$ then $L(G_\pi^1) \neq L(G_\pi^2)$. 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 [4]) which accepts with empty stack. In Oyamaguchi, Honda and Inagaki [16] the decidability of the equivalence problem for these automata has been shown.

COROLLARY 2.1. *The equivalence problem for strong real-time $SD(\pi)$ 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 well-known that strong LL-regular grammars can generate nondeterministic languages. The language $L = \{a^n b^k a^n, a^k b^n c^n \mid n \geqslant 1, k \geqslant 1\}$ 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 have 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 [12], Poplawski [17]).

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

(i) $S \Rightarrow_L^* wA\alpha \Rightarrow_L w\gamma\alpha \Rightarrow_L^* wx$,

(ii) $S \Rightarrow_L^* wA\alpha \Rightarrow_L w\delta\alpha \Rightarrow_L^* wy$,

(iii) $\text{BLOCK}(\gamma\alpha) \cap \text{BLOCK}(\delta\alpha) \neq \phi$

always imply that $\gamma = \delta$.

Notice that if $\text{BLOCK}(\gamma\alpha) \cap \text{BLOCK}(\delta\alpha) \neq \phi$ 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 $LL(\pi)$ for some regular partition π of Σ^*. Notice that a grammar G is $LL(k)$ if G is $LL(\pi_k)$ 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 $\pi = \{B_1, B_2, ..., B_n\}$ be a regular partition of Σ^*. Grammar G is a *strong $LL(\pi)$ grammar* if, for each w_1, w_2, x, $y \in \Sigma^*$; α_1, α_2, γ, $\delta \in V^*$ and $A \in N$, the conditions

(i) $S \Rightarrow_L^* w_1 A\alpha_1 \Rightarrow_L w_1\gamma\alpha_1 \Rightarrow_L^* w_1 x$,

(ii) $S \Rightarrow_L^* w_2 A\alpha_2 \Rightarrow_L w_2\delta\alpha_2 \Rightarrow_L^* w_2 y$,

(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 [17] 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 here are strong.

The language

$$L = \{aa^nba^{2n}b, ba^nba^{2n}, aa^nba^na, ba^nba^nb \mid n \geqslant 0\}$$

is an example of a nondeterministic language which is LL-regular (cf. [12]). Language

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

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

The equivalence problem for $LL(k)$ grammars is decidable (cf. Rosenkrantz and Stearns [18], Aho and Ullman [1] and Olshansky and Pnueli [15]). From [22] we understand that the equivalence problem for special subclasses of the LL-regular grammars has been considered. Here we show that the equivalence problem for LL-regular grammars is decidable.

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. This method for the elimination of ε-productions may require a change in partitions. It transforms an $LL(\pi)$ grammar into an $LL(\pi')$ grammar where π' is defined by

$$\pi' = \{\{\varepsilon\}\} \cup \{aB \mid a \in \Sigma \text{ and } B \in \pi\}.$$

As mentioned above we my assume that the LL-regular grammars under consideration are strong.

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

Proof. Let $G = (N, \Sigma, P, S)$ be an ε-free strong $LL(\pi)$ grammar, where π is a regular partition of Σ^*. We show that G is a strong real-time $SD(\pi)$ grammar for partition $\psi = \{\Sigma\} \cup \{\{A\} \mid A \in N\}$. Without loss of generality we may assume that π is a left congruence. Now consider two derivations

$$S \underset{L}{\overset{*}{\Longrightarrow}} w_1 A \omega_1 \underset{L}{\Longrightarrow} w_1 \alpha\beta\omega_1,$$

$$S \underset{L}{\overset{*}{\Longrightarrow}} w_2 A' \omega_2 \underset{L}{\Longrightarrow} w_2 \alpha\beta'\omega_2$$

with $A \equiv A' \pmod{\psi}$ and $\text{BLOCK}(\beta\omega_1) \cap \text{BLOCK}(\beta'\omega_2) \neq \phi$. From the definition of ψ it follows that $A = A'$. Clearly, if $\text{BLOCK}(\beta\omega_1) \cap \text{BLOCK}(\beta'\omega_2) \neq \phi$ then $\text{BLOCK}(\alpha\beta\omega_1) \cap \text{BLOCK}(\alpha\beta'\omega_2) \neq \phi$ and since G is strong $LL(\pi)$ we have that $\alpha\beta = \alpha\beta'$ and the conditions (1) and (2) of Definition 2.1 are trivially satisfied. It remains to verify that the real-time condition is satisfied. Therefore consider A, B, A', $B' \in N$ and α, $\beta \in V^*$ with $A \to \alpha B$ and $A' \to \alpha B'\beta$ in P, $A \equiv A' \pmod{\psi}$ and derivations

$$S \underset{L}{\overset{*}{\Longrightarrow}} w_1 A \omega_1 \underset{L}{\Longrightarrow} w_1 \alpha B \omega_1,$$

$$S \underset{L}{\overset{*}{\Longrightarrow}} w_2 A' \omega_2 \underset{L}{\Longrightarrow} w_2 \alpha B' \beta \omega_2$$

with $\text{BLOCK}(B\omega_1) \cap \text{BLOCK}(B'\omega_2) \neq \phi$. However, since π is a left congruence we obtain that $\text{BLOCK}(\alpha B \omega_1) \cap \text{BLOCK}(\alpha B' \beta \omega_2) \neq \phi$. Since G is strong $LL(\pi)$ and since $A = A'$ it follows that $\alpha B = \alpha B' \beta$, that is $B = B'$ and $\beta = \varepsilon$. Hence, 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 conversion 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 methods for $LR(0)$ grammar can be used.

Unfortunately the conversion which we use here does not necessarily yield an $LL(1)$ grammar. This has been one of the reasons to introduce a direct parsing algorithm for LL-regular grammars (cf. [12]). In [13] a method has been given which converts an $LL(\pi)$ 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 $LL(\pi)$ grammar G with the method described above. If we were able to obtain from $LL(\pi)$ 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 (decidable) equivalence problem for $LL(1)$ grammars. In the following example we show that our method does not necessarily produce an $LL(1)$ grammar.

EXAMPLE 3.1. Consider CFG G with production set

$$S \to AD,$$

$$A \to aA \mid b,$$

$$D \to a \mid b.$$

Grammar G is $LL(\pi)$ for partition $\pi = \{\{\varepsilon\}, a\Sigma^*, b\Sigma^*\}$. Partition π is a left congruence. For $\pi^R = \{\{\varepsilon\}, \Sigma^* a, \Sigma^* b\}$ we have the sequential machine which is displayed in the following table. The numbers in this table denote the states of the machine.

	0	1	2
a	1	1	1
b	2	2	2

With our method we obtain the following grammar G_π:

$$
\begin{array}{ll}
[1S0] \to [1A1][1D0] & [1A1] \to [1a1][1A1] \\
[1S0] \to [1A2][2D0] & [1A2] \to [1a1][1A2] \\
[2S0] \to [2A1][1D0] & [1A1] \to [1a2][2A1] \\
[2S0] \to [2A2][2D0] & [1A2] \to [1a2][2A2] \\
[1D0] \to [1a0] & [2A1] \to [2b1] \\
[2D0] \to [2b0] & [2A2] \to [2b2]
\end{array}
$$

It is easily verified that G_π is not $LL(k)$, $k \geqslant 0$. Another example for which it can be shown that the method does not produce an $LL(k)$ grammar is the class of grammars which generate the languages in the well-known Kurki–Suonio hierarchy of $LL(k)$ languages. That is, for each $k \geqslant 0$, when the method is applied to the $LL(k+1)$ grammar G

$$
\begin{aligned}
S &\to aSA \mid aA, \\
A &\to b^k d \mid b \mid c,
\end{aligned}
$$

then for each left congruence π such that G is $LL(\pi)$ we have that the resulting grammar G_π is not an LL-grammar.

It is an open problem whether a conversion can be given, based on $LL(\pi)$ grammar G and sequential machine M_π, such that G_π is an $LL(1)$ grammar. It should be mentioned that the example grammars which are given above yield $LL(1)$ languages.

Concluding Remarks

The class of $LL(k)$ grammars is a proper subclass of the class of LL-regular grammars. Therefore we have obtained a new method for deciding $LL(k)$ grammar equivalence. Our method is completely different from other methods. However, we have to use a very strong result on the equivalence of real-time deterministic pushdown automata. Since the class of $LL(k)$ languages is properly included in the class of LL-regular languages our result is more general than the results on $LL(k)$ language equivalence.

The results in Section 2 have merely been given to provide the framework in which the equivalence problem for LL-regular grammars fits. Therefore we have not discussed properties of languages which can be generated by strong real-time SD-regular grammars. Looking at the results from the point of view of the equivalence problem of strict deterministic grammars then we see that, contrary to the situation for real-time strict deterministic grammars, we can allow productions of the form $A \to \alpha B$ and $A' \to \alpha B' \beta$ with $A \equiv A'$ and $\beta \neq \varepsilon$. However, in these cases we have restrictions on the strings which can be generated by B and $B'\beta$ respectively.

ACKNOWLEDGMENTS

I am grateful to Derick Wood (McMaster University) and an unknown referee for making several helpful suggestions.

REFERENCES

1. A. V. AHO AND J. D. ULLMAN, "The Theory of Parsing, Translation, and Compiling," Vols. 1 and 2, Prentice–Hall, Englewood Cliffs, N. J., 1972 and 1973.
2. K. CULIK AND R. COHEN, *LR*-regular grammars—an extension of *LR*(k) grammars, *J. Comput. System Sci.* 7 (1973), 66–96.
3. D. FRIEDE, Transition diagrams and strict deterministic grammars, *in* "Proc. 4th GI Conference on Theoretical Computer Science" (K. Weihrauch, Ed.), Lect. Notes in Comput. Sci. 67, 113–123, Springer-Verlag, 1979.
4. M. A. HARRISON, "Introduction to Formal Language Theory," Addison–Wesley, Reading, Mass., 1978.
5. M. A. HARRISON AND I. M. HAVEL, Strict deterministic grammars, *J. Comput. System Sci.* 7 (1973), 237–277.
6. M. A. HARRISON AND I. M. HAVEL, Real-time strict deterministic grammars, *SIAM J. Comput.* 1 (1972), 333–349.
7. M. A. HARRISON, I. M. HAVEL, AND A. YEHUDAI, An equivalence of grammars through transformation trees, *Theoret. Comput. Sci.* 9 (1979), 173–206.
8. J. E. HOPCROFT AND J. D. ULLMAN, "Formal Languages and Their Relation to Automata," Addison–Wesley, Reading, Mass., 1969.
9. S. JARZABEK AND T. KRAWCZYK, *LL*-regular grammars, *Inform. Process. Lett.* 4 (1975), 31–37.
10. A. J. KORENJAK AND J. E. HOPCROFT, Simple deterministic languages, *in* "Conf. Record of Seventh Annual Symposium on Switching and Automata Theory 1966," pp. 36–46.
11. A. NIJHOLT, On the parsing of *LL*-regular grammars, *in* "Proc. 5th Symposium on the Mathematical Foundations of Computer Science," (A. Mazurkiewicz, Ed.), Lect. Notes in Comput. Sci. 45, pp. 446–452, Springer-Verlag, Berlin, 1976.
12. A. NIJHOLT, *LL*-regular grammars, *Internat. J. Comput. Math.* 8 (1980), 303–318.
13. A. NIJHOLT, "From *LL*-Regular to *LL*(1) Grammars: Transformations, Covers and Parsing," Report IR-61, Vrije Universiteit Amsterdam, May 1980.
14. A. NIJHOLT, "A Framework for Classes of Grammars between the *LL*(k) and *LR*(k) Grammars," TR No. 80-CS-25, McMaster University, Hamilton, Canada.
15. T. OLSHANSKY AND A. PNUELI, A direct algorithm for checking equivalence of *LL*(k) grammars, *Theoret. Comput. Sci.* 4 (1977), 321–349.
16. M. OYAMAGUCHI, N. HONDA, AND Y. INAGAKI, The equivalence problem for real-time strict deterministic languages, *Inform. Contr.* 45 (1980), 90–115.
17. D. A. POPLAWSKI, On *LL*-regular grammars, *J. Comput. System Sci.* 18 (1979), 218–227.
18. D. J. ROSENKRANTZ AND R. E. STEARNS, Properties of deterministic top-down grammars, *Inform. Contr.* 17 (1970), 226–255.
19. E. UKKONEN, A decision method for the equivalence of some non-real-time deterministic pushdown automata, *in* "Proceedings, Twelfth Annual ACM Symp. on Theory of Computing, 1980," pp. 29–38.
20. D. WOOD, Some remarks on the *KH* algorithm for s-grammars, *BIT* 13 (1973), 476–489.
21. D. WOOD, Lecture notes on top-down syntax analysis, *J. Comput. Soc. India* 8 (1978), 1–22.
22. V. V. ZUBENKO, "Simple Pushdown Storage Automata and the Equivalence Problem in Certain Classes of *LL*(π) Grammars" (in Russian). "Theory and Practice of Systems Programming" (Russian), Inst. Kibernet., Akad. Nauk Ukrain. SSR, Kiev, 1979, pp. 60–76.