Theoretical Computer Science 3 (1976) 273–282.

CONCERNING BOUNDED-RIGHT-CONTEXT GRAMMARS*

Thomas G. SZYMANSKI

Department of Electrical Engineering and Computer Science, Princeton University, Princeton, NJ 08540, U.S.A.

Communicated by Michael Harrison
Received 5 December 1975

Abstract. Consider the problem of testing whether a context-free grammar is an (m, n)-BRC grammar. Let $|G|$ denote the size of the grammar G. It is first shown that G is (m, n)-BRC if and only if G is (m_0, n)-BRC where $m_0 = 4 \cdot |G|^2 \cdot (n+1)^2$. Deterministic and nondeterministic algorithms are then presented for testing whether an arbitrary grammar has the (m, n)-BRC property for fixed values of m and n. The running times of both algorithms are low degree polynomials which are independent of m.

1. Introduction and background

A large number of parameterized families of parsing techniques such as the LR(k), LL(k), SLR(k) and (m, n)-BRC methods have appeared in the literature, (see [1] for descriptions of these and other parsing techniques). Since none of the aforementioned methods apply to all context-free grammars, it is natural to ask what is the computational complexity of testing whether a particular parsing method works for a given grammar. In this paper we study this problem for the case of the (m, n)-bounded-right-context (abbreviated (m, n)-BRC) technique introduced in [2] and subsequently studied in [3, 6–8].

Suppose we denote the size of a context-free grammar G by $|G|$. In Section 2 we show that G is (m, n)-BRC if and only if G is (m_0, n)-BRC for $m_0 = 4 \cdot |G|^2 \cdot (n+1)^2$. Of course, the only case of practical interest is the case $n = 1$, for which we see that G is $(m, 1)$-BRC if and only if G is $(16 \cdot |G|^2, 1)$-BRC. Thus the maximum amount m of left context utilizable in parsing a given grammar with the BRC technique is completely determined by the amount n of right context which we are willing to use!

In Section 3 we apply the ideas in the proof of this result to develop asymptotically efficient algorithms for testing whether an arbitrary grammar G has the (m, n)-BRC property. Both nondeterministic and deterministic algorithms are

* Work supported in part by NSF grant DCR74–21939.

presented requiring respectively $O(p(|G|, n))$ and $O(|G|^n \cdot p(|G|, n))$ time where p is a low degree polynomial. Previous algorithms for testing the (m, n)-BRC property have all involved constructing parsers. Such techniques can yield, at best, deterministic test times which grow exponentially with *both* m and n.

In Section 4 we consider some consequences of these results.

The rest of Section 1 is devoted to presenting definitions and background material. Our notation is consistent with that used in [1] and the reader is referred to that source for a more complete treatment of context-free grammars and parsing.

Our first group of definitions deals with context-free grammars.

Definition. A *context-free grammar* is a four-tuple $G = (N, \Sigma, P, S)$ where Σ and N are disjoint finite sets of *terminals* and *nonterminals* respectively, the *start symbol* S is a member of N and the *productions* P are some finite subset of $N \times (N \cup \Sigma)^*$.

Productions of a grammar will be written in the form $A \to \alpha$ rather than (A, α). The following naming conventions will be used throughout this paper:

members of Σ will be denoted by $a, b, c, \ldots$,
members of N will be denoted by $A, B, C, \ldots$,
members of $N \cup \Sigma$ will be denoted by $\ldots, X, Y, Z$,
members of Σ^* will be denoted by $\ldots, x, y, z$,
members of $(N \cup \Sigma)^*$ will be denoted by $\alpha, \beta, \gamma, \ldots$.

The empty string is denoted by λ and the *length* of a string α by $|\alpha|$. If β is a suffix of α we define the (*right*) *quotient* of α by β (written α/β) as the unique string γ such that $\alpha = \gamma\beta$.

The next set of definitions deals with derivations in a grammar.

Definition. Let G be a context-free grammar. If $A \to \alpha$ is a production and β and γ are strings, then we say that $\beta A \gamma$ *derives* $\beta\alpha\gamma$ *in* G (written $\beta A \gamma \Rightarrow_G \beta\alpha\gamma$). The subscript G will be omitted whenever the identity of the grammar in question is clear. If γ consists solely of terminals then we say that $\beta A \gamma$ *derives* $\beta\alpha\gamma$ *in* G *by rightmost derivation* which will be written as $\beta A \gamma \Rightarrow_{rm} \beta\alpha\gamma$. The transitive and reflexive closures of the relations $\Rightarrow$ and $\Rightarrow_{rm}$ will be denoted by $\Rightarrow^*$ and $\Rightarrow^*_{rm}$ respectively.

For convenience, we shall assume that every grammar has been *augmented*, that is, a new start symbol S' has been added to N along with a new production $S' \to S$ where S is the old start symbol of the grammar. We shall also assume that all grammars are *reduced*, that is, every production may be used in the derivation of some terminal string from the start symbol of the grammar. Finally, the symbol \$ will be used as an *endmarker* to delimit the ends of a string being parsed. This symbol will be considered an element of Σ although formally it appears in no production of the grammar.

Next we need the notion of a bounded-right-context grammar. Our development is operational in that we first present a method for constructing parsers and then define a bounded-right-context grammar as a grammar for which the construction succeeds. The following definition is taken from [1].

Definition. Let G be an augmented context-free grammar and l be the length of the longest right side of a production in G. Let m and n be integers. The set of *shift contexts* for G will be denoted by $\mathcal{S}(m, n)$ and will consists of exactly those pairs of strings (α, x) such that

(1) $|\alpha| = m + l$ or else $|\alpha| < m + l$ and α begins with $\$^m$;
(2) $|x| = n$;
(3) there exists a rightmost derivation

$$\$^m S\$^n \underset{rm}{\overset{*}{\Rightarrow}} \beta B y \underset{rm}{\Rightarrow} \beta\theta y$$

in which αx is a substring of $\beta\theta y$ with the rightmost symbol of α lying strictly to the left of the rightmost symbol of $\beta\theta$.

For each nonterminal A of G we define $\mathcal{R}(m, n, A)$, the set of *reduce contexts* for A, as containing exactly those triples (α, β, x) such that

(1) $|\alpha| = m$;
(2) $|x| = n$;
(3) there exists a rightmost derivation

$$\$^m S\$^n \underset{rm}{\overset{*}{\Rightarrow}} \gamma\alpha A x y \underset{rm}{\Rightarrow} \gamma\alpha\beta x y.$$

The $\mathcal{S}$ and $\mathcal{R}$ sets may be used to determine the action of a shift/reduce parser for G. Suppose at some instant of time, the next n input characters form the string x. If $(\alpha, x) \in \mathcal{S}(m, n)$ and the string α is on the top of the parser's pushdown store (i.e. the string formed by concatenating the top $|\alpha|$ characters of the store is equal to α), then the parser should perform a shift move by removing the next input character from the input tape and pushing it into the pushdown store. On the other hand, if in the same configuration $\alpha\beta$ is on top of the store and $(\alpha, \beta, x) \in \mathcal{R}(m, n, A)$, then the parser should reduce the production $A \rightarrow \beta$ by replacing the β on top of the store with the nonterminal A.

A grammar is said to be an (m, n)-*bounded-right-context* (abbreviated (m, n)-BRC) grammar if the shift/reduce parser described above is well-defined. More formally:

Definition. A context-free grammar G is (m, n)-BRC if and only if:

(1) For every pair of distinct productions $A \rightarrow \beta$ and $B \rightarrow \theta$, whenever $(\alpha, \beta, x) \in \mathcal{R}(m, n, A)$ and $(\gamma, \theta, x) \in \mathcal{R}(m, n, B)$, then $\alpha\beta$ is not a suffix of $\gamma\theta$.

(2) For all nonterminals A, if $(\alpha, \beta, x) \in \mathcal{R}(m, n, A)$ then $(\gamma\alpha\beta, x)$ is not in $S(m, n)$ for any γ.

Clause (1) above tests for the presence of reduce/reduce conflicts whereas clause (2) rules out shift/reduce conflicts. The definition given above is quite different from that originally presented in [2] but is nevertheless equivalent as shown in [3] and Lemma 5.6 of [1].

In order to discuss the complexity of problems involving grammars we will need some notion of the size of a grammar.

Definition. The size of a context-free grammar G is the sum of $|A\alpha|$ taken over all productions $A \to \alpha$ of G.

Thus the size of a grammar is nothing more than the number of positions between symbols in the right sides of all productions of the grammar. Notice that the initial and final positions of a right side are both counted. Thus the production $A \to AbS$ contributes four to the size of the grammar, whereas the production $A \to \lambda$ contributes only one.

At this point we recall a key result from [4].

Proposition 1.1. *Let x be a string of length n and let G be a context-free grammar. There exists a nondeterministic finite state automaton $M(G, x)$ such that*

(1) *M has at most $2 \cdot |G| \cdot (n+1)$ states. Moreover, every item of the form $[A \to \beta_1 \cdot \beta_2, u]$ where $A \to \beta_1\beta_2$ is a production of G and u is a suffix of x is a state of M;*

(2) *M has at most $4 \cdot |G| \cdot (n+1)^2$ possible transitions;*

(3) *M may be constructed deterministically from G and x in $O(|G| \cdot (n+1)^3)$ time;*

(4) *If q_0 is the start state of M and δ is the move function of M extended to strings in the usual fashion, then*

$$[A \to \beta_1 \cdot \beta_2, u] \in \delta(q_0, \gamma)$$

if and only if

$$S\$^n \underset{rm}{\Rightarrow}^* \alpha A u w \underset{rm}{\Rightarrow}^* \alpha\beta_1\beta_2 u w$$

with $\gamma = \alpha\beta_1$, $w \in \Sigma^*$ and u some suffix of x.

(5) *every state is accesable from the start state q_0, that is, for every state q there exists a string γ such that $q \in \delta(q_0, \gamma)$.*

We note in passing that the classical "LR(k) parsing machine" for G may be constructed by taking the union of the machines $M(G, x)$ over all strings x of length k, eliminating λ transitions and then applying the well-known subset construction to produce a deterministic finite state device.

Lemma 1.2. *Let $M(G, x)$ be as in Proposition 1.1. Let q and $[A \to \beta_1 \cdot \beta_2, v]$ be states of M. Suppose that $[A \to \beta_1 \cdot \beta_2, v] \in \delta(q, \gamma)$. Then*

(1) *if* $|\gamma| \geqslant |\beta_1|$ *then* β_1 *is a suffix of* γ;
(2) *if* $|\gamma| \leqslant |\beta_1|$ *then* γ *is a suffix of* β_1.

Proof. By (5) of Proposition 1.1, there exists a string γ' such that $q \in \delta(q_0, \gamma')$. Therefore $[A \to \beta_1 \cdot \beta_2, v]$ is a member of $\delta(q_0, \gamma'\gamma)$. By (4) of the same proposition, this implies that β_1 is a suffix of $\gamma'\gamma$. The lemma follows immediately. □

We may now present a characterization of the $\mathscr{S}$ and $\mathscr{R}$ sets in terms of "paths" through the states of the automaton described in Proposition 1.1.

Lemma 1.3. *Let G be an augmented context-free grammar and m and n be positive integers.* $(\alpha, \beta, x) \in \mathscr{R}(m, n, A)$ *if and only if either*

(1) $\alpha \in (N \cup \Sigma - \{\$\})^m$ *and* $[A \to \beta \cdot, x] \in \delta(q, \alpha\beta)$ *for some state q in* $M(G, x)$; *or*

(2) $\alpha = \$^i\alpha'$ *for some* $i > 0$, $\alpha' \in (N \cup \Sigma - \{\$\})^{m-i}$ *and* $[A \to \beta \cdot, x] \in \delta(q_0, \alpha'\beta)$.

Lemma 1.4. *Let G be an augmented context-free grammar and m and n be positive integers. Let l be the length of the longest right side of a production of G.* $(\alpha, x) \in \mathscr{S}(m, n)$ *if and only if either*

(1) $\alpha \in (N \cup \Sigma - \{\$\})^{m+l}$ *and there exists a state q in* $M(G, x)$ *and a production* $B \to \theta$ *in G such that either*

(a) $[B \to \theta \cdot, \lambda] \in \delta(q, \alpha xy)$ *for some* $y \in \Sigma^+$; *or*

(b) $[B \to \theta \cdot, x_2] \in \delta(q, \alpha x_1)$ *for some* $x_1 \in \Sigma^+$, $x_2 \in \Sigma^*$ *with* $x = x_1x_2$;

(2) $\alpha = \$^i\alpha'$ *for some* $i > 0$ *and* $\alpha' \in (N \cup \Sigma - \{\$\})^{m+l-i}$ *and there exists a production* $B \to \theta$ *in G such that in* $M(G, x)$ *either*

(a) $[B \to \theta \cdot, \lambda] \in \delta(q_0, \alpha' xy)$ *for some* $y \in \Sigma^+$; *or*

(b) $[B \to \theta \cdot, x_2] \in \delta(q_0, \alpha' x_1)$ *for some* $x_1 \in \Sigma^+$ *and* $x_2 \in \Sigma^*$ *with* $x = x_1x_2$.

Proof. The proofs of both Lemmas are straightforward applications of Proposition 1.1 to the definitions of the sets $\mathscr{S}$ and $\mathscr{R}$. □

The lemmas just stated provide us with a reasonably efficient means of *constructing* BRC parsers. However, in this paper we are primarily concerned with *testing* for the BRC property without necessarily constructing a parser. It is to this question that we address the sequel.

2. An upper bound on m

In this section we shall show that the maximum amount of left context needed for parsing a BRC grammar is completely determined by the size of the grammar in question and the amount of right context being used.

Theorem 2.1. *Let G be a context-free grammar and let n be any integer. If G is* (m, n)-BRC *for some value of m, then G is* (m_0, n)-BRC *for* $m_0 = 4 \cdot |G|^2 \cdot (n+1)^2$.

Proof. We shall show that if G is not (m_0, n)-BRC then G is not (m, n)-BRC for any value of m. The proof divides into a number of cases based on the reason(s) why the BRC definition is violated.

Case 1. Condition (1) of the BRC definition is not satisfied. Hence for some distinct pair of productions $A \to \beta$ and $B \to \theta$, there exist elements $(\alpha, \beta, x) \in \mathscr{R}(m_0, n, A)$ and $(\gamma, \theta, x) \in \mathscr{R}(m_0, n, B)$ with $\alpha\beta$ a suffix of $\gamma\theta$. We may therefore write $\gamma\theta$ as $\gamma'\alpha\beta$ for some γ'.

If $\alpha\beta$ begins with a \$ then $\gamma' \in \{\$\}^*$. Moreover, for any $i \geqslant 0$,

$$(\$^i\alpha, \beta, x) \in \mathscr{R}(m_0 + i, n, A)$$

and

$$(\$^i\gamma, \theta, x) \in \mathscr{R}(m_0 + i, n, B).$$

Thus G is certainly not (m, n)-BRC for any m.

On the other hand, suppose that $\alpha\beta$ does not begin with a \$. By Lemma 1.3, there exist states p and q in $M(G, x)$ such that

$$[A \to \beta \cdot, x] \in \delta(p, \alpha\beta)$$

and

$$[B \to \theta \cdot, x] \in \delta(q, \gamma'\alpha\beta).$$

Let Q be the state set of $M(G, x)$. Since $|\alpha\beta| \geqslant m_0 = |Q|^2$, we may appeal to the standard "repeated pairs of states" argument[1] and write $\alpha\beta$ as $\eta_1\eta_2\eta_3$ with $\eta_2 \neq \lambda$ such that for all $i \geqslant 0$,

$$[A \to \beta \cdot, x] \in \delta(p, \eta_1\eta_2^i\eta_3)$$

and

$$[B \to \theta \cdot, x] \in \delta(q, \gamma'\eta_1\eta_2^i\eta_3).$$

By Lemmas 1.2 and 1.3 we may therefore conclude that for any $i \geqslant 0$,

$$(\eta_1\eta_2^i\eta_3/\beta, \beta, x) \in \mathscr{R}(m_0 + (i-1) \cdot |\eta_2|, A)$$

and

$$(\gamma'\eta_1\eta_2^i\eta_3/\theta, \theta, x) \in \mathscr{R}(m_0 + (1-1) \cdot |\eta_2|, B).$$

Since $\eta_2 \neq \lambda$, we conclude that G is not (m, n)-BRC for any value of m.

Case 2. Condition (2) of the BRC definition is not satisfied. Hence for some production $a \to \beta$ and element $(\alpha, \beta, x) \in \mathscr{R}(m_0, n, A)$ there exists a string γ for which $(\gamma\alpha\beta, x) \in \mathscr{S}(m_0, n)$.

[1] Very briefly put, there exists a state q' for which $[B \to \theta \cdot, x] \in \delta(q', \alpha\beta)$. Let $k = |\alpha\beta|$ and let the individual characters of the string $\alpha\beta$ be denoted by $X_1, X_2, \ldots, X_k$. There exists a sequence of states $s_0, s_1, \ldots, s_k$ such that $s_0 = p$, $[A \to \beta \cdot, x] = s_k$, and $s_i \in \delta(s_{i-1}, X_i)$ for $0 < i \leqslant k$. Similarly, there exists a sequence of states $t_0, t_1, \cdots, t_k$ such that $t_0 = q'$, $[B \to \theta \cdot, x] = t_k$ and $t_i \in \delta(t_{i-1}, X_i)$ for $0 < i \leqslant k$. Since $k \geqslant |Q|^2$, there must exist distinct integers i and j for which $s_i = s_j$ and $t_i = t_j$. We may then choose $\eta_1 = X_1, \ldots, X_i$, $\eta_2 = X_{i+1}, \ldots, X_j$, and $\eta_3 = X_{j+1}, \ldots, X_k$.

As before, if α begins with a \$, we must have $\gamma \in \{\$\}^*$ and so $(\$^i\alpha, \beta, x) \in \mathscr{R}(m_0 + i, n, A)$ for all $i \geq 0$ and $(\$^i\gamma\alpha\beta, x) \in \mathscr{S}(m_0 + i, n)$ for all $i \geq 0$. Thus G is clearly not (m, n)-BRC for any m.

If α does not begin with a \$, then by Lemma 1.3 there exists a state p in $M(G, x)$ such that $[A \to \beta \cdot, x] \in \delta(p, \alpha\beta)$. Moreover, by Lemma 1.4 there exists a state q in $M(G, x)$ and a production $B \to \theta$ in G such that

$$[B \to \theta \cdot, \lambda] \in \delta(q, \gamma\alpha xy) \quad \text{for some } y \in \Sigma^+$$

or

$$[B \to \theta \cdot, x_2] \in \delta(q, \gamma\alpha x_1) \quad \text{for } x_1 \neq \lambda \text{ and } x = x_1x_2.$$

In either case, since $|\alpha| \geq m_0 = |Q|^2$ we may write α as $\eta_1\eta_2\eta_3$ with $\eta_2 \neq \lambda$ such that

$$[A \to \beta \cdot, \lambda] \in \delta(p, \gamma\eta_1\eta_2^i\eta_3\beta)$$

and either

$$[B \to \theta \cdot, \lambda] \in \delta(q, \gamma\eta_1\eta_2^i\eta_3 xy) \quad \text{for some } y \in \Sigma^+;$$

or

$$[B \to \theta \cdot, x_2] \in \delta(q, \gamma\eta_1\eta_2^i\eta_3 x_1) \quad \text{for some } x_1 \neq \lambda \text{ with } x = x_1x_2.$$

We therefore have

$$(\eta_1\eta_2^i\eta_3, \beta, x) \in \mathscr{R}(m_0 + (i-1)\cdot|\eta_2|, n, A)$$

and

$$(\gamma\eta_1\eta_2^i\eta_3\beta, x) \in \mathscr{S}(m_0 + (i-1)\cdot|\eta_2|, n)$$

for all $i \geq 0$. Since $\eta_2 \neq \lambda$, we conclude that G is not (m, n)-BRC for any m. □

3. Testing the BRC property

In this section we shall devise efficient algorithms for testing whether an arbitrary grammar is a (m, n)-BRC grammar for specified values of m and n. The running times of our algorithms will be independent of m.

Definition. The *rank* of a string x with respect to a grammar G is the minimum value of m such that

(1) For all pairs of distinct productions $A \to \beta$ and $B \to \theta$ whenever $(\alpha, \beta, x) \in \mathscr{R}(m, |x|, A)$ and $(\gamma, \theta, x) \in \mathscr{R}(m, |x|, B)$, $\alpha\beta$ is not a suffix of $\gamma\theta$;

(2) For all nonterminals A, if $(\alpha, \beta, x) \in \mathscr{R}(m, |x|, A)$ then $(\gamma\alpha\beta, x)$ is not a member of $\mathscr{S}(m, |x|)$ for any string γ.

Clearly, if G is not (m, n)-BRC for any m then there exists at least one string x of length n whose rank is undefined. Conversely, if G is (m, n)-BRC then m is greater than or equal to the maximum rank of all terminal strings of length n. The following algorithm provides us with the means for calculating the rank of a string.

Algorithm 3.1:

Input: a reduced context-free grammar G and a terminal string x;

Output: the rank of x with respect to G.

Step 1: Construct $M(G, x)$ with state set Q. Remove all λ transitions from M.

Step 2: Construct a directed graph D whose nodes are the elements of the set $Q \times Q$. The set of edges of D may be defined (using relational notation) as follows:

(a) $(q_0, q_0) D (q_0, q_0)$ where q_0 is the start state of M;

(b) $(q_1, q_2) D (q_3, q_4)$ if and only if for some $Z \in (N \cup \Sigma)$ both $q_3 \in \delta(q_1, Z)$ and $q_4 \in \delta(q_2, Z)$.

Step 3: Define a subset F of the nodes of D and associate a *weight* with each member of F as follows:

(a) $([A \to \beta \cdot, x], [B \to \theta \cdot, x]) \in F$ with weight $-\min(|\beta|, |\theta|)$ if and only if $A \to \beta$ and $B \to \theta$ are distinct productions.

(b) $(q, [A \to \beta \cdot, x]) \in F$ with weight $-|\beta|$ if and only if there exists some production $B \to \theta$ for which either $[B \to \theta \cdot, \lambda] \in \delta(q, v)$ for some $v \in \Sigma^+$ or $[B \to \theta \cdot, x_2] \in \delta(q, vx_1)$ for some $v \in \Sigma^*$ with $x_1 \in \Sigma^+$, $x_2 \in \Sigma^*$ and $x = x_1 x_2$;

(c) $([A \to \beta \cdot, x], q) \in F$ with weight $-|\beta|$ under the same conditions as described in (b).

Note carefully that the weights assigned to the members of F are all negative.

Step 4: Define a subgraph $\bar{D}$ of D such that $\rho \bar{D} \sigma$ if and only if $\rho D \sigma$ and there exists some $\tau \in F$ such that $\sigma D^* \tau$.

Step 5: If $\bar{D}$ contains a cycle, then the rank of x is undefined. Otherwise $\bar{D}$ is acyclic and we may assign a rank to each node of $\bar{D}$ as follows:

(a) the weight of a member of F is simply the weight assigned to it in Step 3;

(b) the weight of a member ρ of $(Q \times Q) - F$ is $1 + \max\{\text{weight}(\sigma) \mid \rho \bar{D} \sigma\}$.

The concept of weight of a node is well-defined because whenever $\rho \in F$ there exists no $\sigma \in Q \times Q$ for which $\rho \bar{D} \sigma$.

Step 6: The rank of x is $\max(\{0\} \cup \{\text{weight}(\rho) \mid \rho \in Q \times Q\})$.

The idea behind the algorithm is to look for the longest left context string needed to resolve a potential violation of the BRC definition with respect to the specified right context string x. The nodes in F represent all possible conflicts, type (a) nodes corresponding to reduce/reduce conflicts and types (b) and (c) nodes corresponding to shift/reduce conflicts. In either case, we attempt to work our way backwards along common access paths in $M(G, x)$, that is, along sequences of transitions labeled with identical strings of characters from $(N \cup \Sigma)$. Each step backwards along these paths is represented by a single step backwards along an edge in the directed graph $\bar{D}$. If such a path can go on indefinitely, then the rank of x is undefined and G is not $(m, |x|)$-BRC for any m. If there is a maximum length to such a path then the rank of x is well-defined and G is $(m, |x|)$-BRC for some m. This m is the length of the longest common access string *exclusive* of that suffix of

the access string forming the potential handle. It is for this reason that the members of F are assigned negative weights by the algorithm.

Lemma 3.2. *Algorithm* 3.1 *may be implemented to run deterministically in the polynomial in both the size of the grammar G and the length of the string x.*

Proof. It is straightforward to verify that each of the steps of the algorithm may be performed in polynomial time. Step 5 may be performed by an easy adaptation of the "topological sort" algorithm of [5]. □

Theorem 3.3. *Let G be a context-free grammar and let m and n be integers. It is possible to test whether G is* (m, n)-BRC *in*

(1) *nondeterministic time* $O(p(|G|, n))$

(2) *determinstic time* $O(|G|^n \cdot p(|G|, n))$

where p is the polynomial mentioned in Lemma 3.2.

Proof. To perform the (m, n)-BRC test nondeterministically, we construct an algorithm which accepts exactly those triples (G, m, n) such that G is *not* (m, n)-BRC. The algorithm works by "quessing" a string x of length n and computing its rank by using Algorithm 3.1. If the rank is undefined or exceeds m then G is not (m, n)-BRC and the algorithm accepts the input triple (G, m, n).

To perform the (m, n)-BRC test deterministically, we simply compute the rank of every terminal string of length n. Since $|\Sigma| \leq |G|$, there are at most $|G|^n$ such strings and the entire test may be performed in $O(|G|^n \cdot p(|G|, n))$ time. □

4. Conclusions

The distinctly different nature of the roles played by the parameters m and n in the (m, n)-BRC definition is revealed by the following result.

Theorem 4.1. *Let* $P(G, m)$ *be the predicate "there exists an n such that G is* (m, n)-BRC". *Let* $Q(G, n)$ *be the predicate "there exists an m such that G is* (m, n)-BRC". *Then*

(1) *P is undecidable*;

(2) *Q is decidable.*

Proof. The undecidability of P follows from the easily established fact that the predicate "there exists an n such that G is $(1, n)$-BRC" is undecidable. The decidability of Q follows from Theorem 2.1 and 3.3. □

It is interesting to compare Theorem 3.3 with the results in [4] in which it is shown that a grammar G can be tested for the LR(n) property in nondeterministic

time $O(|G|^2 \cdot (n+1))$ and deterministic time $O(|G|^{n+2} \cdot (n+1)^3)$. These bounds appear to be less than the ones presented in this paper for (m, n)-BRC testing (the polynomial p mentioned in Lemma 3.2 and Theorem 3.3 seems to be at least quadratic in both of its arguments) despite the fact that every (m, n)-BRC grammar is known to be an LR(n) grammar. We leave open the problem of determining the exact relationship between the complexities of testing the LR(k) and (m, n)-BRC properties when $k = n$. The special case $k = n = 1$ is of particular interest because this is probably the only case of practical significance. Another problem which seems worthy of investigation is determining upper and lower bounds on the complexity of $(1, 1)$-BRC testing.

References

[1] A.V. Aho, and J.D. Ullman, *The Theory of Parsing, Translation and Compiling* (Prentice-Hall, 1972 and 1973).

[2] R.W. Floyd, Bounded context syntactic analysis, *Comm. ACM* **7** (1964) 62–67.

[3] S.L. Graham, On bounded right context languages and grammars, *SIAM J. Comput.* **3** (1974) 224–254.

[4] H.B. Hunt, III, T.G. Szymanski and J.D. Ullman, On the complexity of LR(k) testing, *Comm. ACM.* **18** (1975) 707–716.

[5] D.E. Knuth, *Fundamental Algorithms* (Addison-Wesley, 1968).

[6] J. Loeckx, An algorithm for the construction of bounded-context parsers, *Comm. ACM.* **13** (1970) 297–307.

[7] M.D. Mikunas, R.L. Lancester and V.B. Schneider, Transforming LR(k) grammars to LR(1), SLR(1) and (1, 1) bounded right context grammars, *J. ACM* **23** (1976) 511–533.

[8] M.D. Mikunas, and V.B. Schneider, A parser generating system for constructing compressed compilers, *Comm. ACM.* **16** (1973) 669–676.