JOURNAL OF COMPUTER AND SYSTEM SCIENCES 13, 318–334 (1976)

Complexity Metatheorems for Context-Free Grammar Problems

HARRY B. HUNT, III*

Center for Research in Computing Technology, Harvard University, Cambridge, Massachusetts 02138

AND

THOMAS G. SZYMANSKI†

Department of Electrical Engineering and Computer Science, Princeton University, Princeton, New Jersey 08540

Received September 22, 1975; revised May 5, 1976

Metatheorems are presented which can be used to (i) establish undecidability results about context-free grammar problems, and (ii) establish lower bounds on certain decidable grammar problems. The main undecidability result is obtained through the simulation of a Turing machine which always halts. This technique promises to be applicable in many situations where conventional techniques for proving undecidability do not succeed.

1. INTRODUCTION

This paper is primarily concerned with the problem of establishing lower bounds on the complexity of broad classes of predicates on the context-free grammars.

Section 1 contains an overview of the paper and sufficient background material and definitions for the comprehension of the more technical results.

In Section 2 we develop a metatheorem which gives sufficient conditions for arbitrary predicates on the context-free grammars to be undecidable. Although the idea of such metatheorems is not new, all previous such results deal with predicates on languages rather than predicates on grammars. Specifically, we show the undecidability of any predicate which is true of all grammars which are simultaneously strong LL, SLR, and BRC and false of all grammars which generate languages of unbounded inherent ambiguity.[1] Many undecidability results which had previously be obtained

* Work supported by NSF grant GJ-35570 and U.S. Army Contract DA-31-124-ARO-D-462.

† Work supported by NSF grants GJ-35570 and DCR 74-21939.

[1] A language is of unbounded inherent ambiguity if for any integer k and any grammar generating the language, that grammar produces k distinct derivation trees for some string.

by ad hoc constructions follow directly from this result. More importantly, the proof of this result introduces a new technique for proving undecidability which promises to be applicable in many situations where more conventional techniques cannot be used. Simply put, the technique involves efficiently reducing the membership problem for an arbitrary recursive set to the problem at hand. In this way we can show that the problem is of nonrecursive complexity and must therefore be undecidable. In many situations it is technically far easier to embed the computations of a Turing machine *which is known to eventually halt* than it is to embed the computations of machines not subject to this restriction.

In Section 3 we develop a subrecursive analog of this metatheorem and use it to establish lower bounds for testing membership in certain parameterized hierarchies of grammar classes. For example, any uniform algorithm which decides of a grammar G and integer k whether that grammar is LR(k) requires nondeterministic exponential time. This particular result had previously been shown in [9]. Here we present a metatheorem which yields this result not only for the LR(k) hierarchy but also for most other "natural" hierarchies of grammars.

We assume that the reader is familiar with the basic definitions and results concerning context-free grammars and languages, otherwise see [1] or [10]. The empty string is denoted by λ, the reversal of a string w is denoted by w^{rev}, and the language generated by a grammar G is denoted by $L(G)$.

Moreover, we assume that the reader is acquainted with the basic results of parsing theory as presented, for instance, in [1]. In particular, we assume familiarity with the following parsing methods: strong LL, LL, SLR, LR, and BRC (bounded right context). Throughout the paper we shall only parameterize the above class names when we wish to denote a specific subclass. Under this convenition it would be legitimate to say "It is undecidable whether a grammar is LR, but decidable whether it is LR(k)."

We shall need a few additional definitions associated with grammars.

Definition 1.1. Let $G = (V, \Sigma, P, S)$ be a context-free grammar. The *size* of G (denoted $|G|$) is defined to be the total number of occurrences of terminals and nonterminals in all productions, more formally,

$$|G| = \sum_{A\to\alpha\in P} |A\alpha|.$$

For any $\alpha \in V^*$ and nonnegative integer k, we define

$$\text{FIRST}_k(\alpha) = \{x \in \Sigma^* \mid \alpha \overset{*}{\underset{G}{\Rightarrow}} x\beta \text{ and } |x| = k \text{ for some } \beta \in V^*$$
$$\text{or } \alpha \overset{*}{\underset{G}{\Rightarrow}} x \text{ and } |x| < k\},$$

$$\mathrm{FOLLOW}_k(\alpha) = \{x \in \Sigma^* \mid S \underset{G}{\overset{*}{\Rightarrow}} \gamma\alpha\beta \text{ for some } \gamma, \beta \in V^* \text{ and } x \in \mathrm{FIRST}_k(\beta)\}. \quad \blacksquare$$

The concept of ambiguity arises in several places in this paper.

DEFINITION 1.2. A context-free grammar G is said to be:

(i) *unboundedly ambiguous* if for any positive integer k there exists some string in $L(G)$ having k or more distinct leftmost derivations according to G,

(ii) *ambiguous of degree k* if some string in $L(G)$ has k distinct leftmost derivations, but no string in $L(G)$ has more than k leftmost derivations according to G,

(iii) *unambiguous* if it is ambiguous of degree 1, that is, every string in $L(G)$ has a unique leftmost derivation in G,

(iv) *ambiguous* if it is not unambiguous. ▮

DEFINITION 1.3. A context-free language L is said to be:

(i) of *unbounded inherent ambiguity* if every grammar generating L is unboundedly ambiguous,

(ii) *inherently ambiguous of degree k* if L is generated by some grammar which is ambiguous of degree k but by no grammar which is ambiguous of degree less than k,

(iii) *inherently unambiguous* if L is generated by a grammar which is unambiguous,

(iv) *inherently ambiguous* if every grammar generating L is ambiguous. ▮

It has been shown [13] that there are context-free languages of unbounded inherent ambiguity and of inherent ambiguity k for every $k \geqslant 1$. We shall often apply these terms directly to grammars. Thus the statement "G is inherently ambiguous" means "$L(G)$ is inherently ambiguous."

We shall also need some basic notions concerning computational complexity.

DEFINITION 1.4. A function $f(n)$ is said to be *polynomially bounded* (*exponentially bounded*) if there exists a polynomial $p(n)$ and integer n' such that $f(n) \leqslant p(n)$ ($f(n) \leqslant 2^{p(n)}$) for all $n > n'$. P, NP, E, and NE are the classes of languages accepted by deterministic and nondeterministic, polynomially and exponentially, time bounded Turing machines, respectively. ▮

It is not currently known whether $P = NP$ or $E = NE$. Nevertheless certain languages called *complete* languages reflect the complexity of each entire class.

DEFINITION 1.5. A language L_0 is said to be *NP-hard* (*NE-hard*) if the following condition is true: Given a deterministic polynomially (exponentially) time bounded Turing machine to recognize L_0 and a language L in NP (NE), we can effectively find a deterministic polynomially (exponentially) time bounded Turing machine to recognize L. If in addition, L_0 is a member of NP (NE) then L_0 is said to be *NP-complete* (*NE-complete*). ▮

We shall occasionally deal with various types of transformations between languages.

DEFINITION 1.6. Let Σ and Δ be finite alphabets. A function $f: \Sigma^* \to \Delta^*$ is said to be *log space computable* if there exists a deterministic Turing machine M such that

(1) M has a two-way read-only input tape,

(2) M has a one-way write-only output tape,

(3) M has one or more two-way read-write work tapes,

(4) M when started with x on its input tape produces $f(x)$ on its output tape,

(5) M never uses more than $0(\log |x|)$ tape cells on its work tapes. ▮

We may now define various types of transformations.

DEFINITION 1.7. A language L is *transformable* to a language L' (written $L \leqslant L'$) if there exists a function f such that $x \in L$ if and only if $f(x) \in L'$. If f is computable by a polynomially time bounded deterministic Turing machine then we say that L is *polynomially transformable* to L' (written $L \leqslant_{p\text{-time}} L'$). If f is log space computable we say L is *log space transformable* to L' (written $L \leqslant_{\log} L'$). If in addition $|f(x)| = 0(|x|)$ we say that L is log–lin space transformable to L' (written $L \leqslant_{\log\text{-lin}} L'$). ▮

Notice that $L \leqslant_{\log\text{-lin}} L'$ implies $L \leqslant_{\log} L'$ implies $L \leqslant_{\text{p-time}} L'$. If every language in a set of languages S is transformable to some language L_0, we shall write $S \leqslant L_0$, subscripting the $\leqslant$ if necessary to indicate a particular kind of transformability.

We shall need one technical lemma concerning inherent ambiguity.

LEMMA 1.8. *Let L be a context-free language and h be a homomorphism. Then the degree of inherent ambiguity of $h^{-1}(L)$ is less than or equal to the degree of inherent ambiguity of L.*

Proof. We give a proof modeled after one appearing on [10, p. 172]. Let P be any pushdown automaton (PDA) which accepts L. Construct a PDA P' for $h^{-1}(L)$ as follows: P', on input x, applies h to x one character at a time, buffering the result in its finite control and simulating P on the result. Thus P' accepts x if and only if P accepts $h(x)$. Hence P' accepts exactly $h^{-1}(L)$. Moreover, the number of distinct ways in which P' accepts x is no greater than the number of ways in which P accepts

$h(x)$. Thus, the degree of ambiguity of P' is no greater than the degree of ambiguity of P. Rephrasing this in terms of languages and inherent ambiguity, we have our result. ▌

2. Undecidability Metatheorems for Context-Free Grammar Problems

Several powerful metatheorems for proving the undecidability of arbitrary predicates on the context-free languages have appeared in the literature [5, 7]. The main result of this section is a metatheorem which is applicable to predicates defined on grammars rather than languages.

The reader would be well advised to pause at this point and reflect upon the distinction between language and grammar problems. Decidability questions involving languages must always involve a level of "indirection" as exemplified by "given a grammar G, is $L(G)$ empty?" This indirection is required by the fact that languages are in general infinite objects and as such must always be defined by some sort of finite descriptor. This indirection is not needed in typical grammar questions such as "given a grammar G, is G ambiguous?"

To further clarify this distinction let us contrast the class of recursively enumerable sets, for which no nontrivial predicate is decidable, with the class of type 0 grammars, for which many nontrivial predicates are decidable.

The statement and proof of our metatheorem depend on the following lemma.

Lemma 2.1. *Let M be a deterministic or nondeterministic Turing machine and $T(n)$ be a strictly increasing recursive function such that:*

(i) *M has one semi-infinite tape on which the input is placed left-justified;*

(ii) *M never "falls off" the left end of its tape;*

(iii) *M never performs more than $T(n)$ steps on any input of length n;*

(iv) *M only enters an accepting state when at the extreme right end of the utilized portion of its tape;*

(v) *M can make no move out of an accepting state.*[2]

Let Σ be the input alphabet of M. Let $x \in \Sigma^*$ and let k_x be at least $2 \cdot T(|x|)^2 + 6 \cdot T(|x|) + 5$. Then there exists a grammar $G_{M,x}$ and constants c_1, c_2, and c_3, depending only on M such that:

(1) *$G_{M,x}$ is of size $c_1 + |x|$;*

(2) *$G_{M,x}$ may be constructed from x in deterministic time $c_2 \cdot |x| + c_3$, on a multi-tape Turing machine;*

[2] Conditions (i) through (v) may be assumed without loss of generality for any Turing machine which accepts a recursive set, although the running time $T(n)$ may have to be increased.

(3) *the following are equivalent*:

(a) *M accepts x*,

(b) *$G_{M,x}$ is of unbounded inherent ambiguity*,

(c) *$G_{M,x}$ is not strong* LL(k_x),

(d) *$G_{M,x}$ is not* SLR(k_x),

(e) *$G_{M,x}$ is not* BRC(k_x, k_x). ▌

Proof. Let Q, Q_f, and Γ be respectively the sets of states, final accepting states, and tape symbols of M. Let q_0 be the start state of M. Let \$, ¢, #, a, b, and c be new symbols.

The grammar $G_{M,x}$ includes the following productions

$$\begin{aligned}
&S \rightarrow ABCS \mid \bar{A}\bar{B}\bar{C}S \mid ¢,\\
&A \rightarrow \$,\\
&\bar{A} \rightarrow \$,\\
&B \rightarrow q_0 x \,\#\, D,
\end{aligned}$$

plus many additional productions which depend only on M. The complete construction of $G_{M,x}$ is presented in Appendix I. It should be clear that $G_{M,x}$ is of size $c_1 + |x|$ and may be constructed in time $c_2 \cdot |x| + c_3$ for appropriate constants c_1, c_2, c_3 depending only on M.

Any string derived from B is of the form

$$w_1 \# w_2 \# \cdots \# w_{2t} \#$$

such that

$$\begin{aligned}
&t \text{ is a positive integer},\\
&w_1 = q_0 x,\\
&w_i \in \Sigma^* \cdot (Q - Q_f) \cdot \Sigma^* \qquad \text{for} \quad 1 < i < 2t,\\
&w_{2t} \in Q_f \cdot \Sigma^*,\\
&w_{2i}^{\text{rev}} = w_{2i+1} \qquad \text{for} \quad 1 \leqslant i < t.
\end{aligned}$$

Any string derived from $\bar{B}$ is of the form

$$w_1 \# w_2 \# \cdots \# w_{2t} \#$$

such that

$$\begin{aligned}
&t \text{ is a positive integer},\\
&w_i \in \Sigma^* Q \Sigma^* \qquad \text{for} \quad 1 \leqslant i \leqslant 2t,\\
&w_{2i-1} \underset{M}{\vdash} w_{2i}^{\text{rev}} \qquad \text{for} \quad 1 \leqslant i \leqslant t.
\end{aligned}$$

The nonterminal C derives

$$\{a^i b^j c^l \mid i, j, l \geqslant 1 \text{ and } i = j\},$$

whereas the nonterminal $\bar{C}$ derives

$$\{a^i b^j c^l \mid i, j, l \geqslant 1 \text{ and } i = l\}.$$

We must now show the equivalence of statements (a) through (e). Suppose first that M accepts x. Then for some $m \leqslant T(|x|)$, there exists an accepting computation of M on x which we may write as

$$\text{id}_0 \underset{M}{\vdash} \text{id}_1 \underset{M}{\vdash} \text{id}_2 \underset{M}{\vdash} \cdots \underset{M}{\vdash} \text{id}_m .$$

Let z be the string

$$\text{id}_0 \# \text{id}_1^{\text{rev}} \# \text{id}_1 \# \text{id}_2^{\text{rev}} \# \cdots \# \text{id}_{m-1} \# \text{id}_m^{\text{rev}} \#.$$

Notice that z is derivable from both B and $\bar{B}$.

Define a homomorphism h by

$$\begin{aligned} h(\$) &= \$z, \\ h(a) &= a, \\ h(b) &= b, \\ h(c) &= c, \\ h(¢) &= ¢. \end{aligned}$$

Then

$$h^{-1}(L(G_{M,x})) = \{\$a^i b^j c^l \mid i = j \text{ or } i = l\}^* \cdot \{¢\}.$$

Clearly this is a language of unbounded inherent ambiguity. Since, by Proposition 1.8, an inverse homomorphism cannot increase the degree of inherent ambiguity of a language, we can conclude that $G_{M,x}$ must be a grammar of unbounded inherent ambiguity. Thus $G_{M,x}$ is ambiguous and certainly not BRC(k, k), SLR(k), or strong LL(k) for any value of k, let alone for $k = k_x$. We have thus established that (a) implies (b) through (e).

Suppose then that M does not accept x. Close examination of the productions of $G_{M,x}$ reveal that $G_{M,x}$ is very easily parsed *provided* that the parser can somehow "tell the difference" between the productions $A \to \$$ and $\bar{A} \to \$$. More formally, $G_{M,x}$ is BRC(k_x, k_x), SLR(k_x), and strong LL(k_x) if and only if

$$\text{FOLLOW}_{k_x}(A) \cap \text{FOLLOW}_{k_x}(\bar{A}) = \varnothing .$$

Accordingly, let y be an arbitrary member of $\mathrm{FOLLOW}_{k_x}(A) \cap \mathrm{FOLLOW}_{k_x}(\bar{A})$. Two cases arise depending on whether y contains the symbol a or not. We shall show that both cases lead to contradictions and that y cannot exist. This will prove that $\mathrm{FOLLOW}_{k_x}(A) \cap \mathrm{FOLLOW}_{k_x}(\bar{A})$ is empty and hence that $G_{M,x}$ is $\mathrm{BRC}(k_x, k_x)$, $\mathrm{SLR}(k_x)$, and strong $\mathrm{LL}(k_x)$.

Case 1. y contains one or more a's.

Then we can write y as $y'ay''$ where the indicated a is the leftmost occurrence of a in y. Inspection of $G_{M,x}$ reveals that y' is of the form

$$u_1 \# u_2 \# \cdots \# u_{2t} \#,$$

where t is a positive integer and each u_i is a member of $\Sigma^* Q \Sigma^*$. Moreover, y' must be derivable from both B and $\bar{B}$. Since y' is derivable from B we must have $u_1 = q_0 x$, $u_2^{\mathrm{rev}} = u_3$, $u_4^{\mathrm{rev}} = u_5, \ldots, u_{2t-2}^{\mathrm{rev}} = u_{2t-1}$, and $u_{2t} \in Q_f \Sigma^*$. Since y' is derivable from $\bar{B}$ we must have $u_1 \vdash_M u_2^{\mathrm{rev}}$, $u_3 \vdash_M u_4^{\mathrm{rev}}, \ldots, u_{2t-1} \vdash_M u_{2t}^{\mathrm{rev}}$. Putting these together, there must exist a computation of M on x, namely

$$u_1 \underset{M}{\vdash} u_2^{\mathrm{rev}} \underset{M}{\vdash} u_4^{\mathrm{rev}} \underset{M}{\vdash} \cdots \underset{M}{\vdash} u_{2t}^{\mathrm{rev}}.$$

Moreover, this is an accepting computation since $u_{2t}^{\mathrm{rev}} \in \Sigma^* \cdot Q_f$. Since we have hypothesized that M does *not* accept x, we can conclude that this case cannot occur.

Case 2. y contains no a's.

In this case we must have $|y| = k_x$ and y is the prefix of some string z derivable from B and also a prefix of some $\bar{z}$ derivable from $\bar{B}$. Thus we may write

$$y = \mathrm{FIRST}_{k_x}(z) = \mathrm{FIRST}_{k_x}(\bar{z})$$
$$B \overset{*}{\Rightarrow} z \quad \text{and} \quad \bar{B} \overset{*}{\Rightarrow} \bar{z}.$$

Moreover, it is possible to write

$$y = u_1 \# u_2 \# \cdots \# u_{2t} \# v$$

for some $t \geqslant 0$ and some v containing either 0 or 1 $\#$'s, so that each u_i is a member of $\Sigma^* Q \Sigma^*$.

For convenience, let us rewrite these as

$$B \overset{*}{\Rightarrow} z = u_1 \# u_2 \# \cdots \# u_{2t} \# vz',$$
$$\bar{B} \overset{*}{\Rightarrow} \bar{z} = u_1 \# u_2 \# \cdots \# u_{2t} \# v\bar{z}'$$

for some z' and $\bar{z}'$.

By an argument similar to that used in Case 1, we have

$$\begin{aligned} u_1 &= q_0 x, & u_1 &\underset{M}{\vdash} u_2^{\text{rev}}, \\ u_2^{\text{rev}} &= u_3, & u_3 &\underset{M}{\vdash} u_4^{\text{rev}}, \\ u_{2t-2}^{\text{rev}} &= u_{2t-1}, & u_{2t-1} &\underset{M}{\vdash} u_{2t}^{\text{rev}}. \end{aligned}$$

This, of course, represents a t step computation of M on x, namely,

$$u_1 \underset{M}{\vdash} u_2^{\text{rev}} \underset{M}{\vdash} \cdots \underset{M}{\vdash} u_{2t}^{\text{rev}}.$$

By hypothesis, M never performs more than $T(n)$ steps on an input of length n, and so $t \leqslant T(|x|)$. Each instantaneous description of M operating on x can consist of at most $T(|x|) + 1$ characters because M starts at the end of a semi-infinite tape and can "reach" at most $T(|x|)$ tape cells during a computation (the $+1$ term in the size bound for instantaneous descriptions is due to the single character needed to represent the state and head position of the machine). Thus $|u_1 \# \cdots \# u_{2t} \#| \leqslant 2 \cdot T(|x|)[T(|x|) + 2]$ and $|v|$ is at least $2 \cdot T(|x|) + 5$.

Since $B \overset{*}{\Rightarrow} z$ and u_{2t} is *not* a member of $Q_f \cdot \Sigma^*$, $u_{2t}^{\text{rev}} \#$ must be a prefix of vz'. Since $|u_{2t} \#|$ is at most $T(|x|) + 2$, this means that for some v' we can write

$$v = u_{2t}^{\text{rev}} \# v'$$

with

$$|v'| \geqslant T(|x|) + 3.$$

Thus $z = u_1 \# \cdots \# u_{2'} \# u_{2t}^{\text{rev}} \# v'z'$.

By definition of $G_{M,x}$, there exists some prefix $w \#$ of $v'z'$ such that $u_{2t}^{\text{rev}} \vdash_M w^{\text{rev}}$. Moreover, $|w| \leqslant |u_{2t}| + 1 \leqslant T(|x|) + 2$. Thus w is a proper prefix of v' and so $w \#$ is a prefix of v'. But this means $u_{2t}^{\text{rev}} \# w \#$ is a prefix of v contradicting the definition of v as containing at most 1 occurrence of $\#$. We conclude therefore that this case cannot occur.

At this point we have used the hypothesis that M does not accept x to show that $\text{FOLLOW}_{k_x}(A) \cap \text{FOLLOW}_{k_x}(\bar{A})$ is empty and that $G_{M,x}$ is therefore SLR(k_x), BRC(k_x, k_x), strong LL(k_x), and unambiguous. Thus the negation of (a) implies the negation of (b) through (c) and the theorem is proved. ∎

We may now present the main result of this section.

THEOREM 2.2. *Let Γ be any subset of the context-free grammars such that:*

(1) *Γ includes all grammars which are strong* LL, SLR, *and* BRC,

(2) *Γ includes no grammars of unbounded inherent ambiguity.*

Then it is undecidable whether an arbitrary grammar is a member of Γ.

Proof. Suppose that the membership problem in Γ were decidable. Then there exists some strictly increasing recursive function $f(n)$ which bounds from above the time needed to deterministically decide on a multitape Turing machine whether a grammar of size n is a member of Γ.

Let R be an arbitrary recursive set. Then there exists a deterministic one tape Turing machine M and a recursive function T such that M accepts R and M and T satisfy constraints (i) through (v) of Lemma 2.1.

We shall now describe an efficient recognition algorithm for R. Given a string x, first construct $G_{M,x}$ according to Lemma 2.1 and then test whether $G_{M,x}$ is a member of Γ. If x is a member of R, then M accepts x and $G_{M,x}$ is of unbounded inherent ambiguity and therefore not a member of Γ. Conversely, if x is not a member of R, then M does not accept x and $G_{M,x}$ is strong LL(k), SLR(k), and BRC(k, k) for $k = k_x$. Thus $G_{M,x}$ certainly is a member of Γ. Thus $x \in R$ if and only if $G_{M,x} \notin \Gamma$. The recognition algorithm for R which was just described may be executed deterministically on a multi-tape Turing machine in time $f(c_1 + |x|) + c_2 \cdot |x| + c_3$ where the constants c_1, c_2, and c_3 are the constants of Lemma 2.1 which depend only on M. The last two terms in the expression are, of course, the time needed to construct $G_{M,x}$ from x and the leading term is the time needed to test whether $G_{M,x}$ is a member of Γ.

Since R in the above construction was an *arbitrary* recursive set, we can conclude that any recursive set may be recognized by a deterministic multitape Turing machine in time $f(a + n) + bn + c$ for appropriate constants a, b, and c which depend only on R. Consider now the recursive function $F(n) = f(2n) + n^2$ which is strictly greater than $f(a + n) + bn + c$ almost everywhere regardless of the choice of a, b, and c. Therefore, any recursive set may be recognized in time $F(n)$.

It is well known, however, that for every recursive function $r(n)$, there exists a recursive set which *cannot* be recognized in time $r(n)$ [6]. We must therefore conclude that Γ does not have a decidable membership problem. ▮

The key idea in the proof of Theorem 2.2 is that undecidability results can be derived by embedding arbitrary *halting* computations in the problem at hand. More conventional approaches seek to establish undecidability by embedding arbitrary (and perhaps nonterminating) computations in a given problem. These approaches are not sufficiently powerful to prove Theorem 2.2, as the reader can readily verify by trying to construct a grammar $\bar{G}_{M,x}$ which is ambiguous if Turing machine M accepts string x and LL if M does not.

Theorem 2.2 may be used to show the undecidability of many classes of grammars studied in the literature.

COROLLARY 2.3. *The following predicates are undecidable on the context-free grammars.*

(1) *G is* BRC,

(2) *G is strong* LL,

(3) *G is* LL,

(4) *G is strong* LC,

(5) *G is* LC,

(6) *G is* ELC,

(7) *G is* SLR,

(8) *G is* LALR,

(9) *G is* LR,

(10) *G is* LR-*regular,*

(11) *G is unambiguous,*

(12) *G is ambiguous of degree k_0 for any fixed integer k_0*,

(13) *G is basic* SPM *parsable,*

(14) *G is full* SPM *parsable,*

(15) *G is* FSPA,

(16) *G is* RPP,

(17) $\exists k, t$ *such that G is* LR(k, t),

(18) $\exists k$ *such that G is* LR(k, ∞).

Proof. These classes of grammars all satisfy the conditions of Theorem 2.2. ELC grammars are defined in [2], LR-regular grammars in [3], SPM grammars in [4], FSPA, RPP, and LR(k, ∞) grammars in [14] and LR(k, t) grammars are defined in [12] and [14]. All remaining classes are defined in [1]. ∎

The reader familiar with the history of parsing theory will note that Theorem 2.2 shows that no parsing technique exists which

(1) works for all LR grammars,

(2) works for no ambiguous grammars,

(3) has a decidable "membership" problem.

3. Lower Bounds for Grammar Problems

Many of the more thoroughly explored classes of context-free grammars are actually infinite hierarchies. For example, the class of LR grammars may be considered to be the union or limit of the classes of LR(0), LR(1), LR(2), etc., grammars.

It is interesting to note that although the membership problem for each of the individual subclasses (i.e., LR(k) for any fixed k) is decidable, the membership problem for the general class (LR) is not decidable. The undecidability result given in Theorem 2.2 is useful for establishing this latter fact but is of no use in establishing lower bounds on the membership problem for subclasses. For this reason the current section is devoted to investigating the complexity of the membership problem for parameterized hierarchies of grammar classes.

DEFINITION 3.1. Let $\Gamma = \Gamma(0), \Gamma(1), \Gamma(2),\ldots$ be an infinite hierarchy of classes of grammars. A *uniform algorithm for Γ testing* is an algorithm which takes two inputs,

(i) a context-free grammar G,

(ii) an integer k

and which determines whether G is a member of $\Gamma(k)$. ▌

This paper will not treat complexity issues involving specific members within a hierarchy such as, say, the members LR(3) and LR(17) within the LR hierarchy (such problems are considered in [8, 9]). Instead we restrict our attention to the complexity of uniform algorithms for testing membership in a hierarchy.

Our treatment requires that we be able to encode arbitrary grammars as strings over the fixed alphabet $\{0, 1\}$. We shall leave the specific details of the encoding to the reader.

PROPOSITION 3.2. (i) *Any grammar of size n with vocabulary V can be represented as a binary string of length* $0(n \log |V|)$,

(ii) *Any grammar of size n can be represented as a binary string of size* $0(n \log n)$.

Proof. (i) Each position of the grammar can be represented in $\log |V|$ bits with perhaps an extra bit added to delimit the end of each production.

(ii) Follows from (i) and the realization that no grammar of size n can have a vocabulary with more than n elements. ▌

We shall subsequently denote the binary representation of a grammar G by *REP*(G). We now come to the main result of this section.

THEOREM 3.3. *Let $\Gamma = \Gamma(0), \Gamma(1), \Gamma(2),\ldots$ be an infinite hierarchy of classes of grammars such that*

(1) *$\Gamma(i)$ contains all the grammars which are simultaneously strong* LL(i), SLR(i), *and* BRC(i, i),

(2) *no grammar of $\Gamma(i)$ is of unbounded inherent ambiguity. Let*

$$\Gamma_u = \{\text{REP}(G) \# \nu \mid \nu \textit{ is the unary representation of an integer } i \textit{ and the grammar } G \textit{ is not a member of } \Gamma(i)\}.$$

Let

$$\Gamma_b = \{\text{REP}(G) \# \nu \mid \nu \textit{ is the binary representation of an integer } i \textit{ and the grammar } G \textit{ is not a member of } \Gamma(i)\}.$$

Then

(i) $\text{NP} \leqslant_{\log} \Gamma_u$,

(ii) $\text{NE} \leqslant_{\log} \Gamma_b$.

Proof. The proofs of (i) and (ii) are quite similar. Accordingly, we shall only prove the latter.

Let L be any language in NE. There exists a polynomial $p(n)$ and a nondeterministic one-tape Turing machine M such that

(1) M recognizes L,

(2) M satisfies constraints (i) through (v) of Lemma 2.1 for $T(n) = 2^{p(n)}$.

Let x be a string in Σ^* where Σ is the alphabet of L. Consider the grammar $G_{M,x}$ of Lemma 2.1. Since the vocabulary size of $G_{M,x}$ depends only on M, there must exist constants d_1 and d_2 depending only on M such that $|\text{REP}(G_{M,x})| \leqslant d_1 + d_2 \cdot |x|$. Let $k = 8 \cdot [2^{p(|x|)}]^2 + 5$. Then ν, the binary representation of k is of length at most $2 \cdot p(|x|) + 4$.

It should thus be clear that given any x, we can produce a string $w_{M,x} = \text{REP}(G_{M,x}) \# \nu$ such that

(1) $w_{M,x} \in \Gamma_b$ if and only if x is not a member of L,

(2) $|w_{M,x}|$ is at most $c_1 \cdot |x|^{c_2}$ where c_1 and c_2 depend only on M,

(3) $w_{M,x}$ can be constructed from x in $O(p(|x|))$ time and $O(\log |x|)$ space on a deterministic multitape Turing machine.

We thus have $L \leqslant_{\log} \Gamma_b$ as was to be shown.

This result immediately yields the lower bounds expressed below.

COROLLARY 3.4. *Let Γ be as Theorem* 3.3. *Then*

(i) Γ_u *is* NP-*hard*,

(ii) Γ_b *is* NE-*hard*,

(iii) *there exists a constant $c > 0$ such that any nondeterministic Turing machine which accepts Γ_b uses more than $2^{c|x|}$ time on infinitely many inputs x.*

Proof. Straightforward. ∎

As before, our theorem applies to many well-known classes of grammars.

COROLLARY 3.5. *The following classes of context-free grammars satisfy the conditions of Theorem* 3.3:

(1) BRC,

(2) *strong* LL,

(3) LL,

(4) *strong* LC,

(5) LC,

(6) ELC,

(7) SLR,

(8) LALR,

(9) LR. ∎

We close the section by noting that the lower bounds given by Theorem 3.3 are tight, in that algorithms achieving these run times are known for many hierarchies to which the metatheorem applies. The reader is referred to [9] for further details.

CONCLUSION

We have presented a way of embedding the computations of an always-halting Turing machine in a context-free grammar in such a manner that the grammars was either very "nice" or very "messy" (i.e., easily parsable or of unbounded inherent ambiguity). This embedding gave rise to a powerful undecidability metatheorem for context-free grammar problems as well as specific lower bounds on the running time of any uniform algorithm for deciding membership in many hierarchies of grammars.

APPENDIX I: CONSTRUCTING $G_{M,x}$

Let M satisfy the constraints of Lemma 2.1.
Let Σ be the input alphabet of M,

Γ be the tape alphabet of M,

Q be the state set of M,

Q_f be the set of accepting states of M,

q_0 be the start state of M,

S be the move function of M, that is a function mapping $Q \times \Gamma$ into subsets of $Q \times \Gamma \times \{L, R, S\}$,

x be an element of Σ^*.

We shall assume that $\Sigma \subseteq \Gamma$ and that $\{\$, ¢, \#, a, b, c\}$, Γ and Q are all pairwise disjoint. The blank tape symbol is denoted by $\not{b}$ and is a member of $\Gamma - \Sigma$. All nonterminals of $Q_{M,x}$ are either upper case Roman letters or elements of $(\Gamma \cup \{\#\}) \times Q \times \Gamma$. Nonterminals of the latter category are denoted by $\langle \sigma_1, q, \sigma_2 \rangle$. The complete set of productions of $G_{M,x}$ is broken down for convenience into six groups as follows:

$$\begin{array}{lll}
\text{group 1:} & S \to ABCS \mid \bar{A}\bar{B}\bar{C}S \mid ¢, & \\
& A \to \$, & \\
& \bar{A} \to \$, & \\
& B \to q_0 x \# D; & \\
\text{group 2:} & D \to E \# D \mid qG & \forall q \in Q_f, \\
& E \to \sigma E \sigma \mid qFq & \forall \sigma \in \Gamma, \quad q \in Q_f, \\
& F \to \sigma F \sigma \mid \# & \forall \sigma \in \Gamma, \\
& G \to \sigma G \mid \# & \forall \sigma \in \Gamma; \\
\text{group 3:} & \bar{B} \to \bar{D} \# \bar{B} \mid \lambda & \\
& \bar{D} \to q \# \langle \# q \not{b} \rangle & \forall q \in Q, \\
& \bar{D} \to q\sigma\bar{F}\langle \# q \sigma \rangle & \forall \sigma \in \Gamma, \quad q \in Q, \\
& \bar{D} \to \bar{E}, & \\
& \bar{E} \to \sigma \bar{E} \sigma & \forall \sigma \in \Gamma, \\
& \bar{E} \to \sigma_1 q \sigma_2 \bar{F} \langle \sigma_1 q \sigma_2 \rangle & \forall \sigma_1, \sigma_2 \in \Gamma, \quad q \in Q, \\
& \bar{E} \to \sigma_1 q \# \langle \sigma_1 q \not{b} \rangle & \forall \sigma_1 \in \Gamma, \quad q \in Q, \\
& \bar{F} \to \sigma \bar{F} \sigma \mid \# & \forall \sigma_1 \in \Gamma; \\
\text{group 4:} & \langle \tau q \sigma \rangle \to \sigma' \tau q' & \text{if } (q', \sigma', L) \in \delta(q, \sigma) \text{ and } \tau \in \Gamma, \\
& \langle \tau q \sigma \rangle \to \sigma' q' \tau & \text{if } (q', \sigma', S) \in \delta(q, \sigma) \text{ and } \tau \in \Gamma \cup \{\#\}, \\
& \langle \tau q \sigma \rangle \to q' \sigma' \tau & \text{if } (q', \sigma', R) \in \delta(q, \sigma) \text{ and } \tau \in \Gamma \cup \{\#\}; \\
\text{group 5:} & C \to HI, & \\
& H \to JHb \mid Jb, & \\
& J \to a, & \\
& I \to cI \mid c; & \\
\text{group 6:} & \bar{C} \to \bar{J}\bar{C}c \mid \bar{J}\bar{G}c, & \\
& \bar{G} \to b\bar{G} \mid b, & \\
& \bar{J} \to a. &
\end{array}$$

The productions of group 1 are those that were given in the main text of the paper. Group 2 produces the $w_2 \# w_3 \# \cdots \# w_{2t} \#$ portion of strings derivable from B.

The nonterminal D first produces $(E\#)^{t-1}\,qG$, from which E produces each $w_{2i}^{\text{rev}} \# w_{2i+1}$ substring and qG produces w_{2t}.

Group 3 produces the strings derivable from $\bar{B}$. First $\bar{B}$ generates $(\bar{D}\#)^t$. Each $\bar{D}$ then produces a representation of a single move of the Turing machine M, i.e., a string of the form $w \# y$ such that $w \vdash_M y^{\text{rev}}$. This is normally done by a derivation of the form

$$\begin{aligned} \bar{D} &\Rightarrow \bar{E}, \\ &\overset{*}{\Rightarrow} u\bar{E}u^{\text{rev}}, \\ &\Rightarrow u\sigma_1 q\sigma_2 \bar{F}\langle\sigma_1 q\sigma_2\rangle u^{\text{rev}}, \\ &\overset{*}{\Rightarrow} u\sigma_1 q\sigma_2 v \# v^{\text{rev}}\langle\sigma_1 q\sigma_2\rangle u^{\text{rev}}, \end{aligned}$$

where $u, v \in \Gamma^*$, $\sigma_1, \sigma_2 \in \Gamma$, $q \in Q$. This portion of the grammar is made somewhat messy by the need to handle the cases in which the tape head is at an extreme end of the tape.

Group 4 productions simulate the actual move of the Turing machine.

Group 5 productions generate strings of $\{a^i b^j c^l \mid i, j, l \geqslant 1, i = j\}$. The a's are produced via the nonterminal J so that this portion of the grammar will be BRC.

Finally group 6 produces strings of $\{a^i b^j c^l \mid i, k, l \geqslant 1, i = l\}$.

References

1. A. V. Aho and J. D. Ullman, "The Theory of Parsing, Translation and Compiling," Volumes 1, 2, Prentice-Hall, Englewood Cliffs, N. J., 1972, 1973.
2. B. M. Brosgol, Deterministic translation grammars, *in* "Proceedings of the 8th Annual Princeton Conference on Information Sciences and Systems," pp. 300–306, 1974.
3. K. Culik, II and R. Cohen, LR-regular grammrs—an extension of *LR*(*k*) grammars, *J. Comput. System Sci.* **7** (1973), 66–96.
4. C. N. Fischer, On parsing context-free languages in parallel environments, Ph. D. Thesis, Cornell University, Ithaca, New York, April 1975.
5. S. A. Griebach, A note on undecidable properties of formal languages, *Math. Systems Theory* **2** (1968), 1–6.
6. J. Hartmanis and J. E. Hopcroft, An overview of the theory of computational complexity, *J. Assoc. Comput. Mach.* **18** (1971), 444–475.
7. H. B. Hunt, III and D. J. Rosenkrantz, Computational parallels between the regular and context-free languages, *in* "Proceedings of the 6th Annual ACM Symposium on Theory of Computing," pp. 64–74, 1974.
8. H. B. Hunt, III and T. G. Szymanski, Lower bounds and reductions between grammar problems, in preparation.
9. H. B. Hunt, T. G. Szymanski, and J. D. Ullman, "On the complexity of *LR*(*k*) testing, *Comm. ACM* **18** (1975), 707–716.
10. J. E. Hopcroft and J. D. Ullman, "Formal Languages and Their Relation to Automata," Addison-Wesley, Reading, Mass., 1969.

11. A. J. Korenjak and J. E. Hopcroft, Simple deterministic languages, *in* "IEEE Conference Record of the 7th Annual Symposium on Switching and Automata Theory," pp. 36–46, 1966.
12. D. E. Knuth, On the translation of languages from left to right, *Inform. Contr.* 8 (1965), 609–639.
13. W. F. Ogden, A helpful result for proving inherent ambiguity, *Math. Systems Theory* 2 (1968), 191–194.
14. T. G. Szymanski and J. H. Williams, Non-canonical extensions of bottom-up parsing techniques, *SIAM J. Computing*, to appear.