WIS-CS-75-241

COMPUTER SCIENCES DEPARTMENT
University of Wisconsin
1210 West Dayton Street
Madison, Wisconsin 53706

Received February 20, 1975

ON THE COMPLEXITY OF GRAMMAR AND RELATED PROBLEMS

H. B. Hunt, III
University of Wisconsin - Madison

and

T. G. Szymanski
Princeton University

Technical Report #241

February 1975

ON THE COMPLEXITY OF GRAMMAR AND RELATED PROBLEMS

H. B. Hunt, III[†]
University of Wisconsin - Madison
and
T. G. Szymanski[‡]
Princeton University

1. Introduction

In [1] and [2] a complexity theory for formal languages and automata was developed. This theory implies most of the previously known results and yields many new results as well. Here we develop an analogous theory for several classes of more practically motivated problems. Two such classes, both closely related to formal language and automata theory, suggest themselves - grammar problems and program scheme problems. Here, our primary emphasis is on grammar problems of interest in parsing and compiling. Other problems considered include -

(1) possible techniques for proving non-trivial lower complexity bounds for problems in P ;

(2) the relationship of the complexity of tree automaton equivalence, structural equivalence, and grammatical covering; and

(3) the complexity of the equivalence problem for schemes.

In each case we relate the computational complexity of a problem to its underlying combinatorial structure. The remainder of the paper is divided into four sections.

In Section 2 we consider context-free grammar problems. In 2.1 we show that most of the known undecidability results about context-free grammar problems follow from one simple idea. Roughly, any class of grammars that contains the intersection of the strong LL , and SLR grammars and is contained in any reasonable proper subclass of the context-free grammars (e.g. the unambiguous context-free grammars) is undecidable. Thus there is no need for the special constructions, such as the partial Post Correspondence Problem [3] or the iterated partial Post Correspondence Problem [4], used in the literature. Moreover, all of the known nontrivial lower bounds for decidable grammar problems in [5] also follow from our theory. In 2.2 we present relativizations of the results in 2.1. A general complexity theorem for non-canonical parsing ([6] or [7], pp. 485-487) is also presented.

In Section 3 we consider the problem of proving nontrivial lower complexity bounds (both time and space) for problems in P . Several partial results are obtained. In 3.1 the following is shown - for all integers $k_0 \geq 1$, for all classes of context-free grammars Γ such that the $LL(k_0)$ grammars $\subseteq \Gamma \subseteq$ the $LR(k_0)$ grammars, and for arbitrary context-free grammar G , the predicate "$G \in \Gamma$" requires as much time and space as the predicate "G is an $LL(k_0)$ grammar". In 3.2 results about stack automata in [2] are extended to multi-head finite and pushdown automata as well. Our results reveal two simple ideas that underlie many of the results in [2], [8], [9], [10], [11], [12], [13], [14], and [15]. One interesting corollary is - <u>all</u> nontrivial predicates on the context-free languages and many nontrivial predicates on the deterministic context-free languages, when applied to the pushdown and deterministic pushdown automata, respectively, require as much time and

---

[†]The research of the first author was supported in part by NSF Grant GJ-35570 and by U.S. Army Contract No. DA-31-124-ARO-D-462.

[‡]The research of the second author was supported in part by NSF Grant GJ-35570.

space as the recognition of <u>any</u> 2-way pushdown automaton language. The best known algorithms for 2-way pushdown automaton recognition require $O(n^3)$ operations on RAM's([15] and Section 3.2 below). In 3.3 the results in [2] and the rest of Sections 2 and 3 of this paper are extended to automata and grammars on trees rather than strings. In Section 4 we consider the complexity of several decidable problems about program schemes.

We list several abbreviations, definitions, and lemmas used in the remainder of this paper. We assume that the reader is familiar with the basic definitions and results concerning context-free grammars and languages, otherwise see [7]. We use λ to denote the empty string. The language generated (accepted) by a grammar (automaton) G is denoted by $L(G)$.

The following abbreviations are used throughout the remainder of this paper.

1. cfg - context-free grammar
2. cfl - context-free language
3. PDA - pushdown automaton
4. DFA - deterministic finite automaton
5. NDFA - nondeterministic finite automaton
6. Tm - Turing machine
7. SA - stack automaton
8. DSA - deterministic stack automaton
9. RAM - random access machine
10. BC - bounded context grammars
11. BCP - bounded context parsable grammars [16]
12. BRC - bounded right context grammars
13. SLR - simple LR grammars
14. i.o. - infinitely often
15. a.e. - almost everywhere, when applied to the nonnegative integers almost everywhere means except for a finite set of nonnegative integers.

<u>Def. 1.1</u>: A cfg G is said to be <u>ambiguous</u> if some string $x \in L(G)$ has two distinct left-most derivations, or equivalently, two distinct right-most derivations, or equivalently, two distinct derivation trees. G is said to be <u>inherently ambiguous</u> if all cfg's generating $L(G)$ are ambiguous. ■

<u>Def. 1.2</u>: Let k be a positive integer. A cfg G is said to be <u>ambiguous of degree k</u> if each string $x \in L(G)$ has at most k distinct derivation trees and some string $x \in L(G)$ has at least k distinct derivation trees. G is said to be <u>inherently ambiguous of degree k</u> if every grammar generating $L(G)$ is ambiguous of degree $\geq k$ and some grammar generating $L(G)$ is ambiguous of degree k .

G is said to be <u>infinitely ambiguous</u> if for each positive integer ℓ , there exists a string $x \in L(G)$ such that x has at least ℓ distinct derivation trees. G is said to be <u>infinitely inherently ambiguous</u> if each grammar generating $L(G)$ is infinitely ambiguous. It is known that for all $k \geq 2$ there exists an inherently ambiguous cfg of degree k . Similarly it is known that there exist infinitely inherently ambiguous cfg's [17]. ■

<u>Inherently ambiguous cfl's</u> , <u>inherently ambiguous cfl's of degree k</u> , and <u>infinitely inherently ambiguous cfl's</u> are defined analogously. Thus a cfl L is infinitely inherently ambiguous if every cfg generating L is infinitely inherently ambiguous. A cfl that is <u>not</u> inherently ambiguous is said to be unambiguous.

<u>Def. 1.3</u>: <u>P(NP)</u> is the class of all languages over $\{0,1\}$ accepted by some deterministic (nondeterministic) polynomially time-bounded Tm . <u>PSPACE</u> is the class of all languages over $\{0,1\}$ accepted by some polynomially space-bounded Tm . ■

<u>Def. 1.4</u>: Let Σ,Δ be finite nonempty alphabets. Let $L \subseteq \Sigma^*$ and $M \subseteq \Delta^*$. We say that L is <u>p-reducible</u> to M , written $L \underset{\text{ptime}}{\leq} M$, if there exists a function $f : \Sigma^* \to \Delta^*$ computable by a deterministic polynomially time-bounded Tm such that for all $x \in \Sigma^*$, $x \in L$ iff $f(x) \in M$. If L is p-reducible to M and M is p-reducible to L , then L and M are said to be <u>p-equivalent</u>. L is said to be <u>NP-hard(PSPACE-hard)</u> if all languages in NP(PSPACE) are p-reducible to L . L is said to be <u>NP-complete(PSPACE-complete)</u> if it is NP-hard(PSPACE-hard) and is accepted by some nondeterministic polynomially time-bounded (polynomially space-bounded) Tm . ■

<u>Def. 1.5</u>: A <u>log-space transducer</u> M is a deterministic Tm with a 2-way read-only input tape, a 1-way output tape, and several 2-way read-write work tapes such that M given input x always halts with some string y on its output tape and such that M never uses more than $O(\log|x|)$ tape cells on its work tapes. Let Σ,Δ be finite nonempty alphabets. A function $f : \Sigma^* \to \Delta^*$ is said to be <u>log-space computable</u> if $\exists$ a log-space transducer M such that M , when given input $x \in \Sigma^*$, eventually halts with output $f(x)$. For $L \subseteq \Sigma^*$ and $N \subseteq \Delta^*$ we say that L is <u>log-space reducible</u> to N , written $L \underset{\log}{\leq} N$, if $\exists$ a log-space computable function f such that for all $x \in \Sigma^*$, $x \in L$ iff $f(x) \in N$. If in addition $|f(x)| \leq t(|x|)$ and some log-space transducer that computes f is $O(t(|x|))$time-bounded, we denote this by $L \underset{\substack{\log \\ t(n)(\text{space+time})}}{\leq} N$.

Let <u>Ndtape(log n)</u> denote the class of all languages over $\{0,1\}$ accepted by some nondeterministic log n tape-bounded Tm . A language N is said to be <u>log-complete in Ndtape (log n)</u> if for all $L \in$ Ndtape(log n) , $L \underset{\log}{\leq} N$; and N is accepted by some nondeterministic log n tape-bounded Tm . Similarly N is said to be <u>log-complete in P</u> if for all $L \in P$, $L \underset{\log}{\leq} N$;

and N is accepted by some deterministic polynomially time-bounded Tm. ∎

The reader should note that every log-space transducer is polynomially time-bounded. Thus $L \underset{\log}{\leq} N$ implies $L \underset{ptime}{\leq} N$.

The following proposition lists some of the well-known properties of NP-complete languages, PSPACE—complete languages, etc.

<u>Prop. 1.6</u>:
(1) $P = NP$ iff $\exists$ an NP-complete language L_0 such that $L_0 \in P$.
(2) $P(NP) = PSPACE$ iff $\exists$ a PSPACE-complete language L_0 such that $L_0 \in P(NP)$.
(3) Dtape(log n) = Ndtape(log n) iff $\exists$ a language L_0 such that L_0 is log-complete in Ndtape(log n) and $L_0 \in$ Dtape(log n).
(4) Let $k \geq 1$. $P \subseteq \text{Dtape}([\log n]^k)$ iff $\exists$ a language L_0 such that L_0 is log-complete in P and $L_0 \in \text{Dtape}([\log n]^k)$. ∎

<u>Def. 1.7</u>: A language $L \subseteq \Sigma^*$ is said to be <u>bounded</u> iff $\exists$ strings $w_1, \ldots, w_k \in \Sigma^*$ such that $L \subseteq w_1^* \cdot w_2^* \cdot \ldots \cdot w_k^*$. A language that is not bounded is said to be <u>unbounded</u>. ∎

<u>Prop. 1.8 [18]</u>: A regular set R over $\{0,1\}^*$ is unbounded iff $\exists$ strings r, s, x, and $y \in \{0,1\}^*$ such that $r \cdot (0x + 1y)^* \cdot s \subseteq R$. ∎†

<u>Def. 1.9</u>: Let $A, B \subseteq \Sigma^*$. $A \backslash B = \{y \mid \exists x \in A \text{ and } x \cdot y \in B\}$. $A/B = \{x \mid \exists y \in B \text{ and } x \cdot y \in A\}$. $A \backslash B$ (A/B) is called the <u>left quotient</u> of B with respect to A (the <u>right quotient</u> of A with respect to B). ∎

2. Grammar Problems

Grammar analogues of the general complexity results for formal languages and automata in [1] and [2] are presented. Most known undecidability and subrecursive results about grammar problems follow from our general theorems. The reader should note that there is a difference between problems about the cfl's such as - "for arbitrary cfg G is L(G) regular?", which is undecidable, and problems about cfg's such as - "for arbitrary cfg G is G a regular grammar?", which is decidable deterministically in linear time.

2.1 Grammar Complexity Metatheory

First we state and prove a powerful complexity theorem for context-free language problems.

<u>Thm. 2.1</u>: Let $\mathcal{S}$ be any subset of the finitely inherently ambiguous cfl's over {0,1} such that $\exists L_t \in \mathcal{S}$, where L_t has an unbounded regular subset. Then for arbitrary cfg G, the predicate "$L(G) \in \mathcal{S}$" is undecidable. ∎

<u>Proof</u>: As the reader can verify, the finitely inherently ambiguous cfl's over {0,1} are closed under quotient with single strings on both the left and the right, and under all inverse homomorphisms h^{-1}, where h is defined by $h(0) = 0x$ and $h(1) = 1y$ for some $x, y \in \{0,1\}^*$.

By assumption $\exists L_t \in \mathcal{S}$ such that L_t has an unbounded regular subset. From Prop. 1.8 this implies that $\exists$ strings $r, s, x, y \in \{0,1\}^*$ such that $r \cdot (0x+1y)^* \cdot s \subseteq L_t$. Let $h_1, h_2 : \{0,1\}^* \to \{0,1\}^*$ be the 1-1 homomorphisms defined by $h_1(0) = 0x$, $h_1(1) = 1y$, $h_2(0) = 0x0x$, and $h_2(1) = 0x1y$. Let L_f be some infinitely inherently ambiguous cfl over {0,1}. For all cfg's G a cfg H can be constructed effectively such that -

$$L(H) = L_t \cap \sim r \cdot (0x0x+0x1y)^* \cdot 1y0x \cdot (0x+1y)^* \cdot s +$$
$$r \cdot [h_2(L(G)) \cdot 1y0x \cdot (0x+1y)^* + (0x0x+0x1y)^* \cdot 1y0x \cdot h_1(L_f)] \cdot s.$$

But $L(H) \in \mathcal{S}$ iff $L(G) = \{0,1\}^*$. Thus the existence of a decision procedure for "$L(G) \in \mathcal{S}$" implies the existence of a decision procedure for "$L(G) = \{0,1\}^*$", a predicate well-known to be undecidable.

There are two cases to consider.

<u>Case 1</u>: Let $L(G) = \{0,1\}^*$. Then

$$L(H) = L_t \cap \sim r \cdot (0x0x+0x1y)^* \cdot 1y0x \cdot (0x+1y)^* \cdot s + r \cdot (0x0x+0x1y)^* \cdot 1y0x \cdot (0x+1y)^* s + \ldots = L_t.$$

Thus by assumption $L(H) \in \mathcal{S}$.

<u>Case 2</u>: Let $L(G) \subsetneq \{0,1\}^*$. Then $h_2(L(G)) \subsetneq (0x0x+0x1y)^*$. Let $w \in (0x0x+0x1y)^* - h_2(L(G))$. Let $z = h_2^{-1}(w)$. Since h_2 is 1 - 1, $z \notin L(G)$. Thus, $r \cdot h_2(z) \cdot 1y0x \backslash [L(H)/s] = h_1(L_f) + \mathcal{L}$, where $\mathcal{L} = \{\alpha \mid r \cdot h_2(z) \cdot 1y0x \cdot \alpha \cdot s \in L_t \cap \sim r \cdot [(0x0x+0x1y)^* \cdot 0x1y \cdot (0x+1y)^*] \cdot s\}$. Since h_1 is also 1 - 1, $h_1^{-1}(h_1(L_f)+\mathcal{L}) = L_f + h_1^{-1}(\mathcal{L})$. But $\beta \in h_1^{-1}(\mathcal{L})$ implies that $\alpha = h_1(\beta) \in (0x+1y)^*$. This implies that -

$$r \cdot h_2(z) \cdot 1y0x \cdot \alpha \cdot s \in r \cdot [(0x0x+0x1y)^* \cdot 1y0x \cdot (0x+1y)^*] \cdot s,$$

a contradiction. Thus $L_1^{-1}(\mathcal{L}) = \phi$; and $h_1^{-1}(r \cdot h_2(z) \cdot 1y0x \backslash [L(H)/s]) = L_f$. But noting the closure properties of the finitely inherently ambiguous cfl's mentioned above, this implies that L(H) is infinitely inherently ambiguous. Thus $L(H) \notin \mathcal{S}$. ∎

Thm. 2.1 shows that one simple idea underlies the undecidability of most of the classes of cfl's studied in the literature. The following corollary of 2.1 illustrates its power and applicability.

---

†We sometimes use "+" to denote union. Thus $A+B \equiv A \cup B$.

Thm. 2.2: The following classes of cfl's satisfy the conditions of Theorem 2.1:

1. regular sets;
2. simple precedence languages;
3. operator precedence languages;
4. s-languages;
5. for all $k \geq 1$ the LL(k) languages;
6. LL languages;
7. real-time strict deterministic languages;
8. strict deterministic languages;
9. for all $k \geq 0$ the ELC(k) languages;
10. ELC languages;
11. LR(0) languages;
12. deterministic cfl's;
13. LR Regular languages;
14. RPP languages;
15. LR(1,∞) languages;
16. BCP languages;
17. FPFAP languages;
18. full SPM parsable languages;
19. unambiguous cfl's;
20. for all $k \geq 2$ the inherently ambiguous cfl's of degree equal to (less than or equal to) k; and
21. finitely inherently ambiguous cfl's.

Thus letting Γ denote any of the above classes of the cfl's, the predicate "$L(G)\varepsilon\Gamma$" is undecidable for arbitrary cfg G . ∎†

Thm. 2.2 follows immediately from Thm. 2.1 and known properties of language classes 1-20. Definitions of these classes may be found in [7] (1-6,11,12); [19] (7,8); [20] (9,10); [21] (13); [6] (14-17); and [22] (18).

Let M be any deterministic Tm that always halts on the right end of its tape. Then M is $O(T(n))$ time-bounded for some strictly increasing recursive function $T(n)$. Given an input string x to M , two cfg's $G_1[M,x]$ and $G_2[M,x]$ can be constructed effectively in linear time on a multi-tape Tm such that -

$\mathcal{L}_1 = L(G_1[M,x]) = \#\cdot\{y\cdot\#\cdot z^r\cdot\#|y,\ z$ are i.d.'s of M and $y \underset{M}{\vdash} z\}^*\cdot\$$ and

$\mathcal{L}_2 = L(G_2[M,x]) = \#\cdot x_1\cdot\{\#\cdot y^r\cdot\#\cdot y|y$ is an i.d. of $M\}^*\cdot\{\#\cdot z^r\cdot\#|z$ is an accepting i.d. of $M\}\cdot\$$, where x_1 is the initial i.d. of M or x .

For sufficiently fast increasing functions $T(n)$, no pair of words in $\mathcal{L}_1$ and $\mathcal{L}_2$ has a common prefix of length $\geq c\cdot[T(|x|)]^2$ for some positive integer c depending only on M <u>not</u> x . Moreover, $G_1[M,x]$ and $G_2[M,x]$ are strong LL and SLR grammars.

Prop. 2.3: Let M, $T(n)$, c, and x be as described above. Let $k = c\cdot[T(|x|)]^2$. Then in time $\leq c_1\cdot|x|$ cfg's G_1, G_2 and G_3 each of size $\leq c_2 + |x|$, where c_1 and c_2 are constants depending only upon M not x , can be constructed such that the following are equivalent:

(1) M accepts x ;

(2) $L(G_1)\cap L(G_2) \neq \phi$;

(3) G_3 is inherently ambiguous;

(4) G_3 is ambiguous;

(5) G_3 is not strong LL(k) ;

(6) G_3 is not SLR(k) ; and

(7) G_3 is not LALR(k) . ∎

Proof: G_1 and G_2 are equal to $G_1[M,x]$ and $G_2[M,x]$, respectively. G_3 is the cfg whose productions consist of -

(a) all productions of G_1 and G_2 ;

(b) $S \to AS_1\$S_3|B¢S_2\$S_4$, where S_1 and S_2 are the start symbols of G_1 and G_2 , respectively;

(c) $A \to ¢¢$;

(d) $B \to ¢$;

(e) $S_3 \to aTbcU$;

(f) $T \to aTb|ab$;

(g) $U \to cU|c$;

(h) $S_4 \to aVbWc$;

(i) $V \to aV|a$; and

(j) $W \to bWc|bc$.

We assume that $\{S,A,S_3,B,S_4,T,U,V,W\}$ is disjoint from the union of the nonterminal alphabets of G_1 and G_2 .

If M accepts x then ∃ a string $w\varepsilon L(G_1)\cap L(G_2)$. Thus $L(G_3)\cap¢¢\cdot w\cdot\$\cdot\Sigma^* = ¢¢\cdot w\cdot\$\cdot[\{a^nb^nc^m|n,m \geq 2\}\cup\{a^nb^mc^m|n,m \geq 2\}]$. Since the unambiguous cfl's are closed under intersection with regular sets, this shows that G_3 is inherently ambiguous. Hence a fortiori G_3 is ambiguous and is <u>not</u> strong LL(ℓ) , SLR(ℓ) , or LALR(ℓ) for any choice of ℓ . If M doesnot accept x , then $L(G_1)\cap L(G_2) = \phi$ and no pair of words in $L(G_1)$ and $L(G_2)$ have a common prefix of size k . By inspection of the productions of G_3 , this implies that G_3 is strong LL(k), SLR(k), and LALR(k) . ∎

In what follows $\mathcal{C}$ denotes the intersection of the strong LL and SLR grammars.† Our first major result follows from Prop. 2.3.

---

†Weaker versions of Thm.'s 2.1 and 2.2 appear in [2].

†A more complex construction in Prop. 2.3 allows (i) of 2.4 to be replaced by the intersection of the BRC, strong LL , and SLR grammars $\subseteq \Gamma$.

Thm. 2.4: Let Γ be any class of cfg's such that

(i) $\mathcal{C} \subseteq \Gamma$ and
(ii) $\Gamma \subseteq$ the class of cfg's that are not inherently ambiguous.

Then for arbitrary cfg G , the predicates Π_1 = "$G\in\Gamma$" and Π_2 = "$L(G)\in L(\Gamma) = \{L|L = L(g)$ with $g\in\Gamma\}$" are undecidable. ∎

Proof: Suppose Π_1 is decidable. Then there exists a strictly increasing recursive function f that bounds the time required to decide Π_1 . Let M be any O(T(n)) time-bounded Tm with $T(n) \geq n$ strictly increasing. From 2.3 L(M) is recognizable by some $c_1\cdot n + f(c_2+n)$ time-bounded Tm $\mathcal{M}$, where c_1 and c_2 are constants depending only upon M not x and n = |x| .

$\mathcal{M}$ operates as follows.

1. Given input x , $\mathcal{M}$ constructs G_3 of Prop. 2.3.
2. $\mathcal{M}$ tests if $\Pi_1(G_3)$ is true. If so then $x\notin L(M)$. If not then $x\in L(M)$.

Clearly, step 1 requires at most $c_1\cdot n$ time; and step 2 requires at most $f(c_2+n)$ time. By Prop. 2.3 if $x\notin L(M)$, then $G_3\in\mathcal{C}\subseteq\Gamma$; and if $x\in L(M)$, then G_3 is inherently ambiguous and, hence, $G_3\notin\Gamma$.

Finally for all positive integers a,b, the recursive function $F(n) = n^2 + f(2n)$ is strictly greater than $a\cdot n + f(b+n)$, a.e. Thus if Π_1 is decidable, then every recursive set is accepted by some F(n) time-bounded Tm . But it is well-known that for every recursive function r(n) , there exists a recursive set R that is not recognizable within time r(n) a.e. on any Tm [23]. ∎

Thm. 2.4 shows that one simple idea and construction also underlies the undecidability of most of the classes of cfg's studied in the literature. The following corollary illustrates Thm. 2.4's power and applicability.

Thm. 2.5: The following classes of cfg's satisfy the conditions of Theorem 2.4:

1. strong LL ;
2. LL ;
3. strong LC ;
4. LC ;
5. ELC ;
6. k-Transformable for some k ;
7. SLR ;
8. LALR ;
9. LR ;
10. Floyd-Evans parsable;
11. LR Regular;
12. FPFAP ;
13. $SLR(k,\infty)$, $LR(k,\infty)$ for some k ;
14. RPP ;
15. basic SPM parsable ;
16. full SPM parsable ;
17. unambiguous cfg's ; and
18. the class of cfg's that are not inherently ambiguous.

Thus letting Γ denote any of the above cfg classes, the predicates "$G\in\Gamma$" and "$L(G)\in\{L|L = L(g)$ and $g\in\Gamma\}$" are undecidable for arbitrary† cfg G . ∎

Definitions of these grammar classes may be found in [7] (1-4, 7-10, 17, 18); [6] (12-14); [20] (5); [21] (11); and [22] (15,16).

Subrecursive analogues of Thm. 2.4 also hold. Let $\mathcal{C}(k)$ = the intersection of the strong LL(k) , SLR(k) , and BC(k,k) grammars.

Thm. 2.6: Let $\Gamma = \bigcup_{k=1}^{\infty} \Gamma_k$ be any class of parameterized cfg's such that for all $k \geq 1$, $\mathcal{C}(k) \subseteq \Gamma_k \subseteq$ the class of cfg's that are not inherently ambiguous. Then

(i) $\mathcal{L}_1 = \{(G,\nu)|G$ is a cfg , ν is a unary numeral for the positive integer n , and $G\notin\Gamma_n\} \geq_{\log}$ NP and
(ii) $\mathcal{L}_2 = \{(G,\nu)|G$ is a cfg , ν is a binary numeral for the positive integer n , and $G\notin\Gamma_n\} \geq_{\log\ n(\text{time+space})}$ NDEXP . ∎

The proof of 2.6 is closely related to proofs in [5] and the proof of 2.4 and will not be presented here.

Cor. 2.7: For all classes of cfg's Γ satisfying the conditions of Thm. 2.6,

(i) $\mathcal{L}_1$ is NP-hard; and
(ii) $\exists$a constant $c > 0$ such that any nondeterministic Tm that accepts $\mathcal{L}_2$ requires time $> 2^{cn}$, i.o. ∎

Thm. 2.8: The following classes of cfg's satisfy the conditions of Thm. 2.6.

1. BC ;
2. BRC ;
3. strong LL ;
4. LL ;
5. strong LC ;
6. LC ;
7. ELC ;
8. k-Transformable for some k ;
9. SLR ;
10. LALR ; and
11. LR . ∎

Noting results in [5] the uniform lower time bounds of Thm. 2.8 are fairly tight.

Prop. 2.9: For each of the classes 1-7, 9, and 11 of Thm. 2.8,

---

†The undecidability of classes 15 and 16 was not previously known.

(i) L_1 is in NP ; and

(ii) $\exists$ a constant $d > 0$ such that L_2 is recognizable by some nondeterministic $O(2^{dn})$ time-bounded Tm . ∎

2.2 Relative Decision Problems

Next we consider relative decision problems. For example, if a grammar G is LR(2) , is it decidable if G is an LL grammar? If so how much time is required? Our first results are described in the following table

Scource Class	TARGET CLASS LR	LALR	SLR	LL	strongLL
LR(k)	T	T $k=0$ U $k\geq 1$	T $k=0$ U $k\geq 1$	D[32]	D $k\leq 1$ U $k\geq 2$
LALR(k)	T	T	T $k=0$ U $k\geq 1$	D	D $k\leq 1$ U $k\geq 2$
SLR(k)	T	T	T	D	D $k\leq 1$ U $k\geq 2$
LL(k)	T	T $k\leq 1$ U $k\geq 2$	T $k\leq 1$ U $k\geq 2$	T	T $k\leq 1$ U $k\geq 2$
strongLL(k)	T	T $k\leq 1$ U $k\geq 2$	T $k\leq 1$ U $k\geq 2$	T	T

Here, T denotes trivial; D denotes decidable; and U denotes undecidable.

Typical results include -

(i) For arbitrary LALR(k_0) grammar G with $k_0 \geq 2$, it is undecidable if G is strong LL(k) for some k .

(ii) In the decidable cases a maximal possible k exists. Moreover, the magnitude of $k - k_0$ depends only upon the size of G , the grammar in question. Thus, an LR(k_0) grammar is an LL grammar iff it is $LL(|G|^{|G|+2} + k_0)$.

Bounds for the other pairs of classes will appear in [31].

A noncanonical parsing analogue ([6] or [7] pp. 485-487) of Thm. 2.4 also holds.

Thm. 2.10: Let Γ be any class of cfg's such that the SLR(1,∞) grammars [6] $\subseteq \Gamma \subseteq$ the unambiguous cfg's . Then for arbitrary cfg G , the predicates "$G\in\Gamma$" and "$L(G)\in L(\Gamma)$" are undecidable. ∎

Other relativizations will appear in [31].

3. Lower Bounds for Problems in P

We consider the problem of proving nontrivial lower time and/or space complexity bounds, especially for problems in P . Several partial results are obtained. The "efficient" reducibilities defined in Section 1 are shown to yield insights into the relative complexities of problems in P .

3.1 Lower Bounds for Grammar Problems

Recent results [24] have shown that the strong LL(k) , LL(k) , SLR(k) , and LR(k) properties can all be tested deterministically in polynomial time if k is fixed in advance. In fact the bounds in [24] have been improved to $O(n^{k+2})$ operations in each of these cases [5]. Our intuition suggests that many if not most of the $O(n^{k+1})$ LR(k) items must be considered to decide the LR(k) property, and thus strongly suggests that the amount of time required for LR(k) testing grows exponentially in k . However using results in [9] and [5], we show that such exponential dependence upon k implies that P $\neq$ NP . This suggests that the relative time complexities of strong LL(k) , LL(k) , SLR(k) , and LR(k) testing for fixed k and the relationship between the time complexity for LR(k) and LR(k+1) testing merit investigation.

Thm. 3.1 [9]: P = NP iff there exists a recursive translation σ and a positive integer k , such that for every nondeterministic Tm M_i , which uses time $T_i(n) \geq n$, $M_{\sigma(i)}$ is an equivalent deterministic Tm working in time $O[T_i(n)]^k$. ∎

Thm. 3.2 [5]: Let $C(k)$ represent one of the following grammar classes: the LL(k), LC(k), LR(k), strong LL(k), strong LC(k), or SLR(k) grammars. Then there exists a nondeterministic Tm M and constants a,b,c such that

(a) $L(M) = \{(G,k) | G \text{ is not } C(k)\}$ and

(b) M performs at most $a\cdot|G|^b\cdot k^c$ moves on any input (G,k) . ∎

Combining these theorems we have the following -

Thm. 3.3: If for all integers $\ell \geq 1$ there exists an integer $k \geq 1$ such that LL(k) , LC(k) , LR(k) , strong LL(k) , strong LC(k) , or SLR(k) testing requires time $\geq O(n^{\ell})$ i.o., then P $\neq$ NP . ∎

Proof: From 3.1 P = NP implies that $\forall_\ell \exists$ a constant $d > 0$ such that $L\in$ Nd time(n^ℓ) implies that $L\in$ D time($n^{d\ell}$) for all integers $\ell > 0$. Thus P = NP , 3.1, and 3.2 imply for all $k \geq 1$, that LL(k) , LC(k) , ... , and SLR(k) testing require time at most $O([a\cdot|G|^b k^c]^d) \leq O(k^{cd}|G|^{bd})$ on some deterministic Tm , where b,c,d are constants independent of k . ∎

Thus a proof that the time required for LR(k) testing grows exponentially in k would represent a major breakthrough in theoretical computer science.

Using the "efficient" reducibilities introduced in Section 1 we can, however, discuss the relative complexities of LL(k) and LR(k) testing.

Thm. 3.4: Let Γ be any class of cfg's for which

(i) $\exists k_0 \geq 1$ such that the $LL(k_0)$ grammars $\subseteq \Gamma \subseteq LR(k_0)$ grammars. Then $\{G|G$ is an $LL(k_0)$ grammar$\} \underset{\substack{\log \\ n\log n(\text{space+time})}}{\leq} \{G|G\varepsilon\Gamma\}$.

(ii) $\exists k_0 \geq 1$ such that the strong $LL(k_0)$ grammars $\cap$ $SLR(k_0)$ grammars $\subseteq \Gamma \subseteq$ the strong $LL(k_0)$ grammars $\cup$ $SLR(k_0)$ grammars. Then $\{G|G$ is a strong $LL(k_0)$ grammar$\} \underset{\substack{\log \\ n\log n(\text{space+time})}}{\leq} \{G|G\varepsilon\Gamma\}$. ∎

Proof of (i): Brosgol [20] has shown that for each cfg G , a cfg G' can be constructed "efficiently" such that G' is $LR(k_0)$ iff G is $LL(k_0)$. Moreover, one can easily verify that G' is also $LL(k_0)$ if it is $LR(k_0)$. Thus $G'\varepsilon\Gamma$ iff G is $LL(k_0)$; and the construction of G' from G requires at most O(nlogn) space and time on a log-space transducer. ∎

Informally every class Γ of cfg's satisfying the conditions of (i) or (ii) of 3.4 is as hard to test for as $LL(k_0)$ or strong $LL(k_0)$ testing, respectively. Grammar classes satisfying (i) include the $LC(k_0)$, $ELC(k_0)$, k_0-transformable, and $LR(k_0)$ grammars.

Thm. 3.5: For all integers $k_0 \geq 1$,

(i) $\{G|G$ is a strong $LL(k_0)$ grammar$\} \underset{\substack{\log \\ n\log n(\text{space+time})}}{\leq} \{G|G$ is a strong $LL(k_0+1)$ grammar$\}$;

(ii) $\{G|G$ is an $LL(k_0)$ grammar$\} \underset{\substack{\log \\ n\log n(\text{space+time})}}{\leq} \{G|G$ is an $LL(k_0+1)$ grammar$\}$;

(iii) $\{G|G$ is an $SLR(k_0)$ grammar$\} \underset{\substack{\log \\ n\log n(\text{space+time})}}{\leq} \{G|G$ is an $SLR(k_0+1)$ grammar$\}$; and

(iv) $\{G|G$ is an $LR(k_0)$ grammar$\} \underset{\substack{\log \\ n\log n(\text{space+time})}}{\leq} \{G|G$ is an $LR(k_0+1)$ grammar$\}$. ∎

Thus, informally, increasing k does not decrease the complexity of LR(k) testing.

3.2 Multi-head Finite, Pushdown, and Stack Automata

One simple idea that underlies and unifies much of the recent work on the relationship of time and space complexity classes is presented. This idea unifies and extends many of the results in [2], [8], [9], [10], [11], [12], [13], [14], [15], etc. Many new hardest time and/or space languages for Ndtape (logn), P , the 2-way PDA languages, etc. are presented. We also present strong evidence for the nonlinearity in time of every nontrivial predicate on the cfl's , when applied to the PDA .

Thm. 3.6: Let P be any nontrivial predicate

(1) on the regular sets over $\{0,1\}$ such that $P(\phi)$ is false, then $\{M|M$ is an NDFA (regular grammar) with λ-moves (λ-productions) and $P(L(M))$ is true$\} \underset{\log}{\geq}$ Ndtape (logn);

(2) on the deterministic cfl's over $\{0,1\}$ such that $P(\phi)$ is false, then $\{M|M$ is a deterministic PDA and $P(L(M))$ is true$\} \underset{\log}{\geq} P$;

(3) on the strict deterministic languages over $\{0,1\}$ such that $P(\phi)$ is false, then $\{G|G$ is a strict deterministic grammar and $P(L(G))$ is true$\} \underset{\log}{\geq} P$;

(4) on the cfl's over $\{0,1\}$ such that $P(\phi)$ is false, then $\{M|M$ is a PDA or cfg and $P(L(M))$ is true$\} \underset{\log}{\geq} P$ and $\{M|M$ is a PDA and $P(L(M))$ is true$\} \underset{\substack{\log \\ n\log n(\text{space+time})}}{\geq}$ 2-way PDA languages;

(5) on the 1-way DSA languages over $\{0,1\}$, then $\exists\, c > 0$ such that $\{M|M$ is a 1-way DSA and $P(L(M))$ is true$\}$ requires at least $O(2^{cn})$ time i.o. on any deterministic Tm ;

(6) on the 1-way SA languages over $\{0,1\}$, then $\exists\, c > 0$ such that $\{M|M$ is a 1-way SA and $P(L(M))$ is true$\}$ requires at least $O(2^{cn^2/[\log n]^2})$ time i.o. on any deterministic Tm ;

(7) on the indexed languages over $\{0,1\}$, then $\exists\, r > 0$ such that $\{G|G$ is an indexed or OI-macro grammar and $P(L(G))$ is true$\}$ requires at least $O(2^{n^r})$ time i.o. on any deterministic Tm ; and

(8) on the recursively enumerable sets over $\{0,1\}$ such that $P(\phi)$ is true, then $\{M|M$ is a Tm or type 0 grammar and $P(L(M))$ is true$\}$ is not recursively enumerable. ∎

Proof sketch: Detailed proofs can be found in [25].

(1) It is well-known that the class of languages accepted by 2-way NDFA equals Ndtape(logn) . Let P be any nontrivial predicate on the regular sets such that $P(\phi)$ is false. Since P is nontrivial there exists an NDFA M_0 such that $P(L(M_0))$ is true. Let $L_0 = L(M_0)$. Clearly $L_0 \neq \phi$.

Let M_i be an arbitrary 2-way k-head NDFA with $k \geq 1$. For all $x \in \{0,1\}^*$, an NDFA $M_{i,x}$ with λ-moves can be constructed such that

$$L(M_{i,x}) = \begin{cases} \phi & \text{if } x \notin L(M_i) , \\ L_0 & \text{if } x \in L(M_i) , \end{cases}$$

For each input $x = x_1 \ldots x_n$ to M_i, $M_{i,x}$ is constructed as follows:

(a) All input tape configurations of M_i on x are embedded in $M_{i,x}$'s finite state control.

(b) $|M_{i,x}| \leq c_i \cdot |x|^k \cdot \log(|x|)$, where c_i depends only upon M_i and M_0 not on x.

(c) $M_{i,x}$ simulates M_i on x. If M_i accepts x, then $M_{i,x}$ simulates M_0 on its $(M_{i,x})$'s input. If M_i does not accept x, then $L(M_{i,x}) = \phi$.

(d) For fixed i, the construction of $M_{i,x}$ from M_i and x can be accomplished on a deterministic $\log|x|$ space-bounded transducer.

$M_{i,x}$'s simulation of M_i on input x only involves λ-moves. $M_{i,x}$'s state set includes states of the form $(p, \nu_1, \ldots, \nu_k)$, where p denotes a state of M_i and $\nu_1, \ldots, \nu_k$ are binary numerals for positive integers $n_1, \ldots, n_k$ respectively, with $n_1, \ldots, n_k \leq |x| = n$. State $(p, \nu_1, \ldots, \nu_k)$ signifies that M_i is in state p and that its first input tape head is scanning n_1<u>st</u> character of x, its second input tape head is scanning the n_2<u>st</u> character of x, etc. The construction of $M_{i,x}$ from M_i and x can be accomplished within $O(|x|^k \cdot \log|x|)$ time and with $O(\log n)$ intermediate storage.

But $P(L(M_{i,x}))$ is true iff $x \in L(M_i)$. This follows since $x \notin L(M_i)$ implies that $L(M_{i,x}) = \phi$ and $P(L(M_{i,x}))$ is false by assumption. Otherwise $L(M_{i,x}) = L_0$ and $P(L(M_{i,x}))$ is true by assumption.

(2) Cook [26] has shown that the class of anguages accepted by 2-way multi-head deterministic PDA equals P. The proof is analogous to that of (1) of this theorem and is left to the reader.

(3) The proof of (3) is essentially the same as that of (2) noting the following fact about strict deterministic grammars - for every dpda M with a single final state, the canonical grammar† G_M of M is a strict deterministic grammar [19].

†See [19] for the definition of canonical grammar.

(4) Cook [26] has also shown that the class of languages accepted by 2-way multi-head PDA equals P. The theorem holds for the cfg's as well as the PDA since there exists a deterministic log space transducer M such that M, when given a PDA as input, outputs an equivalent cfg G.

(5) The class of languages accepted by 1-way DSA equals the class of languages accepted by $O(2^{cn\log n})$ time-bounded deterministic Tm's [26]. This, together with known time hierarchy results and a construction like that used in the proof of (1), implies (4).

(6) The class of languages accepted by 1-way SA equals the class of languages accepted by $O(2^{cn^2})$ time-bounded deterministic Tm's [26]. This, together with known time hierarchy results, and a construction like that used in the proof of (1), implies (5).

(7) The algorithms in [27], [28] for converting an arbitrary 1-way nested SA into an equivalent indexed grammar, for converting an arbitrary indexed grammar into an equivalent OI-macro grammar, respectively, can be seen to be executable deterministically in polynomial time.

(8) Let M_i be any arbitrary Tm. For all $x \in \{0,1\}^*$, a Tm $M_{i,x}$ can be constructed effectively such that $L(M_{i,x}) = \begin{cases} \phi & \text{if } x \notin L(M_i), \\ L_0 & \text{otherwise.} \end{cases}$

Here L_0 is some nonempty reset for which $P(L_0)$ is false. Thus $\{M | M$ is a Tm and M diverges on empty input$\}$ is effectively reducible to $\{M | M$ is a Tm and $P(L(M))$ is true$\}$. ∎

Theorem 3.6 shows that every nontrivial predicate on the 1-way 1-head NDFA, deterministic PDA, PDA, DSA, and SA requires as much time and/or space as any language recognizable by a 2-way 1-head NDFA, deterministic PDA, PDA, DSA, and SA, respectively. We present a partial converse.

<u>Thm. 3.7</u>:

(1) $\mathcal{L}_1$ = M M is an NDFA with λ-moves and $L(M) \neq \phi\}$ is the accepted language of some 2-way 2-head NDFA.

(2) $\mathcal{L}_2 = \{M | M$ is a PDA and $L(M) \neq \phi\}$ is the accepted language of some 2-way 1-head PDA.

(3) $\mathcal{L}_3 = \{M | M$ is a 1-way SA and $L(M) \neq \phi\}$ is the accepted language of some 2-way 2-head SA.

(4) $\mathcal{L}_4 = \{M | M$ is a 1-way deterministic PDA and $\lambda \in L(M)\}$ is the accepted language of some 2-way 1-head PDA.

(5) $\mathcal{L}_5 = \{M | M$ is a 1-way DSA and $\lambda \in L(M)\}$ is the accepted language of some 2-way 2-head DSA. ∎

For a proof see [25].

Theorems 3.6 and 3.7 have many corollaries. Here we mention a few of them.

Cor. 3.8 [11]: There exists a language L accepted by some 2-way 2-head NDFA such that L is accepted by some 2-way multi-head DFA iff Dtape(logn) = Ndtape(logn) . ∎

L_1 is one such language; in fact, L_1 is log-complete in Ndtape(logn) . Since the emptiness problem for NDFA is nothing more than the reachability problem for directed graphs, another immediate corollary is -

Cor. 3.9: (i) GAP = {G|G is a directed graph on {1,...,n} for some n , which has a path from vertex 1 to vertex n} is accepted by some 2-way 2-head NDFA .†

(ii) [10] GAP is log-complete in Ndtape(logn). ∎

Cor. 3.10: The language L_2 is log-complete in P . ∎

Noting Thm.'s 3.6 and 3.7, L_2 is a time and space hardest 2-way PDA language. In fact $L_2 \epsilon \mathrm{Dtime}(n^r)$ implies that the 2-way PDA languages $\subseteq \mathrm{Dtime}(n^r[\log n]^r)$ for all $r \geq 1$. This strongly suggests that L_2 requires non-linear time!

Cor. 3.11 [12]: The languages L_6 = {G|G is a cfg and L(G) ≠ ϕ} and L_7 = {G|G is a cfg and L(G) is finite} are log-complete in P . ∎

Cor. 3.12: The language L_4 is log-complete in P . ∎

Cor. 3.12 should be compared with the theorem due to Lewis, Stearns, and Hartmanis [29] that every cfl ε $\mathrm{Dtape}([\log n]^2)$.

Cor. 3.13: $\exists c_1, c_2 > 0$ such that the recognition of L_3 requires time $> 2^{c_1 n^2/(\log n)^2}$, i.o. on any deterministic Tm . Moreover, L_3 is recognizable by some $2^{2^{c_2 n^4}}$ deterministic time-bounded Tm . ∎

Cor. 3.14: $\exists c_1, c_2 > 0$ such that the recognition of L_5 requires time $2^{c_1 n}$ i.o. on any deterministic Tm . Moreover, L_5 is recognizable by some $2^{2^{c_2 n^2 \log n}}$ deterministic time-bounded Tm . ∎

Cor. 3.15: $\exists r_1, r_2 > 0$ such that the recognition of L = {G|G is an indexed [OI-macro] grammar and L(G) ≠ ϕ} requires time $> 2^{n^{r_1}}$ i.o. on any deterministic Tm . Moreover, L is recognizable by some $2^{n^{r_2}}$ deterministic time-bounded Tm . ∎

The upper bounds in Cor.'s 3.13 and 3.14 follow from Thm. 3.6 and results in [30].

Moreover, the emptiness problems for the 1-way DFA , deterministic PDA , and DSA (each without λ-moves) have the same lower complexity bounds as the emptiness problems for the corresponding 1-way nondeterministic automata with λ-moves.

Thm. 3.16: (i) L_1' = {M|M is a DFA and L(M) ≠ ϕ} is log-complete in Ndtape(logn). Moreover, L_1' is recognizable by some 2-way 2-head NDFA.†

ii) L_2' = {M|M is a deterministic PDA with no λ-moves and L(M) ≠ ϕ} is log-complete in P, is a 2-way PDA language, and $\underset{\substack{\log \\ n\log n(\text{space+time})}}{\geq}$ 2-way PDA languages.

iii) $\exists c_1, c_2 > 0$ such that the recognition of L_3' = {M|M is a 1-way DSA with no λ-moves and L(M) ≠ ϕ} requires time $> 2^{c_1 n^2/(\log n)^2}$ i.o. on any deterministic multi-tape Tm. Time $2^{2^{c_2 n^4}}$ suffices. ∎

A proof of 3.16 can be found in [25].

Finally to further illustrate the implications of the results in this section, we present a new and easily understood $O(n^3 \cdot \mathrm{Polynomial}(\log n))$ time-bounded RAM algorithm for arbitrary 2-way PDA language recognition.

Algorithm 3.17: Let L be a fixed 2-way PDA language. Let M be a fixed 2-way PDA such that L(M) = L . Let $x = x_1 \ldots x_n$ be an input to M . To test if xεL(M) = L , the following steps suffice:

(1) Construct a 1-way PDA M_x , as described in the proof of 3.6 such that $L(M_x) \neq \phi$ iff xεL .
(2) Convert M_x into an equivalent context-free grammar G_x .
(3) Test G_x for emptiness.
(4) If $L(G_x) \neq \phi$, then xεL . Otherwise x∉L . ∎

The time required to execute step 1 is O(nlogn) . The time required to convert M_x into an equivalent CFGG_x is $O(n^3(\log n)^3)$. Finally, the time to test G_x for emptiness is

†Graphs are presented by adjacency lists with vertices denoted by binary numerals.

†Jones [10] shows that L_1' is log-complete in Ndtape(logn).

well-known to be $O(n \log n)$ on a logarithmic cost RAM.

3.3 Trees, Structural Equivalence, and Grammatical Covering

The results of the preceeding sections hold for grammars and automata on trees as well as strings. All definitions can be found in [33]-[36]. Our first result extends results in [34].

Thm. 3.18: The following are p-equivalent:
(1) structural equivalence of cfg's;
(2) structural containment of cfg's;
(3) equivalence of nondeterministic top-down tree automata;
(4) containment of nondeterministic top-down tree automata;
(5) equivalence of nondeterministic bottom-up tree automata;
(6) containment of nondeterministic bottom-up tree automata;
(7) equivalence of parenthesis grammars; and
(8) containment of parenthesis grammars. ■[†]

Prop. 3.19: There exists a $2^{P(n)}$, where $P(n)$ is a polynomial, time-bounded algorithm on a deterministic Tm for solving (1)-(8) of Thm. 3.18. ■

Prop. 3.20: If any of the problems (1)-(8) of Thm. 3.18 is not an element of PSPACE, then all of these problems are not elements of PSPACE; and for all positive integers k, P is not a subset of Dtape($[\log n]^k$). ■

In [33] we conjectured that structural equivalence for cfg's requires nonpolynomial space. Prop. 3.20 illustrates the difficulty of proving this conjecture.

The yield of a tree t, denoted by $y(t)$, is defined in [34] and [35]. The yield of a set of trees T is defined by $y(T) = \bigcup_{t \in T} y(t)$. A predicate Π on a class of tree languages C is said to be yield-invariant if for all $T, T' \in C$, $y(T) = y(T')$ implies $\Pi(T) = \Pi(T')$. We allow are trees to have leaves labeled with λ, the empty string.

Thm. 3.21: Let Π be any nontrivial yield-invariant predicate
(i) on the recognizable sets over $\{0,1\}$ such that $\Pi(\phi)$ is true, then $\{M|M$ is a nondeterministic bottom-up (or top-down) tree automaton and $\pi(L(M))$ is true$\} \geq_{\log}$ P; and
(ii) on the context-free dendro-languages [35] such that $\Pi(\phi)$ is true, then $\{G|G$ is a context-free dendrogrammar with λ-productions and $\Pi(L(G))$ is true$\}$ require time $> 2^{n^r}$ i.o. on any multi-tape deterministic Tm for some $r > 0$. ■

Thm. 3.22: (i) Grammatical covering for linear cfg's is PSPACE-complete.
(ii) Grammatical covering for arbitrary cfg's is undecidable. ■

A proof of 3.22 can be found in [33] and [36].

Finally, the undecidability results in Section 2 can be reformulated in terms of trees as well.

4. A Uniform Lower Bound on Scheme Equivalence

In [37] the strong and weak equivalence problems for single variable program schemes were shown to be NP-complete. The definitions of strong equivalence $\equiv$, weak equivalence $\approx$, and interpretations of schemes can be found in [38].

Def. 4.1 [38]: A binary relation $\sim$ on schemes is said to be reasonable if for all schemes S_1, S_2, $S_1 \equiv S_2$ implies $S_1 \sim S_2$ and $S_1 \sim S_2$ implies $S_1 \approx S_2$. ■

Thm. 4.2: For all reasonable relations $\sim$ and
(i) for all fixed non-divergent 2-variable, single-variable, or loop-free program schemes $\mathcal{S}$, $\{S|S$ is a 2-variable, single variable, or loop-free program scheme, respectively, and $S \not\sim \mathcal{S}\}$ is NP-hard; and
(ii) for all fixed monadic or linear monadic recursion schemes $\mathcal{S}$, $\{S|S$ is a monadic or linear monadic recursion scheme, respectively, and $S \not\sim \mathcal{S}\}$ is NP-hard. ■

Proof sketch: We efficiently reduce the well-known NP-complete set $\{f|f$ is a D_3-Boolean form and f is not a tautology$\}$ to the predicate "$S \not\sim \mathcal{S}$". Let f be any arbitrary D_3-Boolean form with n literals and m clauses. For each such f a single variable loop-free and function-free program scheme S_f with two halt statements labeled A and B, respectively, can be constructed deterministically in time bounded by a polynomial in $|f|$ such that the statement labeled B is executable under some interpretation iff f is not a tautology. Let g be a function symbol not appearing in $\mathcal{S}$. Without loss of generality, we assume for all D_3-Boolean forms f that the predicate symbols and the labels appearing in S_f and $\mathcal{S}$ are disjoint. Let S_f' be the scheme that results from (a) replacing all occurrences in $\mathcal{S}$ of the label of the initial statement of $\mathcal{S}$ by A; (b) replacing the statement "A:Halt;" in S_f by the initial statement of the scheme that resulted from $\mathcal{S}$ after (a); and (c) replacing the statement "B:Halt" by "B:x ← g(x); Halt." Then $S_f' \sim \mathcal{S}$ iff f is a tautology. ■

Since there are fixed schemes $\mathcal{S}$ and reasonable relations $\sim$ such that $\{S|S$ is a program

[†]Top-down and bottom-up tree automata are called RFA (root-to-fronter automata) and FRA (frontier-to-root automata), respectively in [34].

scheme and $S \neq \Omega\} \in NP$, our uniform lower bounds are tight.

One immediate corollary of Thm. 4.2 deals with the "degrees of translatability" in [39].

Cor. 4.3: Given a single variable program scheme S, determining S's flowchart degree is an NP-complete problem. ∎

5. Conclusion

We have considered the complexity of a variety of problems from parsing, formal languages, and schemes. In each case we have found close relationships between complexity and underlying combinatorial structure. A complexity theory for grammar problems was presented. A uniform lower bound on the complexity of scheme equivalence was also presented.

Bibliography

[1] H. B. Hunt, III, On the time and tape complexity of languages, Ph.D. Thesis, Cornell University, Aug. 1973.

[2] H. B. Hunt, III, and D. J. Rosenkrantz, Computational parallels between the regular and context-free languages, Proc. 6th An. ACM Symp. on Th. of Comp. (May 1974), 64-73.

[3] D. E. Knuth, On the translation of languages from left to right, Inf. and Cont. 8, 6 (1965), 607-639.

[4] W. F. Ogden, unpublished note (Dec. 1971).

[5] H. B. Hunt, III, T. G. Szymanski, and J. D. Ullman, On the complexity of LR(k) testing, Conf. Rec. 2nd ACM Symp. on Principles of Programming Languages (Jan. 1975), 130-136.

[6] T. G. Szymanski, Generalized bottom-up parsing, Ph.D. Thesis, Cornell University, May 1973.

[7] A. V. Aho and J. D. Ullman, The Theory of Parsing, Translation, and Compiling, Vol.'s 1 and 2, Prentice-Hall, Englewood Cliffs, N.J., 1972 and 1973.

[8] W. J. Savitch, Relationships between nondeterministic and deterministic tape complexity, JCSS 4, 2 (1970), 177-192.

[9] J. Hartmanis and H. B. Hunt, III, The lba problem and its importance in the theory of computing, SIAM-AMS Proc., Vol. 7, Amer. Math. Soc., Providence, R.I., 1974.

[10] N. Jones, Preliminary report: reducibility among combinatorial problems in logn space, Proc. 7th An. Princeton Conf. on Information Sciences and Systems (March 1973), 547-551.

[11] I. H. Sudborough, On tape-bounded complexity classes and multi-head finite automata, Proc. 14th An. IEEE Symp. on Switching and Automata Th. (Oct. 1973), 138-144.

[12] N. D. Jones and W. T. Laaser, Complete problems for deterministic polynomial time, Proc. 6th An. ACM Symp. on Th. of Comp. (May 1974), 40-46.

[13] S. A. Cook, An observation on time-storage tradeoff, Proc. 5th An. ACM Symp. on Th. of Comp. (May 1973), 29-33.

[14] S. A. Cook and R. Sethi, Storage requirements for deterministic polynomial time recognizable languages, Proc. 6th An. ACM Symp. on Th. of Comp. (May 1974), 33-39.

[15] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, Time and tape complexity of pushdown automaton languages, Inf. and Cont. 13, 3 (1968), 186-206.

[16] T. G. Szymanski and J. H. Williams, Noncanonical parsing, Proc. 14th An. IEEE Symp. on Switching and Automata Th. (Oct. 1973), 122-129.

[17] W. F. Ogden, A helpful result for proving inherent ambiguity, Math. Systems Th. 2, 3 (1968), 191-194.

[18] J. E. Hopcroft, On the equivalence and containment problems for context-free languages, Math. Systems Th. 3, 2 (1969), 119-124.

[19] M. A. Harrison and I. M. Havel, Strict deterministic grammars, JCSS 7, 3 (1973), 237-277.

[20] B. M. Brosgol, Deterministic translation grammars, Ph.D. thesis, Harvard University, 1974.

[21] K. Culik, II and R. Cohen, LR-regular grammars-an extension of LR(k) grammars, JCSS 7, 1 (1973), 66-96.

[22] C. N. Fischer, Extended abstract - an approach to parallel parsing, unpublished.

[23] J. Hartmanis and J. E. Hopcroft, An overview of the theory of computational complexity, JACM 18, 13 (1971), 444-475.

[24] H. B. Hunt, III, T. G. Szymanski, and J. D. Ullman, Operations on sparse relations and efficient algorithms for grammar problems, Proc. 15th An. IEEE Symp. on Switching and Automata Th. (Oct. 1974), 127-132.

[25] H. B. Hunt, III, On the complexity of finite, pushdown, and stack automata, Mathematics Research Center Technical Summary Report #1504, University of Wisconsin-Madison, Oct. 1974 (also submitted for publication).

[26] S. A. Cook, Characterizations of pushdown machines in terms of time-bounded computers, JACM 18, 1 (1971), 4-18.

[27] A. V. Aho, Nested stack automata, JACM 16, 3 (1969), 383-407.

[28] M. J. Fischer, Grammars with macro-like productions, Conf. Rec. 9th An. IEEE Symp. on Switching and Automata Th. (Oct. 1968), 131-142.

[29] P. M. Lewis, R. E. Stearns, and J. Hartmanis, Memory bounds for recognition of context-free and context-sensitive languages, IEEE Conf. Rec. on Switching Circuit Th. and Logical Design (Oct. 1965), 191-202.

[30] O. H. Ibarra, Characterizations of some tape and time complexity classes of Tm's in terms of multi-head and auxiliary stack automata, JCSS 5 (1971), 88-117.

[31] H. B. Hunt, III, and T. G. Szymanski, manuscript in preparation.

[32] D. J. Rosenkrantz and R. E. Stearns, Properties of deterministic top-down grammars, Inf. and Con. 17, 3 (1970), 226-256.

[33] H. B. Hunt, III, D. J. Rosenkrantz, and T. G. Szymanski, On the equivalence, containment, and covering problems for the regular and context-free languages (submitted for publication).

[34] J. W. Thatcher, Tree automata: an informal survey, in Currents in the Theory of Computing, A. V. Aho (ed.), Prentice-Hall, Englewood Cliffs, N.J., 1973, 143-172.
[35] W. C. Rounds, Mappings and grammars on trees, Math. Systems Th. 4, 3 (1970), 257-287.
[36] H. B. Hunt, III, D. J. Rosenkrantz, and T. G. Szymanski, The covering problem for linear context-free grammars, Computer Sciences Laboratory Technical Report TR165, Dept. of Electrical Engineering, Princeton University.
[37] R. L. Constable, H. B. Hunt, III, and S. Sahni, On the computational complexity of scheme equivalence (submitted for publication.)
[38] D. C. Luckham, D. M. R. Park, and M. S. Paterson, On formalized computer programs, JCSS 4, 3 (1970), 220-249.
[39] A. K. Chandra, Degrees of translatability and canonical forms in program schemas: part I, Proc. 6th An. ACM Symp. on Th. of Comp. (May 1974), 1-12.