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 non-trivial 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 non-trivial 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 \( \Gamma \) such that the \( LL(k_0) \) grammars \( \subseteq \Gamma \subseteq LR(k_0) \) grammars, and for arbitrary context-free grammar \( G \), 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 - all 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 any 2-way pushdown
automaton language. The best known algorithms
for 2-way pushdown automaton recognition require
O(n^3) operations on RAM's ([16] 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 complex-
ity 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 lan-
guage generated (accepted) by a grammar (auto-
maton) G is denoted by L(G).

The following abbreviations are used through-
out 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. csa - 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 every-
where means except for a finite set of
nonnegative integers.

Def. 1.1: A cfg G is said to be ambiguous
if some string xcl(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 inherently
ambiguous if all cfg's generating L(G) are
ambiguous.

Def. 1.2: Let k be a positive integer. A
cfg G is said to be ambiguous of degree k
if every string xcl(G) has at least k distinct
derivation trees and some string xcl(G) has
at most k distinct derivation trees. G is said to
be inherently ambiguous of degree k, if every
grammar generating L(G) is ambiguous of degree
≥ k and some grammar generating L(G) is ambiguous
of degree k.

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

Inherently ambiguous cfl's, inherently
ambiguous cfl's of degree k, and infinitely
inherently ambiguous cfl's are defined analog-
ously. Thus a cfl L is infinitely inherently
ambiguous if every cfg generating L is in-
finitely inherently ambiguous. A cfl that is
not inherently ambiguous is said to be unam-
biguous.

Def. 1.3: P(NP) is the class of all languages
over {0,1} accepted by some deterministic (non-
deterministic) polynomially time-bounded Tm;
PSPACE is the class of all languages over {0,1}
accepted by some polynomially space-bounded
Tm.

Def. 1.4: Let I, A be finite nonempty alphe-
bets. Let L ⊆ I^* and M ⊆ A^*. We say that
L is p-reducible to M, written L ≤_p M,
if there is a function f : I^* → A^* comput-
able by a deterministic polynomially time-bounded
Tm such that for all x ∈ I^*, xcl(I) iff f(x) ∈ M.
If L is p-reducible to M and M is p-
reducible to L, then L and M are said to be
p-equivalent. L is said to be NP-hard(PSPACE-
hard) if all languages in NP(PSPACE) are p-reducible
to L. L is said to be NP-complete(PSPACE-complete)
if it is NP-hard(PSPACE-hard) and is accepted by
some nondeterministic polynomially time-bounded
(polynomially space-bounded) Tm.

Def. 1.5: A log-space transducer M is a de-
terministic 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 L, A be
finite nonempty alphabets. A function f : I^* → A^* is said to be log-space computable
if a log-space transducer M such that
M, when given input xcl(I), eventually halts
with output f(x). For L ⊆ I^* and M ⊆ A^*
we say that L is log-space reducible to M
written L ⊆_log M, if there is a log-space computable
function f such that for all x ∈ I^*, xcl(I) if f(x) ∈ M.
In addition if |f(x)| ∈ O(|x|) and some log-space transducer that computes
f is O(|f(x)|) time-bounded, we denote this by
L ⊆_log^{time} M.

Def. 1.6: Let Nt = (log n) denote the class of
all languages over {0,1} accepted by some non-
deterministic log n tape-bounded Tm. A lan-
guage N is said to be log-complete in Nt =
(log n) if for all log-tape(log n) tape-bounded
Tm N is accepted by some nondeterministic log n
tape-bounded Tm. Similarly N is said to be
log-complete in P if for all P Tm N is accepted by
some nondeterministic log n tape-bounded Tm.
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 \( L \subseteq \log \)

implies \( L \subseteq N \).

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

Prop. 1.6:
1. \( P = NP \) iff \( L \) is an NP-complete language \( L_0 \) such that \( L_0^{P} \).
2. \( P(NP) = PSPACE \) iff \( L \) is a PSPACE-complete language \( L_0 \) such that \( L_0^{P(NP)} \).
3. \( \text{Dtape}(\log n) = \text{Ndtape}(\log n) \) iff \( L \) is a language \( L_0 \) such that \( L_0^{P} \) is log-complete in \( \text{Dtape}(\log n) \) and \( L_0^{P} \).
4. Let \( k \geq 1 \). \( P \subseteq \text{Dtape}(\log n) \) iff \( L \) is a language \( L_0 \) such that \( L_0^{P} \) is log-complete in \( P \) and \( L_0^{P} \).

Def. 1.7: A language \( L \subseteq \Sigma^* \) is said to be bounded iff 3 strings \( \Sigma^* \) such that \( L \subseteq \Sigma^* \) and \( \Sigma^* \). A language that is not bounded is said to be unbounded.

Prop. 1.8 [10]: A regular set \( \mathcal{R} \) over \((0,1)^* \) is unbounded iff \( 3 \) strings \( r,s,x,y \) such that \( r \cdot (0x + 1y)^* \cdot s \subseteq R \).

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

2. Grammar Problem

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 cfi'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 Metatheorem

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

Thm. 2.1: Let \( \mathcal{G} \) be any subset of the finitely inherently ambiguous cfi's over \((0,1)^* \) such that \( \mathcal{H}_G \).

Proof: As the reader can verify, the finitely inherently ambiguous cfi'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 \( xy \in (0,1)^* \).

By assumption \( \mathcal{H}_G \) is a language. Thus \( \mathcal{H}_G \).

But \( \mathcal{L}(G) \) is a set such that \( \mathcal{L}(G) \).

There are two cases to consider.

Case 1: Let \( \mathcal{L}(G) \subseteq (0,1)^* \). Then \( \mathcal{L}(G) \).

Case 2: Let \( \mathcal{L}(G) \subseteq (0,1)^* \). Then \( \mathcal{L}(G) \).

Thus \( \mathcal{L}(G) \).

This implies that \( \mathcal{L}(G) \).

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

We sometimes use "\( + \)" to denote union. Thus \( A \cdot B = A \cdot B \).
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 EL(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. FPFA 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 \( k \) and \( k+1 \).

Thus letting \( \Gamma \) denote any of the above classes of the cfl's, the predicate \( "L(\Gamma)=?" \) 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 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 -
\[ L_1 = L(G_1[M,x]) = \#^x - \#^z - \#^y - \#^x^y^z^x \] where \( x, y, z \) are i.d.'s of M and \( y, z \) are i.d.'s of M.
\[ L_2 = L(G_2[M,x]) = \#^x - \#^z - \#^y - \#^x^y^z^x \] is an i.d. of M's.
\[ M \rightarrow S, S \rightarrow \#^x, S \rightarrow \#^z \] is an accepting i.d. of M. 

For sufficiently fast increasing functions T(n), no pair of words in \( L_1 \) and \( L_2 \) has a common prefix of length \( \geq c(T(|x|)) \) for some positive integer \( c \) depending only on M not \( x \).

Moreover, \( G_1[M,x] \) and \( G_2[M,x] \) are strong LL and SLR grammars.

Prop. 2.3: Let \( M, T(n), c, a, x \) be as described above. Let \( k = c(T(|x|)) \). Then in time \( \leq c_1|x| \), cfl'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. \( L(G_1) \cap L(G_2) \neq \emptyset \);
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);
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) \( S \rightarrow A_1 \cdots A_n \); where \( A_1 \) and \( A_2 \) are the start symbols of \( G_1 \) and \( G_2 \), respectively;
(c) \( A \rightarrow \#^x \); 
(d) \( B \rightarrow \#^y \);
(e) \( S_3 \rightarrow aBc \); 
(f) \( T \rightarrow aB\# \);
(g) \( U \rightarrow \#B\# \);
(h) \( S_4 \rightarrow aBc \); 
(i) \( V \rightarrow a\# \); and
(j) \( W \rightarrow \# \).

We assume that \( (S_3A,S_3B,S_4U,V,W) \) is disjoint from the union of the nonterminal alphabets of \( G_1 \) and \( G_2 \).

If \( M \) accepts \( x \) then \( 3 \) a string \( w = L(G_1)NL(G_2) \). Thus \( L(G_1)NC-w-S^* = \#^w \#^C \). 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 not strong LL(k), SLR(k), or LALR(k) for any choice of \( k \). If \( M \) does not accept \( x \), then \( L(G_1)NL(G_2) = \emptyset \) 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 strongly LL(k), SLR(k), and LALR(k).

In what follows \( C \) denotes the intersection of the strong LL and SLR grammars.† Our first major result follows from Prop. 2.3.

†A more complex construction in Prop. 2.3 allows \( T \) of 2.4 to be replaced by the intersection of the BRC, strong LL, and SLR grammars - \( T \).
Thm. 2.4: Let $\Gamma$ be any class of cfg's such that
(i) $c < 1$ and
(ii) $\Gamma$ is the class of cfg's that are not inherently ambiguous.

Then for arbitrary cfg $G$, the predicates $\Pi_1^* = "G\uparrow"$ and $\Pi_2 = "L(G) . L(I) = |L| = L(g)"$ with $g \uparrow$ are undecidable.

Proof: Suppose $\Pi_1$ is decidable. Then there exists a strictly increasing recursive function $f$ that bounds the time required to decide $\Pi_1$.
Let $M$ be any $O(T(n))$ time-bounded TM with $T(n) \geq n$ strictly increasing. From 2.2 $L(M)$ is recognizable by some $c_1 \cdot n + f(c_2 \cdot n)$ time-bounded TM $H$, where $c_1$ and $c_2$ are constants depending only on $M$ and $n = |x|$.

$M$ operates as follows.
1. Given input $x$, $M$ constructs $G_3$ of Prop. 2.3.
2. $M$ tests if $\Pi_2(G_3)$ 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 $f(c_2 \cdot n)$ time.

By Prop. 2.3 if $x \notin L(M)$, then $G_3 \notin \Gamma$; 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 $f(n) = n^a + f(2n)$ is strictly greater than $a \cdot n + f(b \cdot n)$, a.e. Thus if $\Pi_1$ 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. LR
9. LALR
10. Floyd-Evans parsable
11. LR Regular
12. FFPP
13. SLR($k, \omega$), LR($k, \omega$) for some $k$
14. RPP
15. basic SPM parsable
16. full SPM parsable
17. unambiguous cfg’s
18. the class of cfg’s that are not inherently ambiguous.

Thus letting $\Gamma$ denote any of the above cfg classes, the predicates "$G\uparrow" and "$L(G) . L(I) = |L| = L(g)$ and $g \uparrow$" are undecidable for arbitrary $G$.

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 $\mathcal{C}(k)$ be the intersection of the strong LL($k$), SLR($k$), and BC($k, k$) grammars.

Thm. 2.6: Let $G = \bigcup_{k \geq 1} \mathcal{C}(k)$ be any class of parameterized cfg’s such that for all $k \geq 1$, $\mathcal{C}(k) \subseteq \Gamma$ the class of cfg’s that are not inherently ambiguous. Then
(i) $L_1 = \{(G, \omega) : G \text{ is a cfg, } \omega \text{ is a unary numeral for the positive integer } n, \text{ and } G(n^m) \geq \text{NP and } 10^n k\log n \text{ timespace}\}$
(ii) $L_2 = \{(G, \omega) : G \text{ is a cfg, } \omega \text{ is a binary numeral for the positive integer } n, \text{ and } G(n^m) \geq \text{NDEXP and } 10^n \text{ timespace}\}$

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 $\Gamma$ satisfying the conditions of Thm. 2.6,
(i) $L_1$ is NP-hard; and
(ii) is constant $c > 0$ such that any nonde terministic TM that accepts $L_2$ requires time $> \omega cn$, i.o.

Thm. 2.8: The following classes of cfg’s satisfy the conditions of Thm. 2.6.
1. BC
2. BNC
3. strong LL
4. LL
5. strong LC
6. LC
7. ELC
8. $k$-Transformable for some $k$
9. SLR
10. LR
11. LALR
12. Floyd-Evans parsable
13. SLR($k, \omega$), LR($k, \omega$) for some $k$
14. RPP
15. basic SPM parsable
16. full SPM parsable

\*The undecidability of classes 15 and 16 was not previously known.
(i) \( L_1 \) is in \( NP \); and
(ii) a constant \( d \geq 0 \) 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.

<table>
<thead>
<tr>
<th>Source Class</th>
<th>LR</th>
<th>LALR</th>
<th>SLR</th>
<th>LL</th>
<th>strongLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR(k)</td>
<td>T</td>
<td>T</td>
<td>k=0</td>
<td>D</td>
<td>k-2</td>
</tr>
<tr>
<td>LALR(k)</td>
<td>T</td>
<td>T</td>
<td>k=0</td>
<td>D</td>
<td>k-2</td>
</tr>
<tr>
<td>SLR(k)</td>
<td>T</td>
<td>T</td>
<td>D</td>
<td>D</td>
<td>k-2</td>
</tr>
<tr>
<td>LL(k)</td>
<td>T</td>
<td>T</td>
<td>k=l</td>
<td>T</td>
<td>k-2</td>
</tr>
<tr>
<td>strongLL(k)</td>
<td>T</td>
<td>T</td>
<td>k=l</td>
<td>T</td>
<td>k-2</td>
</tr>
</tbody>
</table>

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 \( LL(k) \) 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 \( LL(\mid G \mid + k_2 + k_0) \).

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

A noncanonical parsing analogue ([6] or [7] pp. 405-407) of Thm. 2.4 also holds.

Thm. 2.10: Let \( G \) be any class of \( CFG's \) such that the \( SLR(1)^+ \) grammars \( \mid G \mid \leq 1 \) are the unambiguous \( CFG's \). Then for arbitrary \( CFG G \), the predicates \( \text{"get"} \) and \( \text{"get} \mid G \mid \text{"L(k)f(T)"} \) 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 \( O(k^{2.5}) \) 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 \( o \) and a positive integer \( k \) such that for every nondeterministic \( TM \), \( M \), which uses time \( T_{1}(n) \geq n \), \( o(M) \) is an equivalent deterministic \( TM \) working in time \( O(T_{1}(n))^{k} \).

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) \( L(M) = \{G \mid G \text{ is not in } C(k) \} \) and
(b) \( M \) performs at most \( a \cdot |G|^{b \cdot k^c} \) moves on any input \( \{G \} \).

Combining these theorems we have the following.

Thm. 3.3: If for all integers \( k \geq 1 \) there exists an integer \( k_0 \geq 1 \) such that \( LL(k) \), \( LC(k) \), \( LR(k) \), strong \( LL(k) \), strong \( LC(k) \), or \( SLR(k) \) testing requires time \( \Omega(n^{k}) \) i.e., then \( P \neq NP \).

Proof: From 3.1 \( P = NP \) implies that \( a \) a constant \( d > 0 \) such that \( LCd \text{time}(n^{d}) \) implies that \( LCd \text{time}(n^{d}) \) for all integers \( k > 0 \). Thus \( P = NP \), 3.1, and 3.2 imply for all \( k > 1 \), that \( LL(k) \), \( LC(k) \), ..., and \( SLR(k) \) testing require time at most \( O(a \cdot |G|^{b \cdot k^c}) \), \( \Omega(k^{d \cdot |G|^{b \cdot c}}) \) 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 \( \Gamma \) be any class of cfg’s for which

(i) \( k_0 \geq 1 \) such that the \( LL(k_0) \) grammars in \( \Gamma \subseteq LR(k_0) \) grammars. Then \( \forall \Gamma \subseteq LL(k_0) \) grammars \( (G_0) \) is an \( LL(k_0) \) grammar \( \leq (G_0) \). \( \log \frac{\lambda(G_0)}{\lambda(G)} \), \( \log \frac{\lambda(G_0)}{\lambda(G)} \).

(ii) \( k_0 \geq 1 \) such that the strong \( LL(k_0) \) grammars \( \subseteq SLR(k_0) \) grammars \( \subseteq \Gamma \subseteq \) the strong \( LL(k_0) \) grammars \( \subseteq SLR(k_0) \) grammars. Then \( (G_0) \) is a strong \( LL(k_0) \) grammar \( \leq (G_0) \). \( \log \frac{\lambda(G_0)}{\lambda(G)} \), \( \log \frac{\lambda(G_0)}{\lambda(G)} \).

Proof of (i): Brosig [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 \( G' \) is \( \equiv \) \( G \) is \( LL(k_0) \); and the construction of \( G' \) from \( G \) requires at most \( O(\lambda(G)) \) space and time on a log-space transducer.

Informally, every class \( \Gamma \) of cfg’s satisfying the conditions of (i) or (ii) of Section 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 \( LL(k_0) \) and \( SLR(k_0) \) grammars. Theorem 3.4 establishes a hierarchy of the \( LL(k_0) \) and \( LR(k_0) \) grammars.

**Thm. 3.5:** For all integers \( k_0 \geq 1 \),

(i) \( (G_0) \) is a strong \( LL(k_0) \) grammar \( \leq (G_0) \).

(ii) \( (G_0) \) is an \( LL(k_0) \) grammar \( \leq (G_0) \).

(iii) \( (G_0) \) is an \( SLR(k_0) \) grammar \( \leq (G_0) \).

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 \( N \)-tape \( (\log) \), \( P \), the 2-way PDA languages, etc. are presented. We also present strong evidence for the nonlinearity in time of every nontrivial predicate on the cfg’s, when applied to the PDA.

**Thm. 3.6:** Let \( \Gamma \) be any nontrivial predicate

(1) on the regular sets over \( \{0,1\} \) such that \( \Gamma(\epsilon) \) is false, then \( (M) \) is an NDFA (regular grammar) with \( \lambda \)-moves (\( \lambda \)-productions) and \( \lambda(L(M)) \) is true > \( \log \).

(2) on the deterministic cfg’s over \( \{0,1\} \) such that \( \Gamma(\epsilon) \) is false, then \( (M) \) is a deterministic PDA and \( \lambda(L(M)) \) is true > \( \log \).

(3) on the strict deterministic languages over \( \{0,1\} \) such that \( \Gamma(\epsilon) \) is false, then \( (G_0) \) is a deterministic PDA and \( \lambda(L(G_0)) \) is true > \( \log \).

(4) on the cfg’s over \( \{0,1\} \) such that \( \Gamma(\epsilon) \) is false, then \( (M) \) is a PDA and \( \lambda(L(M)) \) is true > \( \log \).

(5) on the 1-way DFA languages over \( \{0,1\} \), then \( 3 \lambda > 0 \) such that \( (M) \) is a 1-way DFA and \( \lambda(L(M)) \) is true > \( \log \).

(6) on the 1-way SA languages over \( \{0,1\} \), then \( 3 \lambda > 0 \) such that \( (M) \) is a 1-way SA and \( \lambda(L(M)) \) is true > \( \log \).

(7) on the indexed languages over \( \{0,1\} \), then \( 3 \lambda > 0 \) such that \( (G_0) \) is an indexed or \( \lambda \)-macro grammar and \( \lambda(L(G_0)) \) is true > \( \log \).

(8) on the recursively enumerable sets over \( \{0,1\} \) such that \( \lambda(\epsilon) \) is true, then \( (M) \) is a TM of type 0 and \( \lambda(L(M)) \) is true > \( \log \).

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 \( N \)-tape \( (\log) \). Let \( P \) be any nontrivial predicate on regular sets such that \( \lambda(\epsilon) \) is false. Since \( \Gamma \) is nontrivial there exists an NDFA \( M_0 \) such that \( \lambda(L(M_0)) \) is true. Let \( L_0 = L(M_0) \). Clearly \( L_0 \neq \emptyset \).
Let $M_1$ be an arbitrary $2$-way $k$-head NFA with $k \geq 1$. For all $x \in \{0,1\}^*$, an NFA $M_{1,x}$ with $\lambda$-moves can be constructed such that

$$L(M_{1,x}) = \begin{cases} \phi & \text{if } x \in L(M_1) \\ L_0 & \text{if } x \notin L(M_1) \end{cases}$$

For each input $x = x_1 \ldots x_n$ to $M_1$, $M_{1,x}$ is constructed as follows:

(a) All input tape configurations of $M_1$ on $x$ are embedded in $M_{1,x}$'s finite state control.

(b) $|M_{1,x}| \leq c_1 \cdot |x|^{k \cdot \log(|x|)}$, where $c_1$ depends only upon $M_1$ and $M_0$ not on $x$.

(c) $M_{1,x}$ simulates $M_1$ on $x$. If $M_1$ accepts $x$, then $M_{1,x}$ simulates $M_0$ on its $(M_1,x)'s$ input. If $M_1$ does not accept $x$, then $L(M_{1,x}) = \phi$.

(d) For fixed $i$, the construction of $M_{1,x}$ from $M_1$ and $x$ can be accomplished on a deterministic $\log |x|$ space-bounded transducer.

$M_{1,x}$'s simulation of $M_1$ on input $x$ only involves $\lambda$-moves. $M_{1,x}$'s state set includes states of the form $(p,v_1 \ldots v_k)$, where $p$ denotes a state of $M_1$ and $v_1 \ldots v_k$ are binary numerals for positive integers $n_1 \ldots n_k$ respectively, with $n_1 \ldots n_k \leq |x| = n$. State $(p,v_1 \ldots v_k)$ signifies that $M_1$ is in state $p$ and that its first input tape head is scanning $v_1$st character of $x$, its second input tape head is scanning the $v_2$st character of $x$, etc. The construction of $M_{1,x}$ from $M_1$ and $x$ can be accomplished within $O(|x|^{k \cdot \log |x|})$ time and with $O(|\log n|)$ intermediate storage.

But $L(M_{1,x}) = \phi$ if $x \in L(M_1)$ and $L(M_{1,x}) = L_0$ otherwise.

Thus $L(M_{1,x})$ is true iff $x \in L(M_1)$.

This follows since $x \in L(M_1)$ implies that $L(M_{1,x}) = \phi$ and $P(L(M_{1,x}))$ is false by assumption. Otherwise $L(M_{1,x}) = L_0$ and $P(L(M_{1,x}))$ is true by assumption.

(2) Cook [26] has shown that the class of languages accepted by $2$-way multi-head deterministic DFA 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 dfa $M$ with a single final state, the canonical grammar of $M$ is a strict deterministic grammar [19].

(4) Cook [26] has also shown that the class of languages accepted by $2$-way multi-head PDA equals $P$. The proof holds for the CFG's as well as the PDA since there exists a deterministic log space transducer $M$ such that $M$ accepts the $L(M)$ of a $2$-way multi-head PDA.

(5) The class of languages accepted by $1$-way DSA equals the class of languages accepted by a $O(2^{n \log n})$ 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 a $O(2^{n^2})$ 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 $D$ macro grammar, respectively, can be seen to be executable deterministically in polynomial time.

(8) Let $M_1$ be any arbitrary TM. For all $x \in \{0,1\}^*$, a TM $M_{1,x}$ can be constructed effectively such that $L(M_{1,x}) = \phi$ if $x \in L(M_1)$, $L_0$ otherwise.

Here $L_0$ is some nonempty set for which $P(L_0) = \phi$. Thus $(M,M)$ is a TM and $M$ diverges on empty input; it 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 NFA, deterministic PDA, PDA, DSA, and SA requires as much time and/or space as any language recognizable by a $2$-way $1$-head NFA, deterministic PDA, PDA, DSA, and SA, respectively. We present a partial converse.

Theorem 3.7:

(1) $L_1 = M \in M$ is an NFA with $\lambda$-moves and $L(M) \neq \phi$ is the accepted language of some $2$-way $2$-head NFA.

(2) $L_2 = (M,M)$ is a PDA and $L(M) \neq \phi$ is the accepted language of some $2$-way $1$-head PDA.

(3) $L_3 = (M,M)$ is a $1$-way SA and $L(M) \neq \phi$ is the accepted language of some $2$-way $2$-head SA.

(4) $L_4 = (M,M)$ is a $1$-way deterministic PDA and $L(M)$ is the accepted language of some $2$-way $1$-head PDA.

(5) $L_5 = (M,M)$ is a $1$-way DSA and $L(M)$ is the accepted language of some $2$-way $2$-head DSA.

See [19] for the definition of canonical grammar.
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 N DFA such that $L$ is accepted by some 2-way multi-head DFA iff $\text{Dptime}^{n}(\log n) = \text{Dtape}(\log n)$.

$L_1$ is one such language; in fact, $L_1$ is log-complete in $\text{Dtape}(\log n)$. 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 $= \{G \mid G \text{ is a directed graph on } (1, \ldots, n) \text{ for some } n, \text{ which has a path from vertex } 1 \text{ to vertex } n\}$ is accepted by some 2-way 2-head NDFA.

(ii) [10] GAP is log-complete in $\text{Dtape}(\log n)$.

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

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

Cor. 3.11 [12]: The languages $L_3 = \{G \mid G \text{ is a cfg and } L(G) \neq \emptyset\}$ and $L_4 = \{G \mid G \text{ is a cfg and } L(G) \text{ is finite}\}$ are log-complete in $P$.

Cor. 3.12: The language $L_5$ is log-complete in $P$.

Cor. 3.12 should be compared with the theorem due to Lewis, Stearns, and Hartmanis [29] that every CFL $\in \text{Dtime}([\log n]^k)$.

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

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

Cor. 3.15: $r_1, r_2 > 0$ such that the recognition of $L = \{G \mid G \text{ is an indexed [01-macro] grammar}\}$ and $L(G) / \emptyset$ requires time $> 2^{r_1 n^2}$, i.o. on any deterministic $T_m$. Moreover, $L$ is recognizable by some $2^{r_2 n^2}$ deterministic time-bounded TM.

The upper bounds in Cor. 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 $\lambda$-moves) have the same lower complexity bounds as the emptiness problems for the corresponding 1-way nondeterministic automata with $\lambda$-moves.

Thm. 3.16: (i) $L_1 = \{M \mid M \text{ is a DFA and } L(M) \neq \emptyset\}$ is log-complete in $\text{Dtape}(\log n)$. Moreover, $L_1$ is recognizable by some 2-way 2-head NDFA.

(ii) $L_2 = \{M \mid M \text{ is a deterministic PDA with no } \lambda \text{-moves and } L(M) \neq \emptyset\}$ is log-complete in $P$, is a 2-way 2-head language, and $L_2$ is a 2-way PDA language. $\log \text{time}(\text{space \_time})$

(iii) $c_3, c_4 > 0$ such that the recognition of $L_3 = \{M \mid M \text{ is a 1-way DSA with no } \lambda \text{-moves}\}$ and $L(M) \neq \emptyset$ requires time $> 2^{c_3 n^2 / (\log n)^2}$, i.o. on any deterministic multi-tape TM. $\log \text{time} 2^{c_4 n^4}$ 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 $O(n^2 \text{-Polynomial}(\log n))$ 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 $x_1 \ldots x_n \in L$, the following steps suffice:

(1) Construct a 1-way PDA $M_x$, as described in the proof of 3.6 such that $L(M_x) \neq \emptyset$ iff $x_1 \ldots x_n \in L$.

(2) Convert $M_x$ into an equivalent context-free grammar $G_x$.

(3) Test $G_x$ for emptiness.

(4) If $L(G_x) \neq \emptyset$, then $x_1 \ldots x_n \in L$. Otherwise $x_1 \ldots x_n \notin L$.

The time required to execute step 1 is $O(n^2 \text{-Polynomial}(\log n))$. The time required to convert $M$ into an equivalent CFG $G_x$ is $O(n^2 \text{-Polynomial}(\log n)^2)$. Finally, the time to test $G_x$ for emptiness is $O(n \log n)$.
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 preceding 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 $\Sigma^k P(k)$.

In [33] we conjectured that structural equiva-
rence 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 $y(T) = \cup y(T)$.

A predicate $\Pi$ on a class of tree languages
$G$ is said to be yield-invariant if for all
$T, T' \in \Pi, y(T) = y(T') \Rightarrow \Pi(T) = \Pi(T')$.

We allow trees to have leaves labeled with
$\lambda$, the empty string.

Thm. 3.21: Let $\Pi$ be any nontrivial yield-
invariant predicate
(i) on the recognizable sets over $\{0,1\}$ such that
$\Pi(M)$ is true, then $\{M| M$ is a non-
deterministic bottom-up (or top-down) tree
automaton and $y(L(M))$ is true} $\geq P$; and
(ii) on the context-free dendro-languages [35]
such that $\Pi(G)$ is true, then $G$ is a context-
free grammar with $x$-productions and
$y(L(G))$ is true} require

time $2^n$ i.e. on any multi-tape deter-
mindistic TM for some $n \geq 0$.

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 Sec-
tion 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 $\equiv$
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
$S_1, S_2, S_1 \sim S_2$ implies $S_1 \equiv S_2$ and $S_1 \equiv S_2$
implies $S_1 = S_2$.

Thm. 4.2: For all reasonable relations $\sim$ and
(i) on all fixed non-divergent 2-variable,
single-variable, or loop-free program schemes
$S$; $[S]$ is a 2-variable, single variable,
or loop-free program scheme, respectively,
and $S \neq \bar{S}$ is NP-hard; and
(ii) for all fixed monadic or linear monadic
recursion schemes $S$; $[S]$ is a monadic
or linear monadic recursion scheme, respec-
tively, and $S \neq \bar{S}$ is NP-hard.

Proof sketch: We efficiently reduce the well-
known NP-complete set $\{f | f$ is a $D_2$-Boolean form
and $f$ is not a tautology\} to the predicate
"$S \neq \bar{S}$". Let $f$ be any arbitrary $D_2$-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 $f$.
Without loss of generality, we assume for all
$D_2$-Boolean forms $f$ that the predicate symbols
and the labels appearing in $S_f$ and $\bar{S}$ are
disjoint. Let $S_f$ be the scheme that results
from (a) replacing all occurrences in $\bar{S}$ of
the label of the initial statement of $\bar{S}$ by
A; (b) replacing the statement "A:Halts," in
$S_f$ by the initial statement of the scheme that
resulted from $\bar{S}$ after (a); and (c) replacing
the statement "B:Halts" by "B: Do $g(x); Halts." Then
$S_f \equiv \bar{S}$ iff $f$ is a tautology.

Since there are fixed schemes $\equiv$ and reason-
able relations $\sim$ such that $[S]$ is a program
scheme and S / \vdash LIP, 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 also was presented.

Bibliography


[33] N. 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).


