COMPUTATIONAL COMPLEXITY OF CURRENT GPSG THEORY

Eric Sven Ristad

MIT Artificial Intelligence Lab
545 Technology Square
Cambridge, MA 02139

and

Thinking Machines Corporation
245 First Street
Cambridge, MA 02142

ABSTRACT

An important goal of computational linguistics has been to use linguistic theory to guide the construction of computationally efficient real-world natural language processing systems. At first glance, generalized phrase structure grammar (GPSG) appears to be a blessing on two counts. First, the precise formalisms of GPSG might be a direct and transparent guide for parser design and implementation. Second, since GPSG has weak context-free generative power and context-free languages can be parsed in $O(n^3)$ by a wide range of algorithms, GPSG parsers would appear to run in polynomial time. This widely-assumed GPSG "efficient parsability" result is misleading: here we prove that the universal recognition problem for current GPSG theory is exponential-polynomial time hard, and assuredly intractable. The paper pinpoints sources of complexity (e.g. metarules and the theory of syntactic features) in the current GPSG theory and concludes with some linguistically and computationally motivated restrictions on GPSG.

1 Introduction

An important goal of computational linguistics has been to use linguistic theory to guide the construction of computationally efficient real-world natural language processing systems. Generalized Phrase Structure Grammar (GPSG) linguistic theory holds out considerable promise as an aid in this task. The precise formalisms of GPSG offer the prospect of a direct and transparent guide for parser design and implementation. Furthermore, and more importantly, GPSG's weak context-free generative power suggests an efficiency advantage for GPSG-based parsers. Since context-free languages can be parsed in polynomial time, it seems plausible that GPSGs can also be parsed in polynomial time. This would in turn seem to provide "the beginnings of an explanation for the obvious, but largely ignored, fact that humans process the utterances they hear very rapidly (Gazdar,1981:155)."[1]

In this paper I argue that the expectations of the informal complexity argument from weak context-free generative power are not in fact met. I begin by examining the computational complexity of metarules and the feature system of GPSG and show that these systems can lead to computational intractability. Next I prove that the universal recognition problem for current GPSG theory is Exp-Poly hard, and assuredly intractable.[2] That is, the problem of determining for an arbitrary GPSG G and input string x whether x is in the language $L(G)$ generated by G, is exponential polynomial time hard. This result puts GPSG-Recognition in a complexity class occupied by few natural problems: GPSG-Recognition is harder than the traveling salesman problem, context-sensitive language recognition, or winning the game of Chess on an $n \times n$ board. The complexity classification shows that the fastest recognition algorithm for GPSGs must take exponential time or worse. One role of a computational analysis is to provide formal insights into linguistic theory. To this end, this paper pinpoints sources of complexity in the current GPSG theory and concludes with some linguistically and computationally motivated restrictions.

2 Complexity of GPSG Components

A generalized phrase structure grammar contains five language-particular components — immediate dominance (ID) rules, metarules, linear precedence (LP) statements, feature co-occurrence restrictions (FCRs), and feature specification defaults (FSDs) — and four universal components — a theory of syntactic features, principles of universal feature instantiation, principles of semantic interpretation, and formal relationships among various components of the grammar.[3]

Syntactic categories are partial functions from features to atomic feature values and syntactic categories. They encode subcategorization, agreement, unbounded dependency, and other significant syntactic information. The set K of syntactic categories is inductively specified by listing the set F of features, the set A of atomic feature values, the function ρ^0 that defines the range of each atomic-valued feature, and a set R of restrictive predicates on categories (FCRs).

The set of ID rules obtained by taking the finite closure of the metarules on the ID rules is mapped into local phrase structure trees, subject to principles of universal feature instantiation, FSDs, FCRs, and LP statements. Finally, local trees are

[1]See also Joshi, "Tree Adjoining Grammars" p.226, in *Natural Language Parsing* (1985) ed. by D. Dowty, L. Karttunen, and A. Zwicky, Cambridge University Press: Cambridge, and "Exceptions to the Rule," *Science News* 128: 314–315.

[2]We use the universal problem to more accurately explore the power of a grammatical formalism (see section 3.1 below for support). Ristad(1985) has previously proven that the universal recognition problem for the GPSG's of Gazdar(1981) is NP-hard and likely to be intractable, even under severe metarule restrictions.

[3]This work is based on current GPSG theory as presented in Gazdar et. al. (1985), hereafter GKPS. The reader is urged to consult that work for a formal presentation and thorough exposition of current GPSG theory.

assembled to form phrase structure trees, which are terminated by lexical elements.

To identify sources of complexity in GPSG theory, we consider the isolated complexity of the finite metarule closure operation and the rule to tree mapping, using the finite closure membership and category membership problems, respectively. Informally, the ***finite closure membership*** problem is to determine if an ID rule is in the finite closure of a set of metarules M on a set of ID rules R. The ***category membership*** problem is to determine if a category or C or a legal extension of C is in the set K of all categories based the function ρ and the sets A, F and R. Note that both problems must be solved by any GPSG-based parsing system when computing the ID rule to local tree mapping.

The major results are that finite closure membership is NP-hard and category membership is PSPACE-hard. Barton(1985) has previously shown that the recognition problem for ID/LP grammars is NP-hard. The components of GPSG theory are computationally complex, as is the theory as a whole.

Assumptions. In the following problem definitions, we allow syntactic categories to be based on arbitrary sets of features and feature values. In actuality, GPSG syntactic categories are based on fixed sets and a fixed function ρ. As such, the set K of permissible categories is finite, and a large table containing K could, in principle, be given.[4] We (uncontroversially) generalize to arbitrary sets and an arbitrary function ρ to prevent such a solution while preserving GPSG's theory of syntactic features.[5] No other modifications to the theory are made.

An ambiguity in GKPS is how the FCRs actually apply to embedded categories.[6] Following Ivan Sag (personal communication), I make the natural assumption here that FCRs apply top-level and to embedded categories equally.

[4]This suggestion is of no practical significance, because the actual number of GPSG syntactic categories is extremely large. The total number of categories, given the 25 atomic features and 4 category-valued features, is:

$$\begin{aligned} |K = K^4| &= 3^{25}((1+3^{25})((1+3^{25})((1+3^{25})(1+3^{25})^1)^2)^3)^4 \\ &= 3^{25}(1+3^{25})^{64} > 3^{1625} > 10^{774} \end{aligned}$$

See page 10 for details. Many of these categories will be linguistically meaningless, but all GPSGs will generate all of them and then filter some out in consideration of FCRs, FSDs, universal feature instantiation, and the other admissible local trees and lexical entries in the GPSG. While the FCRs in some grammars may reduce the number of categories, FCRs are a language-particular component of the grammar. The vast number of categories cited above is *inherent* in the GPSG framework.

[5]Our goal is to identify sources of complexity in GPSG theory. The generalization to arbitrary sets allows a fine-grained study of one component of GPSG theory (the theory of syntactic features) with the tools of computational complexity theory. Similarly, the chess board is uncontroversially generalized to size $n \times n$ in order to study the computational complexity of chess.

[6]A category C that is defined for a feature f, $f \in (F - \texttt{Atom}) \cap \texttt{DOM}(C)$ (e.g. $f = \texttt{SLASH}$), contains an embedded category C_e, where $C(f) = C_e$. GKPS does not explain whether FCR's must be true of C_e as well as C.

2.1 Metarules

The complete set of ID rules in a GPSG is the maximal set that can be arrived at by taking each metarule and applying it to the set of rules that have not themselves arisen as a result of the application of that metarule. This maximal set is called the finite closure (FC) of a set R of lexical ID rules under a set M of metarules.

The cleanest possible complexity proof for metarule finite closure would fix the GPSG (with the exception of metarules) for a given problem, and then construct metarules dependent on the problem instance that is being reduced. Unfortunately, metarules cannot be cleanly removed from the GPSG system. Metarules take ID rules as input, and produce other ID rules as their output. If we were to separate metarules from their inputs and outputs, there would be nothing left to study.

The best complexity proof for metarules, then, would fix the GPSG modulo the metarules and their input. We ensure the input is not inadvertently performing some computation by requiring the one ID rule R allowed in the reduction to be fully specified, with only one 0-level category on the left-hand side and one unanalyzable terminal symbol on the right-hand side. Furthermore, no FCRs, FSDs, or principles of universal feature instantiation are allowed to apply. These are exceedingly severe constraints. The ID rules generated by this formal system will be the finite closure of the lone ID rule R under the set M of metarules.

The (***strict***, resp.) ***finite closure membership problem*** for GPSG metarules is: Given an ID rule r and sets of metarules M and ID rules R, determine if $\exists r'$ such that $r' \sqsupseteq r$ ($r' = r$, resp.) and $r' \in FC(M, R)$.

Theorem 1: Finite Closure Membership is NP-hard

Proof: On input 3-CNF formula F of length n using the m variables $x_1 \ldots x_m$, reduce 3-SAT, a known NP-complete problem, to Metarule-Membership in polynomial time.

The set of ID rules consists of the one ID rule R, whose mother category represents the formula variables and clauses, and a set of metarules M s.t. an extension of the ID rule A is in the finite closure of M over R iff F is satisfiable. The metarules generate possible truth assignments for the formula variables, and then compute the truth value of F in the context of those truth assignments.

Let w be the string of formula literals in F, and let w_i denote the i^{th} symbol in the string w.

1. The ID rules R, A

R : $F \rightarrow$ `<satisfiability>`

A : $[[\mathtt{STAGE}\ 3]] \rightarrow$ `<satisfiable>`

where

`<satisfiable>`	is a terminal symbol
`<satisfiability>`	is a terminal symbol
$F =$	$\{[y_i\ 0] : 1 \leq i \leq m\}$ $\cup\ \{[c_i\ 0] : 1 \leq i \leq \frac{\lvert w \rvert}{3}\}$ $\cup\ \{[\mathtt{STAGE}\ 1]\}$

2. Construct the metarules

(a) m metarules to generate all possible assignments to the variables

$$\forall i, 1 \leq i \leq m$$
$$\begin{array}{c} \{[y_i\ 0], [\mathtt{STAGE}\ 1]\} \rightarrow W \\ \Downarrow \\ \{[y_i\ 1], [\mathtt{STAGE}\ 1]\} \rightarrow W \end{array} \quad (1)$$

(b) one metarule to stop the assignment generation process

$$\begin{array}{c} \{[\mathtt{STAGE}\ 1]\} \rightarrow W \\ \Downarrow \\ \{[\mathtt{STAGE}\ 2]\} \rightarrow W \end{array} \quad (2)$$

(c) $\lvert w \rvert$ metarules to verify assignments
$\forall i, j, k \quad 1 \leq i \leq \frac{\lvert w \rvert}{3},\ 1 \leq j \leq m,\ 0 \leq k \leq 2$,
if $w_{3i-k} = x_j$, then construct the metarule

$$\begin{array}{c} \{[y_j\ 1], [c_i\ 0], [\mathtt{STAGE}\ 2]\} \rightarrow W \\ \Downarrow \\ \{[y_j\ 1], [c_i\ 1], [\mathtt{STAGE}\ 2]\} \rightarrow W \end{array} \quad (3)$$

$\forall i, j, k \quad 1 \leq i \leq \frac{\lvert w \rvert}{3},\ 1 \leq j \leq m,\ 0 \leq k \leq 2$,
if $w_{3i-k} = \overline{x_j}$, then construct the metarule

$$\begin{array}{c} \{[y_j\ 0], [c_i\ 0], [\mathtt{STAGE}\ 2]\} \rightarrow W \\ \Downarrow \\ \{[y_j\ 0], [c_i\ 1], [\mathtt{STAGE}\ 2]\} \rightarrow W \end{array} \quad (4)$$

(d) Let the category $C = \{[c_i\ 1] : 1 \leq i \leq \frac{\lvert w \rvert}{3}\}$. Construct the metarule

$$\begin{array}{c} C[\mathtt{STAGE}\ 2] \rightarrow W \\ \Downarrow \\ \{[\mathtt{STAGE}\ 3]\} \rightarrow \texttt{<satisfiable>} \end{array} \quad (5)$$

The reduction constructs $O(\lvert w \rvert)$ metarules of size $\log(\lvert w \rvert)$, and clearly may be performed in polynomial time: the reduction time is essentially the number of symbols needed to write the GPSG down. Note that the *strict* finite closure membership problem is also NP-hard. One need only add a polynomial number of metarules to "change" the feature values of the mother node C to some canonical value when $C(\mathtt{STAGE}\ ) = 3$ — all 0, for example, with the exception of **STAGE** . Let $F = \{[y_i\ 0] : 1 \leq i \leq m\}\ \cup\ \{[c_i\ 0] : 1 \leq i \leq \frac{\lvert w \rvert}{3}\}$. Then A would be

A : $F[\mathtt{STAGE}\ 3] \rightarrow$ `<satisfiable>`

Q.E.D

The major source of intractability is the finite closure operation itself. Informally, each metarule can more than double the number of ID rules, hence by chaining metarules (i.e. by applying the output of a metarule to the input of the next metarule) finite closure can increase the number of ID rules exponentially.[7]

2.2 A Theory of Syntactic Features

Here we show that the complex feature system employed by GPSG leads to computational intractability. The underlying insight for the following complexity proof is the almost direct equivalence between Alternating Turing Machines (ATMs) and syntactic categories in GPSG. The nodes of an ATM computation correspond to 0-level syntactic categories, and the ATM computation tree corresponds to a full, n-level syntactic category. The finite feature closure restriction on categories, which limits the depth of category nesting, will limit the depth of the corresponding ATM computation tree. Finite feature closure constrains us to specifying (at most) a polynomially deep, polynomially branching tree in polynomial time. This is exactly equivalent to a polynomial time ATM computation, and by Chandra and Stockmeyer(1976), also equivalent to a deterministic polynomial space-bounded Turing Machine computation.

As a consequence of the above insight, one would expect the GPSG Category-Membership problem to be PSPACE-hard. The actual proof is considerably simpler when framed as a reduction from the Quantified Boolean Formula (QBF) problem, a known PSPACE-complete problem.

Let a *specification of K* be the arbitrary sets of features F, atomic features **Atom**, atomic feature values A, and feature cooccurrence restrictions R and let ρ be an arbitrary function, all equivalent to those defined in chapter 2 of GKPS.

The *category membership problem* is: Given a category C and a specification of a set K of syntactic categories, determine if $\exists C'$ s.t. $C' \sqsupseteq C$ and $C' \in K$.

The *QBF problem* is $\{Q_1 y_1 Q_2 y_2 \ldots Q_m y_m F(y_1, y_2, \ldots, y_m) \mid Q_i \in \{\forall, \exists\}$, where the y_i are boolean variables, F is a boolean formula of length n in conjunctive normal form with exactly

[7]More precisely, the metarule finite closure operation can increase the size of a GPSG G worse than exponentially: from $\lvert G \rvert$ to $O(\lvert G \rvert^{2\lvert G \rvert})$. Given a set of ID rules R of symbol size n, and a set M of m metarule, each of size p, the symbol size of *FC(M,R)* is $O(n^{2^m}) = O(\lvert G \rvert^{2\lvert G \rvert})$. Each metarule can match the productions in R $O(n)$ different ways, inducing $O(n+p)$ new symbols per match: each metarule can therefore square the ID rule grammar size. There are m metarules, so finite closure can create an ID rule grammar with $O(n^{2^m})$ symbols.

three variables per clause (3-CNF), and the quantified formula is true}.

Theorem 2: GPSG Category-Membership is PSPACE-hard

Proof: By reduction from QBF. On input formula

$$\Omega = Q_1 y_1 Q_2 y_2 \dots Q_m y_m F(y_1, y_2, \dots, y_m)$$

we construct an instance P of the Category-Membership problem in polynomial time, such that $\Omega \in$ QBF if and only if P is true.

Consider the QBF as a strictly balanced binary tree, where the i^{th} quantifier Q_i represents pairs of subtrees $< T_t, T_f >$ such that (1) T_t and T_f each immediately dominate pairs of subtrees representing the quantifiers $Q_{i+1} \dots Q_m$, and (2) the i^{th} variable y_i is **true** in T_t and **false** in T_f. All nodes at level i in the whole tree correspond to the quantifier Q_i. The leaves of the tree are different instantiations of the formula F, corresponding to the quantifier-determined truth assignments to the m variables. A leaf node is labeled **true** if the instantiated formula F that it represents is true. An internal node in the tree at level i is labeled **true** if

1. $Q_i =$ "$\exists$" and either daughter is labeled **true**, or
2. $Q_i =$ "$\forall$" and both daughters are labeled **true**.

Otherwise, the node is labeled **false**.

Similarly, categories can be understood as trees, where the features in the domain of a category constitute a node in the tree, and a category C immediately dominates all categories C' such that $\exists f \in ((F - \texttt{Atom}) \cap \texttt{DOM}(C))[C(f) = C']$.

In the QBF reduction, the atomic-valued features are used to represent the m variables, the clauses of F, the quantifier the category represents, and the truth label of the category. The category-valued features represent the quantifiers — two category-valued features q_k, q'_k represent the subtree pairs $< T_t, T_f >$ for the quantifier Q_k. FCRs maintain quantifier-imposed variable truth assignments "down the tree" and calculate the truth labeling of all leaves, according to F, and internal nodes, according to quantifier meaning.

Details. Let w be the string of formula literals in F, and w_i denote the i^{th} symbol in the string w. We specify a set K of permissible categories based on A, F, ρ, and the set of FCRs R s.t. the category [[LABEL 1]] or an extension of it is an element of K iff Ω is **true**.

First we define the set of possible 0-level categories, which encode the formula F and truth assignments to the formula variables. The feature w_i represents the formula literal w_i in w, y_j represents the variable y_j in Ω, and c_i represents the truth value of the i^{th} clause in F.

$$\begin{aligned} Atom = \ & \{\text{LEVEL}, \text{LABEL}\} \\ & \cup \{w_i : 1 \le i \le |w|\} \\ & \cup \{y_j : 1 \le j \le m\} \\ & \cup \{c_i : 1 \le i \le \tfrac{|w|}{3}\} \\ F - Atom \ &= \{q_k, q'_k : 1 \le k \le m\} \\ \rho^0(\text{LEVEL}) \ &= \{k : 1 \le k \le m+1\} \\ \rho^0(f) \ &= \{0,1\} \quad \forall f \in Atom - \{\text{LEVEL}\} \end{aligned}$$

FCR's are included to constrain both the form and content of the guesses:

1. FCR's to create strictly balanced binary trees:

$$\forall k,\ 1 \le k \le m,$$
$$[\text{LEVEL } k] \equiv [q_k\ [[y_k\ 1][\text{LEVEL } k+1]]] \& [q'_k\ [[y_k\ 0][\text{LEVEL } k+1]]]$$

2. FCR's to ensure all 0-level categories are fully specified:

$$\begin{aligned} &\forall i,\ 1 \le i \le \tfrac{m}{3}, \\ &\quad [c_i] \equiv [w_{3i-2}]\&[w_{3i-1}]\&[w_{3i}] \\ &\quad [\text{LABEL }] \equiv [c_i] \\ &\forall k,\ 1 \le k \le m, \\ &\quad [\text{LABEL }] \equiv [y_k] \end{aligned}$$

3. FCR's to label internal nodes with truth values determined by quantifier meaning:

$$\begin{aligned} &\forall k,\ 1 \le k \le m, \\ &\text{if } Q_k = \text{"}\forall\text{", then include:} \\ &[\text{LEVEL } k]\&[\text{LABEL } 1] \equiv [q_k\ [[\text{LABEL } 1]]]\&[q'_k\ [[\text{LABEL } 1]]] \\ &[\text{LEVEL } k]\&[\text{LABEL } 0] \equiv [q_k\ [[\text{LABEL } 0]]] \vee [q'_k\ [[\text{LABEL } 0]]] \\ &\text{otherwise } Q_k = \text{"}\exists\text{", and include:} \\ &[\text{LEVEL } k]\&[\text{LABEL } 1] \equiv [q_k\ [[\text{LABEL } 1]]] \vee [q'_k\ [[\text{LABEL } 1]]] \\ &[\text{LEVEL } k]\&[\text{LABEL } 0] \equiv [q_k\ [[\text{LABEL } 0]]]\&[q'_k\ [[\text{LABEL } 0]]] \end{aligned}$$

The category-valued features q_k and q'_k represent the quantifier Q_k. In the category value of q_k, the formula variable $y_k = 1$ everywhere, while in the category value of q'_k, $y_k = 0$ everywhere.

4. one FCR to guarantee that only satisfiable assignments are permitted:

$$[\text{LEVEL } 1] \supset [\text{LABEL } 1]$$

5. FCR's to ensure that quantifier assignments are preserved "down the tree":

$$\begin{aligned} &\forall i, k \quad 1 \le i < k \le m, \\ &[y_i\ 1] \supset [q_k\ [[y_i\ 1]]]\&[q'_k\ [[y_i\ 1]]] \\ &[y_i\ 0] \supset [q_k\ [[y_i\ 0]]]\&[q'_k\ [[y_i\ 0]]] \end{aligned}$$

6. FCR's to instantiate variable assignments into the formula F:

$$\begin{array}{l} \forall i,k\ 1 \le i \le |w| \text{ and } 1 \le k \le m\,, \\ \quad \text{if } w_i = y_k, \text{ then include:} \\ \qquad [y_k\ 1] \supset [w_i\ 1] \\ \qquad [y_k\ 0] \supset [w_i\ 0] \\ \quad \text{else if } w_i = \overline{y_k}, \text{ then include:} \\ \qquad [y_k\ 1] \supset [w_i\ 0] \\ \qquad [y_k\ 0] \supset [w_i\ 1] \end{array}$$

7. FCR's to verify the guessed variable assignments in leaf nodes:

$$\begin{array}{l} \forall i \quad 1 \le i \le \frac{|w|}{3}\,, \\ [c_i\ 0] \ \equiv\ [w_{3i-2}\ 0]\&[w_{3i-1}\ 0]\&[w_{3i}\ 0] \\ [c_i\ 1] \ \equiv\ [w_{3i-2}\ 1] \lor [w_{3i-1}\ 1] \lor [w_{3i}\ 1] \\ [\text{LEVEL}\ m+1]\&[c_i\ 0] \supset [\text{LABEL}\ 0] \\ [\text{LEVEL}\ m+1]\&[c_1\ 1]\&[c_2\ 1]\&\ldots\&[c_{|w|/3}1] \supset [\text{LABEL}\ 1] \end{array}$$

The reduction constructs $O(|w|)$ features and $O(m^2)$ FCRs of size $O(\log m)$ in a simple manner, and consequently may be seen to be polynomial time. $\mathcal{Q.E.D}$

The primary source of intractability in the theory of syntactic features is the large number of possible syntactic categories (arising from finite feature closure) in combination with the computational power of feature co-occurrence restrictions.[8] FCRs of the "disjunctive consequence" form $[f\ \ v] \supset [f_1\ \ v_1] \lor \ldots \lor [f_n\ \ v_n]$ compute the direct analogue of Satisfiability: when used in conjunction with other FCRs, the GPSG effectively must try all n feature-value combinations.

3 Complexity of GPSG-Recognition

Two isolated membership problems for GPSG's component formal devices were considered above in an attempt to isolate sources of complexity in GPSG theory. In this section the recognition problem (RP) for GPSG theory as a whole is considered. I begin by arguing that the linguistically and computationally relevant recognition problem is the universal recognition problem, as opposed to the fixed language recognition problem. I then show that the former problem is exponential-polynomial (Exp-Poly) time-hard.

[8]Finite feature closure admits a surprisingly large number of possible categories. Given a specification $(F, \mathtt{Atom}, A, R, \rho)$ of K, let $a = |\mathtt{Atom}|$ and $b = |F - \mathtt{Atom}|$. Assume that all atomic features are binary: a feature may be +,-, or undefined and there are 3^a 0-level categories. The b category-valued features may each assume $O(3^a)$ possible values in a 1-level category, so $|K^1| = O(3^a(3^a)^b)$. More generally,

$$|K = K^b| = O(3^{(a\sum_{i=0}^{b}\frac{b!}{(b-i)!})}) = O(3^{(a\cdot b!\sum_{i=0}^{b}\frac{1}{i!})}) = O(3^{a\cdot b!\cdot e}) = O(3^{a\cdot b!})$$

where $\sum_{i=0}^{b}\frac{1}{i!}$ converges to $e \approx 2.7$ very rapidly and $a, b = O(|G|)$; $a = 25, b = 4$ in GKPS. The smallest category in K will be 1 symbol (null set), and the largest, maximally-specified, category will be of symbol-size $\log|K| = O(a \cdot b!)$.

3.1 Defining the Recognition Problem

The ***universal recognition problem*** is: given a grammar G and input string x, is $x \in L(G)$?. Alternately, the recognition problem for a class of grammars may be defined as the family of questions in one unkown. This ***fixed language recognition problem*** is: given an input string x, is $x \in L$ for some fixed language L?. For the fixed language RP, it does not matter which grammar is chosen to generate L — typically, the fastest grammar is picked.

It seems reasonable clear that the universal RP is of greater linguistic and engineering interest than the fixed language RP. The grammars licensed by linguistic theory assign structural descriptions to utterances, which are used to query and update databases, be interpreted semantically, translated into other human languages, and so on. The universal recognition problem — unlike the fixed language problem — determines membership with respect to a grammar, and therefore more accurately models the parsing problem, which must use a grammar to assign structural descriptions.

The universal RP also bears most directly on issues of natural language acquisition. The language learner evidently possesses a mechanism for selecting grammmars from the class of learnable natural language grammars $\mathcal{L}_G$ on the basis of linguistic inputs. The more fundamental question for linguistic theory, then, is "what is the recognition complexity of the class $\mathcal{L}_G$?". If this problem should prove computationally intractable, then the (potential) tractability of the problem for each language generated by a G in the class is only a partial answer to the linguistic questions raised.

Finally, complexity considerations favor the universal RP. The goal of a complexity analysis is to characterize the amount of computational resources (e.g. time, space) needed to solve the problem ***in terms of all computationally relevent inputs*** on some standard machine model (typically, a multi-tape deterministic Turing machine). We know that both input string length and grammar size and structure affect the complexity of the recognition problem. Hence, excluding either input from complexity consideration would not advance our understanding.[9]

Linguistics and computer science are primarily interested in the universal recognition problem because both disciplines are concerned with the formal power of a *family* of grammars. Linguistic competence and performance must be considered in the larger context of efficient language acquisition, while computational considerations demand that the recognition problem be characterized in terms of both input string and grammar size. Excluding grammar size from complexity consideration in order

[9]This "consider all relevant inputs" methodology is universally assumed in the formal language and computational complexity literature. For example, Hopcraft and Ullman(1979:139) define the context-free grammar recognition problem as: "Given a CFG $G = (V, T, P, S)$ and a string x in T^*, is x in $L(G)$?". Garey and Johnson(1979) is a standard reference work in the field of computational complexity. All 10 automata and language recognition problems covered in the book (pp. 265-271) are universal, i.e. of the form "Given an instance of a machine/grammar and an input, does the machine/grammar accept the input?" The complexity of these recognition problems is *always* calculated in terms of grammar and input size.

to argue that the recognition problem for a family of grammars is tractable is akin to fixing the size of the chess board in order to argue that winning the game of chess is tractable: neither claim advances our scientific understanding of chess or natural language.

3.2 GPSG-Recognition is Exp-Poly hard

Theorem 3: GPSG-Recognition is Exp-Poly time-hard

Proof 3: By direct simulation of a polynomial space bounded alternating Turing Machine M on input w.

Let $S(n)$ be a polynomial in n. Then, on input M, a $S(n)$ space-bounded one tape alternating Turing Machine (ATM), and string w, we construct a GPSG G in polynomial time such that $w \in L(M)$ iff $\$0w_1 1w_2 2 \ldots w_n n\$n{+}1 \in L(G)$.

By Chandra and Stockmeyer(1976),

$$ASPACE(S(n)) = \bigcup_{c>0} DTIME(c^{S(n)})$$

where $ASPACE(S(n))$ is the class of problems solvable in space $S(n)$ on an ATM, and $DTIME(F(n))$ is the class of problems solvable in time $F(n)$ on a deterministic Turing Machine. As a consequence of this result and our following proof, we have the immediate result that GPSG-Recognition is $DTIME(c^{S(n)})$-hard, for all constants c, or Exp-Poly time-hard.

An alternating Turing Machine is like a nondeterministic TM, except that some subset of its states will be referred to as ***universal states***, and the remainder as ***existential states***. A nondeterministic TM is an alternating TM with no universal states.[10]

The nodes of the ATM computation tree are represented by syntactic categories in K^0 — one feature for every tape square, plus three features to encode the ATM tape head positions and the current state. The reduction is limited to specifying a polynomial number of features in polynomial time; since these features are used to encode the ATM tape, the reduction may only specify polynomial space bounded ATM computations.

The ID rules encode the ATM $\mathtt{Next}_M()$ relation, i.e. $C \rightarrow \mathtt{Next}_M(C)$ for a universal configuration C. The reduction constructs an ID rule for every combination of possible head position, machine state, and symbol on the scanned tape square. Principles of universal feature instantiation transfer the rest of the instantaneous description (i.e. contents of the tape) from mother to daughters in ID rules.

[10]Our ATM definition is taken from Chandra and Stockmeyer(1976), with the restriction that the work tapes are one-way infinite, instead of two-way infinite. Without loss of generality, we use a 1-tape ATM, so

$$\delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma^k \times \{L,R\} \times \{L,R\})$$

Let $\mathtt{Next}_M(C) = \{C_0, C_1, \ldots, C_k\}$. If C is a universal configuration, then we construct an ID rule of the form

$$C \rightarrow C_0, C_1, \ldots, C_k \qquad (6)$$

Otherwise, C is an existential configuration and we construct the $k+1$ ID rules

$$C \rightarrow C_i \qquad \forall i,\ 0 \le i \le k \qquad (7)$$

A universal ATM configuration is labeled accepting if and only if it has halted and accepted, or if all of its daughters are labeled accepting. We reproduce this with the ID rules in 6 (or 8), which will be admissible only if all subtrees rooted by the RHS nodes are also admissible.

An existential ATM configuration is labeled accepting if and only if it has halted and accepted, or if one of its daughters is labeled accepting. We reproduce this with the ID rules in 7 (or 9), which will be admissible only if one subtree rooted by a RHS node is admissible.

All features that represent tape squares are declared to be in the HEAD feature set, and all daughter categories in the constructed ID rules are head daughters, thus ensuring that the Head Feature Convention (HFC) will transfer the tape contents of the mother to the daughter(s), modulo the tape writing activity specified by the next move relation.

Details.

<u>Let</u>

$$\mathtt{Result0}_M(i,a,d) = \begin{cases} [[\text{HEAD0}\ i+1],[i\ \ a],[A\ \ 1]] & \text{if } d = R \\ [[\text{HEAD0}\ i-1],[i\ \ a],[A\ \ 1]] & \text{if } d = L \end{cases}$$

$$\mathtt{Result1}_M(j,c,p,d) = \begin{cases} [[\text{HEAD1}\ j+1],[\Gamma_j\ \ c][\text{STATE}\ \ p]] & \text{if } d = R \\ [[\text{HEAD1}\ j-1],[\Gamma_j\ \ c][\text{STATE}\ \ p]] & \text{if } d = L \end{cases}$$

$$\mathtt{Trans}_M(q,a,b) = \{\langle p,c,d_1,d_2\rangle : \langle\langle q,a,b\rangle,\langle p,c,d_1,d_2\rangle\rangle \in \delta\}$$

where

a	is the read-only (R/O) tape symbol currently being scanned
b	is the read-write (R/W) tape symbol currently being scanned
d_1	is the R/O tape direction
d_2	is the R/W tape direction

The GPSG G contains:

1. Feature definitions

A category in K^0 represents a node of an ATM computation tree, where the features in *Atom* encode the ATM configuration. Labeling is performed by ID rules.

(a) definition of $F, Atom, A$

$$\begin{aligned} F = Atom &= \{\texttt{STATE}, \texttt{HEAD0}, \texttt{HEAD1}, A\} \\ &\cup \{i : 0 \le i \le |w| + 1\} \\ &\cup \{\Gamma_j : 1 \le j \le S(|w|)\} \\ A &= Q \cup \Sigma \cup \Gamma \quad \text{; as defined earlier} \end{aligned}$$

(b) definition of ρ^0

$$\begin{aligned} &\rho^0(A) = \{1,2,3\} \\ &\rho^0(\texttt{STATE}) = Q \quad \text{; the ATM state set} \\ &\rho^0(\texttt{HEAD0}) = \{j : 1 \le j \le |w|\} \\ &\rho^0(\texttt{HEAD1}) = \{i : 1 \le i \le S(|w|)\} \\ &\forall f \in \{i : 0 \le i \le |w| + 1\} \\ &\quad \rho^0(f) = \Sigma \cup \{\$\} \quad \text{; the ATM input alphabet} \\ &\forall f \in \{\Gamma_j : 1 \le j \le S(|w|)\} \\ &\quad \rho^0(f) = \Gamma \quad \text{; the ATM tape alphabet} \end{aligned}$$

(c) definition of **HEAD** feature set

$$\mathbf{HEAD} = \{i : 0 \le i \le |w| + 1\} \cup \{\Gamma_j : 1 \le j \le S(|w|)\}$$

(d) FCRs to ensure full specification of all categories except null ones.

$$\forall f; f \in Atom, \qquad [\texttt{STATE}] \supset [f]$$

2. Grammatical rules

$\forall i, j, q, a, b \ : 1 \le i \le |w|,\ 1 \le j \le S(|w|),\ q \in Q,\ a \in \Sigma,\ b \in \Gamma$

if $\texttt{Trans}_M(q, a, b) \ne \emptyset$, construct the following ID rules.

(a) if $q \in U$ (universal state)

$$\begin{aligned} &\{[\texttt{HEAD0}\ i], [i\ a], [\texttt{HEAD1}\ j], [\Gamma_j\ b], [\texttt{STATE}\ q], [A\ 1]\} \to \\ &\quad \{\texttt{Result0}_M(i, a, d_{1k}) \cup \texttt{Result1}_M(j, c_k, p_k, d_{2k}) : \\ &\quad\quad \langle p_k, c_k, d_{1k}, d_{2k} \rangle \in \texttt{Trans}_M(q, a, b)\} \end{aligned} \tag{8}$$

where all categories on the RHS are heads.

(b) otherwise $q \in Q - U$ (existential state)

$$\begin{aligned} &\forall \langle p_k, c_k, d_{1k}, d_{2k} \rangle \in \texttt{Trans}_M(q, a, b), \\ &\quad \{[\texttt{HEAD0}\ i], [i\ a], [\texttt{HEAD1}\ j], [\Gamma_j\ b], [\texttt{STATE}\ q], [A\ 1]\} \to \\ &\quad\quad \texttt{Result0}_M(i, a, d_{1k}) \cup \texttt{Result1}_M(j, c_k, p_k, d_{2k}) \end{aligned} \tag{9}$$

where all categories on the RHS are heads.

(c) One ID rule to terminate accepting states, using null-transitions.

$$\{[\texttt{STATE}\ h], [1\ \texttt{Y}]\} \to \epsilon \tag{10}$$

(d) Two ID rules to read input strings and begin the ATM simulation. The A feature is used to separate functionally distinct components of the grammar. [A 1] categories participate in the direct ATM simulation, [A 2] categories are involved in reading the input string, and the [A 3] category connects the read input string with the ATM simulation start state.

$$\begin{aligned} &\texttt{START} \to \{[A\ 1]\}, \{[A\ 2]\} \\ &\{[A\ 2]\} \to \{[A\ 2]\}, \{[A\ 2]\} \end{aligned} \tag{11}$$

where all daughters are head daughters, and where

$$\begin{aligned} \texttt{START} = &\ \{[\texttt{HEAD0}\ 1], [\texttt{HEAD1}\ 1], [\texttt{STATE}\ s], [A\ 3]\} \\ &\cup \{[\Gamma_j\ \#] : 1 \le j \le S(|w|)\} \end{aligned}$$

(e) the lexical rules,

$$\begin{aligned} &\forall \sigma, i \quad \sigma \in \Sigma,\ 1 \le i \le |w|, \\ &\quad < \sigma i, \{[A\ 2], [i\ \sigma]\} > \\ &\forall i \ \ 0 \le i \le |w| + 1, \\ &\quad < \$ i, \{[A\ 2], [i\ \$]\} > \end{aligned} \tag{12}$$

The reduction plainly may be performed in polynomial time in the size of the simulated ATM, by inspection.

No metarules or LP statements are needed, although metarules could have been used instead of the Head Feature Convention. Both devices are capable of transferring the contents of the ATM tape from the mother to the daughter(s). One metarule would be needed for each tape square/tape symbol combination in the ATM.

GKPS Definition 5.14 of Admissibility guarantees that admissible trees must be terminated.[11] By the construction above — see especially the ID rule 10 — an [A 1] node can be terminated only if it is an accepting configuration (i.e. it has halted and printed Y on its first square). This means the only admissible trees are accepting ones whose yield is the input string followed by a very long empty string. $\mathcal{Q.E.D}$

[11]The admissibility of nonlocal trees is defined as follows (GKPS, p.104):

Definition: Admissibility

Let R be a set of ID rules. Then a tree t is *admissible from R* if and only if

1. t is terminated, and
2. every local subtree in t is either terminated or locally admissible from some $r \in R$.

3.3 Sources of Intractability

The two sources of intractability in GPSG theory spotlighted by this reduction are null-transitions in ID rules (see the ID rule 10 above), and universal feature instantiation (in this case, the Head Feature Convention).

Grammars with unrestricted null-transitions can assign elaborate phrase structure to the empty string, which is linguistically undesirable and computationally costly. The reduction must construct a GPSG G and input string x in polynomial time such that $x \in L(G)$ iff $w \in L(M)$, where M is a PSPACE-bounded ATM with input w. The 'polynomial time' constraint prevents us from making either x or G too big. Null-transitions allow the grammar to simulate the PSPACE ATM computation (and an Exp-Poly TM computation indirectly) with an enormously long derivation string and then erase the string. If the GPSG G were unable to erase the derivation string, G would only accept strings which were exponentially larger than M and w, i.e. too big to write down in polynomial time.

The Head Feature Condition transfers HEAD feature values from the mother to the head daughters just in case they don't conflict. In the reduction we use HEAD features to encode the ATM tape, and thereby use the HFC to transfer the tape contents from one ATM configuration C (represented by the mother) to its immediate successors $C_0, \ldots, C_n$ (the head daughters). The configurations $C, C_0, \ldots, C_n$ have identical tapes, with the critical exception of one tape square. If the HFC enforced absolute agreement between the HEAD features of the mother and head daughters, we would be unable to simulate the PSPACE ATM computation in this manner.

4 Interpreting the Result

4.1 Generative Power and Computational Complexity

At first glance, a proof that GPSG-Recognition is Exp-Poly hard appears to contradict the fact that context-free languages can be recognized in $O(n^3)$ time by a wide range of algorithms. To see why there is no contradiction, we must first explicitly state the argument from weak context-free generative power, which we dub the efficient parsability (EP) argument.

The *EP argument* states that any GPSG can be converted into a weakly equivalent context-free grammar (CFG), and that CFG-Recognition is polynomial time; therefore, GPSG-Recognition must also be polynomial time. The EP argument continues: if the conversion is fast, then GPSG-Recognition is fast, but even if the conversion is slow, recognition using the "compiled" CFG will still be fast, and we may justifiably lose interest in recognition using the original, slow, GPSG.

The EP argument is misleading because it ignores both the effect conversion has on grammar size, and the effect grammar size has on recognition speed. Crucially, grammar size affects recognition time in all known algorithms, and the only grammars *directly* usable by context-free parsers, i.e. with the same complexity as a CFG, are those composed of context-free productions with atomic nonterminal symbols. For GPSG, this is the set of admissible local trees, and this set is astronomical:

$$O((3^{m!m^{2m+1}}) \qquad (13)$$

in a GPSG G of size m.[12]

Context-free parsers like the Earley algorithm run in time $O(|G'|^2 \cdot n^3)$ where $|G'|$ is the size of the CFG G' and n the input string length, so a GPSG G of size m will be recognized in time

$$O(3^{2\cdot m!m^{2m+1}} \cdot n^3) \qquad (14)$$

The hyper-exponential term will dominate the Earley algorithm complexity in the reduction above because m is a function of the size of the ATM we are simulating. Even if the GPSG is held constant, the stunning derived grammar size in formula 13 turns up as an equally stunning 'constant' multiplicative factor in 14, which in turn will dominate the real-world performance of the Earley algorithm for all expected inputs (i.e. any that can be written down in the universe), every time we use the derived grammar.[13]

Pullum(1985) has suggested that "examination of a suitable 'typical' GPSG description reveals a ratio of only 4 to 1 between expanded and unexpanded grammar statements," strongly implying that GPSG is efficiently processable as a consequence.[14] But this "expanded grammar" is not adequately expanded, i.e. it is not composed of context-free productions with unanalyz-

[12]As we saw above, the metarule finite closure operation can increase the ID rule grammar size from $|R| = O(|G|)$ to $O(m^{2m})$ in a GPSG G of size m. We ignore the effects of ID/LP format on the number of admissible local trees here, and note that if we expanded out all admissible linear precedence possibilities in *FC(M,R)*, the resultant 'ordered' ID rule grammar would be of size $O(m^{2m}!)$. In the worst case, every symbol in *FC(M,R)* is underspecified, and every category in K extends every symbol in the *FC(M,R)* grammar. Since there are

$$O(3^{m\cdot m!})$$

possible syntactic categories, and $O(m^{2m})$ symbols in *FC(M,R)*, the number of admissible local trees (= atomic context-free productions) in G is

$$O((3^{m\cdot m!})^{m^{2m}}) = O(3^{m!m^{2m+1}})$$

i.e. astronomical. Ristad(1986) argues that the minimal set of admissible local trees in GKPS' GPSG for English is considerably smaller, yet still contains more than 10^{20} local trees.

[13]The compiled grammar recognition problem is at least as intractable as the uncompiled one. Even worse, Barton(1985) shows how the grammar expansion increases both the space *and* time costs of recognition, when compared to the cost of using the grammar directly.

[14]This *substantive* argument is somewhat strange coming from a co-author of a book which advocates the purely *formal* investigation of linguistics: "The universalism [of natural language] is, ultimately, intended to be entirely embodied in the formal system, not expressed by statements made in it."GKPS(4). It is difficult to respond precisely to the claims made in Pullum(1985), since the abstract is (necessarily) brief and consists of assertions unsupported by factual documentation or clarifying assumptions.

able nonterminal symbols.[15] These informal tractability arguments are a particular instance of the more general EP argument and are equally misleading.

The preceding discussion of how intractability arises when converting a GPSG into a weakly equivalent CFG does not in principle preclude the existence of an efficient compilation step. If the compiled grammar is truly fast and assigns the same structural descriptions as the uncompiled GPSG, and it is possible to compile the GPSG in practice, then the complexity of the universal recognition problem would not accurately reflect the real cost of parsing.[16] But until such a suggestion is forthcoming, we must assume that it does not exist.[17,18]

[15]"Expanded grammar" appears to refer to the output of metarule finite closure (i.e. ID rules), and this expanded grammar is tractable only if the grammar is directly usable by the Earley algorithm exactly as context-free productions are: all nonterminals in the context-free productions must be unanalyzable. But the categories and ID rules of the metarule finite closure grammar do not have this property. Nonterminals in GPSG are decomposable into a complex set of feature specifications and cannot be made atomic, in part because not all extensions of ID rule categories are legal. For example, the categories -0001VP[+VFORM PAS] and VP[+INV, VFORM FIN] are not legal extensions of VP in English, while VP[+INV, +AUX, VFORM FIN] is. FCRs, FSDs, LP statements, and principles of universal feature instantiation — all of which contribute to GPSG's intractability — must all still apply to the rules of this expanded grammar.

Even if we ignore the significant computational complexity introduced by the machinery mentioned in the previous paragraph (i.e. theory of syntactic features, FCRs, FSDs, ID/LP format, null-transitions, and metarules), *GPSG will still not obtain an efficient parsability result.* This is because the Head Feature Convention alone ensures that the universal recognition problem for GPSGs will be NP-hard and likely to be intractable. Ristad(1986) contains a proof. This result should not be surprising, given that (1) principles of universal feature instantiation in current GPSG theory replace the metarules of earlier versions of GPSG theory, and (2) metarules are known to cause intractability in GPSG.

[16]The existence or nonexistence of efficient compilation functions does not affect either our scientific interest in the universal grammar recognition problem or the power and relevance of a complexity analysis. If complexity theory classifies a problem as intractable, we learn that something more must be said to obtain tractability, and that any efficient compilation step, if it exists at all, must itself be costly.

[17]Note that the GPSG we constructed in the preceding reduction will actually accept *any* input x of length less than or equal to $|w|$ if and only if the ATM M accepts it using $S(|w|)$ space. We prepare an input string x for the GPSG by converting it to the string $\$0x_1 1x_2 2 \ldots x_n n\$n+1$ e.g. *abadee* is accepted by the ATM if and only if the string $\$0a1b2a3d4e5e6\$7$ is accepted by the GPSG. Trivial changes in the grammar allows us to permute and "spread" the characters of x across an infinite class of strings in an unbounded number of ways, e.g. $\$0\gamma_1 x_i i \gamma_2 \ldots \gamma_a x_1 1 \gamma_b \ldots \gamma_n \$n+1$ where each γ_i is a string over an alphabet which is distinct from the σi alphabet. Although the flexibility of this construction results in a more complicated GPSG, it argues powerfully against the existence of any efficient compilation procedure for GPSGs. Any efficient compilation procedure must perform more than an exponential polynomial amount of work (GPSG-Recognition takes at least Exp-Poly time) on at least an exponential number of inputs (all inputs that fit in the $|w|$ space of the ATM's read-only tape). More importantly, the required compilation procedure will convert *any* exponential-polynomial time bounded Turing Machine into a polynomial-time TM for the class inputs whose membership can be determined within a arbitrary (fixed) exp-poly time bound. Simply listing the accepted inputs will not work because both the GPSG and TM may accept an infinite class of inputs. Such a compilation procedure would be extremely powerful.

[18]Note that compilation illegitimately assumes that the compilation step

4.2 Complexity and Succinctness

The major complexity result of this paper proves that the fastest algorithm for GPSG-Recognition must take more than exponential time. The immediately preceding section demonstrates exactly how a particular algorithm for GPSG-Recognition (the EP argument) comes to grief: weak context-free generative power does not ensure efficient parsability because a GPSG G is weakly equivalent to a very large CFG G', and CFG size affects recognition time. The rebuttal does *not* suggest that computational complexity arises from representational succinctness, either here or in general.

Complexity results characterize the amount of resources needed to solve instances of a problem, while succinctness results measure the space reduction gained by one representation over another, equivalent, representation.

There is no casual connection between computational complexity and representational succinctness, either in practice or principle. In practice, converting one grammar into a more succinct one can either increase or decrease the recognition cost. For example, converting an instance of context-free recognition (known to be polynomial time) into an instance of context-sensitive recognition (known to be PSPACE-complete and likely to be intractable) can significantly speed the recognition problem if the conversion decreases the size of the CFG logarithmically or better. Even more strangely, increasing ambiguity in a CFG can speed recognition time if the succinctness gain is large enough, or slow it down otherwise — unambiguous CFGs can be recognized in linear time, while ambiguous ones require cubic time.

In principle, tractable problems may involve succinct representations. For example, the iterating coordination schema (ICS) of GPSG is an unbeatably succinct encoding of an infinite set of context-free rules; from a computational complexity viewpoint, the ICS is utterly trivial using a slightly modified Earley algorithm.[19] Tractable problems may also be verbosely represented: consider a random finite language, which may be recognized in essentially constant time on a typical computer (using a hash table), yet whose elements must be individually listed. Similarly, intractable problems may be represented both succinctly and nonsuccinctly. As is well known, the Turing machine for any arbitrary r.e. set may be either extremely small or monstrously big. Winning the game of chess when played on an $n \times n$ board is likely to be computationally intractable, yet the chess board is not intended to be an encoding of another representation, succinct or otherwise.

is free. There is one theory of primitive language learning and use: conjecture a grammar and use it. For this procedure to work, grammars should be easy to test on small inputs. The overall complexity of learning, testing, and speech must be considered. Compilation speeds up the speech component at the expense of greater complexity in the other two components. For this linguistic reason the compilation argument is suspect.

[19]A more extreme example of the unrelatedness of succinctness and complexity is the absolute succinctness with which the dense language Σ^* may be represented — whether by a regular expression, CFG, or even Turing machine — yet members of Σ^* may be recognized in constant time (i.e. always accept).

Tractable problems may involve succinct or nonsuccinct representations, as may intractable problems. The reductions in this paper show that GPSGs are not merely succinct encodings of some context-free grammars; they are inherently complex grammars for some context-free languages. The heart of the matter is that GPSG's formal devices are computationally complex and can encode provably intractable problems.

4.3 Relevance of the Result

In this paper, we argued that *there is nothing in the GPSG formal framework that guarantees computational tractability:* proponents of GPSG must look elsewhere for an explanation of efficient parsability, if one is to be given at all. The crux of the matter is that the complex components of GPSG theory interact in intractable ways, and that weak context-free generative power does not guarantee tractability when grammar size is taken into account. A faithful implementation of the GPSG formalisms of GKPS will provably be intractable; expectations computational linguistics might have held in this regard are not fulfilled by current GPSG theory.

This formal property of GPSGs is straightforwardly interesting to GPSG linguists. As outlined by GKPS, "an important goal of the GPSG approach to linguistics [is] the construction of theories of the structure of sentences under which significant properties of grammars and languages fall out as theorems as opposed to being stipulated as axioms (p.4)."

The role of a computational analysis of the sort provided here is fundamentally positive: it can offer significant formal insights into linguistic theory and human language, and suggest improvements in linguistic theory and real-world parsers. The insights gained may be used to revise the linguistic theory so that it is both stronger linguistically and weaker formally. Work on revising GPSG is in progress. Briefly, some proposed changes suggested by the preceding reductions are: unit feature closure, no FCRs or FSDs, no null-transitions in ID rules, metarule unit closure, and no problematic feature specifications in the principles of universal feature instantiation. Not only do these restrictions alleviate most of GPSG's computational intractability, but they increase the theory's linguistic constraint and reduce the number of nonnatural language grammars licensed by the theory. Unfortunately, there is insufficient space to discuss these proposed revisions here — the reader is referred to Ristad(1986) for a complete discussion.

Acknowledgments. Robert Berwick, Jim Higginbotham, and Richard Larson greatly assisted the author in writing this paper. The author is also indebted to Sandiway Fong and David Waltz for their help, and to the MIT Artificial Intelligence Lab and Thinking Machines Corporation for supporting this research.

5 References

Barton, G.E. (1985). "On the Complexity of ID/LP Parsing," *Computational Linguistics,* 11(4): 205-218.

Chandra, A. and L. Stockmeyer (1976). "Alternation," 17th *Annual Symposium on Foundations of Computer Science,*: 98-108.

Gazdar, G. (1981). "Unbounded Dependencies and Coordinate Structure," *Linguistic Inquiry* 12: 155-184.

Gazdar, G., E. Klein, G. Pullum, and I. Sag (1985). *Generalized Phrase Structure Grammar.* Oxford, England: Basil Blackwell.

Garey, M, and D. Johnson (1979). *Computers and Intractability.* San Francisco: W.H. Freeman and Co.

Hopcroft, J.E., and J.D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation.* Reading, MA: Addison-Wesley.

Pullum, G.K. (1985). "The Computational Tractability of GPSG," Abstracts of the 60th Annual Meeting of the Linguistics Society of America, Seattle, WA: 36.

Ristad, E.S. (1985). "GPSG-Recognition is NP-hard," A.I. Memo No. 837, Cambridge, MA: M.I.T. Artificial Intelligence Laboratory.

Ristad, E.S. (1986). "Complexity of Linguistic Models: A Computational Analysis and Reconstruction of Generalized Phrase Structure Grammar," S.M. Thesis, MIT Department of Electrical Engineering and Computer Science. (In progress).