The Generative Power of Categorial Grammars and Head-Driven Phrase Structure Grammars with Lexical Rules

Bob Carpenter*
Carnegie Mellon University

In this paper, it is shown that the addition of simple and linguistically motivated forms of lexical rules to grammatical theories based on subcategorization lists, such as categorial grammars (CG) or head-driven phrase structure grammars (HPSG), results in a system that can generate all and only the recursively enumerable languages. The proof of this result is carried out by means of a reduction of generalized rewriting systems. Two restrictions are considered, each of which constrains the generative power of the resulting system to context-free languages.

1. Introduction

In recent grammatical theories, there has been an increasing trend toward the lexicalization of syntactic information. This is particularly evident in the case of categorial grammars (CG) and head-driven phrase structure grammars (HPSG), where a small number of highly schematic syntactic rules are assumed to apply universally. With this assumption of a universal syntax, the task of explaining the variations between languages must be carried out in the lexicon. Rather than assuming that the lexicon is simply an unstructured list associating words with syntactic categories, organization is usually imposed by means of hierarchical inheritance systems, linking theories relating lexical semantics to grammatical categories and finally with lexical rules. It is the lexical rule component of these grammars that we investigate in this paper.

We present a straightforward formalization of categorial grammars with lexical rules based in part on the systems of Dowty (1978, 1979), Bach (1984), Keenan and Faltz (1985), Keenan and Timberlake (1988), Hoeksema and Janda (1988), and the HPSG lexical rule systems of Flickinger et al. (1985), Flickinger (1987), and Pollard and Sag (1987). We show that even using such a simple form of lexical rules, any recursively enumerable language can be recognized. Thus the addition of lexical rules leads to systems in which it is not possible to effectively decide whether a string is accepted by a grammar. We first introduce a pared-down formalism that captures the way in which heads combine with complements in CG and HPSG. We then provide examples to motivate a very straightforward and natural system of lexical rules.

To show that arbitrary recursively enumerable languages may be generated by the resulting system, we provide a reduction of generalized rewriting systems (Type 0 grammars). Though the details of our reduction are different, our method is reminiscent of that used by Uszkoreit and Peters (1986) to show that context-free grammars with metarules could generate arbitrary recursively enumerable languages. The analogy between CG lexical rules and GPSG metarules is strengthened by the fact that

* Computational Linguistics Program, Philosophy Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA; e-mail: carp@lcl.cmu.edu

GPSG as presented in Gazdar et al. (1985) restricts the application of metarules to lexical phrase structure rules.

Along the way to proving that categorial grammars with lexical rules can generate arbitrary recursively enumerable languages, we consider two restrictions that have the effect of reducing the generative power of the system to context-free languages. The first of these restrictions limits the recursive application of lexical rules, while the second puts a bound on the number of complements that can be specified by a category.

2. CGs and HPSGs

In Generalized Phrase Structure Grammar (GPSG) as presented in Gazdar et al. (1985), each lexical head category is assigned a simple integer subcategorization value. Similarly, each lexical phrase structure rule specifies the possible value(s) of the subcategorization feature occurring on its head daughter. In both CG and HPSG, the subcategorization value and correspondingly indexed phrase structure rule are replaced with a lexical encoding of head/complement structure by means of a subcategorization or complement list. The exact characterization of the notion of head has been the subject of some debate, but which items are labeled as heads is not important here; we use the term **head** to refer to any category that specifies its complements. Thus heads in our sense may be categories traditionally classified as specifiers or adjuncts. The subcategorization list of a category determines the number, form, and order of its complements. The only syntactic rule scheme that we consider is one that allows a head to combine with a complement. The result of such a construction is a category just like the head, only with the complement removed from its subcategorization list. Many extended categorial grammar systems and the HPSG system allow more sophisticated rules than this to deal with adjuncts, coordination, and long-distance dependencies, but we only need the simple head-complement construction to show that the addition of lexical rules leads to undecidability.

We now turn to a more formal presentation of a framework that incorporates the core of both CG and HPSG. We begin with a finite set BasCat of **basic categories** out of which complex categories are constructed. The collection of basic categories is usually assumed to contain "saturated" syntactic categories, including nouns, noun phrases, sentences, and so forth, but the choice of basic categories is ultimately up to the grammar writer. We are not concerned with the details of any particular syntactic analysis here. The full set Cat of **categories** is defined to be the least collection of objects of the form $b[c_0, \ldots, c_{n-1}]$ where $b \in \mathsf{BasCat}$ is a basic category and $c_i \in \mathsf{Cat}$ for $i < n$. The categories between the brackets specify the **arguments** of the category, with the assumption being that these arguments must be attached in the order in which they occur on the subcategorization list. While our notation follows the usage of HPSG and Unification Categorial Grammar (Calder et al. 1988), it should be clear how it relates to more traditional categorial grammar notation (see Bar-Hillel et al. 1960).

We assume the following schematic phrase structure rule that allows heads to combine with a single complement:

Definition 1

$$b[c_1, \ldots, c_n] \quad \longrightarrow \quad b[c_0, \ldots, c_n] \quad c_0$$

In most categorial grammars and in HPSG, there are rules in which the complement category precedes the head, but we do not even need this much power. For instance, a simple transitive verb might be given the category $s[np[], np[]]$, a noun phrase the

category $np[]$, a noun $n[]$, and a determiner the category $np[n[]]$. Thus the rule instance $s[np[]] \longrightarrow s[np[], np[]]\ \ np[]$ allows the combination of a transitive verb with an object noun phrase, while the rule $np[] \longrightarrow np[n[]]\ \ n[]$ would allow a determiner followed by a noun to be analyzed as a noun phrase.

Let Rule be the set of all instances of the schematic rule applied to Cat. Note that this set is totally determined by the choice BasCat of basic categories.

A **lexicon** for our simple grammar formalism consists of a finite relation between categories and basic expressions:

Definition 2
$\mathsf{Lex} \subseteq \mathsf{BasExp} \times \mathsf{Cat}$.

We write $e := c$ if $\langle e, c \rangle \in \mathsf{Lex}$. There is no restriction preventing a single expression from having more than one lexical entry. What we have with Lex and Rule is a simple phrase structure grammar, albeit one with an infinite set of rules. We interpret admissibility (well-formedness or grammaticality) in this phrase structure grammar in the usual way (see Hopcroft and Ullman 1979:79–87). In particular, we take $\mathcal{L}_{\mathsf{Lex}}(c)$ to be the set of strings of basic expressions of category c. By mutual recursion over Cat we define the $\mathcal{L}_{\mathsf{Lex}}(c)$ as the minimal sets such that:

Definition 3

- $e \in \mathcal{L}_{\mathsf{Lex}}(c)$ if $\langle e, c \rangle \in \mathsf{Lex}$
- $e_1 e_2 \in \mathcal{L}_{\mathsf{Lex}}(c[c_1, \ldots, c_n])$ if $e_1 \in \mathcal{L}_{\mathsf{Lex}}(c[c_0, c_1, \ldots, c_n])$ and $e_2 \in \mathcal{L}_{\mathsf{Lex}}(c_0)$.

It should be fairly obvious at this point how our simple categorial grammar formalism represents the core of both categorial grammars and the subcategorization component of HPSG. It should also be noted that such a grammar and lexical assignment reduces to a context-free grammar. This is because only finitely many subcategories of the lexical categories may ever be used in a derivation and thus only finitely many instances of the application scheme are necessary for a finite categorial lexicon. Somewhat surprisingly, the converse result also holds (Bar-Hillel et al. 1960); every context-free language is generated by some categorial grammar in the form we have presented. This latter result can be deduced from the fact that every context-free language can be expressed with a Greibach normal form grammar where every production is of the form $C_0 \longrightarrow aC_1 \cdots C_n$ where $n \geq 0$, the C_i are nonterminal category symbols, and a is a terminal expression (see Hopcroft and Ullman 1979). Taking a basic category for every nonterminal of the context-free grammar and a lexical entry of the form $a := C_0[C_1[], \ldots, C_n[]]$ for every production of the above form in the context-free grammar, we produce a categorial grammar that can be shown by induction on derivation length to generate exactly the same language as the Greibach normal form context-free grammar.

Not only is recognition decidable for context-free languages, but Earley's (1970) algorithm is known to decide them in $\mathcal{O}(n^3)$ time where n is the length of the input string (in fact, general CFG parsing algorithms can be constructed from matrix multiplication algorithms with slightly better worst-case asymptotic performance than Earley's algorithm [Valiant 1975]). Unfortunately, the situation is quite different after the addition of lexical rules.

Before going on, it should be noted that one of the primary reasons for employing categorial grammars is its natural relation to a compositional semantics (Montague 1970). While we do not consider the semantic effects of lexical rules here, the interested

reader is urged to consult Carpenter (1991) for details of how a higher-order typed semantics can be naturally added to the system of lexical rules presented below.

3. Adding Lexical Rules

The form of lexical rules that we propose to add to the basic categorial system are what Keenan and Faltz (1985) have termed **valency affecting operations**. These operations allow the permutation, addition, or subtraction of complements and the modification of the head or functor category. Our operations do not have any overt morphological effects, and are thus often referred to as **zero morphemes**. The same general lexical rule format has been proposed by virtually everyone considering the lexicon from a categorial perspective (see Dowty 1978, 1979; Bach 1984; Keenan and Faltz 1985; Keenan and Timberlake 1988; and Hoeksema and Janda 1988). Moortgat (1987) and Aone and Wittenburg (1990) have presented systems that allow extended categorial grammars to operate at both the morphological and syntactic levels, but the operations that can be carried out by their extended sets of rules produce results very similar to the lexical operations we allow here. Our results are especially relevant in light of recent work in HPSG, which admits lexical rules that do the same work as the ones employed here (see Flickinger 1987 and Pollard and Sag 1987). In the tradition of Montague, Dowty (1979) allowed arbitrary well-defined operations to be applied to lexical entries, but none of the rules he considered fall outside of the scope of the system presented here. The lexical generalizations studied by Bach (1984) led him to employ lexical redundancy rules that are expressed by composing basic functions that pick out the head or tail of a subcategorization list as well as the subcategorization list with either the head or tail removed. More formally, Bach allowed arbitrary concatenations and compositions of the following functions to be applied to subcategorization lists:

Definition 4

$$\begin{aligned} \text{FIRST}(x_1 \cdots x_n) &= x_1 \\ \text{RREST}(x_1 \cdots x_n) &= x_2 \cdots x_n \\ \text{LAST}(x_1 \cdots x_n) &= x_n \\ \text{LREST}(x_1 \cdots x_n) &= x_1 \cdots x_{n-1} \end{aligned}$$

After we present our lexical rule system, it should be obvious that Bach's system allows exactly the same operations to be expressed as we do. The only difference is that we recast Bach's functional rules in terms of simple pattern-matching. It is conjectured in Hoeksema and Janda (1988) that the resulting system is a proper subset of the context-sensitive languages. We show here that this conjecture could not be further from the truth, as all of the recursively enumerable languages can be generated using these operations. Keenan and Timberlake (1988) also present a collection of lexical redundancy rules that would seem to admit the system presented here as a natural generalization. In particular, they present an analysis of passive almost identical to the one presented below.

The restriction placed on the form of our rules is similar to the restriction on GPSG metarules; we only allow one variable over sequences of categories (see Gazdar et al. 1985). This allows the manipulation of arguments a specified distance from either end of the subcategorization list, but not arbitrary arguments. With obliqueness specified in terms of subcategorization order (see Dowty 1982 and Pollard and Sag 1987), lexical rules are able to specify operations on arguments based on their obliqueness. In

general, a lexical rule is of the form:

Definition 5
$b[c_0, \ldots, c_n, \$, d_0, \ldots, d_m] \Rightarrow b'[c'_0, \ldots, c'_{n'}, \$, d'_0, \ldots, d'_{m'}]$

where \$ is taken to be a variable ranging over sequences of categories (as in Ades and Steedman 1982). More precisely, we assume that the lexical rules are given by a finite relation:

Definition 6
$\mathsf{LexRule} \subseteq (\mathsf{BasCat} \times \mathsf{Cat}^* \times \mathsf{Cat}^*) \times (\mathsf{BasCat} \times \mathsf{Cat}^* \times \mathsf{Cat}^*)$.

Thus a rule of the form in (5) would be formally represented as:

Definition 7
$\langle\langle b, \langle c_0, \ldots, c_n\rangle, \langle d_0, \ldots, d_m\rangle\rangle, \langle b', \langle c'_0, \ldots, c'_{n'}\rangle, \langle d'_0, \ldots, d'_{m'}\rangle\rangle\rangle \in \mathsf{LexRule}$.

The intended interpretation of an element of LexRule is that if a basic expression is assigned to a category that matches its left-hand side, then it is also assigned to a category that matches its right-hand side. We generate the final (possibly infinite) lexicon LexRule(BasLex) by closing the basic lexicon under the lexical rules. More formally, we define LexRule(BasLex) to be the least relation such that:

Definition 8

- $\mathsf{BasLex} \subseteq \mathsf{LexRule}(\mathsf{BasLex})$, and
- if $\sigma := b[c_0, \ldots, c_n, e_0, \ldots, e_k, d_0, \ldots, d_m] \in \mathsf{LexRule}(\mathsf{BasLex})$ and $b[c_0, \ldots, c_n, \$, d_0, \ldots, d_m] \Rightarrow b'[c'_0, \ldots, c'_{n'}, \$, d'_0, \ldots, d'_{m'}] \in \mathsf{LexRule}$ then $\sigma := b'[c'_0, \ldots, c'_{n'}, e_0, \ldots, e_k, d'_0, \ldots, d'_{m'}] \in \mathsf{LexRule}(\mathsf{BasLex})$.

Thus, a grammar is determined by the specification of finite sets BasLex and LexRule, over some given finite sets BasExp and BasCat of basic expressions and categories. In the undecidability proof that follows, it is only really necessary to consider lexical rules that modify the subcategorization list; a slightly modified proof goes through if lexical rules are prohibited from affecting the basic head category.

We now take the time to motivate the full power of this lexical rule system. A standard example of the application of a lexical rule is passivization, which in our system can be stated in the form:

Example 9
$s[np[]_1, \$, np[]_2] \Rightarrow s[\$, pp_{by}[]_2, np[]_1]$ (Passive)

The intuitive reading of this rule is that the first argument of any verb can become the last argument (subject) and the last argument becomes a prepositional *by*-phrase; the subscripts, while not an actual part of the rule, indicate this swapping of arguments and their syntactic markings. For instance, with a ditransitive verb category such as $s[np[]_1, np[]_2, np[]_3]$, the result of applying the passive rule would be $s[np[]_2, pp_{by}[]_3, np[]_1]$. While we cannot account for the fact that the subject occurs before the verb rather than after it in the simplified system presented here, this category reflects the fact that the number, order, and category of arguments is permuted as a result of passivization. With the stripped-down system presented here, a second lexical rule is required for

passives without the prepositional argument. An operation such as detransitivization can be stated by the rule:

Example 10
$s[np[], \$] \Rightarrow s[\$]$ (Detransitivization)

The effect of detransitivization is simply to remove the most oblique argument from the subcategorization list. A nominalization rule might be stated in the form:

Example 11
$s[\$] \Rightarrow n[\$]$ (Nominalization)

Applying nominalization to a verbal lexical entry produces a nominal lexical entry with the same arguments. We can capture causative verbs with the following lexical rule:

Example 12
$s[\$] \Rightarrow s[\$, np[]]$ (Causative)

The causative rule adds another noun phrase argument for the subject in the least oblique position and increases the obliqueness of the existing arguments. A rule for dative shift could be expressed as:

Example 13
$s[np[]_1, pp_{dat}[]_2, np_3[]] \Rightarrow s[np_2[], np_1[], np_3[]]$ (Dative)

The point of these examples is that very simple lexical regularities such as passives, causatives, and detransitives motivate a system of lexical rules that switch, add, and delete elements from the subcategorization lists of lexical entries.

Lexical rules of the form we have here could also be used for what are traditionally considered to be syntactic operations, such as a rule for headless relatives (see Carpenter 1991):

Example 14
$s[\$, np[]] \Rightarrow n[\$, n[]]$ (Headless Relativization)

This rule has the effect of transforming a verbal category into a nominal modifier category with the same complements. Of course, to achieve the effect we desire, we must be able to mark verbs for their inflectional form and number. It should be apparent from these examples that the formalization of lexical rules presented here is particularly simple and motivated by a wide variety of seemingly lexical regularities. Further applications of lexical rules of the form we use here may be found in Dowty (1978, 1979), Bach (1984), Hoeksema and Janda (1988), Keenan and Timberlake (1988), Flickinger (1987), and Pollard and Sag (1987).

4. Finite Closure

We will now consider a restriction on the application of lexical rules that has the result of restricting the resulting system to a finite set of lexical entries. In the context of GPSG, Thompson (1982) restricted metarules to be nonrecursive so that they could not apply to rules that they had a hand in generating. This means that starting with a finite set of rules and metarules, only a finite number of rules would result. A similar

restriction allows Aone and Wittenburg (1990) to pre-compile the results of closing a categorial lexicon under a set of morphological operations. Let FinClos(LexRule,BasLex) be the **finite closure** of the set of basic lexical entries under the metarules with the restriction that no rule can apply to its own output. More formally, we define FinClos(LexRule,BasLex) inductively to be the least relation such that:

Definition 15

- BasLex $\subseteq$ FinClos(LexRule, BasLex)
- If $e := c \in$ FinClos(LexRule $- R$, BasLex) and $R \in$ LexRule maps c to c' then $e := c' \in$ FinClos(LexRule, BasLex).

This restriction ensures that a lexical rule never applies to its own output, as a lexical rule R can only be applied to a lexical entry that is derived without application of R. We thus have:

Theorem 1 (Finite Closure)
If LexRule and BasLex are finite
then FinClos(LexRule,BasLex) is finite.

Proof
Trivial by induction on the cardinality of LexRule. ■

The force of this result is that if we are willing to restrict our lexical rules to nonrecursive applications, then we have a finite lexicon and hence generate only a context-free language. But Carpenter (1991) argues for recursive lexical rules and provides examples from the English verbal system for which recursive rule application seems necessary.

5. Argument Complexity Bounds

Before moving on to the undecidability result, we define a notion of category complexity and show that the result of closing a finite lexicon under a finite number of lexical rules leads to a lexicon for which there is an upper bound on the complexity of the complements in lexical categories. Placing an upper bound on the complexity of entire categories in the lexicon restricts the system to context-free languages and ensures decidability.

Our complexity metric is based purely on the number of complements for which a category subcategorizes and not the complexity of the complements themselves. The **complexity** of a category $b[c_1, \ldots, c_n]$ is defined to be n, which we write as:

Definition 16
$Comp(b[c_1, \ldots, c_n]) = n$.

While it is possible to have a lexical rule such as $b[\$] \Rightarrow b[c, \$]$, which allows the derivation of categories of unbounded complexity, the complements of categories derived by lexical rules are bounded in their complexity.

Theorem 2
For any finite base lexicon BasLex and finite set LexRule of lexical rules, there is a bound k such that if $e := b[c_1, \ldots, c_n] \in$ LexRule(BasLex) then $Comp(c_i) < k$ for $1 \leq i \leq n$.

Proof
Because BasLex is finite, there is an upper bound on argument complexity for any complement category assigned by BasLex. Similarly, since there are only a finite number of rules in LexRule there is an upper bound on the complexity of complements specified in the outputs of any rule. Taken together, these facts imply that there is a bounded complement complexity in the result of closing BasLex under LexRule, since any complement category assigned by LexRule(BasLex) must have been a complement category either in BasLex or in the complement list of one of the output rules in LexRule. ■

This result shows that the lexical rules cannot modify the structure of complements other than by completely replacing them with one of a finite number of alternatives.

We now note that if we restrict the complexity of lexical categories themselves, we wind up with a context-free grammar. Let LexRule(BasLex)(n) be the set of lexical entries with categories of complexity less than or equal to n, so that $\langle e, c\rangle \in$ LexRule(BasLex)(n) if and only if $\langle e, c\rangle \in$ LexRule(BasLex)(n) and $Comp(c) \leq n$.

Theorem 3
The language generated by LexRule(BasLex)(n) is context-free.

Proof
Using the previous theorem, we know that there is a bound k on the size of the complements in any lexical entry in LexRule(BasLex), and thus there must only be a finite number of lexical entries with complexity of less than or equal to n. Consequently, LexRule(BasLex)(n) is finite and thus a standard finitary categorial grammar lexicon that generates a context-free language. ■

6. Generative Power

As we said in the introduction, we characterize the generative power of our system by the reduction of generalized rewriting systems to our head-complement grammars with lexical rules. Before doing this, we review the basic definition of a generalized rewriting system. A **generalized rewriting system** is a quadruple $G = \langle V, s, T, R\rangle$ where V is a finite set of **nonterminal** category symbols, $s \in V$ is the **start** symbol, T is a set of **terminal** symbols, and $R \subseteq (V^* \times V^*) \cup (V \times T)$ is a finite set of **rewriting** rules and **lexical insertion** rules, which are usually expressed in the forms:

Definition 17

- $v_1 \cdots v_n \longrightarrow u_1 \cdots u_m$ where $v_i, u_j \in V$
- $v \longrightarrow t$ where $v \in V$ and $t \in T$.

String rewriting is defined so that:

Definition 18
$\rho\sigma\rho' \longrightarrow \rho\tau\rho'$

if $\rho, \rho' \in (V \cup T)^*$ and if $(\sigma \longrightarrow \tau) \in R$ is a rule. The language $\mathbf{L}(G)$ generated by a general rewriting system G is defined to be

Definition 19
$\mathbf{L}(G) = \{\sigma \in T^* \mid s \xrightarrow{*} \sigma\}$

where s is the start symbol and $\stackrel{*}{\longrightarrow}$ is the usual transitive closure of the $\longrightarrow$ relation. It is well known that:

Theorem 4
A language L is recursively enumerable if and only if there is a generalized rewriting system $G = \langle V, s, T, R\rangle$ such that $L = \mathbf{L}(G)$.

Proof
See Hopcroft and Ullman (1979:221–223). ■

We now present the fundamental result of this paper, which states that the languages that can be characterized by categorial grammars with lexical rules are exactly the recursively enumerable languages.

Theorem 5 (R.E.-Completeness)
A language S is recursively enumerable if and only if there is a finite lexicon BasLex and finite set LexRule of lexical rules such that S is the set of strings assigned to the category $s[]$ by LexRule(BasLex).

Proof
It is trivial to show that the languages generated by our system are recursively enumerable; standard breadth-first search mechanisms that interleave lexical and syntactic derivations in order of complexity can be seen to enumerate all analyses.

Conversely, suppose that we have a recursively enumerable language S and that the generalized rewriting system $G = \langle V, s, T, R\rangle$ is such that $S = \mathbf{L}(G)$ is the set of strings generated by G. We show how to construct a categorial grammar using lexical rules that assigns the set S of expressions to some distinguished basic category.

We begin by assuming that:

Definition 20
$\mathsf{BasCat} = V \cup \{\#, \mathbf{s}\}$.

We take a basic category for every nonterminal symbol in the generalized rewriting system along with two special symbols; the # is used as a delimiter in representing sequences of nonterminals by means of circular queues, while the **s** is used as the distinguished category of the grammar (note that s, not **s**, is the distinguished start symbol of G). Our claim is that the lexicon in (21) and lexical rules in (22) generate exactly the same language as G.

Definition 21

- $t := v[]$ if $(v \longrightarrow t) \in R$
- $t := v[\#, s]$ if $(v \longrightarrow t) \in R$

Definition 22

- $v_1[\$, v_2] \Longrightarrow v_1[v_2, \$]$ if $v_2 \in V \cup \{\#\}$
- $v[\$, v_1, \ldots, v_n] \Longrightarrow v[\$, u_1, \ldots, u_m]$ if $(v_1 \cdots v_n \longrightarrow u_1 \cdots u_m) \in R$
- $v[\#, v, \$] \Longrightarrow \mathbf{s}[\$]$ if $v \in V$

We represent an arbitrary string $v_1 v_2 \cdots v_n \in V^*$ by means of the categorial grammar category $v[\#[], v_1[], \ldots, v_n[]]$. In what follows, we omit empty subcategorization lists, so that the above category would be abbreviated to $v[\#, v_1, \ldots, v_n]$. Note that with this encoding, there are as many representations of a string as there are nonterminals $v \in V$.

By repeated application of the first lexical rule in (22), from a lexical entry

$$t := v[\#, v_1, \ldots, v_m, v_{m+1}, \ldots, v_n]$$

we can generate a lexical entry of the form:

$$t := v[v_{m+1}, \ldots, v_n, \#, v_1, \ldots, v_m].$$

The application of this rule is the key to allowing an arbitrary string in the middle of a subcategorization list to move to the end so that it may be modified by a lexical rule. The # symbol keeps track of the true beginning of the sequence being derived.

Suppose that the generalized rewriting system allows the one-step derivation:

$$x_1 \cdots x_i v_1 \cdots v_n y_1 \cdots y_j \longrightarrow x_1 \cdots x_i u_1 \cdots u_m y_1 \cdots y_j.$$

If this rewriting is possible, then $(v_1 \cdots v_n \longrightarrow u_1 \cdots u_m) \in R$ so that by a combination of the first lexical rule and second lexical rule we can carry out the following lexical derivation:

$$\begin{aligned} t &:= v[\#, x_1, \ldots, x_i, v_1, \ldots, v_n, y_1, \ldots, y_j] \\ t &:= v[y_1, \ldots, y_j, \#, x_1, \ldots, x_i, v_1, \ldots, v_n] \\ t &:= v[y_1, \ldots, y_j, \#, x_1, \ldots, x_i, u_1, \ldots, u_m] \\ t &:= v[\#, x_1, \ldots, x_i, u_1, \ldots, u_m, y_1, \ldots, y_j]. \end{aligned}$$

A simple induction then gives us the result that if

$$x_1 \cdots x_i \xrightarrow{*} y_1 \cdots y_j$$

then we can derive a lexical entry of the form

$$t := v[\#, y_1, \ldots, y_j]$$

from a lexical entry of the form

$$t := v[\#, x_1, \ldots, x_i].$$

Considering the first lexical entry, and our last observation, if

$$s \xrightarrow{*} v_1 \cdots v_n$$

according to the generalized rewriting system, then we can derive the lexical entry

$$t := v[\#, v_1, \ldots, v_n]$$

from the lexical entry

$$t := v[\#, s]$$

if $(v \longrightarrow t) \in R$. Now suppose that $(v_i \longrightarrow t_i) \in R$ for $1 \leq i \leq n$ so that $t_1 \cdots t_n \in \mathbf{L}(G)$. Beginning with the basic lexical entry

$$t_1 := v_1[\#, s]$$

representing the lexical rewriting $(v_1 \longrightarrow t_1) \in R$, we can derive the lexical entry:

$$t_1 := v_1[\#, v_1, \ldots, v_n].$$

From this last entry and our last lexical rule, we may derive a lexical entry:

$$t_1 := \mathrm{s}[v_2, \ldots, v_n].$$

Furthermore, since we have $t_i := v_i$ for $2 \leq i \leq n$ because $(v_i \longrightarrow t_i) \in R$, we can assign $t_1 t_2 \ldots t_n$ to the category $\mathbf{s}[]$ by repeated application of the categorial grammar rule scheme. In fact, for $1 \leq i \leq n$ we have $t_1 \cdots t_i$ assigned to $\mathbf{s}[v_{i+1}, \ldots, v_n]$. Thus every string that belongs to the language generated by G can also be generated using our categorial grammar with lexical rules.

It simply remains to notice that we have $t_1 := \mathbf{s}[v_2, \ldots, v_n]$ if and only if $t_1 := v_1[\#, v_1, \ldots, v_n]$, which holds if and only if $s \xrightarrow{*} v_1 \cdots v_n$ and $v_1 \longrightarrow t_1$. The only way to derive $v_i[]$ from t_i is by using a lexical entry which is only possible if $(v_i \longrightarrow t_i) \in R$. Furthermore, the only derivations of category $\mathbf{s}[]$ in the categorial grammar must be derived in this manner, as the only way **s** can arise is by lexical rule application in the above situation. Also note that while the # category can be removed by applying some lexical rule, there are no expressions that are assigned to #[] by our grammar. Thus the only way that the string $t_1 \ldots t_n$ can be assigned to category $\mathbf{s}[]$ is by following a lexical derivation that directly mirrors a derivation in G. ■

The fundamental idea behind our reduction is that the complement specification on a categorial grammar category can be used to simulate the intermediate stages of a generalized rewriting system derivation. The only complication arises from the fact that categorial grammar lexical rules operate on the ends of subcategorization lists, while generalized rewriting systems are allowed to operate on arbitrary substrings.

7. Conclusion

The system presented here for lexical rules in a simplified form of categorial grammar with only one head-complement rule scheme has proven to generate arbitrary recursively enumerable languages. The inevitable conclusion is that if we want a natural and effectively decidable lexical rule system for categorial grammars or head-driven

phrase structure grammars, then we must place restrictions on the system given here or look to state lexical rules at completely different levels of representation which themselves provide the restrictiveness desired, such as in terms of some finite set of thematic roles and grammatical relations, as is done in Lexical Function Grammar (LFG) (see Bresnan 1982; Levin 1987; Bresnan and Kanerva 1989). The common assumption that lexical rules can perform arbitrary operations on subcategorization lists based on obliqueness is simply not restrictive enough to yield an effective recognition algorithm.

Acknowledgments

The primary thanks go to Kevin Kelly, who suggested using the circular queue idea for representing the tape of a Turing machine, an idea that I adapted here. Stuart Shieber originally conjectured that the undecidability result would hold. I would also like to thank two anonymous referees for comments. Finally, I owe thanks to Robert Levine, Alex Franz, Mitzi Morris, and Carl Pollard for providing helpful comments on a previous incarnation of the undecidability proof.

References

Ades, A. E.; and Steedman, M. J. (1982). "On the order of words." *Linguistics and Philosophy* 4:517–558.

Aone, C.; and Wittenburg, K. (1990). "Zero morphemes in unification-based combinatory categorial grammar." In *Proceedings, 28th Annual Meeting of the Association for Computational Linguistics*. Pittsburgh, PA.

Bach, E. (1984). "Some generalizations of categorial grammars." In *Varieties of Formal Semantics: Proceedings of the Fourth Amsterdam Colloquium 1982*, edited by F. Landman and F. Veltman, 1–23. Dordrecht: Foris.

Bar-Hillel, Y.; Gaifman, C.; and Shamir, E. (1960). "On categorial and phrase structure grammars." *Bulletin of the Research Council of Israel* **9F**:1–16.

Bresnan, J. (ed.) (1982). *The Mental Representation of Grammatical Relations*. Cambridge, MA: The MIT Press.

Bresnan, J.; and Kanerva, J. M. (1989). "Locative inversion in Chicheŵa: A case study in factorization in grammar." *Linguistic Inquiry* **20**(1):1–50.

Calder, J.; Klein, E.; and Zeevat, H. (1988). "Unification categorial grammar: A concise, extendable grammar for natural language processing." In *Proceedings, 12th International Conference on Computational Linguistics*. Budapest.

Carpenter, B. (1991). "Categorial grammar, lexical rules and the English predicative." In *Formal Grammar: Theory and Implementation*, Vancouver Studies in Cognitive Science, Volume 2, edited by R. Levine. Vancouver: University of British Columbia Press.

Dowty, D. R. (1982). "Grammatical relations and Montague grammar." In *The Nature of Syntactic Representation*, edited by P. Jacobson and G. K. Pullum. Dordrecht: D. Reidel.

Dowty, D. R. (1979). *Word Meaning and Montague Grammar*. Synthese Language Library, Volume 7. Dordrecht: D. Reidel.

Dowty, D. R. (1978). "Lexically governed transformations as lexical rules in a Montague grammar." *Linguistic Inquiry* **9**:393–426.

Earley, J. (1970). "An efficient context-free parsing algorithm." *Communications of the ACM* **13**:94–102.

Flickinger, D. (1987). *Lexical Rules in the Hierarchical Lexicon*. Doctoral dissertation, Stanford University, Stanford, CA.

Flickinger, D., Pollard, C., and Wasow, T. (1985). "Structure-sharing in lexical representation." In *Proceedings, 23rd Annual Meeting of the ACL*.

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

Hoeksema, J.; and Janda, R. D. (1988). "Implications of process morphology." In *Categorial Structures and Natural Language Structures*, edited by R. Oehrle, E. Bach, and D. Wheeler. Dordrecht: D. Reidel.

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

Keenan, E. L.; and Faltz, L. M. (1985). *Boolean Semantics*, Synthese Language Library, Volume 23. Dordrecht: D. Reidel.

Keenan, E. L.; and Timberlake, A. (1988). "Natural language motivations for extending categorial grammar." In *Categorial Structures and Natural Language Structures*, edited by R. Oehrle, E. Bach, and D. Wheeler. Dordrecht: D. Reidel.

Levin, L. (1987). "Toward a linking theory of relation changing rules in LFG."

Technical report CSLI-87-115, Center for the Study of Language and Information, Stanford University.

Montague, R. (1970). "Universal grammar." *Theoria* **36**:373–398.

Moortgat, M. (1987). "Compositionality and the syntax of words." In *Foundations of Pragmatics and Lexical Semantics*, edited by J. Groenendijk, D. de Jongh, and M. Stokhof. Dordrecht: Foris.

Oehrle, R.; Bach, E.; and Wheeler, D., eds. (1988). *Categorial Grammars and Natural Language Structures*. Dordrecht: D. Reidel.

Pollard, C. J., and Sag, I. A. (1987). *Information-Based Syntax and Semantics: Volume I — Fundamentals*. CSLI Lecture Notes, Volume 13. Stanford: Center for the Study of Language and Information.

Thompson, H. (1982). "Handling metarules in a parser for GPSG." In *The 21st Annual Meeting of the Association for Computational Linguistics*. 26–37.

Uszkoreit, H., and Peters, P. S. (1986). "On some formal properties of metarules." *Linguistics and Philosophy* **9**(4):477–494.

Valiant, L. (1975). "General context-free recognition in less than cubic time." *Journal of Computer and Systems Science* **10**:308–315.