CATEGORY STRUCTURES

Gerald Gazdar

Cognitive Studies Programme, University of Sussex, Brighton BN1 9QN, U.K.

Geoffrey K. Pullum

Cowell College, University of California, Santa Cruz, Santa Cruz, California 95064, USA

Robert Carpenter, Ewan Klein

Centre for Cognitive Science, University of Edinburgh, Edinburgh EH8 9LW, U.K.

Thomas E. Hukari

Department of Linguistics, University of Victoria, Victoria, B.C., Canada V8W 2Y2

Robert D. Levine

Department of Linguistics, University of British Columbia, Vancouver, B.C., Canada V6T 1W5

This paper outlines a simple and general notion of syntactic category on a metatheoretical level, independent of the notations and substantive claims of any particular grammatical framework. We define a class of formal objects called "category structures" where each such object provides a constructive definition for a space of syntactic categories. A unification operation and subsumption and identity relations are defined for arbitrary syntactic categories. In addition, a formal language for the statement of constraints on categories is provided. By combining a category structure with a set of constraints, we show that one can define the category systems of several well-known grammatical frameworks: phrase structure grammar, tagmemics, augmented phrase structure grammar, relational grammar, transformational grammar, generalized phrase structure grammar, systemic grammar, categorial grammar, and indexed grammar. The problem of checking a category for conformity to constraints is shown to be solvable in linear time. This work provides in effect a unitary class of data structures for the representation of syntactic categories in a range of diverse grammatical frameworks. Using such data structures should make it possible for various pseudo-issues in natural language processing research to be avoided. We conclude by examining the questions posed by set-valued features and sharing of values between distinct feature specifications, both of which fall outside the scope of the formal system developed in this paper.

The notion **syntactic category** is a central one in most grammatical frameworks. As Karttunen and Zwicky (1985) observe, traditional "parsing" as taught for languages like Latin involved little more than supplying a detailed description of the grammatical category of each word in the sentence to be parsed. Phrase structure grammars are entirely concerned with assigning terminal strings to categories and determining dominance and precedence between constituents on the basis of their categories. In a classical transformational grammar (TG), the objects transformations manipulate are primarily strings of syntactic categories (and, to a lesser extent, of terminal symbols). This is just as true of recent TG work.

Although the use of syntactic categories is not a logical prerequisite of generative grammar (see Levy and Joshi (1978)), no linguistic approach known to us dispenses with them altogether. In view of this, it is perhaps surprising that linguists have not attempted to explicate the concept "syntactic category" in any gen-

0362-613X/88/010001-19$03.00

eral way, i.e., independently of particular systems of notation and the associated substantive assumptions about grammar.

In this paper we offer an explicit metatheoretical framework in which a notion of "syntactic category" receives a precise definition. The framework is intended to facilitate analysis and comparison of the underlying concepts of different theories, freed from the notational and sociological baggage that sometimes encumbers the original presentations in the literature. Viewed from the standpoint of implementation, it can be regarded as providing a unitary data structure for categories that can be used in the implementation of a number of superficially different grammatical frameworks.

We begin by defining in section 1 a space of categories broad enough to encompass the objects employed as syntactic categories in a range of diverse types of generative grammar. Then, in section 2, we present the syntax and semantics for L_C, a formal language for defining constraints on categories. In the succeeding section we provide illustrative definitions of the grammatical categories used in a number of frameworks. We cover simple phrase structure grammar in section 3.1; tagmemics in section 3.2; Harman's (1963) augmented phrase structure grammar in section 3.3; relational grammar and arc pair grammar in section 3.4; $\overline{X}$ syntax, TG, and the government-binding (GB) framework in section 3.5; generalized phrase structure grammar (GPSG) in section 3.6; systemic grammar in section 3.7; categorial grammar in section 3.8; and Aho's (1968) indexed grammar in section 3.9. We then go on to consider some relevant computational complexity matters (section 4). Finally, we discuss two issues that do not arise in any of these approaches, and which fall outside the scope of the simple theory that we present, namely the use of sets as values of features (section 5) and values shared between distinct feature specifications (section 6). These issues are important in the context of the category systems employed in functional unification grammar (FUG), lexical functional grammar (LFG), and the PATR II grammar formalism.

Our goal in this paper is not an empirical one, but rather one which is analogous to that of Montague's "Universal Grammar" (1970) (see Halvorsen and Ladusaw (1977) for a useful introduction) which attempts to give a general definition of the notion "possible language" in terms applicable to, but not limited to, the study of human languages. We have the much more modest goal of characterizing one rather simple and general notion of "possible syntactic category", and of exploring the range of linguistic approaches that it will generalize to, its formal properties, and its limitations. As will become evident below, our exercise is complementary in certain respects to that of Pereira and Shieber (1984) and Shieber (1987) and to recent work of Rounds and his associates on the development of a logic for the description of the notions of syntactic category that are embodied in functional unification grammar and PATR II (see Kasper and Rounds (1986), Moshier and Rounds (1987), Rounds and Kasper (1986)).

We do not concern ourselves with the appearance or representational details of a given theory of categories (or any of the other aspects of the linguistic framework in question, e.g., its rule system), but only with its underlying semantics—the issue of what set-theoretic (or other nonlinguistic) objects provide categories with their interpretation. We are content with being able to exhibit an isomorphism between one of the theories of categories permitted by our framework and the concrete example we are considering; we need not demonstrate identity. Hence we have deliberately refrained from specifying a formal language for *representing* categories and features. To the extent that we need to produce exemplificatory features or categories for inspection, we may use the conventional notation of the approach in question, or the ordinary notations of set theory, or an informal labeled graph notation introduced below, but we do not offer a representational formalism for categories that has a significance of its own.

In the framework we provide, it is possible to define the category systems of a wide variety of apparently very different approaches to natural language syntax simply by defining two primitive typing functions, and by varying the constraints stated on the categories that they induce. The exercise of expressing the content of various specific linguistic approaches in such terms immediately calls attention to certain interesting formal issues. For example, we reconstruct below the notion of a list-valued (or stack-valued) feature in terms of category-valued features, which automatically allows operations defined on categories such as unification to apply to lists without special redefinition.

An interesting fact that emerges from the view taken here is that on the matter of syntactic categories, there is somewhat more commonality among the diverse approaches currently being pursued than there appears to be when those approaches are viewed in the formalisms used by their practitioners. The various syntactic frameworks that we examine below can be seen to share a great deal of their underlying substantive claims about the information content of the category label of a constituent. Our explication of these underlying commonalities may make somewhat easier the task of the computational linguist attempting to implement a system on the basis of some grammatical framework, or attempting to decide which approach to implement in the first place.

In order to prepare for some of the definitions that follow, we will briefly and informally sketch some of our assumptions about features and categories and the terminology we shall use for talking about them. A **category** is a set of **feature specifications** meeting certain conditions to be defined below. A feature specification is an attribute-value pair $\langle f, v \rangle$ where the attribute f (the **feature**) is atomic (i.e., given by some finite list, and regarded as unanalyzable) and the value v is either

atomic or **complex**. Here we shall assume just one type of complex value, namely a category (but see below in section 5).

An example of an atom-valued feature specification would be ⟨SINGULAR,+⟩ (which many grammarians would write as [+SINGULAR]); intuitively, it might mark singular number (though, of course, the interpretation it actually has depends on the role it plays in the grammar). An example of a complex feature specification, with a category as the value, would be ⟨AGREEMENT, {⟨SINGULAR, +⟩, ⟨GENDER, FEM⟩, ⟨PERSON, 3⟩}⟩; intuitively, it might be used to convey that the value of the AGREEMENT feature is a category representing the combination of singular number, feminine gender, and third person. In the following sections, we will always use SMALL CAPITALS for feature names, and we will generally replace "−" and "+", which are standard usage in the linguistic literature for the atomic values of a binary feature, by 0 and 1 respectively.

As we have said, a category is a set of feature specifications meeting certain conditions. We will now specify these. We do not require that every feature name be represented in each category, but we do require that each occurrence of a feature be paired with exactly one value in any set of specifications; thus {⟨SINGULAR,+⟩, ⟨SINGULAR,−⟩} could not be a category. Hence a category can be modeled as a partial function $C{:}F \rightarrow V$, where F is a set of features and V is the set of values. An equivalent alternative would be to treat categories as total functions into a range that includes an element $\perp$ that can stand as the value where the corresponding partial functions would fail to assign a value. Note that we use the term 'range' here, and subsequently, to refer to a set that includes all the values that a partial function or family of partial functions might take given appropriate domain elements, rather than just the set of values that it does take when we fix a particular domain for a function.

It may be helpful to think of a category as having the structure of an unordered tree, and we will introduce a type of diagram below which exhibits this structure overtly. Often, however, the idea of categories as partial functions will be crucial, so it should be kept in mind throughout.

Since the set V of values may include categories, the definition of the entire set of categories has to be given recursively. Moreover, it has to allow for the possibility that not all values are compatible with all features. Thus, for example, in a given feature system, ⟨GENDER, ∅⟩ and ⟨PERSON, *plural*⟩ might be coherent objects but mnemonically perverse, whereas in another feature system, they might simply be ill-formed. We shall show how these issues can be resolved in the coming sections. We will not, however, give a constructive definition of the set of categories for each grammatical framework we consider. Instead, given our comparative and metatheoretical goals, it turns out to be more convenient to define a category system as a pair $\langle \Sigma, C \rangle$ where Σ is a **category structure**, which defines a set of potential categories (see section 8), and C is a set of constraints expressed in L_C, a language for which the category structure defines the models (see section 9). The actual categories in the system are then to be construed as that subset of the potential categories defined in Σ, each member of which satisfies every constraint listed in C.

1 Defining Category Structures

In this section we define the notion of a **category structure**, which is basically a choice of primitives: a list of features, and a range of possible values for each. Here and throughout the paper we will frequently use "2" to denote the set {0, 1} (the context will make it clear when "2" represents an integer and when it represents a set). We will write A^B for the set of total functions from B into A, $A^{(B)}$ for the set of partial functions from B into A, $\mathcal{P}(A)$ for the power set of A, $|A|$ for the cardinality of A, and $\Delta(f)$ for the domain of a (partial) function f (if f is a partial function than $\Delta(f)$ is the set of items to which f assigns a value).

A category structure Σ is a quadruple $\langle F, A, \tau, \rho \rangle$ where F is a finite set of features, A is a finite set of atoms, τ is a function in 2^F, and ρ is a function from $\{f | \tau(f) = 0\}$ into $\mathcal{P}(A)$. The function τ partitions F into two sets: the set of type 0 features $F^0 = \{f | \tau(f) = 0\}$, and the set of type 1 features $F^1 = \{f | \tau(f) = 1\}$. We will write τ as τ_0 when $F = F^0$. Type 0 features take atomic values and type 1 features take categories as values. The function ρ assigns a range of atomic values to each feature of type 0.

The set of categories K is recursively defined in terms of $\langle F, A, \tau, \rho \rangle$, in a way very similar to that used in Pollard (1984, p. 299ff), though Pollard's assumptions differ on some important details. A relatively informal presentation will suffice here. We will refer to the set of partial functions from F^0 into A that are consistent with ρ as the type 0 categories. We first define the set of pure type 0 categories of Σ as those containing only type 0 feature specifications. Then we build up K via a series of approximations we will refer to as **levels**, finally taking the infinite union of all the levels to obtain K itself:

(1) a. $\emptyset$ is a category at level 0
   b. If α is a type 0 category and β is a category containing only type 1 features whose values are categories at level n, then $\alpha \cup \beta$ is a category at level $n + 1$.
   c. K is the set of all categories at all levels $n \geq 0$.

Given the way K is built up, the induction step in (1b) being restricted to union of finite partial functions, it should be clear that K is a recursive set.

We can define certain relations and operations on the space K of possible categories. Thus, we can give a

constructive definition for **unification** (symbolized $\sqcup$) as a binary operation on categories.

(2) Definition: **unification**
- (i) if $\langle f, v\rangle \in \alpha$ but $\beta(f)$ is undefined, then $\langle f, v\rangle \in \alpha \sqcup \beta$;
- (ii) if $\langle f, v\rangle \in \beta$ but $\alpha(f)$ is undefined, then $\langle f, v\rangle \in \alpha \sqcup \beta$;
- (iii) if $\langle f, v_i\rangle \in \alpha$ and $\langle f, v_j\rangle \in \beta$ and $\tau(f) = 1$, then if $v_i \sqcup v_j$ is undefined, $\alpha \sqcup \beta$ is also undefined, else $\langle f, v_i \sqcup v_j\rangle \in \alpha \sqcup \beta$;
- (iv) if $\langle f, v_i\rangle \in \alpha$ and $\langle f, v_j\rangle \in \beta$ and $\tau(f) = 0$, then if $v_i = v_j$, $\langle f, v_i\rangle \in \beta \sqcup \beta$, else $\alpha \sqcup \beta$ is undefined.
- (v) nothing else is in $\alpha \sqcup \beta$.

We can then use unification to define the **subsumes** relation between categories (where 'subsumes' means 'is more general/underspecified than', or 'is extended by'). We symbolize 'subsumes' with '$\sqsubseteq$', and define it as follows.

(3) Definition: **subsumption**
α subsumes β ($\alpha \sqsubseteq \beta$) if and only if $\beta = \alpha \sqcup \beta$.

Thus α subsumes β if and only if β is the unification of α and β. When α subsumes β then we may refer to β as an *extension* of α. If $\alpha \sqcup \beta$ is undefined, then $\beta = \alpha \sqcup \beta$ fails, and α does not subsume β. From this it follows that, if α and β are categories, then $\alpha = \beta$ if and only if $\alpha \sqsubseteq \beta$ and $\beta \sqsubseteq \alpha$. The following theorem is provable by induction on category levels.

(4) Theorem:
α subsumes β if and only if
(i) $\forall f \in (\Delta(\alpha) \cap F^0)\ [\alpha(f) = \beta(f)]$ and
(ii) $\forall f \in (\Delta(\alpha) \cap F^1)\ [\alpha(f) \sqsubseteq \beta(f)]$.

2 The Constraint Language L_C

We now provide an interpreted formal language, L_C, for expressing specific constraints on categories. Constraints are statements that can be true or false of a category. By requiring satisfaction of the constraint, a constraint can be used to delimit a subspace within the set K induced by a given category structure Σ, to serve as the grammatical categories for a particular type of grammar.

It should be noted that our goals in formulating L_C are slightly different from those of Rounds and his associates: L_C is a language for formulating constraints on well-formed categories, not a language whose expressions are intended for use in place of categories. To put it rather crudely, our language is for category definition whereas Rounds' is (in part) for category manipulation. However, the languages look rather similar syntactically, and where they overlap, the semantics is essentially the same.

We define two types of constraint: **basic** and **complex**. If f is an element of F, and a is an element of A, then there are just two distinct types of well-formed basic constraint:

(5) a. f
b. f:a (where $\tau(f) = 0$)

Informally, (5a) constrains a category to contain some specification for the feature f; thus, the constraint "BAR" says that every syntactic category satisfying it has as one of its elements a pair $\langle$BAR, $n\rangle$. This does not entail that every value of every category-valued feature contained in the category must contain BAR; a basic constraint applies to the "top level" of the tree-like structure of a category. Likewise, (5b) says of a category satisfying it that it has as one of its elements the pair $\langle f, a\rangle$. Note that the only thing a basic constraint can require of a type 0 feature beyond saying that it must be present (defined) is that it have a particular atomic value, and that a basic constraint cannot require anything of a type 1 feature at all beyond demanding its presence.

Turning to complex constraints, we now continue the list (5), giving the syntax for each type of complex constraint together with an informal indication of its semantics. Assume that f is an element of F^1, and ϕ and ψ are themselves well-formed basic or complex constraints, and that we are considering the interpretation of the constraints with respect to some fixed category structure Σ and some category α.

(5) c. f: ϕ 'f is defined in α and its value satisfies ϕ'
- d. $\neg\phi$ 'α does not satisfy ϕ'
- e. $\phi \vee \psi$ 'α satisfies either ϕ or ψ'
- f. $\phi \wedge \psi$ 'α satisfies both ϕ and ψ'
- g. $\phi \rightarrow \psi$ 'either α does not satisfy ϕ or α does satisfy ψ'
- h. $\phi \leftrightarrow \psi$ 'α satisfies either both or neither of ϕ and ψ'
- i. $\Box\phi$ 'α satisfies ϕ, and all values of type 1 features in α satisfy $\Box\phi$'
- j. $\Diamond\phi$ 'either α satisfies ϕ or some value of a type 1 feature in α satisfies $\Diamond\phi$'

Constraints of the forms (5a) through (5h) are fairly straightforward, but constraints like those shown in (5i) and (5j) need a little more discussion. They introduce modality into our language. Their purpose is to allow for recursive constraints to be imposed on successively embedded layers of category values. As indicated, a category α satisfies $\Box\phi$ provided that, firstly, α satisfies ϕ and secondly, whenever α assigns a category β to a type 1 feature f, β satisfies $\Box\phi$. This may appear to introduce a circularity, but it does not: categories are finite, and within any category there will be a level so deeply embedded in the tree structure that there are no more category values within it; at that point $\Box\phi$ is true if ϕ is, thus ending the recursion.

Our choice of notation in (5i) is quite deliberate: in effect, constraints of the form (5i) express universal quantification over embedded accessible categories in

the way that the familiar necessity operator $\Box$ of modal logic enforces universal quantification over accessible worlds in the standard semantics. The possibility operator in (5j) is, as usual, the dual of the necessity operator: $\Diamond\phi$ says of a category α satisfying it that either α satisfies ϕ, or there exists a category-value β assigned to a type 1 feature f by α such that β satisfies $\Diamond\phi$.

As a simple example of the sort of work a complex constraint in L_C might do in a grammatical theory, consider the constraint that is known as the "Case filter" in recent TG (see Chomsky 1980, p. 25). Stated informally as "*N, where N has no Case", the constraint appears to require every occurrence of the feature complex characterizing the category N, i.e., every occurrence of [+N, −V], to co-occur with a feature called "Case". The constraint can be stated in L_C as (6).

(6) $\Box(((\text{N}:1) \wedge (\text{V}:0)) \rightarrow \text{CASE})$

Here and from now on, we use parentheses in the obvious way wherever it is necessary prevent ambiguity in the statement of constraints.

The account of L_C given thus far will suffice for a reading of this paper, but those readers who would like to see the semantics given more formally may turn to the appendix.

To recapitulate, a theory of categories Θ in our sense is a pair $\langle\Sigma, C\rangle$, where Σ is a category structure and C is a set of sentences of L_C. The set of categories determined by Θ is the maximal subset K_C of K determined by Σ such that each member of K_C satisfies every member of C.

3 Illustrative Applications

We will now illustrate the application of the apparatus developed thus far by reconstructing the category systems used in a number of well-known grammatical frameworks that linguists have developed, most of them frameworks that have been used in natural language processing systems at one time or another.

3.1 SIMPLE PHRASE STRUCTURE GRAMMAR

The case of simple phrase structure grammar is trivial, but will serve as an introduction to the form of later sections, and as a straightforward example of the use of a type 0 feature.

The set of categories used in a simple phrase structure grammar is just some finite set of atomic categories $\{a_1, \ldots, a_n\}$, for example, {S, NP, VP, Det, N, V}. So we fix values for F, A, τ, and ρ as in (7):

(7) a. $F = \{\text{LABEL}\}$
b. $A = \{a_1, \ldots, a_n\}$
c. τ_0
d. $\rho = \{\langle\text{LABEL}, A\rangle\}$

Thus, for example, we might have A = {S, NP, VP, Det, N, V}, and thus have ρ(LABEL) as the same set. In addition, we need the following constraint, to make sure that every category does indeed have a specification for the solitary type 0 feature LABEL, i.e., to exclude the empty set from counting as a category:

(8) LABEL

Obviously, we can now show that the category inventory for any simple phrase structure grammar is representable. We let θ be the bijection defined by $\theta(a_i) = \{\langle\text{LABEL}, a_i\rangle\}$, and the result is immediate. Thus there is a bijection from the set of simple phrase structure grammar categories to the categories admitted by the category structure (7) under the constraint (8). As is evident, the set of categories induced is finite, and of cardinality $n = |A|$.

3.2 TAGMEMICS

It may be that there are more published syntactic analyses of languages in the framework of tagmemics than in any other theoretical framework ever developed. Since the early 1960s, those who have followed the work of Kenneth Pike, including a very large number of field linguists working for the Summer Institute of Linguistics, have produced analyses of hundreds of languages, mostly non-Indo-European. Moreover, Postal (1964, p. 33) remarks that "these languages are, for the most part, exotic enough so that the tagmemic descriptions of them may very well be the only ones done."

Tagmemics describes syntactic structure in terms of TAGMEMES, which are notated in the form $A{:}b$, where A is said to represent a SLOT and b a FILLER. For example, Elson and Pickett (1962) represent (part of) the structure of English prepositional phrases and intransitive clauses with tagmemic formulæ (i.e., rules) similar to the following (we simplify very slightly):

(9) a. LraPhr = +R:prep +A:mNc
b. mNc = +Lim:ar ±M:aj +H:nc
c. iCl = +S:mNc +iP:v_3

The informal explication of these is: (9a) one type of location relater-axis phrase consists of an obligatory relater slot filled by a preposition followed by an obligatory axis slot filled by a modified count noun phrase; (9b) one type of modified count noun phrase consists of an obligatory limiter slot filled by an article followed by an optional modifier slot filled by an adjective followed by an obligatory head slot filled by a count noun; (9c) one type of intransitive clause consists of an obligatory subject slot filled by a modified common noun phrase followed by an obligatory intransitive predicate slot filled by a verb of class 3. Thus the left hand side of a formula (before the equality sign) consists of an atomic label, and the right hand side is a string of tagmemes, which are ordered triples $\langle a, b, c\rangle$ where a is an indication of optional (±) or obligatory (+) status, b is a slot or function name, and c is a filler or category label.

One way of representing tagmemes in our terms is to employ a type 0 feature bearing the slot name, taking as value an atomic label identifying the filler. Thus we set up correspondences like the following:

(10) a. R:prep {⟨R, prep⟩}
b. A:mNc {⟨A, mNc⟩}
c. Lim:ar {⟨LIM, ar⟩}
d. M:aj {⟨M, aj⟩}
e. H:nc {⟨H, nc⟩}
f. S:mNc {⟨S, mNc⟩}
g. P:v_3 {⟨P, v_3⟩}

Left hand sides of formulæ can be seen as implicit schematizations over slot names. For example, (9b) says that for any slot name σ, a constituent labelled {⟨σ, mNc⟩} may have the immediate constituent analysis seen on the right hand side of the equation.

A category structure representing a set of categories including all those seen in the above illustrative examples is given in (11).

(11) a. F = {R, A, LIM, M, H, S, P}
b. A = {LraPhr,prep,mNc,ar,aj,nc,v_1,v_2,v_3}
c. τ_0
d. ρ = {⟨R,{prep}⟩,⟨A,{mNc}⟩,⟨LIM,{ar}⟩,⟨M,{aj}⟩,⟨H, {nc}⟩,⟨S,{mNc}⟩,⟨P,{v_1,v_2,v_3}⟩}

This artificially tiny fragment does not show much of the structure that would be revealed in a larger fragment, with more word classes and phrases types, but it will suffice to show how we could set up a category structure that provided isomorphic correspondents to the categories employed in a tagmemic description. Moreover, there is an unclarity about whether there is more to a tagmemic formula than has been illustrated here; as discussed by Postal (1964), there are some remarks about the treatment of agreement in Elson and Pickett (1962) that imply either finite schematization or additional representational devices of an unclarified sort. We will not explore this topic here.

Postal (1964) is probably right in saying that tagmemics appears to be only notationally distinct from context-free phrase structure grammar. Longacre (1965) claims that "[b]y bringing together function and set in the tagmeme" tagmemics ensures that "function is at once kept in focus and made amenable to formal analysis." Under our reconstruction, "functions" like "subject" or "modifier" are "made amenable to formal analysis" simply by incorporating them into the feature structure of categories, making it clear that little was at stake in the debate between Postal and Longacre over the content of tagmemics. It is clear that the number of categories defined by a category structure for tagmemics will be bounded from above by $|F| \cdot |A|$, and thus finite. The question of whether tagmemics reduces to context-free grammar therefore turns on whether tagmemic formulæ can in all cases be reduced to context-free rules. This seems likely, but such issues are not the focus of our attention in this paper.

3.3 HARMAN'S AUGMENTED PHRASE STRUCTURE GRAMMAR

Harman (1963) presents a proposal that involves augmenting the ordinary category inventory (S, NP, VP, etc.) of simple phrase structure grammar by attaching "an unordered sequence of zero or more (up to N for some finite N) subscripts" to a category. Abbreviatory conventions are then used to manage large sets of rules over the resultant vocabulary. Note that the indices stand for the members of a set rather than a sequence, and that there is only a finite number of them.

To formalize Harman's proposal in the present framework, we again use LABEL as the feature that identifies major syntactic categories in the traditional sense, and we set up a finite number of type 0 features $F_1, \ldots, F_n$ to correspond to the presence (value 1) or absence (value 0) of each of the n different subscripts. The set of feature specifications for these features reconstructs the characteristic function of the set of indices. The category structure is as follows:

(12) a. F = {LABEL, $F_1, \ldots, F_n$}
b. A = $\{a_1, \ldots, a_m\} \cup 2$
c. τ_0
d. ρ = {⟨LABEL, $\{a_1, \ldots, a_m\}$⟩, ⟨F_1, 2⟩, . . ., ⟨F_n, 2⟩}

We now have to guarantee that every category has a value for LABEL and a value for each F_i in F. We therefore impose the following constraint:

(13) LABEL $\wedge$ F_1 $\wedge$ . . . $\wedge$ F_n

The resultant specification induces a finite set of categories, of cardinality $m \cdot 2^n$.

Harman's system is more than just a historical curiosity. More recent works are found that use almost exactly the same sort of syntactic categories. For example, the use made of syntactic features in one influential variety of augmented phrase structure grammar, the Prolog-based definite clause grammar (DCG) formalism of Pereira and Warren (1980) closely resembles that of Harman. However, it is clear that the full power of the DCG formalism can, in principle, be used to exploit features with structured values and value-sharing (see section 6 on the latter).

3.4 RELATIONAL AND ARC PAIR GRAMMAR

Relational grammar (RG) Perlmutter and Postal (1977) and arc pair grammar (APG) Johnson and Postal (1980), (henceforth *J&P*) appear to make relatively little use of grammatical category information, expressing most grammatical rules as conditions on arcs representing grammatical relations between nodes (in RG) or as conditions on relations between such arcs (in APG) rather than on the labeling of nodes. Nonetheless, J&P make clear that nodes are assigned grammatical category labels in APG, and since APG is essentially a formalized elaboration of RG ideas, we will assume that much the same is true in RG, though the RG literature so far has not made such aspects of the approach

explicit. Syntactic category labels are not entirely without utility in RG and APG, since, for example, agreement rules crucially make reference to categorial properties like number, gender, and person, and the proper formulation of agreement rules has been a topic of some interest in RG and APG research.

As defined in *J&P*, an **arc** is an ordered pair $\langle R(\langle a, b\rangle), c_1 \ldots c_k\rangle$ where $R(\langle a, b\rangle)$ indicates that b (the second or **head** node) bears the grammatical relation named by the "relational sign" R to a (the first or **tail** node), and c_1 through c_k are the representational strata Ladusaw (1985) at which this holds. In APG, categories are assigned to nodes by means of arcs in which the relational sign is L; such arcs are referred to as L arcs. The head of an L arc is simply an atomic label from a set of "grammatical category nodes" (called **GNo** by *J&P*) that is given by listing.

Two types of grammatical category are recognized in APG: Major categories such as CI (clause), Nom (nominal), and V (verb), and minor categories such as Feminine, Singular, Third-Person, etc. A general constraint (Pair Network Law 31, the Major Category Exclusiveness Law) prevents a node from being the tail of two distinct arcs with heads in the set **Major** (*J&P*, 202), i.e., the set of grammatical category nodes that represent major categories. We can obtain the effect of this law simply by assuming a type 0 feature LABEL which takes values in the set of Major categories.

In the case of minor categories, APG permits multiple atomic elements from **GNo** to be attached by L arcs to a single tail node (*J&P*). Thus a node might be the tail of L arcs whose head nodes are the atoms Nom, Feminine, Singular, and Third-Person, representing a third person singular feminine noun or noun phrase. It is easy to represent such sets of labels attached to a single node using type 0 features. We can represent the set of elements of **GNo** assigned to a given tail node by including a category corresponding to the characteristic function of that set, as with the indices in Harman's system. So we fix values for F, A, τ, and ρ as shown in (14):

(14) a. $F = \{\text{LABEL}, \text{F}_1, \ldots, \text{F}_n\}$
b. $A = \{a_1, \ldots, a_m\} \cup 2$
c. τ_0
d. $\rho = \{\langle \text{LABEL}, \{a_1, \ldots, a_m\}\rangle, \langle \text{F}_1, 2\rangle, \ldots, \langle \text{F}_n, 2\rangle\}$

Here **Major** $= \{\text{F}_1, \ldots, \text{F}_n\}$, and **GNo** $= \{\text{F}_1, \ldots, \text{F}_n\} \cup \{a_1, \ldots, a_m\}$. The constraint needed is the following:

(15) $\text{F}_1 \wedge \ldots \wedge \text{F}_n$

This has the effect of requiring every category to include the characteristic function of a set (of minor categories, in the APG sense). However, we do not need to guarantee that every category has a specification for LABEL, as *J&P* specifically leaves it open whether there are nonterminal nodes with no associated grammatical categories; the absence of any grammatical category node will be reconstructed in our terms as that function ζ that is undefined for LABEL and which assigns 0 to each $\text{F}_i \in F$.

It can be shown that the category system just defined adequately represents category labelling in APG, in the sense that there exists a bijection θ between (a) nonterminal nodes together with their grammatical category L arcs in an admissible APG syntactic representation and (b) admissible categories induced by the category structure in (14) and the constraint in (15).

From an arbitrary well-formed APG pair network we can extract the set X of arcs it contains (*J&P*), and the set N of nodes associated with X. Since we are not concerned with coordinates, we can discard the coordinate sequences and consider just the incomplete arcs to which the arcs in X correspond. By Theorem 1 (*J&P*), all and only the terminal nodes in N are heads of L arcs. Extracting just the arcs with terminal nodes as heads gives us the set of L arcs from X; and discarding those with heads not in **GNo** gives us just the L arcs with grammatical category labels as their heads. The members of this set can be partitioned into equivalence classes having the same tail node (since by definition no arc has more than one tail). For convenience of reference we can call these equivalence classes **category-labelled nodes**.

Theorem. There is a bijection from APG category-labelled nodes to categories admitted by (14) and (15).

Proof. Consider an arbitrary category-labelled node κ with tail n. By PN Law 31, the Major Category Exclusiveness Law, exactly one arc in κ has a head which is in **Major**. Let θ_1 be the bijection established by $\theta_1(\text{L}(n, \alpha)) = \alpha$, and let θ_2 be the bijection established by $\theta(\alpha) = \langle \text{LABEL}, \alpha\rangle$ iff $\alpha \in$ **Major** and $\langle \alpha, 1\rangle$ otherwise. The category corresponding to κ will be the smallest set that contains $\theta_1\theta_2(A)$ for all arcs A in κ and contains $\langle \text{F}_i, 0\rangle$ for all F_i in F that are not in the range of θ_1. Since θ_1 and θ_2 are bijections, their product $\theta_1\theta_2$ is a bijection. The correspondence in the opposite direction is obvious. A node that is the tail of no L arcs will be mapped by $\theta_1\theta_2$ to ζ, and other nodes will be mapped onto categories in which the values of the features record the details of the category-labelling L arcs in κ together with (redundantly) information about which one is the major category, the mapping yielding a unique result in each case.■

The set of APG (and, we assume, RG) categories induced is finite, and *ceteris paribus* is of cardinality $m \cdot 2^n$; it will be much smaller once further conditions on coocurrence of minor categories are imposed (Masculine and Feminine presumably cannot both be mapped to 1 in a category, for example). It is of interest that despite the utterly different grammatical formalism and theoretical background associated with it, the APG notion of syntactic category can be seen to be almost identical to that of Harman's augmented phrase struc-

ture grammar, nodes without LABEL values contributing the only relevant difference.

3.5 $\overline{X}$ SYNTAX, TRANSFORMATIONAL GRAMMAR, GOVERNMENT-BINDING

In the great majority of contemporary works in transformational grammar (TG), including those representing what is known as "government-binding" (GB) Chomsky (1981), the conception of grammatical categories follows what is called "the X-bar convention" Jackendoff (1974) Hornstein (1977) or "X-bar syntax". "X-bar" is often notated $\overline{X}$ or X′, or as X^1, X^2, etc., the superscript numeral denoting the number of bars or **bar level**.) The central idea of X-bar syntax is that phrasal categories are "projected" from lexical categories. Given a lexical category X, the related phrasal nodes are assumed to be $\overline{X}(= X' = X^1)$, $\overline{\overline{X}}(= X'' = X^2)$, and so on.

Representing phrasal categories as founded on lexical categories in this way amounts to treating categories as non-atomic, the distinction between lexical categories and the various levels of phrasal category being tantamount to a feature specification distinction. Bar level is not treated in terms of features in most works using X-bar notation, probably because of the tradition in TG (and related work in segmental phonology) restricting features to the values {−, +}. Thus Bresnan (1975) treats categories as ordered pairs $\langle i, M\rangle$ where i is a natural number representing the bar level and M is a matrix of feature specifications, and the same formalization is used by Lasnik and Kupin (1977). Here we simply integrate bar level information with the rest of the feature system.

Although the origins of the X-bar proposal (Harris 1951) do not take such a feature analysis of categories any further, but treat lexical categories as atomic, it is always assumed in current instantiations of X-bar syntax that lexical categories themselves have a feature analysis. In much TG, it is presupposed that the lexical categories N, A, V, and P are to be analyzed in terms of two binary features N and V.[1] Lasnik and Kupin (1977) is a fairly explicit formulation of this type of category system. They assume a maximum bar level of three. To characterize their system of categories, we fix our values for F, A, τ, and ρ as in (16), and impose the constraint in (17).

(16) a. $F = \{\text{N}, \text{V}, \text{BAR}\}$
b. $A = \{0, 1, 2, 3\}$
c. τ_0
d. $\rho = \{\langle N, 2\rangle, \langle \text{V}, 2\rangle, \langle \text{BAR}, A\rangle\}$

(17) $\text{N} \wedge \text{V} \wedge \text{BAR}$

This yields a system of 16 categories, four at each bar level.

Jackendoff (1977) proposes a version of X-bar syntax in which lexical categories are distinguished from one another by means of the features [±SUBJ], [±OBJ], [±COMP], and [±DET] rather than by [±N] and [±V]. He does not provide an explicit definition of his full set of categories, but he gives enough detail for it to be deducible. To define Jackendoff's system of categories, we fix our values for F, A, τ, and ρ in the manner shown below:

(18) a. $F = \{\text{SUBJ}, \text{COMP}, \text{DET}, \text{OBJ}, \text{BAR}\}$
b. $A = \{0, 1, 2, 3\}$
c. τ_0
d. $\rho = \{\langle \text{SUBJ}, 2\rangle, \langle \text{COMP}, 2\rangle, \langle \text{DET}, 2\rangle, \langle \text{OBJ}, 2\rangle, \langle \text{BAR}, A\rangle\}$

To get the exact set of permissible categories, we need to make sure that SUBJ, OBJ, COMP, and BAR are defined in all categories, and that DET is only specified in [-COMP], [-OBJ] categories. The following set of L_C constraints will achieve this.

(19) a. $\text{SUBJ} \wedge \text{OBJ} \wedge \text{COMP} \wedge \text{BAR}$
b. $\text{DET} \rightarrow ((\text{COMP}{:}0) \wedge (\text{OBJ}{:}0))$

We can now obtain a bijection between Jackendoff's X-bar categories and the admissible categories induced by F, A, and the constraints listed in (19). We define a mapping θ between the Jackendoff's own category abbreviations and the admissible categories with respect to (19a) and (19b), as follows (we schematize by writing X with n bars as X^n, $0 \leq n \leq 3$):

(20) a. $\theta(V^n) = \{\langle \text{SUBJ}, 1\rangle, \langle \text{OBJ}, 1\rangle, \langle \text{COMP}, 1\rangle, \langle \text{BAR}, n\rangle\}$
b. $\theta(M^n) = \langle \text{SUBJ}, 1\rangle, \langle \text{OBJ}, 1\rangle, \langle \text{COMP}, 0\rangle, \langle \text{bar}, \text{n}\rangle\}$
c. $\theta(P^n) = \{\langle \text{SUBJ}, 0\rangle, \langle \text{OBJ}, 1\rangle, \langle \text{COMP}, 1\rangle, \langle \text{BAR}, \text{n}\rangle\}$
d. $\theta(\text{Prt}^n) = \{\langle \text{SUBJ}, 0\rangle, \langle \text{OBJ}, 1\rangle, \langle \text{COMP}, 0\rangle, \langle \text{BAR}, \text{n}\rangle\}$
e. $\theta(N^n) = \{\langle \text{SUBJ}, 1\rangle, \langle \text{OBJ}, 0\rangle, \langle \text{COMP}, 1\rangle, \langle \text{BAR}, n\rangle\}$
f. $\theta(\text{Art}^n) = \{\langle \text{SUBJ}, 1\rangle, \langle \text{OBJ}, 0\rangle, \langle \text{COMP}, 0\rangle, \langle \text{DET}, 1\rangle, \langle \text{BAR}, n\rangle\}$
g. $\theta(Q^n) = \{\langle \text{SUBJ}, 1\rangle, \langle \text{OBJ}, 0\rangle, \langle \text{COMP}, 0\rangle, \langle \text{DET}, 0\rangle, \langle \text{BAR}, n\rangle\}$
h. $\theta(A^n) = \{\langle \text{SUBJ}, 0\rangle, \langle \text{OBJ}, 0\rangle, \langle \text{COMP}, 1\rangle, \langle \text{BAR}, n\rangle\}$
i. $\theta(\text{Deg}^n) = \{\langle \text{SUBJ}, 0\rangle, \langle \text{OBJ}, 0\rangle, \langle \text{COMP}, 0\rangle, \langle \text{DET}, 1\rangle, \langle \text{BAR}, n\rangle\}$
j. $\theta(\text{Adv}^n) = \{\langle \text{SUBJ}, 0\rangle, \langle \text{OBJ}, 0\rangle, \langle \text{COMP}, 0\rangle, \langle \text{DET}, 0\rangle, \langle \text{BAR}, N\rangle\}$

An example of a category admitted in Jackendoff's system would be $\{\langle \text{BAR}, 3\rangle, \langle \text{SUBJ}, 1\rangle, \langle \text{OBJ}, 0\rangle, \langle \text{COMP}, 1\rangle\}$, which can be more perspicuously presented in the graphic form given in (21).

BAR	3
SUBJ	1
OBJ	0
COMP	1

As is evident, the set of categories induced by Jackendoff's system has a cardinality of 40, ten at each bar level.

Sets of categories as small as this are clearly insufficient for the description of natural languages. All transformational grammarians seem to agree that references to distinctions of tense, mood, voice, person, number, gender, case, pronominality, definiteness, *wh*-ness, and many other morphological and syntactic distinctions are in fact needed in a grammar. As pointed out by Pullum (1985), some statements in the TG literature suggest that further features are provided for the expression of such distinctions but are restricted to lexical (⟨BAR, 0⟩) categories. However, it is easy to find examples in the literature of additional features like definiteness, case, *wh*-ness, and many others, being assigned to phrasal nodes as well. In marked contrast to a work such as Stockwell, Schachter and Partee (1973), recent TG has not been explicit about such matters. Allowing for twenty binary morphosyntactic features (a modest estimate if any serious effort at coverage is to be made) and allowing them only on lexical categories would increase the cardinality of the set of categories to about $4 \cdot 10^6$ in the case of Lasnik and Kupin's system and to over 10^7 in the case of Jackendoff's.

In one respect, what we have said so far may not adequately capture the conception of categories found in recent TG and GB works. These works generally make considerable and crucial use of co-indexing of nodes, using indices taken from an infinite set such as the integers. If the index on a node is taken to be part of the structure of the category labelling that node Chomsky (1970), which is not the only view one could take, then the number of distinct categories becomes infinite. This does not mean it becomes difficult to represent. Indexing of this sort can be represented directly in the present framework without adding an infinite set of additional atoms such as the natural numbers. We add a type 0 feature OF (with ρ(OF) = {0}) and a type 1 feature SUCCESSOR to the feature system and use this to build the set of indices. Thus the index "3" would be represented as shown in (22), where category-valued feature specifications are shown with pointers to categories in their value positions.

In some recent TG, more than one indexing system is employed. Thus Rouveret and Vergnaud (1980, p. 160) "postulate that each verbal complex in a structure is identified by some integer p and each [-N] element in the verbal complex p bears the superscript p." This superscripting system is distinct from the subscripting system maintained to indicate anaphoric linkage or binding, and neither places an upper bound on the number of indices. Hence it would not be sufficient to have a single type 1 feature. Two further type 1 features SUBSCRIPT and SUPERSCRIPT could be used, each taking category values representing indices with SUCCESSOR and OF.

It may seem implausible to suppose that anyone would choose in practice to handle indexing via a feature system such as that just suggested. Nonetheless, it would clearly be possible, which shows that one can incorporate integer indices into the structure of categories in terms of a finite number of features and a finite number of atoms, which might not initially have been evident.

3.6 GENERALIZED PHRASE STRUCTURE GRAMMAR

The generalized phrase structure grammar framework (GPSG), as set out in Gazdar, Klein, Pullum, and Sag (1985), (henceforth *GKPS*), differs from the examples considered so far in that it makes extensive use of features that are permitted to have categories as their values.[2]

For concreteness, we suggest how the set of categories for the *GKPS* version of GPSG would be reconstructed in the framework presented here (see *GKPS* pp. 245–6, for the complete lists where we abbreviate with ". . .").

(24) a. F = {SUBJ, N, COMP, BAR, ... AGR, SLASH}
b. A = {0, 1, 2, . . ., *for*, *that*, . . .}
c. τ = {⟨SUBJ, 0⟩, ⟨N, 0⟩, ⟨V, 0⟩, ⟨COMP, 0⟩, ⟨BAR, 0⟩, . . . ⟨AGR, 1⟩, ⟨SLASH, 1⟩}
d. ρ = {⟨SUBJ, 2⟩, ⟨N, 2⟩, ⟨V, 2⟩, ⟨COMP, {*for*, *that*, . . .}⟩, . . ., ⟨BAR, {0, 1, 2}⟩}

We add to this, for each feature $f \in F^1$, the following constraint:

(25) $\Box \neg (f: \Diamond f)$

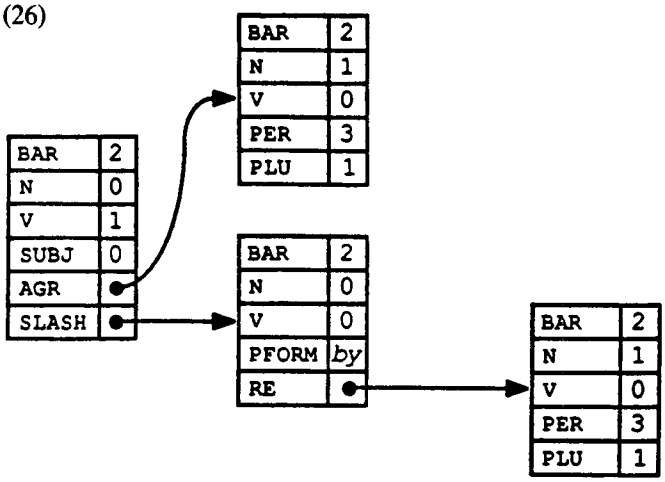
This prevents a category-valued feature f from being specified anywhere within the value of an occurrence of f. An example of a moderately complex category with more than one category-valued feature that nonetheless obeys (25) is shown in (26).

(22)

Constraints are necessary to ensure that the value of SUCCESSOR does not contain anything but SUCCESSOR or OF specifications. To this end, we constrain each feature $f \in F^0$ (except OF) as shown in (23a), and in addition we impose (23b) and (23c):

(23) a. $\Box \neg$ (SUCCESSOR: f)
b. $\Box \neg$ (SUCCESSOR $\wedge$ OF)
c. $\Box \neg$ (SUCCESSOR: $\neg$ OF $\neg$ SUCCESSOR)

(26)



The constraint (25) restricts us to exactly the set of legal *GKPS* categories.[3] The total *GKPS* category set is finite, but naturally, it is extremely large (Ristad (1986) calculates that it is in excess of 10^{774}). It is clear that the set of *GKPS* categories is vastly too large to be precompiled and stored–and indeed, no implementation that we know of has attempted this.

3.7 SYSTEMIC GRAMMAR

Systemic grammar, originally known as "scale and category" grammar, has its origins in the work of Halliday (1961) and is widely known among computational linguists through Winograd (1972) and other works, and it has recently received rigorous formalization in the hands of Patten and Ritchie (1987). Tree structures in systemic grammar tend to be flat, more structural information being expressed through categories than in most other approaches Hudson (1971). Categories in systemic grammar are simply bundles of feature specifications: there is "nothing in systemic theory corresponding to the distinction between "features"—such as [+past] —and "categories"—such as NP and S—in TG theory" Hudson (1971, p. 48). A set of well-formed categories in a systemic grammar is defined by a **system network**, which "is in effect a body of rules, in symbolic form, which specify precisely how features can combine with each other: in other words, which features can appear together in the paradigmatic description of a single item, and which cannot" Hudson (1971).

We will not discuss rules for forming systemic networks (and hence categories) here, but will instead refer the reader to the presentation in Winograd (1983), where a system network expressing category information for the English pronominal form is provided as an example of the notational techniques used in systemic grammar for specifying a set of categories. We reproduce this in Figure 1.

The content of Figure 1 can be reconstructed straightforwardly as a category structure subject to a set of L_C constraints (for a closely related analysis of this

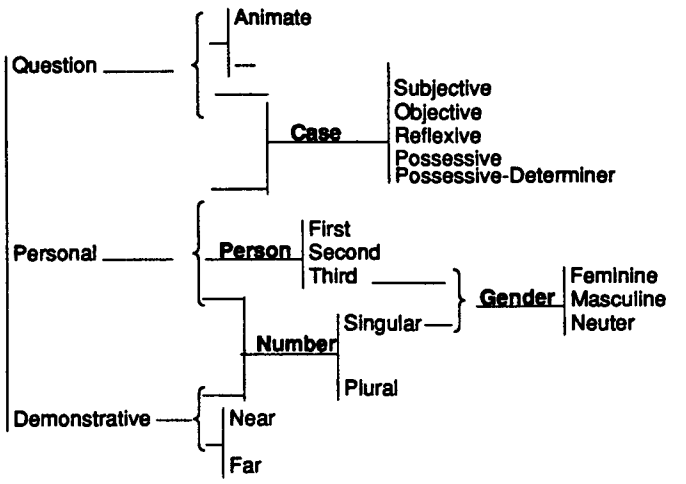


Figure 1: Systemic Network for English Pronouns

example, developed independently, see Mellish (1986). The following is the category structure that we need:

(27) a. F = {PRONOUN, CASE, PERSON, GENDER, NUMBER, ANIMACY, PROXIMITY}
b. A = {*question, personal, demonstrative, subjective, objective, reflexive, possessive, possessive-determiner, first, second, third, feminine masculine, neuter, singular, plural*}
c. τ_0
d. ρ = {⟨PRONOUN, {*question, personal, demonstrative*}⟩,
⟨CASE, {*subjective, objective, reflexive, possessive, possessive-determiner*}⟩,
⟨PERSON, {*first, second third*}⟩,
⟨GENDER, {*feminine, masculine, neuter*}⟩,
⟨NUMBER, {*singular, plural*}⟩,
⟨ANIMACY, 2⟩,
⟨PROXIMITY, 2⟩}

The constraints that must be imposed are the following:

(28) a. PRONOUN
b. (PRONOUN:*question*) ↔ (CASE ∧ ¬ PERSON ∧ ¬ NUMBER ∧ ANIMACY ∧ ¬ PROXIMITY)
c. (PRONOUN:*personal*) ↔ (CASE ∧ PERSON ∧ NUMBER ∧ ¬ ANIMACY ∧ ¬ PROXIMITY)
d. (PRONOUN:*demonstrative*) ↔ (¬ CASE ∧ ¬ PERSON ∧ NUMBER ∧ ¬ ANIMACY ∧ PROXIMITY)
e. GENDER ↔ (PRONOUN ∧ (PERSON:*third*) ∧ (NUMBER: *singular*))

Note that this description of the pronominal system of English is artificially complicated by its isolation from the rest of the grammar. If it were embedded in the context of a definition of a wider class of categories (for example, the English noun class network given by Winograd (1983), it would be modified by the elimination of (28a) and the relaxation of (28b-d) to simple conditionals.

The structure seen in this example employs only type 0 features. For example, the category it defines for a pronoun like *herself* would be (29).

(29)

PRONOUN	*personal*
CASE	*reflexive*
PERSON	*third*
NUMBER	*singular*
GENDER	*feminine*

Interestingly, however, systemic grammar as formalized by Hudson (1971), at least is not limited to type 0 features. Hudson explicitly permits recursive growth of feature structures in order to count constituents (see pp. 60-62). This could be reconstructed here by using a type 1 feature in roughly the manner we employed SUCCESSOR, above. Such a use of type 1 features immediately makes the size of the category set infinite.

3.8 CATEGORIAL GRAMMAR

Categorial grammar originates with work by Lesniewski and Adjukiewicz in the 1940s (see van Benthem, Buszkowski and Marciszewski (1986), Haddock, Klein and Morrill (1987) and Oehrle, Bach and Wheeler (1987) for recent work and references to the earlier literature). The set of categories used is infinite. It is often defined as the smallest set containing some set of basic categories $\{a_1, \ldots, a_n\}$, and closed under the operation of forming from two categories α and β a new category $\alpha|\beta$.

To reconstruct the category system for categorial grammar, we define Σ as shown in (30).

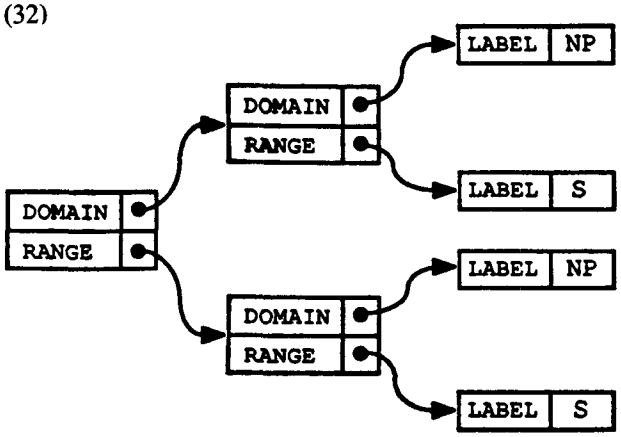
(30) a. $F = \{$LABEL, DOMAIN, RANGE$\}$
b. $A = \{a_1, \ldots, a_n\}$
c. $\tau = \{\langle$LABEL, 0$\rangle$, $\langle$DOMAIN, 1$\rangle$, $\langle$RANGE, 1$\rangle\}$
d. $\rho = \{\langle$LABEL, $A\rangle\}$

We then add the following:

(31) a. $\Box$(DOMAIN $\leftrightarrow \neg$ LABEL)
b. $\Box$(DOMAIN $\leftrightarrow$ RANGE)

We can now represent any category allowed in the simple form of categorial grammar considered so far. For example, the category (S|NP)|(S|NP) can be represented as shown graphically in (32).

(32)



To show formally that we have captured the content of the category system of categorial grammar, we can exhibit a bijection between the categorial grammar categories and the admissible categories induced by F, A, and the constraints defined above. We define a mapping θ between the categorial grammar categories and the admissible categories with respect to (31a) and (31b), as follows:

(33) a. $\theta(a_i) = \langle$LABEL, $a_i\rangle$ where $a_i \in A$
b. $\theta(\alpha|\beta) = \{\langle$DOMAIN, $\theta(\beta)\rangle$, $\langle$RANGE, $\theta(\alpha)\rangle\}$ where α and β are categories.

A simple structural induction argument suffices to show that θ is indeed bijective. The smallest category will be of the type a_i, and corresponds to $\{\langle$LABEL, $a_i\rangle\}$. The next step up yields a category of the form $a_i|a_j$, which corresponds to:

(34) $\{\langle$DOMAIN, $\{\langle$LABEL, $a_j\rangle\}\rangle$, $\langle$RANGE, $\{\langle$LABEL, $a_i\rangle\}\rangle\}$.

Each further step replaces a_i or a_j by a non-basic category and will clearly yield a unique result. It can be seen immediately that the mapping θ has an inverse.

The categories defined thus far are non-directional, in the sense that a complex category can combine with an argument either to its left or its right. However, most definitions assume directional categories Bach (1984). This further specification can be easily incorporated by introducing a new feature name DIRECTION which takes values in 2. We then add a constraint that categories taking values for DOMAIN also take a value for DIRECTION, thus determining the directionality of the category.

(35) $\Box$(DOMAIN $\leftrightarrow$ DIRECTION)

The translation function is then:

(36) a. $\theta(a_i) = \{\langle$LABEL, $a_i\rangle\}$
b. $\theta(\alpha/\beta) = \{\langle$DOMAIN, $\theta(\beta)\rangle$, $\langle$RANGE, $\theta(\alpha)\rangle$, $\langle$DIRECTION, 1$\rangle\}$
c. $\theta(\alpha/\beta) = \{\langle$DOMAIN, $\theta(\beta)\rangle$, $\langle$RANGE, $\theta(\alpha)\rangle$, $\langle$DIRECTION, 0$\rangle\}$

This translation function is again a bijection, for the same reasons as before. Clearly we could employ an analogous move to subsume the α/β vs. $\alpha//\beta$ category distinction employed in Montague (1973).

In some recent work on categorial grammar, it makes sense to think of expressions being assigned to infinite sets of categories rather than to a single category, but we will not pursue the implications of such a move here (see van Benthem (1986c) for relevant discussion).

3.9 INDEXED GRAMMAR

Indexed grammars are a generalization of phrase structure grammars due originally to Aho (1968). Like categorial grammar and some of the other frameworks previously mentioned, it uses an infinite category set. In the formulation presented in Gazdar (1985), an indexed grammar category consists of an atomic label and a

possibly empty list (or stack) of atomic indices drawn from a finite set.

There is a familiar technique for encoding lists or stacks in a notation which relies on the fact that lists can be decomposed into an initial element and the residual list (see, for example, Shieber (1984)). Thus, we add new elements INDEX and LIST to the set F:

(37) a. $F = \{\text{LABEL}, \text{INDEX}, \text{LIST}\}$
b. $A = \{a_1, \ldots, a_m\} \cup \{0, i_1, \ldots, i_n\}$
c. $\tau = \{\langle \text{LABEL}, 0\rangle, \langle \text{INDEX}, 0\rangle, \langle \text{LIST}, 1\rangle\}$
d. $\rho = \{\langle \text{LABEL}, \{a_1, \ldots, a_m\}\rangle, \langle \text{INDEX}, \{0, i_1, \ldots, i_n\}\rangle\}$

A list of indices of the form (38a) is represented as (38b).

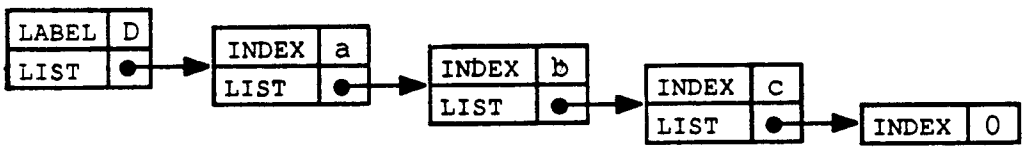
(38) a. $[j_0, j_1, \ldots, j_k]$
b. $\{\langle \text{LIST}, \{\langle \text{INDEX}, j_0\rangle, \langle \text{LIST}, \{\langle \text{INDEX}, j_1\rangle, \langle \text{LIST}, \ldots \{\langle \text{INDEX}, j_k\rangle, \langle \text{LIST}, \{\langle \text{INDEX}, 0\rangle\}\rangle\} \ldots\rangle\}$

In addition, we need the following constraints:

(39) a. LABEL $\wedge$ LIST
b. $\Box \neg$ (LABEL $\wedge$ INDEX)
c. $\Box \neg$ (LIST: $\neg$ INDEX)
d. $\Box \neg$ (LIST $\wedge$ INDEX:0)

The first requires that at the top level, an indexed category has a label and a list of indices. The second disallows INDEX from co-occurring with LABEL, enforcing the constraint recursively downward. The third requires that if LIST is defined anywhere, then INDEX is defined in its value. And the last, also enforced recursively downward, requires that if INDEX has the value 0, LIST is not defined (so the end of the list of indices is unambiguously flagged by INDEX having the value 0). A category bearing an "empty" list of indices is thus one whose value for LIST is $\{\langle \text{INDEX}, 0\rangle\}$. An example of a category allowed by these constraints is shown in (40).

(40)



Indexed grammar as originally formalized by Aho uses lists of atomic indices as part of the composition of categories. It is also possible in the framework we have defined to allow features to have lists of categories as their values. This is in fact proposed in the literature by Shieber (1984) and Pollard (1985). To extend an indexed grammar to permit *GKPS*-style categories in place of atomic indices, one can simply make INDEX a type 1 feature, add the *GKPS* category structure and constraints to the indexed grammar category structure and constraints, and then exempt LIST (but, crucially, not INDEX) from being subject to the constraint schema in (25). The resultant type of grammar, assuming that the limitations on rules in indexed grammars are maintained, is equivalent to indexed grammar as originally defined, since the distinction between atomic indices and indices taken from a finite set of categories has no language-theoretic implications.

Given the representability of list-valued features as category-valued features in the present framework, the definitions of subsumption and unification automatically apply to lists without the need for any redefinition. If the empty category is used as the end marker for lists then two lists of different lengths will unify if one is a prefix of the other. Depending upon the linguistic interpretation of lists, this may or may not be what one wants. In our illustration, we use an atomic end marker that will block prefix unification.

4 Computational Complexity of Category Checking[4]

The **checking problem** for categories is the problem of determining whether a category is legal given a fixed set of constraints, or more precisely, of determining for an arbitrary category α and a fixed formula ϕ of L_C whether α satisfies ϕ. It is a special case of the problem of determining whether some arbitrary model satisfies some fixed formula of a logic.

Theorem. The checking problem for categories is solvable in linear time.

Proof. Assuming a category structure $\Sigma = \langle F, A, \tau, \rho\rangle$, we represent a category in K as a partially labelled, unordered tree with all nodes except the root labelled from $F \cup A$, all nodes labelled from A being terminals. The category $\varnothing$ corresponds to a single unlabelled node; $\langle f, a\rangle$ for $f \in F^0$ and $a \in A$ corresponds to a node labelled f with daughter labelled a; and $\langle f, \{\sigma_1, \ldots, \sigma_k\}\rangle$ for $f \in F^1$ and $n \geq 0$ corresponds to a node labelled f with the first elements of σ_1 through σ_k as its daughter(s). Let T be such a tree, and let ϕ be a fixed formula of L_C. We check T for satisfaction of ϕ by annotating each node of T with the complete list of all subexpressions of ϕ, and working from the frontier to the root recording at each node which subexpressions are satisfied by the subtree rooted there. At each point the checking is local: only the current node and its daughters (if any) need be examined. Even for a subformula like $\Box\psi$, all that must be verified at a node q as we work up the tree is that ψ is satisfied at q and $\Box\psi$ is recorded as satisfied at each daughter node. The conclusion of the procedure will be to determine whether or not ϕ itself is true at the root of T, and thus whether T is well-formed. If ϕ has s subformulæ and T has n nodes, the time taken is bounded by sn (the number of steps

required if every subformula is evaluated at every node), and thus linear in n, the size of the input. ■

Of somewhat less interest than the checking problem is the **universal checking problem**, that of determining for an arbitrary input pair $\langle \phi, \alpha \rangle$, ϕ a formula and α a category, whether α satisfies ϕ. The difference is that here ϕ is not held constant; the task is analogous not to checking the legality of a category within a selected grammatical framework, but rather to a kind of framework-design oversight role, switching frameworks with every input and evaluating the given category relative to the proffered constraint. We note, however, that the universal checking problem only calls for, at worst, quadratic time. To see this, simply note that we can use the algorithm sketched above, and take account of s as well as n as part of the size of the input. The worst case is where s and n contribute about equally to the size of the product sn, i.e., where $s \approx n$. Then $sn \approx ((s + n)/2)^2 = (s + n)^2/4$, which varies with the square of the input size $s + n$.

For some special cases, both the checking problem and the universal checking problem are of course much easier. For example, if only type 0 features are permitted, checking is decidable in real time by a simple inspection of the finite number of $\langle f, a \rangle$ pairs, regardless of whether ϕ is part of the input or not.

Note that the much harder **satisfiability problem**, that of determining for an arbitrary formula ϕ whether there exists a category α that satisfies it, is of even less interest in the present context. When a grammatical framework intended for practical use is devised, the constraints on its category system are formulated to delimit a particular set of categories already well understood and exemplified. There is no practical interest in questions about arbitrary formulæ of L_C for which no one has ever considered what a satisfying category would be like.

We would expect the satisfiability problem for L_C to be PSPACE-complete, like the satisfiability problem for most modal logics. Ristad (1986, p. 33-4) proves a PSPACE-hardness result for what he calls "GPSG Category-Membership", specifically with respect to the *GKPS* framework, and this can immediately be seen to be extendable to the satisfiability result for L_C (as mentioned in footnote 3, L_C is in effect a language for the statement of feature cooccurrence restrictions, and can be used in the same way that Ristad uses the *GKPS* FCR formalism). The problem he considers, despite the misleading name he gives it, is the analog of satisfiability, not of checking; it asks whether there exists an extension of a given category that satisfies a given set of FCRs, and since the given category might be ∅, this is equivalent to satisfiability. Satisfiability is NP-complete even for simple propositional logic, so as soon as it is appreciated that a language for stating constraints on categories is in effect a logic with categories as its models, the complexity of satisfiability for category constraints comes as no surprise. Checking of *GKPS* categories, on the other hand, which Ristad does not consider, can be done very fast, as a corollary of the theorem above.

5 Sets as Values

All the syntactic approaches that we have considered so far distinguish syntactic categories from structural description of expressions in a fairly transparent fashion. In FUG Kay (1979, 1985), LFG Kaplan and Bresnan (1982), and work by Shieber and others on PATR II Shieber (1984), this traditional distinction disappears almost entirely. Thus, in LFG, syntactic categories and the structural descriptions known as f-structures are exactly the same kind of object. In FUG, not only is there no formal distinction between categories and structural descriptions, but even the distinction between structural descriptions and grammars disappears. At first sight, LFG f-structures seem likely to be the trivial case of a set of categories observing no constraints on admissibility at all. We simply take F to be the LFG set of f-structure attribute names, and A to be the LFG set of atomic f-structure values (the "simple symbols" and "semantic forms"). So, following this reasoning, the set of LFG f-structures would be just K, modulo the appropriate typing. However, this is not the case, for reasons that will emerge below.

The first problem we consider is that at least two of the frameworks just mentioned permit sets as feature values. In one sense we already permit sets as values since type 1 features have categories as their values, and categories are sets. Categories are a rather special kind of set, however, namely partial function from features to values. Suppose we merely wanted to have a model for a set of atoms. Then, as we saw in our discussion of APG, we can model such a set by constructing the set's characteristic function. But modelling a set that way, whilst perfectly adequate for APG categories, has a consequence that may not always be acceptable: two sets on the same domain will unify just in case they are exactly the same set. Given certain quite natural interpretations of a feature system making use of sets, this may not be what we want.

An alternative strategy then, and one which is also consistent with our framework, is to model sets as partial functions into a single value range (as opposed to total functions into a two value range). For example, the subset of the authors of this paper with British addresses could be represented as a partial function on the domain {*Gazdar*, *Pullum*, *Carpenter*, *Klein*, *Hukari*, *Levine*}, namely the function {⟨*Carpenter*, 1⟩, ⟨*Gazdar*, 1⟩, ⟨*Klein*, 1⟩,} instead of the following total (characteristic) function on the same domain: {⟨*Carpenter*, 1⟩, ⟨*Gazdar*, 1⟩, ⟨*Hukari*, 0⟩, ⟨*Klein*, 1⟩, ⟨*Levine*, 0⟩, ⟨*Pullum*, 0⟩}. Then unification of the partial functions amounts to union of the corresponding sets.

This is fine if our intended interpretation of the set is

conjunctive, i.e., if {*a*, *b*, *c*} means that *a* holds and *b* holds and *c* holds (Carpenter has a British address and Klein has a British address and Gazdar has a British address). But if our intended interpretation is disjunctive, then we want the unification operation to give us intersection, not union. FUG actually uses set-valued attributes with a disjunctive interpretation Kay (1979). And, in a discussion of possible enhancements to the PATR II formalism, Karttunen (1984) provides a number of very relevant examples that illustrate the issues that arise when a unification-based formalism is augmented in order to encompass disjunction.

As Chris Barker has pointed out to us, a perverse variant of the approach to conjunctively interpreted sets outlined above serves to handle the disjunctive interpretation of sets of atoms. We map the set {Accusative, Dative} into the partial function {⟨NOMINATIVE, 0⟩, ⟨ABLATIVE, 0⟩, ⟨GENITIVE, 0⟩} on the domain {ACCUSATIVE, DATIVE, NOMINATIVE, ABLATIVE, GENITIVE}. Now unification (and hence union) of such complement-specifying partial functions gives us an operation equivalent to intersection applied to the original sets. Thus the unification of {⟨NOMINATIVE, 0⟩, ⟨ABLATIVE, 0⟩, ⟨GENITIVE, 0⟩} (standing for {Accusative, Dative}) with {⟨NOMINATIVE, 0⟩, ⟨ACCUSATIVE, 0⟩, ⟨GENITIVE, 0⟩} (standing for {Ablative, Dative}) gives us {⟨NOMINATIVE, 0⟩, ⟨ABLATIVE, 0⟩, ⟨GENITIVE, 0⟩, ⟨ACCUSATIVE, 0⟩} which stands for {Dative}.

Clearly, the present approach could be generalized to directly allow a type of feature that would take sets of atoms as values. The price to be paid for this, in a metatheoretical exercise such as the one we are engaged in, would be that the definition of unification becomes dependent on the intended interpretation of such features: the relevant clause needs to use union if the interpretation is conjunction, and intersection if the interpretation is disjunction.

An altogether more serious issue arises when we consider the possibility of attributes taking sets of categories as values. We could represent such sets in a manner analogous to the treatment of lists, but with a special marking (given in terms of special attribute-value pairs) indicating that the list representation in question is to be interpreted as a set. The trouble with this is that the identity conditions for the resulting objects are no longer transparent. Two structurally distinct lists may or may not count as identical, depending on whether or not they are both representing sets, and that in turn will depend on whether particular attributes appear in certain relevant structural positions. Likewise, our existing definitions of unification and subsumption would simply fail to provide one with intuitively reasonable results, and its seems unlikely that they could be made to do so without further formal contortions. This whole strategy seems contrived and inelegant.

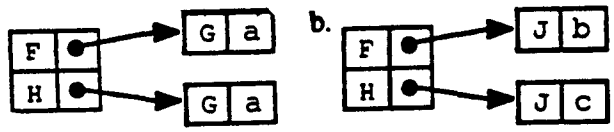
The alternative is, again, to introduce a new type of feature, one taking sets of categories as its values, and some recent works have done just this. Sabimana (1986) proposes a feature ARG which takes a set of categories as its value. The feature appears on elements that correspond semantically to predicates, and its value is the set containing the categories that correspond semantically to the arguments of that predicate. The Japanese Phrase Structure Grammar (JPSG) of Gunji (in press) goes further in that it restricts itself entirely to such features (together with atom-valued features, of course) and does not employ simple category-valued features at all.

Both FUG and LFG also permit category-set values, in effect, though the interpretation they assign to the resulting objects is, once again, different. FUG's interpretation is, as with atom sets, disjunctive. On this interpretation, unification of two sets of categories can be defined as the set of categories each of whose members is the unification of a pair in their Cartesian product (again, see Karttunen (1984) for relevant discussion of this kind of approach). In LFG, sets of categories acting as values for single attributes are used in the analysis of adjuncts (and possibly coordination) and the interpretation is intendedly conjunctive Kaplan and Bresnan (1982). Under this interpretation, there is, in general, no unique unification to be had, although one can define an operation to provide one with a set of possible unifications. In Gunji (in press), where a conjunctive interpretation is assigned to category-set values, the non-uniqueness problem is sidestepped by defining **unify** as a predicate of category pairs, rather than as an operation.

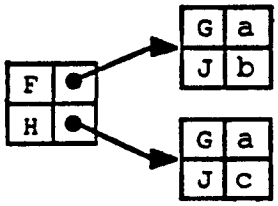
In view of all these considerations, we have opted for simplicity over generality and simply excluded set valued features from our purview.

6 Shared Values

One property that FUG and PATR II have in common, which sets them apart from the simpler grammar type discussed earlier in this paper, is the option of letting two or more distinct features share the same value. Thus, FUG functional descriptions allow one instance of a value to simultaneously be the value of more than one (instance of an) attribute. Consequently, the implicit hierarchy, represented graphically, does not respect the single-mother requirement that is built deep into our definitions. Of course, two category-valued features within a category may contingently have identical values, but this is not the same as **sharing** the same value (except in common parlance, perhaps). Kasper and Rounds (1986) refer to the distinction as one of type identity versus token identity. If we take a category, containing two contingently identical category-values, and unify it with a second category, then the contingent identity may not be preserved in the result. Consider, for example, the result of unifying these two categories:



where the values of F and H are identical in the first but not in the second. The result is:



and here the values of F and H are no longer identical. If the original common value had been genuinely shared, then no unification would have been possible (see also Shieber (1985) where the term "reentrancy" is used in this connection).

There is an alternative way of thinking about the problem of shared values, and that is to reconstruct it in terms of indexing: every value carries an index, and two structurally identical values are the very same thing if and only if they bear the same index. An integer indexing of this sort can be represented in the present framework as we have already see in section 4.5 above. However, a coindexing reconstruction would not be a sensible way of thinking about shared values in the present context since such a use of indices makes nonsense of structurally defined unification, subsumption, and so on. For two intuitively identical structures to unify, it would not be sufficient for them to exhibit the same internal patterns of coindexed values. Rather, they would need in addition to manifest the very same choice of indices. Clearly, this is not what one wants, as choice of index is completely arbitrary, and structures differing only in identity of the integers selected as indices should be regarded as equivalent.

To achieve a semantics for shared-value category formalisms, it is necessary to move beyond the partial function-based category structures that provide the basis for our semantics, and thus depart from the particular category constraint logic that it induces. Like set values, shared values are simply beyond the scope of the rather parsimonious theory of categories developed here.[5] The reader interested in pursuing richer approaches should consult Pereira and Shieber (1984) for a domain-theoretic account of the semantics of categories in LFG, PATR II, and GPSG; Ait-Kaci and Nasr (1986), who capture shared values with a coreference relation on the nodes of the tree; Kasper and Rounds (1986), Moshier and Rounds (1987), and Rounds and Kasper (1986) for a finite state automaton-based logic and semantics for categories in FUG and PATR II; and van Benthem (1986a, b) for an interesting foundational discussion and application of such an automaton-based semantics.

7 Conclusion

We have developed and applied a general framework for defining syntactic categories, including categories in which features can have categories as their value, which latter possibility turns out to subsume the possibility of a feature taking as its value a list of indices or categories, drawn from either a finite or an infinite set. The unitary way in which we have characterized these diverse systems is intended to assist in the exploration and comparison of grammatical formalisms. Questions concerning whether particular rule types and operations on categories that are familiar from one approach to grammar can be carried over unproblematically to another approach, and questions concerning the implementation difficulties that arise when a given formalism is adopted, can in many cases be settled in a straightforward and familiar way, namely by reducing them to previously encountered types of question.

The grammatical frameworks we have considered as examples fall into a five-class typology which we can now explicate. The first class contains the frameworks that use only atom-valued features (simple phrase structure grammar, Harman's augmented phrase structure grammar; RG and APG); the second contains the special case of *GKPS*, which uses category-valued features but imposes a constraint which prevents them from having effects on expressive power that could not ultimately by simulated by atom-valued features; the third contains the frameworks that use just a single category-valued feature (our key example being indexed grammar); the fourth contains frameworks making use of more than one category-valued feature (an example being categorial grammar); and the fifth includes those frameworks that fall outside the scheme we have developed in that their categories are not representable as finite partial functions constrained by statements in L_C (LFG, FUG, PATR II, etc.).

It is not at all clear which of these five classes of approaches will prove the most suitable for implementing natural language processing systems in the long term. In this paper, we hope to have made somewhat clearer the nature of the issues at stake. We hope also to have done something more: for the first four classes, we have provided what is in effect a unitary type of data structure for the representation of their syntactic categories. Thinking in terms of such data structures should make it possible for pseudo-issues in natural language processing research to be avoided in a large class of circumstances, to the point that even a decision in mid-project to change the grammatical framework from one linguistic approach to another need not entail any fundamental redesign of what are in most frameworks the basic objects of syntactic representation.

Appendix

In this appendix we restate the semantic rules for L_C more precisely. All well-formed expressions of L_C have

the same kind of denotation—they denote truth values (i.e., members of 2) relative to the category structure Σ and a category α determined by Σ. If ϕ is a well-formed expression of L_C, then we use $[\![\phi]\!]_{\Sigma,\alpha}$ to stand for the denotation of ϕ with respect to the category structure Σ and category α. If $[\![\phi]\!]_{\Sigma,\alpha} = 1$ then we shall say that α SATISFIES ϕ. The formal statement of our semantic rules is the following, where a, f, ϕ, and ψ are as above.

(A1) a. $[\![f]\!]_{\Sigma,\alpha} = 1$ iff $\alpha(f)$ is defined.
b. $[\![f{:}a]\!]_{\Sigma,\alpha} = 1$ iff $\alpha(f) = a$.
c. $[\![f{:}\phi]\!]_{\Sigma,\alpha} = 1$ iff $[\![\phi]\!]_{\Sigma,\alpha(f)} = 1$.
d. $[\![\neg\phi]\!]_{\Sigma,\alpha} = 1$ iff $[\![\phi]\!]_{\Sigma,\alpha} = 0$.
e. $[\![\phi \vee \psi]\!]_{\Sigma,\alpha} = 1$ iff $[\![\phi]\!]_{\Sigma,\alpha} = 1$ or $[\![\psi]\!]_{\Sigma,\alpha} = 1$.
f. $[\![\phi \wedge \psi]\!]_{\Sigma,\alpha} = 1$ iff $[\![\phi]\!]_{\Sigma,\alpha} = 1$ and $[\![\psi]\!]_{\Sigma,\alpha} = 1$.
g. $[\![\phi \rightarrow \psi]\!]_{\Sigma,\alpha} = 1$ iff $[\![\phi]\!]_{\Sigma,\alpha} = 0$ or $[\![\psi]\!]_{\Sigma,\alpha} = 1$.
h. $[\![\phi \leftrightarrow \psi]\!]_{\Sigma,\alpha} = 1$ iff $[\![\phi]\!]_{\Sigma,\alpha} = [\![\psi]\!]_{\Sigma,\alpha}$.
i. $[\![\Box\phi]\!]_{\Sigma,\alpha} = 1$ iff $[\![\phi]\!]_{\Sigma,\alpha} = 1$ and for all f in $F^1 \cap \Delta(\alpha), [\![\Box\phi]\!]_{\Sigma,\alpha(f)} = 1$.
j. $[\![\Diamond\phi]\!]_{\Sigma,\alpha} = 1$ iff $[\![\phi]\!]_{\Sigma,\alpha} = 1$ or for some f in $F^1 \cap \Delta(\alpha), [\![\Diamond\phi]\!]_{\Sigma,\alpha(f)} = 1$.

Note that if $\alpha \sqsubseteq \beta$ and α satisfies ϕ, it does NOT follow in L_C that β satisfies ϕ (compare Rounds and Kasper (1986), Theorem 6). For example, we have $\varnothing \sqsubseteq \{\langle F, a\rangle\}$ and $\varnothing$ satisfies $\neg F$, but $\{\langle F, a\rangle\}$ does not. Likewise, the fact that both α and β satisfy some constraint ϕ does not entail that $\alpha \sqcup \beta$ will satisfy ϕ, even if $\alpha \sqcup \beta$ is defined. The desire to incorporate negation whilst maintaining an upward closure property lead Moshier and Rounds (1987) to set aside a classical semantics for their feature description language and postulate an intuitionistic se mantics that, in effect, quantifies over possible extensions.

We will write $\models\phi$ to mean that for every category structure Σ and category α in Σ, α satisfies ϕ. Given this, we can list some valid formulæ and valid formula schemata of the logic of category constraints.

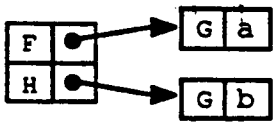
(A2) a. $\models(f{:}a) \rightarrow f$ (for all $a \in \rho(f), f \in F^0$)

This simply says that if a feature has an atomic value, then it has a value. We also have all the valid formulæ of the standard propositional calculus, which we will not list here. Furthermore, we have the following familiar valid modal formulæ.

(A2) b. $\models\Box\phi \leftrightarrow \neg\Diamond\neg\phi$
c. $\models\Box(\phi \rightarrow \phi)$
d. $\models\Box\phi \rightarrow \phi$
e. $\models\phi \rightarrow \Diamond\phi$
f. $\models\Box(\phi \wedge \psi) \leftrightarrow (\Box\phi \wedge \Box\psi)$
g. $\models\Diamond(\phi \vee \psi) \leftrightarrow (\Diamond\phi \vee \Diamond\psi)$
h. $\models\Box\phi \rightarrow \Box\Box\phi$

Here, (A2h) shows us that our logic at least contains **S4** (we follow the nomenclature of Hughes and Cresswell (1968) throughout). But we do not have $\models\Diamond\phi \rightarrow \Box\Diamond\phi$, and so our logic does not contain **S5**. To see this, consider the following category, assuming F is a category-valued feature: $\{\langle F, \varnothing\rangle\}$. This category satisfies $\Diamond F$ but not $\Box\Diamond F$.

The category $\{\langle F, \{\langle G, a\rangle\}\rangle, \langle H, \{\langle G, b\rangle\}\rangle\}$ (graphically represented in (50), below) provides us with an analogous falsifying instance for $\models\Diamond\Box\phi \rightarrow \Box\Diamond\phi$ when we set $\phi = (G{:}a)$.

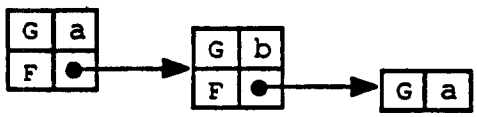


This shows that our logic does not contain **S4.2**. Interestingly, the converse of this constraint IS valid, hence:

(A2) i. $\models\Box\Diamond\phi \rightarrow \Diamond\Box\phi$

This is easy to demonstrate: if α satisfies $\Box\Diamond\phi$ then $\Diamond\phi$ must hold in all the categories that terminate α, and if $\Diamond\phi$ holds in those categories, then ϕ and $\Box\phi$ hold in them as well. So $\Box\phi$ holds in at least one category in α, and thus α must satisfy $\Diamond\Box\phi$. This shows that our logic at least contains **K1** and, as a consequence, is not contained by **S5**.

However, our logic cannot contain **K2**, since the latter contains **S4.2**. Nor does it contain **K1.2** since the latter's characteristic axiom, namely $\models\phi \rightarrow \Box(\Diamond\phi \rightarrow \phi)$ is shown to be invalid by the category $\{\langle G, a\rangle, \langle F, \{\langle G, b\rangle, \langle F, \{\langle G, a\rangle\}\rangle\}\rangle\}$ (shown in (51), below) when set set $\phi = (G{:}a)$.



In fact, our logic does not merely contain **K1**, it also contains **K1.1**, whose characteristic axiom is:

(A2) j. $\models\Box(\Box(\phi \rightarrow \Box\phi) \rightarrow \phi) \rightarrow \phi)$

Hughes and Cresswell note that **K1.1** 'is characterized by the class of all finite partial orderings, i.e., finite frames in which ***R*** [the accessibility relation] is reflexive, transitive, and antisymmetrical' Hughes and Cresswell ((1984), p. 162). So it should be no surprise, given the basis for our semantics, that our logic turns out to include **K1.1**. This logic, also known as **S4Grz** (after Grzegorczyk (1967)), 'is decidable, for every nontheorem of **S4Grz** is invalid in some *finite* weak partial ordering' (Boolos (1979, p. 167).

Two further valid formula schemata of L_C have some interest, before we conclude the list of valid formulæ in (A2):

(A2) k. $\models\Diamond\neg f$ (for all $f \in F^1$)
l. $\models(f{:}\phi) \rightarrow \Diamond\phi$ (for all $f \in F^1$)

The first of these follows from the fact that categories are finite in size and thus ultimately grounded in categories that contain no category-valued features: f must be false of these terminating embedded categories, and hence $\Diamond\neg f$ must be true of the category as a whole.

The second states that if a category is defined for a category-valued feature whose value satisfies ϕ, then the category as a whole satisfies $\Diamond\phi$.

(A2) m. $\models(f{:}\phi) \rightarrow f$ (for all $f \in F^1$)
n. $\models((f{:}\phi) \wedge (f{:}\psi)) \leftrightarrow (f{:}\phi \wedge \psi)$ (for all $f \in F^1$)
o. $\models((f{:}\phi) \vee (f{:}\psi)) \leftrightarrow (f{:}\phi \vee \psi)$ (for all $f \in F^1$)

It is worth considering the valid formulæ one would get in certain restricted classes of category structures. Suppose we consider category structures which contain only atom-valued features (i.e., $F = F^0$). In this case, as one would expect, the modal logic collapses into the propositional calculus and the relevant notion of validity (call it $\models_0$) gives us the following:

(A5) $\models_0\phi \leftrightarrow \Box\phi$

The converse case, where we only permit category-valued features (i.e. $F = F^1$), is uninteresting, since it is not distinct from the general case. We can always encode atom-valued features as (sets of) category-valued features and subject the latter to appropriate constraints, as follows. For every feature specification $\langle f, a\rangle$ such that $f \in F^0$ and $a \in \rho(f)$, we introduce a new type 1 feature fa and use the presence of $\langle fa, \varnothing\rangle$ to encode the presence of $\langle f, a\rangle$ and likewise absence to encode absence. Then, for each pair of atoms a and b in $\rho(f)$, we require the new features to satisfy $\Box\neg(fa \wedge fb)$. And to constrain each new feature fa to have the empty set as its value, we stipulate $\Box\neg(fa{:}g)$ for every feature g.

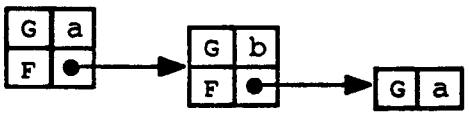
However, consider validity in category structures containing at most one category-valued feature (call this kind of validity $\models_1$). With this restriction, the **S4.2** axiom considered earlier becomes valid:

(A6) $\models_1 \Diamond\Box\phi \rightarrow \Box\Diamond\phi$

In addition, we get (A7).

(A7) $\models_1 \Box(\Box\phi \rightarrow \Box\psi) \vee \Box(\Box\psi \rightarrow \Box\phi)$

This means that this restricted logic at least contains **K3**, but it cannot contain **K4**, since $\models_1\phi \rightarrow (\Diamond\Box\phi \rightarrow \Box\phi)$ is falsified by the category $\{\langle \mathrm{G}, a\rangle, \langle \mathrm{F}\ \{\langle \mathrm{G}, b\rangle, \langle \mathrm{F}, \{\langle \mathrm{G}, a\rangle\}\}\rangle\}$ when we set $\phi = (\mathrm{G}{:}\ a)$.



In fact it must also contain **K3.1**, in view of the validity of (A2j) above, and this logic, also known as **S4.3Grz**, is characterized by finite linear orderings Hughes and Cresswell (1984). This is the characterization we would expect given the character of the $\models_1$ restriction on the form of permissible categories, since with only one category-valued feature, there is at most one path through the structure of a category and so the partial order becomes a linear order. These observations concerning the logic induced by category structures where $|F^1| = 1$ are of some potential relevance to the study of indexed grammars whose categories can be construed as being restricted in just this way (see section 4.9, above).

Acknowledgments

Chris Barker has contributed substantively to the research reported here, and we offer him our gratitude. We are also grateful to Edward Briscoe, Jeremy Carroll, Roger Evans, Joseph Halpern, David J. Israel, Ronald M. Kaplan, William Keller, James Kilbury, William A. Ladusaw, Christopher Mellish, Richard E. Otte, Fernando Pereira, P. Stanley Peters, Carl J. Pollard, Stephen Pulman, William Rounds, Stuart M. Shieber, Henry Thompson and Manfred Warmuth for their generous assistance during the research reported in this paper. Though in some respects they have contributed substantially, they should not be associated with any errors that the paper may contain. In addition, we thank Calvin J. Pullum, who is responsible for the diagrams, and we acknowledge partial research support from the following sources: the UCSC Syntax Research Center (Gazdar, Hukari, Levine, Pullum); grants from the (U.K.) SERC and ESRC (Gazdar); NSF Graduate Fellowship RCD-8651747 (Carpenter); NSF grants BNS-85 11687 and BNS-85 19708 (Pullum).

References

Aho, Alfred V. 1968 Indexed Grammars. *Journal of the Association for Computing Machinery* 15: 647–671.

Ait-Kaci, Hassan; and Nasr, Roger. 1986 *Proceedings of the 13th Annual ACM Conference on Principles of Programming Languages*: 219–228. Association for Computing Machinery.

Bach, Emmon. 1984 Some Generalizations of Categorial Grammar. In Landman, Fred; and Veltman, Frank, Eds., *Varieties of Formal Semantics: Proceedings of the 4th Amsterdam Colloquium, September 1982*, Foris, Dordrecht, Holland: 1–23.

van Benthem, Johan. 1986a Semantic Automata. In Groenendijk, Joroen; de Jongh, Dick; and Stokhof, Martin, Eds., *Information, Interpretation and Inference*. Foris, Dordrecht, Holland. Reprinted in van Benthem, Johan. 1986 *Essays in Logical Semantics*. D. Reidel, Dordrecht, Holland: 151–176. [Also published as CSLI Report 85-27, Center for the Study of Language and Information, Stanford, 1985]

van Benthem, Johan. 1986b Towards a Computational Semantics: In Cooper, Robin; Engdahl, Elisabet; and Gardenfors, P., Eds., *Proceedings of a Workshop on Generalized Quantifiers, Lund 1985*. D. Reidel, Dordrecht, Holland.

van Benthem, Johan. 1986c Categorial Grammar. In Johan van Benthem. 1986 *Essays in Logical Semantics*. D. Reidel, Dordrecht, Holland: 123–150.

van Benthem, Johan; Buszkowski, W.; and Marciszewski, W., Eds., *Categorial Grammar*. John Benjamin, Amsterdam, Holland.

Boolos, George. 1979 *The Unprovability of Consistency*. Cambridge University Press, Cambridge, England.

Bresnan, Joan W. 1975 Transformations and Categories in Syntax. In Butts, Ronald; and Hintikka, Jaakko, Eds., *Basic Problems in Methodology and Linguistics*. D. Reidel, Dordrecht, Holland: 283–304.

Chomsky, Noam. 1970 Remarks on Nominalization. In Jacobs, R.; and Rosenbaum, P., Eds., *Readings in English Transformational Grammar*. Ginn, Waltham, Massachusetts: 11–61.

Chomsky, Noam. 1980 On Binding. *Linguistic Inquiry* 11: 1–46.

Chomsky, Noam. 1981 *Lectures on Government and Binding*. Dordrecht: Foris.

Elson, Benjamin; and Pickett, Velma. 1962 *An Introduction to Morphology and Syntax*. Summer Institute of Linguistics, Santa Ana, California.

Gazdar, Gerald. 1985 Applicability of Indexed Grammars to Natural Languages. Center for the Study of Language and Information, Stanford, California: Report No. CSLI-85-34.

Gazdar, Gerald; Klein, Ewan; Pullum, Geoffrey K.; and Sag, Ivan A. 1985 *Generalized Phrase Structure Grammar*. Harvard University Press, Cambridge, Massachusetts.

Grzegorczyk, Andrzej. 1967 Some Relational Systems and the Associated Topological Spaces. *Fundamentae Mathematicae* 60: 223–231.

Haddock, Nicholas; Klein, Ewan; and Morrill, Glyn, Eds., 1987 *Categorial Grammar, Unification Grammar and Parsing*. Edinburgh Working Papers in Cognitive Science 1, Edinburgh, Scotland.

Halliday, Michael A. K. 1961 Categories of the Theory of Grammar. *Word* 17:241–292.

Halvorsen, Per-Kristian; and Ladusaw, William A. 1979 Montague's 'Universal Grammar': an Introduction for the Linguist. *Linguistics and Philosophy* 3: 185–223.

Harman, Gilbert H. 1963 Generative Grammars without Transformation Rules: a Defense of Phrase Structure. *Language* 39: 597–616.

Harris, Zellig S. 1951 *Methods in Structural Linguistics*. University of Chicago Press, Chicago, Illinois.

Hendriks, Herman. 1986 *Foundations of GPSG Syntax*. Doctoraalscriptie Wijsbegeerte, University of Amsterdam, Amsterdam, Holland.

Hornstein, Norbert. 1977 S and X′ Convention. *Linguistic Analysis* 3: 137–176.

Hudson, Richard A. 1971 *English Complex Sentences*. North Holland, Amsterdam, Holland.

Hughes, G. E.; and Cresswell, Max J. 1968 *An Introduction to Modal Logic*. Methuen, London, England.

Hughes, G. E.; and Cresswell, Max J. 1984 *A Companion to Modal Logic*. Methuen, London, England.

Jackendoff, Ray. 1974 *Introduction to the $\overline{X}$ Convention*. Indiana University Linguistics Club, Bloomington, Indiana.

Jackendoff, Ray. 1977 *$\overline{X}$ Syntax: A Study of Phrase Structure*. MIT Press, Cambridge, Massachusetts.

Johnson, David E.; and Postal, Paul M. 1980 *Arc Pair Grammar*. Princeton University Press, Princeton, New Jersey.

Kaplan, Ronald; and Bresnan, Joan. 1982 Lexical-Functional Grammar: a Formal System for Grammatical Representation. In J. W. Bresnan, Ed., *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, Massachusetts: 173–281.

Karttunen, Lauri. 1984 Features and Values. *Proceedings of the 10th International Conference on Computational Linguistics and the 22nd Annual Meeting of the Association for Computational Linguistics*. Stanford University, California: 28–33.

Karttunen, Lauri; and Zwicky, Arnold M. 1985 Introduction to Dowty, D.R.; Karttunen, L.; and Zwicky, A.M., Eds., *Natural Language Parsing*. Cambridge University Press, Cambridge, England: 1–25.

Kasper, Robert T.; and Rounds, William C. 1986 A Logical Semantics for Feature Structures. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*: 257–266.

Kay, Martin. 1979 Functional Grammar. In Chiarrello, Christine et al., Eds., *Proceedings of the 5th Annual Meeting of the Berkeley Linguistics Society*: 142–158.

Kay, Martin. 1985 Parsing in Functional Unification Grammar. In Dowty, D.R.; Karttunen, L.; and Zwicky, A.M., Eds., *Natural Language Parsing*. Cambridge University Press, Cambridge, England: 251–278.

Ladusaw, William A. 1985 A Proposed Distinction Between *Levels* and *Strata*. Presented to the Annual Meeting of the Linguistic Society of America, Seattle, Washington. Memo no. SRC-85-04, Syntax Research Center, University of California, Santa Cruz, California.

Lasnik, Howard; and Kupin, Joseph J. 1977 A Restrictive Theory of Transformational Grammar. *Theoretical Linguistics* 4: 173–196.

Levy, Leon S.; and Joshi, Aravind, K. 1978 Skeletal Structural Descriptions. *Information and Control* 39: 192–211.

Longacre, Robert E. 1965 Some Fundamental Insights of Tagmemics. *Language* 41:65–76.

Mellish, Christopher. 1986 Implementing Systemic Classification by Unification. Manuscript, University of Sussex.

Montague, Richard. 1970 Universal Grammar. In Thomason, Richmond H., Ed., *Formal Philosophy*. Yale University Press, New Haven, Connecticut: 222–246.

Montague, Richard. 1973 The Proper Treatment of Quantification in Ordinary English. In Thomason, Richmond H., Ed., *Formal Philosophy*. Yale University Press, New Haven, Connecticut: 247–270.

Moshier, M. D., and Rounds, William C. 1987 A Logic for Partially Specified Data Structures. *Proceedings of the ACM Conference on Principles of Programming Languages*, Munich.

Oehrle, Richard T.; Bach, Emmon; and Wheeler, Deirdre W., Eds., 1987 *Categorial Grammars and Natural Language Structures*, D. Reidel, Dordrecht, Holland.

Patten, Terry; and Ritchie, Graeme. 1987 A Formal Model of Systemic Grammar. In Kempen, Gerard, Ed., *Natural Language Generation: Recent Advances in AI, Psychology and Linguistics*. Kluwer, Amsterdam, Holland.

Pereira, Fernando C. N.; and Shieber, Stuart M. 1984 The Semantics of Grammar Formalisms Seen as Computer Languages. In *Proceedings of the 10th International Conference on Computational Linguistics and the 22nd Annual Meeting of the Association for Computational Linguistics*: 123–129.

Pereira, Fernando C. N.; and Warren, David H. D. 1980 Definite Clause Grammars for Language Analysis—a Survey of the Formalism and a Comparison with Augmented Transition Networks. *Artificial Intelligence* 13: 231–278.

Perlmutter, David M.; and Postal, Paul M. 1977 Toward a Universal Characterization of Passivization. In Whistler, Kenneth et al., Eds., *Proceedings of the 3rd Annual Meeting of the Berkeley Linguistics Society* 394–417. Reprinted in: Perlmutter, David M., Ed., *Studies in Relational Grammar 1*. University of Chicago Press, Chicago, Illinois.

Pollard, Carl J. 1984 *Generalized Phrase Structure Grammars, Head Grammars, and Natural Languages*. Ph.D. dissertation, Stanford University.

Pollard, Carl. 1985 Phrase Structure Grammar Without Metarules. Goldberg, Jeffrey; MacKaye, Susannah; and Wescoat, Michael, Eds., *Proceedings of the West Coast Conference on Formal Linguistics, Volume Four*. Stanford Linguistics Association, Stanford, California: 246–261.

Postal, Paul M. 1964 *Constituent Structure: A Study of Contemporary Models of Syntactic Description*. Publication 30 of the Indiana University Research Center in Anthropology, Folklore, and Linguistics, Bloomington, Indiana.

Pullum, Geoffrey K. 1985 Assuming Some Version of X-Bar Theory. In Eilfort, William D.; Kroeber, Paul D.; Peterson, Karen L., Eds., *CLS 21, Part 1: Papers from the General Session at the Twenty-First Regional Meeting*. Chicago Linguistic Society, Chicago, Illinois: 323–353.

Ristad, Eric Sven. 1986 Computational Complexity of Current GPSG Theory. *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*: 30–39.

Rounds, William C.; and Kasper, Robert T. 1986 A Complete Logical Calculus for Record Structures Representing Linguistic Informa-

tion. *Proceedings of the 15th Annual Symposium on Logic in Computer Science*. Cambridge, Massachusetts.

Rouveret, Alain; and Vergnaud, Jean-Roger. 1980 Specifying Reference to the Subject: French Causatives and Conditions on Representations. *Linguistic Inquiry* 11: 97–202.

Sabimana, Firmard. 1986 *The Relational Structure of the Kirundi Verb*. D.Phil. dissertation, Indiana University, Bloomington, Indiana.

Shieber, Stuart. 1984 The Design of a Computer Language for Linguistic Information. In *Proceedings of the 10th International Conference on Computational Linguistics and the 22nd Annual Meeting of the Association for Computational Linguistics*: 362–366.

Shieber, Stuart. 1985 Criteria for Designing Computer Facilities for Linguistic Analysis. Linguistics 23: 189–211.

Shieber, Stuart. 1987 Separating Linguistic Analyses from Linguistic Theories. In Whitelock, Peter J. et al., Eds., *Linguistic Theory and Computer Applications*. Academic Press, London.

Stockwell, Robert P.; Schacter, Paul; Partee, Barbara H. 1973 *The Major Syntactic Structures of English*. Holt, Rinehart and Winston, New York, New York.

Winograd, Terry. 1972 *Understanding Natural Language*. Academic Press, New York, New York.

Winograd, Terry. 1983 *Language as a Cognitive Process: Volume 1 Syntax*. Addison-Wesley, Reading, Massachusetts.

Notes

1. Bresnan (1975) correctly attributes the [±N, ±V] feature system to lectures delivered by Chomsky at the 1974 Linguistic Institute in Amherst, Massachusetts. In some works, e.g., Jackendoff (1977) and Gazdar, Klein, Pullum, and Sag (1985), Chomsky (1970) is wrongly given as the source. The latter work does, however, contain the following relevant comment: "we might just as well eliminate the distinction of feature and category, and regard all symbols of the grammar as sets of features" (p. 208).
2. As Hendriks (1986) has noted, the definition of categories given in *GKPS* "is a bit of a mess from a formal point of view" (1986, p. 19). Definition 1 reads as follows: "ρ^0 is a function from F to $POW(A)$ such that for all $f \in (F-Atom)$, $\rho^0(f) = \{\{\}\}$" (*GKPS*, p. 36). But $\{\{\}\}$ is not in the power set of A; "$POW(A)$" should be replaced by "$POW(A) \cup \{\{\{\}\}\}$". Parts of the text and examples following Definition 1 assume correctly that it ends "$\rho^0(f) = \{\{\}\}$", but other parts assume incorrectly that it ends "$\rho^0(f) = \{\}$". If the latter version were adopted, Definition 4 would fail to add category-valued feature specifications in the desired way (since the condition "$\exists C' \in \rho^{n-1}(f)[C' \subseteq C]$" would never be satisfied where $n = 1$.)
3. The "feature cooccurrence restrictions" (FCRs) of *GKPS* form part of the definition of **admissible tree** rather than being part of the definition of categories. However, every *GKPS* FCR can be expressed in L_C, and the translation is trivial.
4. We are indebted to Joseph Halpern for his help with the material in this section.
5. One of our referees has suggested that our semantics can be made to handle sharing by introducing an equality predicate into L_C, marking shared value situations with special nonce features, and then using conditional constraints triggered by these features to impose identical values on the relevant features. But we have been unable to get any scheme of this kind to work in the general case. There appears to be no upper bound to the number of nonce features that may be required, and moreover, unification ceases to behave in an intuitively reasonable manner.