Talking About Trees

Patrick Blackburn
Department of Philosophy, Rijksuniversiteit Utrecht
Heidelberglaan 8, 3584 CS Utrecht. Email: patrick@phil.ruu.nl

Claire Gardent
GRIL, Université de Clermont Ferrand, France, and
Department of Computational Linguistics, Universiteit van Amsterdam
Spuistraat 134, 1012 VB Amsterdam. Email: claire@mars.let.uva.nl

Wilfried Meyer-Viol
Centrum voor Wiskunde en Informatica
Kruislaan 413, 1098 SJ Amsterdam. Email: W.Meyer.Viol@cwi.nl

Abstract

In this paper we introduce a modal language L^T for imposing constraints on trees, and an extension $L^T(L^F)$ for imposing constraints on trees decorated with feature structures. The motivation for introducing these languages is to provide tools for formalising grammatical frameworks perspicuously, and the paper illustrates this by showing how the leading ideas of GPSG can be captured in $L^T(L^F)$.

In addition, the role of modal languages (and in particular, what we have called *layered modal languages*) as constraint formalisms for linguistic theorising is discussed in some detail.

1 Introduction

In this paper we introduce a modal language L^T for talking about trees, and an extension $L^T(L^F)$ for talking about trees decorated with feature structures. From a logical point of view this is a natural thing to do. After all, the trees and feature structures used in linguistics are simple graphical objects. To put it another way, they are merely rather simple Kripke models, and modal languages are probably the simplest languages in which non-trivial constraints can be imposed on such structures. Moreover the approach is also linguistically natural: many of the things linguists need to say about trees (and feature structures) give rise to modal operators rather naturally, and indeed our choice of modalities has been guided by linguistic practice, not logical convenience.

There are several reasons why we think this path is an interesting one to explore, however two are of particular relevance to the present paper.[1] First, we believe that it can lead to relatively simple and natural formalisations of various grammatical frameworks. In our view, neither simplicity nor naturalness are luxuries: unless a formalisation possesses a high degree of clarity, it is unrealistic to hope that it can offer either precise analyses of particular systems or informative comparisons of different frameworks. We believe our approach has the requisite clarity (largely because it arose by abstracting from linguistic practice in a rather direct manner) and much of this paper is an attempt to substantiate this. Second, L^T can be combined in a very natural way with feature logics to yield simple systems which deal with configurational concepts, complex categories and their interaction. The key idea is to perform this combination of logics in a highly constrained way which we have called *layering*. Layering is a relatively new idea in modal logic (in fact the only paper devoted exclusively to this topic seems to be [Finger and Gabbay 1992]), and it seems to provide the right level of expressive power needed to model many contemporary grammar formalisms.

The paper is structured as follows. In section 2 we define the syntax and semantics of L^T, our modal language for imposing constraints on tree structure.

---

[1] Lurking in the background are two additional, rather more technical, reasons for our interest. First, we believe that being *explicit* about tree structure in our logical object languages (instead of, say, coding tree structure up as just another complex feature) may make it easier to find computationally tractable logics for linguistic processing. Second, we believe that logical methods may interact fruitfully with the mathematical literature on tree admissibility (see [Peters and Ritchie 1969], [Rounds 1970] and [Joshi and Levy 1977]). However we won't explore these ideas here.

In section 3 we put L^T to work, showing how it can be used to characterise the parse trees of context free phrase structure grammars. In section 4 we consider how the structured categories prevalent in modern linguistic formalisms are dealt with. Our solution is to introduce a simple feature logic L^F for talking about complex categories, and then to layer L^T across L^F. The resulting system $L^T(L^F)$ is capable of formulating constraints involving the interaction of configurational and categorial information. Section 5 illustrates how one might use this expressive power by formulating some of the leading ideas of GPSG in $L^T(L^F)$. We conclude the paper with some general remarks on the use of modal languages as constraint formalisms in linguistics.

2 The language L^T

The primitive alphabet of the language $L^T(Prop)$ contains the following items: two constant symbols s and t, some truth functionally adequate collection of Boolean operators,[2] two unary modalities $\downarrow$ and $\uparrow$, a binary modality $\Rightarrow$, a modality of arbitrary positive arity $\bullet$, the left bracket (and the right bracket). In addition we have a set of propositional symbols $Prop$. We think of the symbols in $Prop$ as given to us by the linguistic theory under consideration; different applications may well result in different choices of $Prop$. To give a very simple example, $Prop$ might be $\{S, NP, VP, N, V, DET, CONJ\}$.

The wffs of $L^T(Prop)$ are defined as follows. First, all elements of $Prop$ are $L^T(Prop)$ wffs, and so are the constant symbols s and t. Second, if ϕ, ψ and $\phi_1, \ldots, \phi_n$ $(n \geq 1)$ are $L^T(Prop)$ wffs, then so are $\neg\phi$, $(\phi \wedge \psi)$, $\uparrow\phi$, $\downarrow\psi$, $(\phi \Rightarrow \psi)$ and $\bullet(\phi_1, \ldots, \phi_n)$. Third, nothing else is an $L^T(Prop)$ wff. In what follows, we will assume that some choice of $Prop$ has been fixed and drop all subsequent mention of it. That is, we'll usually speak simply of the language L^T.

The semantics of L^T is given in terms of finite ordered trees. We regard finite trees $\mathbf{T}$ as quadruples of the form $\langle \Gamma, >, \Theta, root \rangle$, where Γ is a *finite* set of nodes, $>$ is the 'Mother of' relation between nodes ('$u > v$' means 'u is the mother of v'), Θ ($\subseteq \Gamma$) is the set of terminal nodes and $root$ is the root node of the tree. As for the precedence ordering, we define:

Definition 2.1 (Finite ordered trees)
A *finite ordered tree* is a pair $\mathbf{O} = \langle \mathbf{T}, \lambda \rangle$ where $\mathbf{T}$ is a tree $\langle \Gamma, >, \Theta, root \rangle$ and λ is a function assigning to each node u in Γ a finite sequence of nodes of Γ. For all nodes $u \in \Gamma$, $\lambda(u)$ must satisfy two constraints. First, $\lambda(u)$ must contain no repetitions. Second, $\lambda(u) = \langle u_1, \ldots, u_k \rangle$ iff $u > u_1, \ldots, u > u_k$

and there is no node u' in Γ such that $u > u'$ and u' does not occur in the sequence $\lambda(u)$. □

(In short, the repetition-free sequence assigned to a node u by λ consists of all and only the nodes immediately dominated by u; the sequence gives us a precedence ordering on these nodes. Note that it follows from this definition that $\lambda(u)$ is the null-sequence iff u is a terminal node.) Next, we define a *model* $\mathbf{M}$ to be a pair $\langle \mathbf{O}, V \rangle$ where $\mathbf{O}$ is a finite ordered tree $\langle \mathbf{T}, \lambda \rangle$ and V is a function from $Prop$ to the powerset of Γ. That is, V assigns to each propositional symbol a set of nodes. Given any model $\mathbf{M}$ and any node u of $\mathbf{M}$, the satisfaction relation $\mathbf{M}, u \models \phi$, (that is, the model $\mathbf{M}$ satisfies ϕ at node u) is defined as follows:

$\mathbf{M}, u \models p$	*iff*	$u \in V(p)$, for all $p \in Prop$
$\mathbf{M}, u \models s$	*iff*	$u = root$
$\mathbf{M}, u \models t$	*iff*	$u \in \Theta$
$\mathbf{M}, u \models \neg\phi$	*iff*	not $\mathbf{M}, u \models \phi$
$\mathbf{M}, u \models \phi \wedge \psi$	*iff*	$\mathbf{M}, u \models \phi$ and $\mathbf{M}, u \models \psi$
$\mathbf{M}, u \models \downarrow\phi$	*iff*	$\exists u' \in \Gamma(u > u'$ and $\mathbf{M}, u' \models \phi)$
$\mathbf{M}, u \models \uparrow\phi$	*iff*	$\exists u' \in \Gamma(u' > u$ and $\mathbf{M}, u' \models \phi)$
$\mathbf{M}, u \models \phi \Rightarrow \psi$	*iff*	$\forall u' \in \Gamma(\mathbf{M}, u' \models \phi$ implies $\mathbf{M}, u' \models \psi)$
$\mathbf{M}, u \models \bullet(\phi_1, \ldots, \phi_k)$	*iff*	$length(\lambda(u)) = k$ and $\mathbf{M}, \lambda(u)(i) \models \phi_i$, for all $1 \leq i \leq k$

(In the satisfaction clause for $\bullet$, $\lambda(u)(i)$ denotes the ith element of the sequence assigned to u by λ.) If ϕ is satisfied at all nodes of a model $\mathbf{M}$, then we say that ϕ is *valid* on $\mathbf{M}$ and write $\mathbf{M} \models \phi$. The notion of validity has an important role to play for us. As we shall see in the next section, we think of a grammar G as being represented by an L^T wff ϕ^G. The trees admitted by the grammar are precisely those models on which ϕ^G is valid.

Note that L^T is just a simple formalisation of linguistic discourse concerning tree structure. First, $\downarrow$ and $\uparrow$ enable us to say that a daughter node, or a mother node, do (or do not) bear certain pieces of linguistic information. For example, $\downarrow\phi$ insists that the information ϕ is instantiated on some daughter node. Second, $\Rightarrow$ enables general constraints about tree structure to be made: to insist that $\phi \Rightarrow \psi$ is to say that any node in the tree that bears the information ϕ must also bear the information ψ. Finally $\bullet$ enables admissibility conditions on local trees to be stated. That is, $\bullet$ is a modal operator that embodies the idea of local tree admissibility introduced by [McCawley 1968].[3] It enables us to insist that a node

---

[2]In what follows we treat $\neg$ and $\wedge$ as primitive, and the other Boolean symbols such as $\vee$ (disjunction), $\rightarrow$ (material implication), $\leftrightarrow$ (material equivalence), $\top$ (constant true) and $\bot$ (constant false) as abbreviations defined in the familiar manner.

[3]As well as McCawley's article, see also [Gazdar 1979]. Our treatment of node admissibility conditions has been heavily influenced by Gazdar's paper and later work in the GPSG tradition (for example, [Gazdar *et al.* 1985]).

must immediately and exhaustively dominate nodes bearing the listed information, and in the following section we'll see how to put it to work.[4]

Although the design of L^T was guided by linguistic considerations, with the exception of $\bullet$ it turns out to be a rather conventional language.[5] In particular, $\Rightarrow$ is what modal logicians call *strict implication*, and $\downarrow$ and $\uparrow$ together form a particularly simple example of what modal logicians like to call a 'tense logic'.

In what follows we will occasionally make use of the following (standard) modal abbreviations: $\Box\phi =_{def} \top \Rightarrow \phi$ (that is, ϕ is satisfied at all nodes) and $\Uparrow\phi =_{def} \neg \uparrow\neg\phi$ (that is, ϕ is true at the mother of the node we are evaluating at, if in fact this node has a mother).

3 Talking about trees

In this section we show by example how to use L^T to formulate node admissibility conditions that will pick out precisely the parse trees of context free grammars. Consider the context free phrase structure grammar $G = \langle S, N, T, P \rangle$ where S is the start symbol of G; where N, the set of non-terminal symbols, is $\{S, NP, VP, N, V, DET, CONJ\}$; where T, the set of terminal symbols, is $\{the, a, man, woman, donkey, beat, stroke, and, but\}$; and where P, the set of productions, is:

$$
\begin{aligned}
S &\longrightarrow NP\ VP \mid S\ CONJ\ S \\
NP &\longrightarrow DET\ N \\
VP &\longrightarrow V\ NP \\
N &\longrightarrow man \mid woman \mid donkey \\
V &\longrightarrow beat \mid stroke \\
DET &\longrightarrow the \mid a \\
CONJ &\longrightarrow and \mid but
\end{aligned}
$$

Let's consider how to capture the parse trees of this grammar by means of constraints formulated in L^T. The first step is to fix our choice of *Prop*. We choose it to be $N \cup T$, that is, all the terminal and non-terminal symbols of G. The second step is to capture the effect of the productions P. We do so as follows. Let ϕ^P be the conjunction of the following wffs:

$$
\begin{aligned}
S &\Rightarrow \bullet(NP, VP) \vee \bullet(S, CONJ, S) \\
NP &\Rightarrow \bullet(DET, N) \\
VP &\Rightarrow \bullet(V, NP) \\
N &\Rightarrow \bullet(man) \vee \bullet(woman) \vee \bullet(donkey) \\
V &\Rightarrow \bullet(beat) \vee \bullet(stroke) \\
DET &\Rightarrow \bullet(the) \vee \bullet(a) \\
CONJ &\Rightarrow \bullet(and) \vee \bullet(but)
\end{aligned}
$$

Note that each conjunct licences a certain information distribution on local trees. Now, any parse tree for G can be regarded as an L^T model, and because each conjunct in ϕ^P mimics in very obvious fashion a production in G, each node of an L^G parse tree is licenced by one of the conjuncts. To put it another way, ϕ^P is valid on all the G parse trees.

However we want to capture *all and only* the parse trees for G.

This is easily done: we need merely express in our language certain general facts about parse trees. First, we insist that each node is labeled by at least one propositional symbol: $\Box(\bigvee_{p\in N\cup T} p)$ achieves this. Second, we insist that each node is labeled by at most one propositional symbol: $(p \Rightarrow \bigwedge_{q\in((N\cup T)\setminus\{p\})} \neg q)$ achieves this. Third, we insist that the root of the tree be decorated by the start symbol S of the grammar: $s \Rightarrow S$ achieves this. Fourth, we insist that non-terminal symbols label only nonterminal nodes: $\bigwedge_{p\in N}(p \Rightarrow \neg t)$ achieves this. Finally, we insist that terminal symbols label terminal nodes: $\bigwedge_{p\in T}(p \Rightarrow t)$ achieves this. Call the conjunction of these five wffs ϕ^U; note that, modulo our choice of the particular sets N and T, ϕ^U expresses *universal* facts about parse trees.

Now, for the final step: let ϕ^G be $\phi^P \wedge \phi^U$. That is, ϕ^G expresses both the productions P of G and the universal facts about parse tree structure. It is easy to see that for any model M, M $\models \phi^G$ iff M is (isomorphic to) a parse tree for G. Indeed, it is not hard to see that the method we used to express G as a formula generalises to any context free phrase structure grammar.

4 Trees decorated with feature structures

The previous section showed that L^T is powerful enough to get to grip with interesting 'languages' in the sense of formal language theory; but although natural language syntacticians may use tools borrowed from formal language theory, they usually have a rather different conception of language than do computer scientists. One important difference is that linguists typically do *not* consider either non-terminal or terminal symbols to be indivisible atoms, rather they consider them to be conglomerations of 'lower level' information called *feature structures*. For example, to say that a node bears the information NP is to say that the node actually bears a lot

---

[4]The reader will doubtless be able to think of other interesting operators to add. For example, adding operators $\downarrow^*$ and $\uparrow^*$ which explore the transitive closures of the daughter-of and mother-of relations respectively enables GB discourse concerning command relations to be modeled; while weakening the definition of $\bullet$ to ignore the precedence ordering on sisters, and adding a new unary modality to take over the task of regulating linear precedence, permits the ID/LP format used in GPSG to be naturally incorporated. Such extensions will be discussed in [Blackburn *et al.* forthcoming], however for the purposes of the present paper we will be content to work with the simpler set of operators we have defined here.

[5]Indeed, on closer inspection, even $\bullet$ can be regarded as an old friend in disguise; such logical issues will be discussed in [Blackburn *et al.* forthcoming].

of lower level information: for example, the features $+N$, $-V$, and BAR 2. Moreover (as we shall see in section 5) the constraints that a tree must satisfy in order to be accepted as well formed will typically involve this lower level information. The feature co-occurrence restrictions and feature instantiation principles of GPSG are good examples of this.

Is it possible to extend our framework to deal with such ideas? More to the point, is there a *simple* extension of our framework that can deal with complex categories? *Layered modal languages* provide what is needed. Semantically, instead of associating each tree node with an atomic value, we are going to associate it with a feature structure. Syntactically, we are going to replace the propositional symbols of L^T with wffs capable of talking about this additional structure. To be more precise, instead of defining the wffs of L^T over a base *Prop* consisting of primitive propositional symbols, we are going to define them over a base of modal formulas. That is, we will use a language with two 'layers'. The top layer L^T will talk about tree structure just as before, whereas the base layer (or feature layer) L^F will talk about the 'internal structure' associated with each node.[6]

Clearly the first thing we have to do is to define our feature language L^F and its semantics. We will assume that the linguistic theory we are working with tells us what features and atoms may be used. That is, we assume we have been given a signature $\langle \mathcal{F}, \mathcal{A} \rangle$ where both $\mathcal{F}$ and $\mathcal{A}$ are non-empty finite or denumerably infinite sets, the set of *features* and the set of *atomic information* respectively. Typical elements of $\mathcal{F}$ might be CASE, NUM, PERSON and AGR; while typical elements of $\mathcal{A}$ might be *genitive, singular, plural, 1st, 2nd,* and *3rd.*

The language L^F (of signature $\langle \mathcal{F}, \mathcal{A} \rangle$) contains the following items: all the elements of $\mathcal{A}$ (which we will regard as propositional symbols), a truth functionally adequate collection of Boolean connectives, and all the elements of $\mathcal{F}$ (which we will regard as one place modal operators). The set of wffs of L^F is the smallest set containing all the propositional symbols (that is, all the elements of $\mathcal{A}$) closed under the application of the Boolean and modal operators (that is, the elements of $\mathcal{F}$). Thus a typical wff of L^F might be the following:

$$\langle \text{AGR} \rangle \langle \text{PERSON} \rangle 3rd \wedge \langle \text{CASE} \rangle genitive.$$

Note that this wff is actually something very familiar, namely the following Attribute Value Matrix:

$$\begin{bmatrix} \text{AGR} & [\text{PERSON} \ 3rd] \\ \text{CASE} & genitive \end{bmatrix}$$

Indeed the wffs of L^F are nothing but straightforward 'linearisations' of the traditional two dimensional AVM format. Thus it is unsurprising that the semantics of L^F is given in terms of *feature structures*:

Definition 4.1 (Feature structures)
A feature structure of signature $\langle \mathcal{F}, \mathcal{A} \rangle$ is a triple F of the form $\langle W, \{R_f\}_{f \in \mathcal{F}}, V \rangle$, where W is a non-empty set, called the set of *points*; for all $f \in \mathcal{F}$, R_f is a binary relation on W that is a partial function; and V is a function that assigns to each propositional symbol (that is, each $\alpha \in \mathcal{A}$), a subset of W.[7] $\square$

Our satisfaction definition for L^F wffs is as follows. For any $\mathbf{F} = \langle W, \{R_f\}_{f \in \mathcal{F}}, V \rangle$ and any point $w \in W$:

$\mathbf{F}, w \models \alpha$	iff	$w \in V(\alpha)$, for all $\alpha \in \mathcal{A}$
$\mathbf{F}, w \models \neg \phi$	iff	not $\mathbf{F}, w \models \phi$
$\mathbf{F}, w \models \phi \wedge \psi$	iff	$\mathbf{F}, w \models \phi$ and $\mathbf{F}, w \models \psi$
$\mathbf{F}, w \models \langle f \rangle \phi$	iff	$\exists w'(wR_f w'$ and $\mathbf{F}, w' \models \phi)$

With L^F and its semantics defined, we are ready to define a language for talking about trees decorated with feature structures: the language $L^T(L^F)$, that is, *the language L^T layered over the language L^F.* That is, we choose *Prop* to be L^F and then make the L^T wffs on top of this base in the usual way.[8] As a result, we've given an 'internal structure' (namely, a modal structure, or AVM structure) to the propositional symbols of L^T. This is the syntactical heart of layering.

Definition 4.2 (Feature decorated trees)
By a (finite ordered) *feature structure decorated tree* (of signature $\langle \mathcal{F}, \mathcal{A} \rangle$) is meant a triple $\langle O, D, d \rangle$ where O is a finite ordered tree, D is a function that assigns to each node u of O a feature structure (of signature $\langle \mathcal{F}, \mathcal{A} \rangle$), and d is a function that assigns to each node u of O a point of $D(u)$. That is, $d(u) \in D(u)$.[9] $\square$

It is straightforward to interpret $L^T(L^F)$ wffs on feature structure decorated trees: indeed all we have

---

[6] As was mentioned earlier, at present there is relatively little published literature on layered modal languages. The most detailed investigation is that of [Finger and Gabbay 1992], while [de Rijke 1992] gives a brief account in the course of a general discussion on the nature of modal logic.

[7] For detailed discussion of this definition see [Blackburn 1991, 1992] or [Blackburn and Spaan 1991, 1992]. For present purposes it suffices to note that it includes as special cases most of the well known definitions of feature structures, such as that of [Kasper and Rounds 1986].

[8] This is worth spelling out in detail. The wffs of the language $L^T(L^F)$ (of signature $\langle \mathcal{F}, \mathcal{A} \rangle$) are defined as follows. First, all L^F wffs (of signature $\langle \mathcal{F}, \mathcal{A} \rangle$) are $L^T(L^F)$ wffs, and so are the constant symbols s and t. Second, if ϕ, ψ and $\phi_1, \ldots, \phi_n$ are $L^T(L^F)$ wffs then so are $\neg \phi$, $\phi \wedge \psi$, $\uparrow \phi$, $\downarrow \psi$, $\phi \Rightarrow \psi$ and $\bullet(\phi_1, \ldots, \phi_n)$. Third, nothing else is an $L^T(L^F)$ wff.

[9] In a number of recent talks Dov Gabbay has advocated the idea of 'fibering' one set of semantic entities over another. This is precisely what's going on here: we're fibering trees over feature structures. Fibered structures are the natural semantic domains for layered languages.

to do is alter the base clause of the L^T definition. So, let $\mathbf{M} = \langle \mathbf{O}, D, d \rangle$ be a feature structure decorated tree, and u be any node in $\mathbf{O}$. Then for all wffs $\phi \in L^F$:

$$\mathbf{M}, u \models \phi \quad \textit{iff} \quad D(u), d(u) \models \phi.$$

In short, when in the course of evaluating an $L^T(L^F)$ wff at a node u we encounter an L^F wff (that is, when we reach 'atomic' level) we go to the feature structure associated with u (that is, $D(u)$), and start evaluating the L^F wff at the point $d(u)$. This change at the atomic level is the only change we need to make: all the other clauses (that is, the clauses for s and t, the Boolean operators, for $\Rightarrow$, $\downarrow$, $\uparrow$, and $\bullet$) are unchanged from the L^T satisfaction definition given in section 2.

To close this section, a general comment. $L^T(L^F)$ is merely one, rather minimalist, example of a layered modal language. The layering concept offers considerable flexibility. By enriching either the L^T component, the L^F component, or both, one can tailor constraint languages for specific applications. Indeed, it's worth pointing out that one is not forced to layer L^T over a modal language at all. One could perfectly well layer L^T across a first order feature logic or over a fragment of such a first order logic (such as the Schönfinkel Bernays fragment explored in [Johnson 1991]),[10] and doubtless the reader can imagine other possibilities. That said, we're struck by the simplicity of purely modal layered languages such as $L^T(L^F)$, and we believe that there are good theoretical reasons for being interested in modal approaches (these are discussed at the end of the paper). Moreover, as we shall now see, even the rather simple collection of operators offered by $L^T(L^F)$ are capable of imposing interesting constraints on syntactic structures.

5 $L^T(L^F)$ and linguistic theory

At this stage, it should be intuitively clear why $L^T(L^F)$ is well suited for modeling contemporary linguistic theories. On the one hand, the L^T part of the language lets us talk directly about tree structure, thus clearly it is a suitable tool for imposing constraints on constituent structure. On the other hand, the L^F part of the language permits the description of *complex* (rather than atomic) categories; and nowadays the use of such categories is standard. The aim of this section is to give a concrete illustration of how $L^T(L^F)$ can be used to model modern linguistic theories. The theory we have chosen for this purpose is GPSG. In what follows we sketch how some of the leading ideas of GPSG can be captured using $L^T(L^F)$ wffs.

5.1 Complex categories

One of the fundamental ideas underlying GPSG (and indeed many other contemporary syntactic theories) is that a linguistic category is a complex object consisting of feature specifications, where feature specifications are feature/value pairs, and a value is either an atom or is itself a category. In $L^T(L^F)$, this idea is easily modeled since $L^T(L^F)$ contains L^F, a language specifically designed for talking about feature structures. To give a simple example, consider the following complex category:

$$\begin{bmatrix} \text{NOUN} & - \\ \text{VERB} & + \\ \text{BAR} & \text{two} \end{bmatrix}$$

This is naturally represented by the following L^F wff:

$$\neg noun \wedge verb \wedge \langle \text{BAR} \rangle two$$

where the attribute BAR is represented by a modality and the atomic symbols and Boolean features are represented by propositional symbols. This wff is satisfied at any point w in a feature structure such that *noun* is false at w, *verb* is true at w, and the propositional information *two* is reachable by making a BAR transition from w.

5.2 Admissibility constraints on local trees

The heart of GPSG is a collection of interacting principles governing the proper distribution of features within syntactic trees. Central to this theory is the concept of admissibility constraints on local trees. Very roughly,[11] the idea is that a local tree is admissible if it is a projection of an immediate dominance rule (that is, each node in the tree corresponds in some precisely defined way to exactly one category in the rule) and it satisfies all of the grammar principles; these include feature co-occurrence restrictions (FCRs), feature specification defaults (FSDs), linear precedence (LP) statements, and universal feature instantiation principles (UIPs). In what follows, we show how $L^T(L^F)$ can be used to model some of these admissibility conditions on local trees: section 5.2.1 shows how to model phrase structure restrictions and section 5.2.2 concentrates on FCRs. Finally, in section 5.2.3 we sketch an $L^T(L^F)$ treatment of the GPSG UIPs.

5.2.1 Phrase structure restrictions

In GPSG, restrictions on constituent structure are expressed by a set of ID/LP statements. As the name indicates, I(mmediate) D(ominance) statements encode immediate dominance restrictions on local trees (for instance, the ID rule $A \rightarrow B, C$ licenses any local tree consisting of a mother node labeled with category A and exactly two daughter nodes labeled with

---

[10] Layering over first order languages is treated in [Finger and Gabbay 1992].

[11] For a more precise formulation of the constraints on tree admissibility, see [Gazdar *et al.* 1985, page 100].

categories B and C respectively), whereas LP statements define a linear precedence relation between sister nodes (for example, the LP statement $C \prec B$ states that in any local tree with sisters labeled B and C, the C node must precede the B node).

Strictly speaking, such restrictions cannot be modeled in $L^T(L^F)$. The reason for this is trivial. As has already been pointed out, the satisfaction definition for • makes use of both the immediate dominance and linear precedence relations. In a full-blooded attempt to model GPSG, we would probably define a variant modal operator o of • that did not make use of the precedence relation, and introduce an additional modal operator (say ▷) to control precedence. However, having made this point, we shall not pursue the issue further. Instead, we will show how the present version of $L^T(L^F)$ allows for the encoding of phrase structure rules involving complex categories.

As was shown in section 3, rules involving atomic categories can be modeled in a fairly transparent way using $\Rightarrow$, • and $\vee$. For instance,

$$S \quad \Rightarrow \quad \bullet(NP, VP) \vee \bullet(S, CONJ, S)$$

captures the import of the following two phrase structure rules:

$$S \longrightarrow NP\ VP$$
$$S \longrightarrow S\ CONJ\ S$$

In these rules the information associated with each node of the tree is propositional in nature, that is, non-structured. However because $L^T(L^F)$ allows one to peer into the internal structure of nodes, this way of modeling phrase structure rules extends straightforwardly to rules involving complex categories: it suffices to replace the propositional symbols by L^F wffs. For example, the phrase structure rule:

$$\begin{bmatrix} \text{NOUN} & - \\ \text{VERB} & + \\ \text{BAR} & two \end{bmatrix} \rightarrow$$
$$\begin{bmatrix} \text{NOUN} & - \\ \text{VERB} & + \\ \text{BAR} & zero \\ \text{SUBCAT} & trans \end{bmatrix} \begin{bmatrix} \text{NOUN} & + \\ \text{VERB} & - \\ \text{BAR} & two \end{bmatrix}$$

can be formulated as the following $L^T(L^F)$ wff:

$(\neg noun \wedge verb \wedge \langle \text{BAR}\rangle two) \Rightarrow$

$\bullet((\neg noun \wedge verb \wedge \langle \text{BAR}\rangle zero \wedge \langle \text{SUBCAT}\rangle trans),$
$\quad (noun \wedge \neg verb \wedge \langle \text{BAR}\rangle two))$

That is, the L^F wffs give the required 'internal structure' in the obvious way.

5.2.2 Feature co-occurrence restrictions

FCRs encode restrictions on the distribution of features *within* categories. More specifically, they express conditional or bi-conditional dependencies between feature specifications occurring within the same category. For instance, the FCR:

$$[\text{INV} +] \supset [\text{AUX} +, \text{VFORM } fin] \quad \text{(FCR1)}$$

states that any category with feature specification INV + must also contain the feature specifications AUX + and VFORM fin. In other words, any inverted constituent must be a finite auxiliary.

FCRs are naturally expressed in $L^T(L^F)$ by using the $\Rightarrow$ connective. Thus, FCR1 can be captured by means of the following schema:

$$inv \Rightarrow (aux \wedge \langle \text{VFORM}\rangle fin)$$

This says that for any ordered tree and any node u in this tree, if the feature structure associated with u starts with the point w and inv is true at w, then aux is also true at w and furthermore, the propositional information fin is reachable from w by making a $\langle \text{VFORM}\rangle$ transition to some other node w'.

5.2.3 Universal principles

In this section, we show that $L^T(L^F)$ allows us to axiomatize the main content of GPSG three feature instantiation principles namely, the foot feature principle, the head feature convention and the control agreement principle.

Consider first the foot feature principle (FFP). This says that:

> Any foot feature specification which is instantiated on a daughter in a local tree must also be instantiated on the mother category in that tree. [Gazdar *et al.* 1985, page 81] [12]

So, assume that our GPSG theorising has resulted in signature $\langle \mathcal{F}, \mathcal{A}\rangle$ which includes the feature FOOT. We capture the FFP by means of the following schema:

$$\langle \text{FOOT}\rangle\phi \Rightarrow \Uparrow\langle \text{FOOT}\rangle\phi.$$

This says that for any node u, if the information ϕ is reachable by making a FOOT transition in the feature structure associated with u, then it must also be possible to obtain the information ϕ by making a FOOT transition in the feature structure associated with the mother of u. That is, FOOT information percolates up the tree. So for instance, if three sister nodes u_1, u_2 and u_3 of a tree bear the information $\langle \text{FOOT}\rangle\phi_1$, $\langle \text{FOOT}\rangle\phi_2$ and $\langle \text{FOOT}\rangle\phi_3$ respectively, then the feature structure associated with the mother node must bear the information $\langle \text{FOOT}\rangle\phi_1 \wedge \langle \text{FOOT}\rangle\phi_2 \wedge \langle \text{FOOT}\rangle\phi_3$. Incidentally, it then follows from the semantics of L^F that this node bears the information $\langle \text{FOOT}\rangle(\phi_1 \wedge \phi_2 \wedge \phi_3)$. That is, the three pieces of foot information are unified.

---

[12]This axiom is actually a simplified version of the FFP in that it ignores the distinction between inherited and instantiated features. See section 5.3 for discussion of this point.

26

Consider now the head feature convention (HFC). A simplified version of the HFC can be stated as follows:[13]

> Any head features carried by the head daughter is carried by the mother and vice-versa.

Assuming a signature $\langle \mathcal{F}, \mathcal{A} \rangle$ which includes the feature HEAD-FEATURE and the atomic information *head*, we capture the HFC by means of the following schema:

$$(head \wedge \langle \text{HEAD-FEATURE} \rangle \phi) \Rightarrow \Uparrow \langle \text{HEAD-FEATURE} \rangle \phi$$
$$\wedge$$
$$(head \wedge \Uparrow \langle \text{HEAD-FEATURE} \rangle \phi) \Rightarrow \langle \text{HEAD-FEATURE} \rangle \phi$$

The first conjunct says that whenever the feature structure associated with a node u marks it as a *head* node, and the information ϕ is reachable by making a HEAD-FEATURE transition, then one can also reach the same information ϕ by making a HEAD-FEATURE transition in the feature structure associated with the mother of u. The second conjunct works analogously to bring HEAD-FEATURE information down to the head daughter.

Finally, we sketch how the effect of the more elaborate control agreement principle (CAP) can be captured. GPSG formulates CAP by making use of the Montagovian semantic type assignments. As we haven't discussed semantics, we're going to assume that the relevant type information is available inside our feature structures. With this assumed, our formulation of CAP falls into three steps: first, defining the notions of controller and controllee (or *target* in GPSG terminology); second, defining the notion of a control feature; and third, defining the instantiation principle. We consider each in turn. Controller and controllee are defined as follows:[14]

> A category C is controlled by another category C' in a constituent C_0 if one of the following situations obtains at a semantic level: either C is a functor that applies to C' to yield a C_0, or else there is a control mediator C'' which combines with C and C' in that order to yield a C_0. [Gazdar *et al.* 1985, page 87]

Further, a control mediator is a head category whose semantic type is $\langle VP, \langle NP, VP \rangle \rangle$ where VP denotes the type of an intransitive verb phase and NP that of a generalised quantifier. The first step is to formulate the notions of controller and controllee.

---

[13]The exact formulation of the HFC implies that only *free* feature specifications are taken into account. See section 5.3 for discussion of this point.

[14]Again this is somewhat simplified in that the final GPSG definition of control only takes into account so-called χ-features so as to ignore perturbations of semantic types introduced by the presence of instantiated features.

We do this with the following three wffs (a and b are metavariables over semantic types, and np and vp correspond to the NP and VP above):

$$\bullet(\langle \text{TYPE} \rangle a/b, \langle \text{TYPE} \rangle a)$$
$$\Rightarrow \bullet(controllee, controller)$$

$$\bullet(\langle \text{TYPE} \rangle a, \langle \text{TYPE} \rangle a/b)$$
$$\Rightarrow \bullet(controller, controllee)$$

$$\bullet(\top, \langle \text{TYPE} \rangle vp/(np/vp), \top)$$
$$\Rightarrow \bullet(controller, \top, controllee)$$

Control features are SLASH and AGR and are not mutually exclusive. The problem is to decide which should actually function as control feature when both of them are present on the controllee category. In effect, in case of conflict (cf. [Gazdar *et al.* 1985, 89]), SLASH is the control feature if it is inherited, else AGR is. As we have no way to distinguish here between inherited and instantiated feature values, we will (again) give a simplified axiomatisation of control features, namely:

$$\langle \text{SLASH} \rangle \phi \quad \Rightarrow \quad \langle \text{CONTROL\_FEAT} \rangle \phi$$
$$(\neg \langle \text{SLASH} \rangle \top \wedge \langle \text{AGR} \rangle \phi) \quad \Rightarrow \quad \langle \text{CONTROL\_FEAT} \rangle \phi$$

Finally, we turn to the CAP itself. This says that the value of the control feature of the controllee is identical with the category of the controller. In $L^T(L^F)$:

$$(\downarrow (controller) \wedge \downarrow (controllee)) \Rightarrow$$
$$\downarrow ((controller \wedge \phi) \leftrightarrow$$
$$\downarrow (controllee \wedge \langle \text{CONTROL\_FEAT} \rangle \phi))$$

5.3 Discussion

In the preceding sections, we showed how $L^T(L^F)$ could be used to capture some of the leading concepts of GPSG. Although the account involves many simplifications and shortcomings, the examples should illustrate how to use $L^T(L^F)$: one expresses linguistic principles as $L^T(L^F)$ wffs, and only those (decorated) trees validatings all these wffs are considered well-formed. What we hope to have shown is that $L^T(L^F)$ is a very natural language for expressing the various types of theoretical constructs developed in GPSG and, more generally, in most modern theories of grammar. Complex categories can be described using the L^F part of the language while general information concerning the geometry of trees and the distribution of feature specifications within those trees can be stated using the full language. More specifically, the bullet operator $\bullet$ provides an easy way to

27

express phrase structure constraints while the strict implication operator ⇒ allows one to express various types of constraints on the distribution of features in trees. When used to connect two L^F wffs, ⇒ expresses generalisations over the internal structure of categories (as illustrated in section 5.2.2 on FCRs), whereas when used together with ↑, ↓ and • it allows information sharing between feature structures associated with different nodes in the tree (cf. section 5.2.3).

As already repeatedly mentioned, there remain many shortcomings in our approach to modeling GPSG. To close this discussion let's consider them a little more closely; this will lead to some interesting questions about the nature of linguistic theorising.

The first type of shortcoming involves lack of expressivity in $L^T(L^F)$ and is illustrated by the impossibility of expressing ID/LP statements (cf. section 5.2.1). As already indicated, we don't regard such shortcomings as a failure of the general modal approach being developed here. With a slightly different choice of modal language, an adequate modeling of ID/LP statements could be attained. More generally, we think it is important to explore a wide range of modal languages for linguistic theorising, for we believe that it may be possible to usefully classify differing linguistic theories in terms of the different modal operators required to formalise them. A theoretical justification for our confidence will be given in the following section; here we'll simply say that we think this is a feasible way of addressing the questions raised in [Shieber 1988] concerning the comparative expressivity and computational complexity of grammatical formalisms.

The second type of shortcoming is more serious and potentially far more interesting. Two cases in point are (i) the distinction made in GPSG between *instantiated* and *inherited* features and (ii) the GPSG notion of a *free* feature. Briefly, inherited features are features whose presence on categories in trees is directly determined by an ID rule whereas instantiated features are non-inherited features (cf. [Gazdar *et al.* 1985, page 76]). Furthermore, given a category C occurring in a tree τ such that τ is a projection of some ID rule that satisfies the FFP and the CAP, a feature specification is said to be free in C iff is is compatible with the information contained in C (cf. [Gazdar *et al.* 1985, page 95] for a more precise definition of free features). The problem in both cases is that *derivational* information needs to be taken into account. In the first case, the source of the feature specification must be known (does it stem from an ID rule or from some other source?). In the second case, we must know that both CAP and FFP are already being satisfied by the category under consideration. There is an essentially dynamic flavour to these ideas, something that goes against the grain of the essentially static tree descriptions offered by $L^T(L^F)$. Whether this dynamic aspect is

in fact required, and how it could best be modeled, we leave here as open research questions.

6 But why modal languages?

To close this paper we wish to discuss an issue that may be bothering some readers: why were *modal* languages chosen as the medium for expressing constraints on trees and feature structures? A reader unfamiliar with the developments that have taken place in modal logic since the early 1970's, and in particular, unfamiliar with the emergence of *modal correspondence theory*, may find the decision to work with modal languages rather odd; surely it would be more straightforward to work in (say) some appropriate first order language? However we believe that there are general reasons for regarding modal languages as a particularly natural medium for linguistic theorising, and what follows is an attempt to make these clear.

The first point that needs to be made about modal languages is that they are nothing but extremely simple languages for talking about graphs. Unfortunately, the more philosophical presentations of modal logic tend to obscure this rather obvious point. In such presentations the emphasis is on discussing such ideas as 'possible worlds' and 'intensions'. Such discussions have their charms, but they make it very easy to overlook the fact that the mathematical structures on which these ideas rest are extremely simple: just sets of nodes decorated with atomic information on which a transition relation is defined. Kripke models are nothing but graphs.

The second point is even more important. Modal languages are not some strange alternative to classical languages; rather, they are relatively constrained fragments of such languages. If a problem has been modelled in a modal language then it has, *ipso facto*, been modeled in a classical language; and moreover, it has been modeled in a very resource conscious way. The point deserves a little elaboration. Ever since the early 1970's, one of the most important branches of research in technical modal logic has been modal correspondence theory (see [van Benthem 1984] and references therein), the systematic study of the interrelationships between modal languages on the one hand, and various classical logics (first order, infinitary, and second order) on the other. Modal correspondence theory rests on the following simple observation. It is usually possible to view modal operators as logical 'macros'; essentially modal operators are a prepackaging of certain forms of quantification that are available in classical languages. To give a simple example, we might view a statement of the form ↑ϕ as a shorthand for the first order expression $\exists y(y > x \wedge \varphi(y))$, where $\varphi(y)$ is a certain first order wff called the *standard translation* of ϕ.[15] For

---

[15]This is somewhat impressionistic; for the full story consult [van Benthem 1984]. For a discussion of the fun-

present purposes the details aren't particularly important; the key point to note is that the † operator is essentially a neat notation which embodies a limited form of first order quantificational power: namely the ability to quantify over mother nodes. More generally, modal languages eschew the quantificational power that classical languages achieve through the use of variables and binding, in favour of a variable free syntax in which quantification is performed using operators. Expressive power is traded for syntactic simplicity.

The relevance of these points for linguistics should be clear. Linguistic theorizing makes heavy use of graph structures; trees and feature structures are obvious examples. Thus modal languages *can* be used as constraint formalisms; what correspondence theory tells us is that they are particularly interesting ones, namely formalisms that mesh neatly with the linguists' quest for revealing descriptions using the weakest tools possible.

References

[Blackburn 1991] Blackburn, P.: 1991, Modal Logic and Attribute Value Structures. To appear in *Diamonds and Defaults*, edited by M. de Rijke, Studies in Logic, Language and Information, Kluwer.

[Blackburn and Spaan 1991] Blackburn, P. and Spaan, E.: 1991, On the Complexity of Attribute Value Logics. *Proceedings of the Eighth Amsterdam Colloquium*, edited by P. Dekker and M. Stokhof, Philosophy Department, Amsterdam University, The Netherlands.

[Blackburn and Spaan 1992] Blackburn, P. and Spaan, E.: 1992, A Modal Perspective on the Computational Complexity of Attribute Value Grammar. To appear in *Journal of Logic, Language and Information*.

[Blackburn 1992] Blackburn, P.: 1992, Structures, Languages and Translations: the Structural Approach to Feature Logic. To appear in *Constraints, Language and Computation*, edited by C. Rupp, M. Rosner and R. Johnson, Academic Press.

[Blackburn *et al.* forthcoming] Blackburn, P., Gardent, C., and Meyer-Viol, W.: Modal Phrase Structure Grammars. In preparation.

damental correspondences involved in feature logic see [Blackburn 1992].

[de Rijke 1992] de Rijke, M.: 1992, What is Modal Logic? In *Logic at Work*, proceedings of the Applied Logic Conference, CCSOM, University of Amsterdam, 1992.

[Finger and Gabbay 1992] Finger, M. and Gabbay, D.: 1992, Adding a Temporal Dimension to a Logic System. *Journal of Logic, Language and Information*, 1, pp. 203–233.

[Gazdar 1979] Gazdar, G.: 1979, Constituent Structures. Manuscript, Sussex University.

[Gazdar *et al.* 1985] Gazdar, G.: Klein, E., Pullum, G., and Sag, S.: 1985, *Generalised Phrase Structure Grammar*. Basil Blackwell.

[Johnson 1991] Johnson, M.: 1991, Features and Formulas, *Computational Linguistics*, 17, pp. 131–151.

[Joshi and Levy 1977] Joshi, A. and Levy, S.: 1977, Constraints on Structural Descriptions: Local Transformations. *SIAM Journal of Computing*, 6, pp. 272–284.

[Kasper and Rounds 1986] Kasper, R. and Rounds, W.: 1986, A logical semantics for feature structures. *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, Columbia University, New York, pp. 257–266.

[McCawley 1968] McCawley, J.: 1968, Concerning the Base Component of a Transformational Grammar. *Foundations of Language*, 4, pp. 55–81.

[Peters and Ritchie 1969] Peters, S. and Ritchie, R.: 1969, Context-Sensitive Immediate Constituent Analysis – Context-Free Languages Revisited. *Proceedings ACM Symposium on Theory of Computing*, Association for Computing Machinery, pp. 1– 10.

[Rounds 1970] Rounds, W.: 1970, Tree-Oriented Proofs of Some Theorems in Context-Free and Indexed Languages. *Proceedings ACM Symposium on Theory of Computing*, Association for Computing Machinery, pp. 210 – 216.

[Shieber 1988] Shieber, S.: 1988, Separating Linguistic Analyses from Linguistic Theories. In *Natural Language Parsing and Linguistic Theories*, edited by U. Reyle and C. Rohrer, Reidel.

[van Benthem 1984] van Benthem, J.: 1984, Correspondence Theory, in *Handbook of Philosophical Logic*, 2, edited by D. Gabbay and F. Guenthner, Reidel.