Linguistics, Logic, and Finite Trees

Patrick Blackburn* and Wilfried Meyer-Viol†

Abstract

A modal logic is developed to deal with finite ordered binary trees as they are used in (computational) linguistics. A modal language is introduced with operators for the 'mother of', ' first daughter of' and 'second daughter of' relations together with their transitive reflexive closures. The relevant class of tree models is defined and three linguistic applications of this language are discussed: context free grammars, command relations, and trees decorated with feature structures. An axiomatic proof system is given for which completeness is shown with respect to the class of finite ordered binary trees. A number of decidability results follow.

Many computational linguists use ideas from logic to analyse and develop syntactical frameworks. Their interest is not confined to the (fairly obviously applicable) tools offered by proof and complexity theory: there is a growing perception that the mathematical ontologies underlying linguistic theorising are interesting in their own right, and that the grammatical formalisms that deal with them should have an explicitly formulated semantics. In short, model theory is increasingly seen as valuable.

Nonetheless, this model theoretic turn has been curiously one sided. It has primarily been directed to the analysis of the role played by *feature structures* in unification formalisms, and has tended to ignore other important aspects of linguistic ontologies. For example, Generalised Phrase Structure Grammar (GPSG) and Lexical Functional Grammar (LFG) are commonly regarded as unification formalisms, yet they both make important use of *finite trees* in addition to feature structures. However most model theoretic work in feature logic says nothing about this additional structure.

Such one-sidedness is unfortunate. For a start, one of the most interesting aspects of GPSG and LFG is precisely the fact that they deal with hybrid ontologies. How do the different components fit together? What sort of expressive power is needed for talking about composite structures? To ignore the presence

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

†Centrum voor Wiskunde en Informatica, Kruislaan 413, 1098 SJ Amsterdam. Email: wilfried@cwi.nl

of trees in such systems is to ignore some of the most interesting issues. Moreover, many syntactic theories (notably Chomsky's Government and Binding (GB) theory, probably the dominant contemporary syntactic paradigm) are firmly tree based. If the insights of such frameworks are to be reconciled with the insights of unification tradition, it seems important to extend the model theoretic perspective to the role played by finite trees.

Some recent papers have made a start in this direction. Rogers and Vijay-Shankar (1992) and Vijay-Shankar (1992) propose various languages for describing trees. Their motivation is to combine insights from the Tree Adjoining Grammar and Unification traditions. Kracht (1993a) uses a modal 'orientation language' over the parse trees of context free grammars to relate GB and GPSG. Finally, Blackburn, Gardent and Meyer-Viol (1993) introduce a modal language for talking about trees, and, by 'layering' this language across a feature logic, give an account of some of the leading ideas of GPSG.

The present paper builds on Blackburn, Gardent and Meyer-Viol (1993). We discuss two linguistic ontologies, namely *finite ordered binary trees*, and *finite ordered binary trees fibred over feature structures*, formulate languages for talking about them, and prove a number of results. We proceed as follows. In the first section we introduce finite ordered binary trees and a simple modal language $\mathcal{L}$ for talking about them. In the second section we outline three linguistic applications of $\mathcal{L}$: talking about the parse trees of context free languages, talking about GB command relations, and, by 'layering' $\mathcal{L}$ across a feature logic, modeling aspects of GPSG. In the third section the technical work begins: we introduce and discuss an axiomatisation of the logic of finite ordered binary trees, and in the fourth section prove it to be complete. In the fifth section we discuss some of the consequences of this completeness result, notably the decidability of the logic, and, building on the work of Finger and Gabbay (1992), the decidability of the 'layered language'. We close the paper by noting a number of directions for future work.

1 The language $\mathcal{L}$ and its semantics

Trees can be found on the pages of most syntax textbooks; they are an important part of the ontologies posited in most grammatical theories. In this section we isolate a linguistically important class of trees, namely the *finite ordered binary trees* together with a *collection of unary relations*, and define a simple modal language $\mathcal{L}$ for talking about it.

First some terminology. We assume that the reader knows what a *finite tree* is, and that such locutions as 'a node w_1 immediately dominates node w_2', or equivalently 'w_1 is the mother of w_2' or equivalently 'w_2 is a daughter of w_1' are understood. Next, certain special tree nodes are important. Every tree has a unique node called the *root* which has the property that it is not the daughter of any node. Moreover, a non-empty subset Θ of the nodes of any finite tree

are *terminal nodes*, that is, nodes without daughters. A tree is a *binary* tree if no node in the tree immediately dominates more than two nodes.

In linguistics trees are typically thought of as *ordered*. For binary trees this means the following: the daughters of any node (of which there are at most two) are uniquely classified as being either the *first daughter* or the *second daughter*. We think of the first daughter as preceding the second daughter (if in fact there is a second daughter) and when finite ordered binary trees are drawn this is conventionally represented by placing the first daughter to the left of the second.

Such finite ordered binary trees lie at the heart of much syntactic analysis. We will usually present them as tuples of the form

$$\langle W, \succ_1, \succ_2, root, \Theta \rangle.$$

Here W is a finite, non-empty set, the set of tree nodes; Θ ($\subseteq W$) contains all and only the tree's terminal nodes; and *root* is the (unique) root node of the tree. As for $\succ_1$ and $\succ_2$, these are binary relations defined as follows: for all $w, w' \in W$, $w \succ_1 w'$ iff w' is the first daughter of w; and $w \succ_2 w'$ iff w' is the second daughter of w. Note that both $\succ_1$ and $\succ_2$ are partial functions, for any node in an ordered binary tree has at most one first daughter and at most one second daughter. Further note that if $w \succ_2 w'$ then there exists a unique w'' such that $w \succ_1 w''$. Moreover $w'' \neq w'$.

This presentation of finite ordered binary trees enables us to recover four other binary relations we will make heavy use of, namely $\succ$, $\prec$, $\succ_*$ and $\prec_*$. We define $\succ$ to be $\succ_1 \cup \succ_2$, thus $w \succ w'$ means that w' is a daughter of w. We define $\prec$ to be the converse relation of $\succ$, thus $w \prec w'$ means that w' is the mother of w. Note that $\prec$ is a partial function that is defined on all nodes except *root*. As to $\succ_*$ and $\prec_*$, they are the reflexive transitive closures of $\succ$ and $\prec$ respectively. Sometimes it is convenient to make some of these defined relations explicit in our presentations. For example, when defining satisfaction it will be natural to present finite ordered binary trees $\mathbf{O}$ as tuples of the form

$$\langle W, \succ_1, \succ_2, \prec, \succ_*, \prec_*, root, \Theta \rangle.$$

Finite ordered binary trees are clearly Kripke *frames*, but how does syntactic theory give rise to Kripke *models*? The answer is not hard to find. Linguists annotate trees with further information. For example, nodes may be marked by the symbols S, NP or VP, indicating that they are sentential nodes, noun phrase nodes, or verb phrase nodes respectively. Indeed nodes often bear multiple annotations. For instance, a noun phrase node might be marked +N, BAR-2 and CASE-GENITIVE. In short, the structures underlying much linguistic theorising can be seen as finite ordered binary trees *together with a collection of unary relations on the tree nodes*, and such structures are Kripke models. Incidentally, while trees are used by a wide range of grammatical theories, different theories

typically use different unary relations. Thus the frames we have isolated are common to many theories, but the models built over them will vary.

Now that we know the structures we're interested in, how shall we talk about them? In this paper we explore the use of propositional modal language called $\mathcal{L}$.[1] The primitive alphabet of $\mathcal{L}(\mathbf{Prop})$ consists of following items: a non-empty set of propositional symbols **Prop**, a truth functionally adequate collection of Boolean operators, five primitive unary 'diamond' modalities $\downarrow^1$, $\downarrow^2$, $\uparrow$, $\downarrow^*$ and $\uparrow^*$, and the punctuation symbols (and). For any choice of **Prop**, the wffs of $\mathcal{L}(\mathbf{Prop})$ are built up in the usual way: all elements of **Prop** are wffs; if ϕ and ψ wffs, then so are $\neg\phi$, $(\phi \wedge \psi)$, $\downarrow^1\phi$, $\downarrow^2\phi$, $\uparrow\phi$, $\downarrow^*\phi$ and $\uparrow^*\phi$; and nothing else is a wff. We normally suppose that some choice of **Prop** has been fixed and speak simply of the language $\mathcal{L}$.

We use a number of defined modalities. Firstly, we define a new unary diamond by stipulating that $\downarrow\phi$ is to be $\downarrow^1\phi \vee \downarrow^2\phi$. Second, we define the dual 'box' operators to our diamonds in the usual fashion: $\Downarrow^1\phi$ is defined to be $\neg\downarrow^1\neg\phi$, $\Downarrow^2\phi$ is defined to be $\neg\downarrow^2\neg\phi$, $\Downarrow\phi$ is defined to be $\neg\downarrow\neg\phi$, $\Uparrow\phi$ is defined to be $\neg\uparrow\neg\phi$, $\Downarrow^*\phi$ is defined to be $\neg\downarrow^*\neg\phi$, and $\Uparrow^*\phi$ is defined to be $\neg\uparrow^*\neg\phi$. Third, we define two nullary modalities (or constants) as follows: s is defined to be $\Uparrow\bot$ and t is defined to be $\Downarrow\bot$. We sometimes call $\downarrow^1$, $\downarrow^2$, $\downarrow$ and $\uparrow$ *basic operators.*

This language can talk about trees in linguistically interesting ways. The atomic symbols of **Prop** will enable us to talk about node labels. Intuitively, an atomic symbol (say, NP) will be true at a node iff that node is labeled with the corresponding property (that is, iff it is a noun phrase node). The modal operators are to 'move us round' the trees in a natural manner: $\downarrow^1$ and $\downarrow^2$ will look for information at first and second daughters respectively, $\uparrow$ will look for information at mothers, and $\downarrow^*$ and $\uparrow^*$ will explore the 'dominates' and 'is dominated by' relations respectively.

Let's make this precise. The standard interpretation for our language is in terms of Kripke models built over arbitrary Kripke frames of the form

$$\langle W, R_{\succ_1}, R_{\succ_2}, R_{\prec}, R_{\succ_*}, R_{\prec_*} \rangle,$$

that is, frames with a transition relation for each modality. However, because we are interested in linguistic applications, we are only interested in models built over frames

$$\mathbf{O} = \langle W, \succ_1, \succ_2, \prec, \succ_*, \prec_*, root, \Theta \rangle$$

that are presentations of finite ordered binary trees. That is, our intended semantics is in terms of those pairs $\langle \mathbf{O}, V \rangle$ where $\mathbf{O}$ is such a presentation and V is a map from **Prop** to $Pow(W)$. In what follows we reserve the terms 'Kripke model' and 'model' for our intended structures.

[1] Modal languages seem a natural choice for linguistic applications: although extremely simple, they offer significant expressive power. For further discussion see Blackburn, Gardent and Meyer-Viol (1993).

Now for the satisfaction definition. For any model $\mathbf{M}$ $(= \langle \mathbf{O}, V \rangle)$, for any node w of $\mathbf{M}$, and for any wff ϕ we define:

$$
\begin{array}{lll}
\mathbf{M}, w \models p & \text{iff} & w \in V(p), \text{ for all } p \in \mathbf{Prop} \\
\mathbf{M}, w \models \neg\phi & \text{iff} & \mathbf{M}, w \not\models \phi \\
\mathbf{M}, w \models \phi \wedge \psi & \text{iff} & \mathbf{M}, w \models \phi \text{ and } \mathbf{M}, w \models \phi \\
\mathbf{M}, w \models \downarrow^1 \phi & \text{iff} & \exists w'(w \succ_1 w' \text{ and } \mathbf{M}, w' \models \phi) \\
\mathbf{M}, w \models \downarrow^2 \phi & \text{iff} & \exists w'(w \succ_2 w' \text{ and } \mathbf{M}, w' \models \phi) \\
\mathbf{M}, w \models \uparrow \phi & \text{iff} & \exists w'(w \prec w' \text{ and } \mathbf{M}, w' \models \phi) \\
\mathbf{M}, w \models \downarrow^* \phi & \text{iff} & \exists w'(w \succ_* w' \text{ and } \mathbf{M}, w' \models \phi) \\
\mathbf{M}, w \models \uparrow^* \phi & \text{iff} & \exists w'(w \prec_* w' \text{ and } \mathbf{M}, w' \models \phi).
\end{array}
$$

If $\mathbf{M}, w \models \phi$ then we say ϕ is *true* in $\mathbf{M}$ at w, or ϕ is *satisfied* in $\mathbf{M}$ at w. For any wff ϕ, if there is a model $\mathbf{M}$ and a node w in $\mathbf{M}$ such that $\mathbf{M}, w \models \phi$, then we say that ϕ is *satisfiable*. If ϕ is true at all nodes in a model $\mathbf{M}$ then we say it is *valid in the model* $\mathbf{M}$. The notion of validity in a model has an important role to play for us. As discussed in the next section, we think of grammars G as $\mathcal{L}$ wffs ϕ^G. The trees admitted by the grammar are precisely those models in which ϕ^G is valid. Another important concept is validity: if a wff ϕ is valid in all models then we say it is *valid* and write $\models \phi$.

The satisfaction definition clearly captures the intended interpretation of $\mathcal{L}$. Also, note what the defined operators mean. In particular, note that:

$$
\begin{array}{lll}
\mathbf{M}, w \models s & \text{iff} & w = root \\
\mathbf{M}, w \models t & \text{iff} & w \in \Theta \\
\mathbf{M}, w \models \downarrow \phi & \text{iff} & \exists w'(w \succ w' \text{ and } \mathbf{M}, w' \models \phi).
\end{array}
$$

That is, s and t are constants true at only the root node and terminal nodes respectively, while $\downarrow$ looks for information at daughter nodes. Other useful defined operators abound, for $\mathcal{L}$ is very expressive over its intended models. For example, we can define the *universal modality*: $[U]\phi =_{def} \uparrow^*(s \wedge \Downarrow^* \phi)$. This says that ϕ is true at all points in a model, thus the modality allows universal constraints on grammatical well-formedness to be stated in the object language; see Blackburn and Spaan (1993) for further discussion.

Before considering applications, there is an aspect of $\mathcal{L}$'s semantics that is worth discussing, namely the way we have restricted our discussion to *binary* trees. Actually, from a logical point of view the important point is not that our trees are binary, but that there is a fixed upper bound on their branch factor. We could easily extend $\mathcal{L}$ to permit third daughters, fourth daughters, ..., n-th daughters to be talked about using unary operators $\downarrow^3$, $\downarrow^4$, ..., $\downarrow^n$, and the results of this paper generalise straightforwardly to such extensions. However if we do *not* have a fixed upper bound, matters are different. We briefly discuss the issue in the paper's conclusion.

Linguistically the restriction to binary trees is reasonable, though not uncontroversial. In GB (see Chomsky (1981)) binary branching trees are widely

considered to be fundamental. On the other hand, most versions of GPSG (see Gazdar *et al* (1985)) regard co-ordinations as 'flat'. For example, 'John and Sue and Bill and Lou and Butch and Peggy-Sue' would be represented by a node with six daughters. As no upper bound can be placed on the number of conjuncts, such versions of GPSG place no upper bound on the branch factor.

2 Three linguistic applications

The aim of this section is to give the reader a taste of $\mathcal{L}$ in action. We give three examples. First we show that $\mathcal{L}$ can pick out the parse trees of any context free phrase structure grammar. Second we show that $\mathcal{L}$ can express many of the 'command relations' used in GB. Third, we combine $\mathcal{L}$ with a 'feature logic' in a particularly simple fashion. The resulting 'layered language' enables some of the more important ideas of GPSG to be captured.

2.1 Context free grammars

We show here how to construct $\mathcal{L}$ formulas that will distinguish the parse trees of a given context free grammar from all other models. Strictly speaking, we only show how to capture the parse trees of any context free grammar that rewrites no symbol to more than two symbols. However, as far as weak generative capacity is concerned, nothing is lost; for any context free language can be generated by a grammar in Chomsky Normal Form, and such grammars have this property. Moreover, it will be clear from our discussion that by adding to $\mathcal{L}$ finitely many of the operators $\downarrow^3$, $\downarrow^4$, ..., $\downarrow^{n-1}$, $\downarrow^n$ mentioned in the previous section, any context free grammar whatsoever can be handled directly.

So suppose $\mathcal{G} = \langle S, \mathcal{N}, \mathcal{T}, \mathcal{P} \rangle$ is a context free grammar with start symbol S, set of non-terminal symbols $\mathcal{N}$, set of terminal symbols $\mathcal{T}$, and set of productions $\mathcal{P}$. We can assume that all these sets are finite and that $\mathcal{N}$, $\mathcal{T}$ and $\{S\}$ are pairwise disjoint. Our goal is to define a wff $\phi^{\mathcal{G}}$ that is valid on all and only the parse trees of the grammar $\mathcal{G}$.

The first step is to specify our modal language; that is, to find a suitable choice of **Prop**. For any context free grammar $\mathcal{G}$ there is an obvious choice. If the grammar contains no epsilon productions (that is, the grammar rewrites no symbol to the null string), take **Prop** to be $\mathcal{N} \cup \mathcal{T} \cup \{S\}$. On the other hand, if the grammar contains epsilon productions choose some distinct new symbol ϵ and take **Prop** to be $\mathcal{N} \cup \mathcal{T} \cup \{S\} \cup \{\epsilon\}$.

The next step is to capture the effect of the productions in $\mathcal{G}$. As we assume that $\mathcal{G}$ contains only binary rewrites, every production has one of the following three forms:

$$X \longrightarrow YZ \quad \text{or} \quad X \longrightarrow Y \quad \text{or} \quad X \longrightarrow \epsilon,$$

where $X \in \mathcal{N} \cup \{S\}$ and $Y, Z \in \mathcal{N} \cup \mathcal{T} \cup \{S\}$. For each production $P \in \mathcal{P}$ we form the corresponding wff ϕ^P, namely:

$$X \rightarrow (\downarrow^1 Y \wedge \downarrow^2 Z) \quad \text{or} \quad X \rightarrow \downarrow^1 Y \quad \text{or} \quad X \rightarrow \downarrow^1 \epsilon.$$

Let $\phi^{\mathcal{P}}$ be $\bigvee_{P \in \mathcal{P}} \phi^P$. Clearly $\phi^{\mathcal{P}}$ is valid on every parse tree for $\mathcal{G}$.

But we want to define an $\mathcal{L}$ formula that is valid on all *and only* the parse trees for $\mathcal{G}$, so we have a little more work to do: we must express in $\mathcal{L}$ certain general facts about parse trees. First, we insist that each node of a parse tree must be labeled by at least one element of **Prop**: $\bigvee_{p \in \mathbf{Prop}} p$ achieves this. Second, we insist that each node of a parse tree must be labeled by at most one element of **Prop**: $p \rightarrow \bigwedge_{q \in \mathbf{Prop} \setminus \{p\}} \neg q$ achieves this. Third, we insist that the root node of a parse tree must 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 of parse trees: $\bigwedge_{p \in \mathcal{N}} (p \rightarrow \neg t)$ achieves this. Fifth, we insist that terminal symbols label only terminal nodes of parse trees: $\bigwedge_{p \in \mathcal{T}} (p \rightarrow t)$ achieves this. Finally, we insist that ϵ labels only terminal nodes: $\epsilon \rightarrow t$ achieves this. Call the conjunction of these wffs $\phi^{\mathcal{U}}$.

Now we can define the required formula: let $\phi^{\mathcal{G}}$ be $\phi^{\mathcal{P}} \wedge \phi^{\mathcal{U}}$. Clearly for any model $\mathbf{M}$, $\mathbf{M} \models \phi^{\mathcal{G}}$ iff $\mathbf{M}$ is (isomorphic to) a parse tree for $\mathcal{G}$.

The most important point to note about this logical reconstruction of $\mathcal{G}$ is the way that something essentially *procedural* (namely rewrite rules on strings) have been turned into something *declarative* (namely a collection of axioms regulating the local structure of parse trees). This is not a new idea in linguistics. It is standard in both GB and GPSG to insist that phrase structure 'rewrite rules' should *not* be thought of as operations on strings (or indeed, as operations at all) but rather as *node admissibility conditions on trees*. In GB this is a consequence of the idea that representations, not rules, are fundamental. In GPSG it is a consequence of an explicitly model theoretic stance towards syntactic theorising: the primary task facing the linguist is to construct a logical theory characterising precisely those structures exhibited by grammatical sentences. How these structures are to be constructed is an interesting, but separate, concern.

It's also worth remarking that even the idea of turning rewrite rules into *modal* axioms is not novel: essentially the same thing was done by McCawley in 1968. McCawley doesn't use the terminology of modal logic, nonetheless the heart of his insight is that the rewrite arrow of formal language theory can be regarded as a modal operator interpreted over tree based Kripke models. Thus our choice of a modal language for talking about trees was not capricious. Modal operators arise quite naturally in the transition from the 'rule based' perspective on linguistic structure to the declarative model theoretic perspective, a theme that recurs in contemporary work on feature logic.

2.2 Command relations

The tree relations we have been considering are rather conventional. The 'mother of' and 'daughter of' relations, together with their transitive closures, are fundamental relations that occur in practically any research area concerned with trees. However many linguistic frameworks, and in particular GB, work with a much richer collection of tree relations. Perhaps the prime examples of such relations are the *command relations* of GB. Intuitively, command relations deal with the sphere of influence of syntactic mechanisms, and they play an essential role in various components of GB theory. Here we will investigate how $\mathcal{L}$ copes with command relations. Our point of departure is the definition given by Barker and Pullum (1990).

Definition 2.1 (Command Relations on Tree Models) *Let* $\mathbf{M}$ *be a model and* P *be a unary relation on the nodes of* $\mathbf{M}$. *The* P-command relation C_P *on* $\mathbf{M}$ *is defined as* $\{\langle w, v\rangle : \forall k(k \neq w \ \& \ k \succ_* w \ \& \ k \in P \Rightarrow k \succ_* v)\}$. □

P is called the *generating property* of the command relation C_P. The set $\{v \in W : C_P(w, v)\}$ is called the *P-command domain* of w in $\mathbf{M}$. Note that because the P-command relations are defined on *trees*, when $C_P(w, v)$ holds in $\mathbf{M}$, then all P-nodes properly dominating w dominate v and, consequently, the (unique) *lowest* such P-node dominates v. Conversely if the lowest P-node dominating w dominates v, then all P-nodes dominating w do. So the P-command domain of w is determined by the lowest P-node properly dominating w. By checking the definition we see that, if P is the empty set or if P is the singleton set $\{root\}$, then the P command domain of w is the entire tree.

Given command relations C_P, are the obvious modalities corresponding to these relations expressible in $\mathcal{L}$? To be more precise, it would be pleasant to have at our disposal for each P of interest a modality $\langle P\rangle$ with the following semantics:

$$\mathbf{M}, w \models \langle P\rangle\psi \ \text{ iff } \ \exists v(C_P(w, v) \ \& \ \mathbf{M}, v \models \psi).$$

Are such modalities definable in $\mathcal{L}$? The answer is 'yes' whenever there is an $\mathcal{L}$ formula ϕ^P which holds in $\mathbf{M}$ on all and only all nodes in P. For suppose we have such a formula ϕ^P. Then it is clear that defining

$$\langle P\rangle\psi \ =_{def} \ \Uparrow\Uparrow^*((s \vee \phi^P) \rightarrow \downarrow^*\psi)$$

has the required effect. (The presence of s ensures that if no ϕ^P node dominates the point of evaluation then the command domain will be the entire tree; this is the approach adopted by Pullum and Barker.) The dual operator $[P]$ can also be defined:

$$[P]\psi \ =_{def} \ \uparrow\uparrow^*((s \vee \phi^P) \wedge \Downarrow^*\psi).$$

Because of the correspondence between the relations P and the wffs ϕ^P which pick them out, it is natural to use the $\mathcal{L}$ formulas themselves to label the modalities. That is, if ϕ is the wff corresponding to P we will write $\langle\phi\rangle\psi$ rather

than $\langle P\rangle\psi$. Using this 'command modality' notation, here are some examples of expressible command relations:

1. $\langle\top\rangle\phi$ insists that ϕ holds in the *smallest* command domain of w, for $C_\top(w,v)$ iff v is dominated by the mother of m.

2. $\langle\bot\rangle\phi$ insists that ϕ holds in the *largest* command domain of w, for $C_\bot(w,v)$ iff v is dominated by *root*.

3. $\langle\downarrow^2\top\rangle\phi$ insists that ϕ holds somewhere in what is known as the *c-command* domain of w. Note that $C_{\downarrow^2\top}(w,v)$ iff v is dominated by the first *branching* node dominating w.

4. $\langle\mathrm{S}\rangle\phi$ insists that ϕ holds in the S-command domain of w. Note that the generating property of this relation consists of all the nodes in the tree decorated with S for 'Sentence'.

The last two examples are typical of the command relations used in GB. The terminology *c*-command is short for 'in construction with' and plays an important role in the GB account of anaphoric binding; the classic account is Reinhart (1981). For further discussion of S-command, *Sentence* command, see Barker and Pullum (1990).

A full discussion of command relations would take us too far afield (for a thorough investigation see Kracht (1992, 1993a, 1993b)) but it is worth noting that a number of the characterizing properties of command relations discussed by Barker and Pullum follow straightforwardly from the semantics of our language. For example, consider the *intersection property*, which can be formulated as $C_{\phi\vee\psi} = C_\phi \cap C_\psi$: the intersection of command relations corresponds to union over their generating properties. This immediately follows from the fact that

$$\Uparrow\Uparrow^*((s \vee \phi \vee \psi) \rightarrow \downarrow^*\chi)$$

is logically equivalent to

$$\Uparrow\Uparrow^*((s \vee \phi) \rightarrow \downarrow^*\chi) \wedge \Uparrow\Uparrow^*((s \vee \psi) \rightarrow \downarrow^*\chi).$$

When written in command modality notation this boils down to the fact that $\langle\phi\vee\psi\rangle\chi \leftrightarrow \langle\phi\rangle\chi \wedge \langle\psi\rangle\chi$ is logically valid.

2.3 Trees decorated with feature structures

When linguists decorate trees they do not usually do so in the manner familiar from formal language theory. For example, when a linguist decorates a node with the information NP this is usually a way of insisting that it possesses a whole bundle of properties. Sometimes it is possible to represent this information bundle using only Boolean combinations of propositional variables, but for

theories such as GPSG this would be most unnatural. GPSG envisages 'atomic level information' as a structured entity, a so-called feature structure. The purpose of the present section is to show how the intuition concerning structured atomic information that underlies GPSG can be captured. First we define a standard notion of feature structure and present a language $\mathcal{L}^F$ for talking about them. We then show how to 'layer' $\mathcal{L}$ over $\mathcal{L}^F$, yielding a language $\mathcal{L}(\mathcal{L}^F)$ capable of formulating the central ideas of GPSG in a natural way.

A *feature structure* is a labeled decorated directed graph. The elements of feature structures will be called *points*, and to say that feature structures are *labeled directed* graphs simply means that one can envisage the points being linked by arrows bearing a label. The most important constraint on feature structures concerns these labeled arrows: there is no point in any feature structure from which two distinct arrows bearing the same label emerges. In effect, labeled arrows are representations of partial functions; these partial functions are usually called *features*. As to the *decorations*, we envisage the points of feature structures being adorned with pieces of information of linguistic interest.

Let's make this precise. We assume that the linguistic theory we are working with tells us what features and decorations may be used. That is, we assume that our linguistic theorising gives us a signature $\langle \mathcal{F}, \mathcal{D} \rangle$ where both $\mathcal{F}$ and $\mathcal{D}$ are non-empty denumerable sets, the set of *features* and the set of *decorations* respectively. Typical elements of $\mathcal{F}$ might be CASE, NUMBER, PERSON and AGREEMENT; while typical elements of $\mathcal{D}$ might be *genitive*, *singular*, *plural*, *1st*, *2nd*, and *3rd*. We now define:

Definition 2.2 (Feature structures) *A feature structure of signature $\langle \mathcal{F}, \mathcal{D} \rangle$ is a triple* $\mathbf{F}$ *of the form $\langle U, \{R_f\}_{f \in \mathcal{F}}, \{Q_d\}_{d \in \mathcal{D}} \rangle$, where U is a non-empty set; for all $f \in \mathcal{F}$, R_f is a binary relation on U that is a partial function; and for each $d \in \mathcal{D}$, Q_d is a unary relation on U.* □

As we have defined them feature structures are multimodal Kripke models; and indeed the language $\mathcal{L}^F$ we now introduce will be the obvious modal language for talking about them, with the R_f serving to interpret its modalities, and the Q_d interpreting its propositional symbols. However we shall continue to call these entities feature structures, reserving the words 'model' and 'Kripke model' for the objects that interpret our language of trees $\mathcal{L}$.

The language $\mathcal{L}^F$ (of signature $\langle \mathcal{F}, \mathcal{D} \rangle$) contains the following items: all the elements of $\mathcal{D}$ (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).[2] The set of wffs of $\mathcal{L}^F$ is the smallest set containing all the propositional symbols (that is, all the elements of $\mathcal{D}$) closed under the Boolean and modal operators (that is, the

[2] Our discussion of $\mathcal{L}^F$ and its semantics is of necessity somewhat brief. For a more leisurely account see Blackburn and Spaan (1993).

elements of $\mathcal{F}$). Thus a typical wff of $\mathcal{L}^F$ might be the following:

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

In passing, a computational linguist would probably write this wff as follows:

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

Such two dimensional wffs are called Attribute Value Matrices, and they are essentially a perspicuous notation for $\mathcal{L}^F$ wffs.

$\mathcal{L}^F$ wffs are interpreted on feature structures as follows. For any feature structure $\mathbf{F}$ (that is, $\langle U, \{R_f\}_{f \in \mathcal{F}}, \{Q_d\}_{d \in \mathcal{D}}, \rangle$) and any point $u \in U$:

$$\begin{array}{lll} \mathbf{F}, u \models d & \textit{iff} & u \in Q_d, \textit{ for all } d \in \mathcal{D} \\ \mathbf{F}, u \models \neg\phi & \textit{iff} & \textit{not } \mathbf{F}, u \models \phi \\ \mathbf{F}, u \models \phi \wedge \psi & \textit{iff} & \mathbf{F}, u \models \phi \textit{ and } \mathbf{F}, u \models \psi \\ \mathbf{F}, u \models \langle f \rangle \phi & \textit{iff} & \exists u'(u R_f u' \textit{ and } \mathbf{F}, u' \models \phi). \end{array}$$

If $\mathbf{F}, u \models \phi$ then we say that ϕ is *satisfied* (or *true*) in $\mathbf{F}$ at u. The most obvious definition of validity in $\mathcal{L}^F$ is as follows: a wff ϕ is valid iff it is satisfied at all points in all feature structures. However, many feature structures are linguistically uninteresting, so we will confine our attention to *finite point-generated feature structures* (or finite *rooted* feature structures). These are finite feature structures $\mathbf{F}$ that contain a point u_0 such that any other point u of $\mathbf{F}$ can be reached by making a finite number of feature transitions from u_0. (Such a point u_0 is said to *generate* $\mathbf{F}$.) Thus an alternative definition suggests itself: an $\mathcal{L}^F$ wff is valid iff it is satisfied at all points in all finite, point generated feature structures. Actually, as we shall later see, both definitions yield the same set of validities.

With $\mathcal{L}^F$ and its semantics defined, we are ready to define a language for talking about the structures underlying GPSG: trees decorated with feature structures. The language is called $\mathcal{L}(\mathcal{L}^F)$, that is, the language $\mathcal{L}$ *layered over* the language $\mathcal{L}^F$, and it is defined in a very simple way. We simply choose **Prop** to be $\mathcal{L}^F$ and then construct the $\mathcal{L}$ wffs over this base in the usual way. As a result, we've given an 'internal structure' (namely, a modal structure) to the propositional symbols of $\mathcal{L}$. For further discussion see Blackburn, Gardent and Meyer-Viol (1993).

Syntactically that's all there is to it; what about the semantics? There is a straightforward interpretation for $\mathcal{L}(\mathcal{L}^F)$ in terms of the following entities:

Definition 2.3 (Feature decorated trees) *By a (finite, ordered, binary)* feature structure decorated tree *(of signature $\langle \mathcal{F}, \mathcal{D} \rangle$) is meant a triple $\langle \mathbf{O}, Z, z \rangle$ where $\mathbf{O}$ is the presentation of a finite ordered binary tree, Z is a function that assigns to each node u of $\mathbf{O}$ a finite, point-generated feature structure (of signature $\langle \mathcal{F}, \mathcal{D} \rangle$), and z is a function that assigns to each node u of $\mathbf{O}$ a point $z(u) \in Z(u)$ that generates $Z(u)$.* □

Two comments about feature decorated trees are in order. First of all, they seem to do justice to the ideas of GPSG. An examination of Gazdar *et al* (1985) suggests that they are a natural mathematical embodiment of the ontology underlying GPSG (modulo the fact that we are working only with binary trees). Second, in a number of recent talks Dov Gabbay has emphasized the importance of 'fibred semantics' for combined logics. By a combined logic he means a layered language of the sort exemplified by $\mathcal{L}(\mathcal{L}^F)$; and feature decorated trees are a nice example of what he means by a fibred semantics — the definition fibres a tree over a collection of feature structures. Finger and Gabbay (1992) gives a very clear and detailed account of such systems and proves a number of useful results which we shall make use of later.

To interpret $\mathcal{L}(\mathcal{L}^F)$ wffs on feature structure decorated trees, all we have to do is alter the base clause of the satisfaction definition for $\mathcal{L}$. Let $\mathbf{M} = \langle \mathbf{O}, Z, z \rangle$ be a feature structure decorated tree, and w any node in $\mathbf{O}$. Then for all $\mathcal{L}^F$ wffs ϕ, $\mathbf{M}, w \models \phi$ iff $Z(w), z(w) \models \phi$. If $\mathbf{M}$ $(= \langle \mathbf{O}, Z, z \rangle)$ is a feature structure decorated tree, w is a node in $\mathbf{O}$, ϕ is an $\mathcal{L}(\mathcal{L})^F$ wff and $\mathbf{M}, w \models \phi$ then we say that ϕ is *satisfied* (or *true*) in $\mathbf{M}$ at w. If ϕ is satisfied at all nodes w of all feature structure decorated trees, then we say that ϕ is *valid*.

To sum up: when in the course of evaluating an $\mathcal{L}(\mathcal{L}^F)$ wff at a node w we encounter an $\mathcal{L}^F$ wff (that is, when we reach what used to be the 'atomic' level) we jump into the feature structure associated with w (that is, $Z(w)$), and start evaluating the $\mathcal{L}^F$ wff at $z(w)$. This atomic level change is the only change needed: the remaining clauses are those that were given in the satisfaction definition for $\mathcal{L}$. In short, the 'top' layer of language $\mathcal{L}$ moves us round trees in the familiar way, while the 'bottom' layer of language $\mathcal{L}^F$ moves us round feature structures.

Having assembled the machinery, let's briefly discuss how GPSG puts it to work. One of the central insights of GPSG is that by making systematic use of the structured information embodied in feature structures, it is possible to give elegant accounts of many troublesome grammatical phenomena. First, context free rules in GPSG makes use of feature information. The GPSG analogue of $VP \longrightarrow V\ NP$ (*'A verb phrase is made up of a verb followed by a noun phrase'*) might be something like:

$$\begin{aligned} \neg noun \wedge verb \wedge \langle \text{BAR} \rangle two \quad \rightarrow \quad & \downarrow^1(\neg noun \wedge verb \wedge \langle \text{BAR} \rangle zero) \wedge \\ & \downarrow^2(noun \wedge \neg verb \wedge \langle \text{BAR} \rangle two). \end{aligned}$$

For the purposes of the present discussion it is the *form* of this wff that is important: the 'outer' (or $\mathcal{L}$) level of this wff has the same form as the wffs we used to capture the parse trees of context free grammars. But because of the 'inner' (or $\mathcal{L}^F$) level it reaches inside the feature structures and insists that certain conditions must hold there.

Secondly, and more interestingly, GPSG imposes a number of global restrictions on the way feature structures can be distributed over trees. The simplest

of these is the *foot feature principle*. When a natural language is analysed GPSG style, certain information is classified as 'foot information'. For a feature decorated tree to be acceptable, foot information must be 'passed up' to the feature structure associated with the mother node.[3] This is essentially to demand the validity of the following $\mathcal{L}(\mathcal{L}^F)$ wff:

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

Now, every word in a GPSG lexicon is associated with feature information, that is, with an $\mathcal{L}^F$ wff. If a certain word bears foot information then the foot feature principle forces this information to trickle up the tree from any terminal node where the word is inserted. It is quite possible that this foot information is incompatible with other information present in the structure. If this happens the structure is judged 'bad' that is, as not being the representation of a grammatical sentence. It is in this manner that 'feature passing' allows more refined accounts of grammaticality to be developed.

In short, GPSG is essentially a collection of axioms stipulating which feature decorated trees correspond to grammatical structures and which do not. Some of the theory's constraints are essentially a generalisation of the idea of phrase structure grammars to the richer setting of feature decorated trees; but in addition there is a set of global constraints, such as the foot feature convention, which act as a further filter on ungrammaticality. The way these various principles interact enables GPSG to give a neat account of a variety of phenomena in a wide range of languages. Many of these principles can be formalised in $\mathcal{L}(\mathcal{L}^F)$. For further discussion of GPSG from the present perspective see Blackburn, Gardent and Meyer-Viol (1993).

We close this section with a general warning. The reader should *not* conclude from our discussion that layered modal languages or feature structure decorated trees are all there is to modelling grammatical frameworks that make use of feature structures. This is simply false. For a start, different theories use feature structures in different ways. For example, Lexical Functional Grammar (LFG) (see Kaplan and Bresnan (1982)) uses them to model grammatical relations such as subject and object, and Head Driven Phrase Structure Grammar (HPSG) (see Pollard and Sag (1987)) uses them for a wide variety of tasks. Moreover, while both GPSG and LFG have composite ontologies made up of trees and feature structures, the theories 'glue' these building blocks together differently. Like GPSG, substantial parts of the LFG formalism can be viewed as a certain combination of modal logics; but the combination in question is not the simple layering + fibering idea embodied in $\mathcal{L}(\mathcal{L}^F)$, and the resulting systems have very different logical properties.

[3] As far as later versions of GPSG are concerned this is something of a simplification; in such systems feature passing involves the use of defaults. For a discussion of some of the logical issues raised see Evans (1987).

3 The axiomatisation

In this section we present an axiomatisation of the logic of finite ordered binary trees. We will later prove that this axiomatisation (which we call **Lot**) is complete. As axioms we take any suitable axiomatisation of propositional calculus, together with all instances of the schemas B1–B10, E1–E4, D1–D9 and F1–F3 below. As rules of inference we take modus ponens (if ϕ and $\phi \rightarrow \psi$ are provable then so is ψ) and the rule of necessitation in $\Downarrow^1$, $\Downarrow^2$, $\Uparrow$, $\Downarrow^*$ and $\Uparrow^*$. That is, if ϕ is provable then so are $\Downarrow^1\phi$, $\Downarrow^2\phi$, $\Uparrow\phi$, $\Downarrow^*\phi$ and $\Uparrow^*\phi$. Formal proofs are finite sequences of wffs built using the axioms and rules of inference in the usual way. If a wff ϕ is provable then we write $\vdash \phi$ and say that ϕ is a theorem.

With these generalities to hand, let us examine the details. The axioms split naturally into four groups. First of all, there are the axioms for the basic operators.

B1 $\Downarrow^1(\phi \rightarrow \psi) \rightarrow (\Downarrow^1\phi \rightarrow \Downarrow^1\psi)$

B2 $\Downarrow^2(\phi \rightarrow \psi) \rightarrow (\Downarrow^2\phi \rightarrow \Downarrow^2\psi)$

B3 $\Uparrow(\phi \rightarrow \psi) \rightarrow (\Uparrow\phi \rightarrow \Uparrow\psi)$

B4 $\downarrow^1\phi \rightarrow \Downarrow^1\phi$

B5 $\downarrow^2\phi \rightarrow \Downarrow^2\phi$

B6 $\uparrow\phi \rightarrow \Uparrow\phi$

B7 $\phi \rightarrow \Downarrow^1\uparrow\phi$

B8 $\phi \rightarrow \Downarrow^2\uparrow\phi$

B9 $\phi \rightarrow \Uparrow\downarrow\phi$

B10 $\downarrow^2\top \rightarrow \downarrow^1\top$

Most of this is familiar. B1–B3 are universally valid modal principles, while B4–B6 reflect the partial functional (or 'deterministic') nature of the $\succ_1$, $\succ_2$ and $\prec$ relations. B7 and B8 are familiar from tense logic: they record the fact that both the converse of $\succ_1$ and the converse of $\succ_2$ are contained in $\prec$. B9 is closely related and says that the converse of $\prec$ is contained in $\succ_1 \cup \succ_2$ (to see this, recall that $\downarrow\phi$ is shorthand for $\downarrow^1\phi \vee \downarrow^2\phi$). Finally, B10 takes account of the fact that in ordered binary trees, the existence of a $\succ_2$ successor to some node entails the existence of a $\succ_1$ successor to that same node.

The next group of axioms deals with the transitive closure operators and their interactions with the basic operators.

E1 $\Downarrow^*\phi \leftrightarrow (\phi \wedge \Downarrow\Downarrow^*\phi)$

E2 $\Uparrow^*\phi \leftrightarrow (\phi \wedge \Uparrow\Uparrow^*\phi)$

E3 $\Downarrow^*(\phi \rightarrow \Downarrow\phi) \rightarrow (\phi \rightarrow \Downarrow^*\phi)$

E4 $\Uparrow^*(\phi \rightarrow \Uparrow\phi) \rightarrow (\phi \rightarrow \Uparrow^*\phi)$

These are familiar from temporal logic and Propositional Dynamic Logic; see Goldblatt (1992). They reflect the fact that $\succ_*$ and $\prec_*$ are the reflexive transitive closures of $\succ$ and $\prec$ respectively.

The intended meaning of the defined symbols was discussed in the introduction; the next group of axiom pins these down:

D1 $s \leftrightarrow \Uparrow\bot$

D2 $t \leftrightarrow \Downarrow\bot$

D3 $\downarrow\phi \leftrightarrow \downarrow^1\phi \vee \downarrow^2\phi$

D4 $\Downarrow^1\phi \leftrightarrow \neg\downarrow^1\neg\phi$

D5 $\Downarrow^2\phi \leftrightarrow \neg\downarrow^2\neg\phi$

D6 $\Uparrow\phi \leftrightarrow \neg\uparrow\neg\phi$

D7 $\Downarrow\phi \leftrightarrow \neg\downarrow\neg\phi$

D8 $\Downarrow^*\phi \leftrightarrow \neg\downarrow^*\neg\phi$

D9 $\Uparrow^*\phi \leftrightarrow \neg\uparrow^*\neg\phi$

Finally we turn to the axioms that give the system its flavour, namely those that reflect the fact that our intended models are all *finite*.

F1 $\downarrow^*t$

F2 $\uparrow^*s$

F3 $\phi \rightarrow \downarrow^*(\phi \wedge \Downarrow\Downarrow^*\neg\phi)$

F1 and F2 are straightforward: no matter where we are in a finite tree we are only a finite number of $\prec$ steps away from the root node (which is what F1 says) and a finite number of $\succ$ steps away from at least one terminal node (which is what F2 says). More interesting is F3. Roughly speaking, it says that if ϕ holds at any node in a tree, then this node dominates a ϕ node not dominating any other ϕ nodes. It is this axiom that will enable us to maintain the finiteness of the tree constructed in the completeness proof. As it plays such a crucial role, let's look at it more closely.

All instances of F3 are valid in our intended semantics. For suppose some wff ϕ is true at a point w_1 in a model $\mathbf{M}$. Now, either w_1 dominates a distinct

node w_2 such that $\mathbf{M}, w_2 \models \phi$, or this is not the case. If this is *not* the case then we are through: for it is immediate that $\mathbf{M}, w_1 \models \phi \wedge \Downarrow\Downarrow^*\neg\phi$, and as $w_1 \succ_* w_1$ it follows that $\mathbf{M}, w_1 \models \downarrow^*(\phi \wedge \Downarrow\Downarrow^*\neg\phi)$, and we have verified the consequent of the axiom. So suppose that there *is* a point w_2 such that $w_1 \neq w_2$, $w_1 \succ_* w_2$, and $\mathbf{M}, w_2 \models \phi$. Now we ask: does w_2 dominate a distinct point w_3 such that $\mathbf{M}, w_3 \models \phi$? If the answer is 'no' then it follows that $\mathbf{M}, w_1 \models \downarrow^*(\phi \wedge \Downarrow\Downarrow^*\neg\phi)$ and we have verified the consequent of the axiom. On the other hand, if the answer is 'yes', then we repeat the question, asking whether there is a distinct w_4 ... in short, we keep asking whether or not there is a lower node that satisfies ϕ, and as soon as we get the answer 'no' we have our desired result. And we *must* eventually get the answer 'no', for as we are working with *finite* trees, our original point w_1 dominates only finitely many nodes.

Note that all instances of $\phi \rightarrow \uparrow^*(\phi \wedge \Uparrow\Uparrow^*\neg\phi)$, the mirror image of F3, are also valid. However we don't need them as axioms: its an easy exercise to show that they are all derivable in **Lot**. This derivability of the mirror image reflects a fairly obvious fact about our models. When we look downwards in a tree we may see complex branching structure, and a special axiom (namely F3) is needed to cope with this. However when we look upwards we see a nice regular linear structure, and the deductive power we already have copes successfully.[4]

The discussion of this section has established that **Lot** is sound: the axioms are valid, and the rules of inference clearly preserve validity. We are ready to turn to the question of its completeness.

4 The completeness proof

We begin with the following observation: a completeness result for **Lot** must be a weak completeness result, for as we are working only over *finite* trees there is an obvious compactness failure. For example, let Φ be $\{p, \uparrow p, \uparrow\uparrow p, \uparrow\uparrow\uparrow p, \ldots\}$. Any finite subset of Φ has a model, but it is impossible to satisfy all the wffs of Φ in the same model.

In fact **Lot** is weakly complete and the proof falls into two parts. In the first part ('Preliminaries') we define the basic entities we use to build our model, prove a number of results about them, and finally state and prove the Truth Lemma that we shall use. Much of this material is familiar from the literature on temporal logics for programs and Propositional Dynamic Logic. We have given fairly complete proof details, but occasionally the reader may find it useful to consult Goldblatt (1992) or van Benthem and Meyer-Viol (to appear).

In the subsequent part ('Building the model') we turn to the heart of the proof. The problem is this: we need to build a model, but this model must be based on a *finite* tree. An inductive construction suggests itself, but can it be

[4] It's also interesting to note (as Jan van Eijck pointed out to us) that F1 can be derived from the other axioms. F3 plays a key role in its derivation.

shown to terminate after a finite number of steps? By making use of axiom F3 it is possible to guarantee this.

4.1 Preliminaries

The first notion we need is that of a *closure* of a set of sentences. Recall that a set of sentences Σ is said to be closed under subformulas iff for all $\phi \in \Sigma$, if ψ is a subformula of ϕ then $\psi \in \Sigma$. We need to work with closures that offer more structure than just closure under subformulahood, thus, following Fisher and Ladner (1979) we define:

Definition 4.1 (Closures) *If Σ is a set of formulas, $Cl(\Sigma)$ is defined to be the smallest set of sentences containing Σ that is closed under subformulas and satisfies the following additional properties:*

1. *$\uparrow^*\phi \in Cl(\Sigma)$ implies $\uparrow\uparrow^*\phi \in Cl(\Sigma)$.*
2. *$\downarrow^*\phi \in Cl(\Sigma)$ implies $\downarrow\downarrow^*\phi \in Cl(\Sigma)$.*
3. *$\downarrow^1\top$, $\downarrow^2\top$, $\uparrow\top$, $\uparrow^*s$ and $\downarrow^*t \in Cl(\Sigma)$.*
4. *If $\phi \in Cl(\Sigma)$ and ϕ is not of the form $\neg\psi$ then $\neg\phi \in Cl(\Sigma)$.*

$Cl(\Sigma)$ is called the closure *of Σ. Note that if Σ is finite then $Cl(\Sigma)$ is finite.* □

In fact, because of the failure of compactness already noted (and because our ultimate goal is to build a *finite* tree) we shall only be interested in finite closures. The next step is to pick out the subsets of $Cl(\Sigma)$ needed for building models. Following van Benthem and Meyer-Viol (to appear) we define:

Definition 4.2 (Atoms) *If Σ is a set of formulas, then $At(\Sigma)$ consists of all the maximal consistent subsets of $Cl(\Sigma)$. That is, $A \in At(\Sigma)$ means that A is consistent, and, if B is a consistent subset of $Cl(\Sigma)$ such that $A \subseteq B$, then $A = B$. The elements of $At(\Sigma)$ are called* atoms. □

For example, if Σ is the set of all formulas then elements of $At(\Sigma)$ are just the usual maximal consistent sets of sentences. More generally, the following relationship holds between atoms and the familiar notion of maximal consistent set of sentences:

Lemma 4.3 *Let $\mathcal{M}$ be the set of all maximal consistent sets of sentences, and Σ be some set of sentences. Then $At(\Sigma) = \{M \cap Cl(\Sigma) : M \in \mathcal{M}\}$.*

Proof: Straightforward. □

The next two lemmas guarantee that we have a plentiful supply of atoms, and that atoms have nice properties. Direct proofs of these facts are straightforward, or one can observe that analogues of these results are familiar properties of maximal consistent sets of sentences and use the previous lemma to transfer them to atoms.

Lemma 4.4 *If $\phi \in Cl(\Sigma)$ and ϕ is consistent then there is an $A \in At(\Sigma)$ such that $\phi \in A$.* □

Lemma 4.5 *For any set of sentences Σ, and any $A \in At(\Sigma)$:*

1. *If $\phi \in Cl(\Sigma)$ then $\phi \in A$ iff $\neg\phi \notin A$.*
2. *If $\phi \wedge \psi \in Cl(\Sigma)$ then $\phi \wedge \psi \in A$ iff $\phi \in A$ and $\psi \in A$*
3. *If $\phi \rightarrow \psi$ and $\phi \in Cl(\Sigma)$, then $\phi \rightarrow \psi$ and $\phi \in A$ implies $\psi \in A$.*
4. *If $\uparrow^*\phi \in Cl(\Sigma)$ then $\uparrow^*\phi \in A$ iff $\phi \in A$ or $\uparrow\uparrow^* A \in A$.*
5. *If $\downarrow^*\phi \in Cl(\Sigma)$ then $\downarrow^*\phi \in A$ iff $\phi \in A$ or $\downarrow\downarrow^* A \in A$.*
6. *$\uparrow^* s$, $\downarrow^* t$ and $\top \in A$.* □

In the completeness proof that follows we shall work with closures of *finite* sets of sentences Σ. Let us assume from now on that Σ always denotes a finite set of sentences.

In the finite case $At(\Sigma)$ has a very pleasant structure. We can enumerate all the singly negated formulas in $Cl(\Sigma)$ as one list $\neg\sigma_1, \ldots, \neg\sigma_n$ (we call this the *negative enumeration*) and all the non-negated formulas in $Cl(\Sigma)$ as another list $\sigma_1, \ldots, \sigma_n$ (we call this the *positive enumeration*), in such a way that the i-th item on the negative enumeration is the negation of the i-th item on the positive enumeration. Note that for any formula ϕ in $Cl(\Sigma)$ there is a formula ψ_i $(1 \leq i \leq n)$ such that ϕ is logically equivalent to ψ_i and ψ_i occurs on either the negative or positive enumerations at the i-th place. Define a *pointwise selection* from the negative and positive enumerations to be the result of choosing, for each i $(1 \leq i \leq n)$, a wff from one of the enumerations. Further, note that the conjunction of all the wffs in a consistent pointwise selection is logically equivalent to the conjunction of all the wffs in some atom, and conversely.

Lemma 4.6 *Suppose $At(\Sigma) = \{A_1, \ldots, A_n\}$. Then $\vdash \bigwedge A_1 \vee \cdots \vee \bigwedge A_n$.*

Proof: Simple propositional logic: use the tautology $\alpha \leftrightarrow ((\alpha \wedge \beta) \vee (\alpha \wedge \neg\beta))$ repeatedly. □

We now define the finite analogs of the canonical model:

Definition 4.7 *For all A, B in $At(\Sigma)$ define:*

1. *$A >_1 B$ iff $\bigwedge A \wedge \downarrow^1 \bigwedge B$ is consistent.*
2. *$A >_2 B$ iff $\bigwedge A \wedge \downarrow^2 \bigwedge B$ is consistent.*
3. *$A > B$ iff $A >_1 B$ or $A >_2 B$.*
4. *$A < B$ iff $\bigwedge A \wedge \uparrow \bigwedge B$ is consistent.*

5. $A >_* B$ *iff* $\bigwedge A \wedge \downarrow^* \bigwedge B$ *is consistent.*

6. $A <_* B$ *iff* $\bigwedge A \wedge \uparrow^* \bigwedge B$ *is consistent.*

Let $\mathcal{C}^\Sigma$ *be* $At(\Sigma)$ *together with these six relations.* □

$\mathcal{C}^\Sigma$ is not generally a tree; nonetheless, as the following sequence of lemmas shows, it does have a number of useful properties.[5]

Lemma 4.8 *For all* A, B *in* $At(\Sigma)$*:*

1. $A >_1 B$ *implies* $B < A$.

2. $A >_2 B$ *implies* $B < A$.

3. $A < B$ *implies* $B >_1 A$ *or* $B >_2 A$.

4. $A < B$ *iff* $B > A$.

5. $A >_* B$ *iff there is a finite sequence of atoms* $A = A_1 > \cdots > A_n = B$.

6. $A <_* B$ *iff there is a finite sequence of atoms* $A = A_1 < \cdots < A_n = B$.

Proof: These results are standard, or simple variations on standard results. *1* and *2* are proved using B7 and B8 respectively in the manner familiar from tense logic; *3* is a minor variation on this theme, making use of B9 and D3; while *4* follows immediately from *1* – *3*. The proofs of *5* and *6* make use of E1–E4 in the standard fashion; see Goldblatt (1992) or van Benthem and Meyer-Viol (to appear) for further details. □

Lemma 4.9 *For any atom* A, $s \in A$ *iff no formula of the form* $\uparrow\phi$ *is in* A*; and* $t \in A$ *iff no formula of the form* $\downarrow^1\phi$ *or* $\downarrow^2\phi$ *is in* A.

Proof: We prove the second equivalence. Suppose that $t \in A$ and further suppose for the sake of a contradiction that a formula of the form $\downarrow^1\phi$ or $\downarrow^2\phi$ is in A; without loss of generality we suppose that it is a formula of the form $\downarrow^1\phi$. As $t \in A$ we have $\Downarrow\bot \in A$ by D2, but as $\downarrow^1\phi \in A$ we have that $\downarrow^1\bot \in A$, which contradicts the consistency of A. This proves the left to right direction.

For the right to left direction, suppose no formula of the form $\downarrow^1\phi$ or $\downarrow^2\phi$ is in A, and further suppose for the sake of a contradiction that $t \notin A$. Now $\downarrow^* t \in A$ (by Lemma 4.5 clause 6) and by Lemma 4.5 clause 5 this means that either $t \in A$ or $\downarrow\downarrow^* t \in A$. But $t \notin A$, thus $\downarrow\downarrow^* t \in A$. That is, either $\downarrow^1\downarrow^* t$ or $\downarrow^2\downarrow^* t$ is in A, which contradicts our original assumption. We conclude that t must be in A after all, the required result. □

[5] It is perhaps worth making an aside for readers familiar with the approach of Goldblatt (1992). One can also regard $\mathcal{C}^\Sigma$ as arising by *filtrating the canonical model*; this is Goldblatt's approach and it is probably the standard one. We have a slight preference for the present approach (developed in detail in van Benthem and Meyer-Viol (to appear)) because it deals with finite structures right from the start.

Lemma 4.10 *Suppose* $\diamond\phi \in Cl(\Sigma)$, *where* $\diamond \in \{\downarrow^1, \downarrow^2, \downarrow, \uparrow, \downarrow^*, \uparrow^*\}$. *Then for all atoms* A *and* B *we have that* $AR^{\diamond}B$ *and* $\phi \in B$ *implies* $\diamond\phi \in A$, *where* $R^{\diamond}$ *denotes the relation on the canonical graph corresponding to the operator* $\diamond$.

Proof: We treat the case for $\downarrow^*$; the others are similar. Suppose $\downarrow^*\phi \in Cl(\Sigma)$ and further suppose that $A >_* B$ and $\phi \in B$. Thus we have that $\bigwedge A \wedge \downarrow^* \bigwedge B$ is consistent, and as $\phi \in B$ we have $\bigwedge A \wedge \downarrow^*\phi$ is consistent. As $\downarrow^*\phi \in Cl(\Sigma)$ and A is an atom (which means it is *maximal* consistent in $Cl(\Sigma)$), $\downarrow^*\phi \in A$. □

Lemma 4.11 *For all atoms* A *and* B:

1. *If* $A >_1 B$ *and* $\downarrow^1\phi \in A$ *then* $\phi \in B$.
2. *If* $A >_2 B$ *and* $\downarrow^2\phi \in A$ *then* $\phi \in B$.
3. *If* $A < B$ *and* $\uparrow\phi \in A$ *then* $\phi \in B$.

Proof: We treat the third case. Suppose $A < B$ and $\uparrow\phi \in A$. Thus $\bigwedge A \wedge \uparrow \bigwedge B$ is consistent, and as $\uparrow\phi \in A$ we have that $\uparrow\phi \wedge \uparrow \bigwedge B$ is consistent. It is an easy consequence of B6, the functionality axiom for $\uparrow$, that $\vdash \uparrow\gamma \wedge \uparrow\lambda \rightarrow \uparrow(\gamma \wedge \lambda)$; thus it follows that $\uparrow(\phi \wedge \bigwedge B)$ is consistent, which in turn means that $\phi \wedge \bigwedge B$ is consistent. But $\phi \in Cl(\Sigma)$ (as $\uparrow\phi \in A$), so as B is an atom, $\phi \in B$.

The other two cases are similar, and make use of B4 and B5. □

Definition 4.12 *Let* $\mathbf{O}$ $(= \langle W, \succ_1, \succ_2, \succ, root, \Theta\rangle)$ *be the presentation of a finite ordered binary tree and* Σ *a finite set of sentences. By a* decoration of $\mathbf{O}$ by $At(\Sigma)$ *is meant a mapping* $h : W \longrightarrow At(\Sigma)$. *Suppose* h *is a decoration with the following properties:*

1. *For all* $w, w' \in T$, *if* $w \succ_1 w'$ *then* $h(w) >_1 h(w')$,
2. *For all* $w, w' \in T$, *if* $w \succ_2 w'$ *then* $h(w) >_2 h(w')$,
3. $s \in h(root)$,
4. $t \in h(w)$, *for all* $w \in \Theta$.

Then h *is a* sensible decoration. □

Lemma 4.13 *Let* h *be a sensible decoration of* $\mathbf{O}$ *by* $At(\Sigma)$. *Then, for all nodes* w *and* w' *of* $\mathbf{O}$, $w \succ w'$ *implies* $h(w) > h(w')$ *and* $w \prec w'$ *implies* $h(w) < h(w')$.

Proof: Suppose $w \succ w'$. Thus $w' \succ_1 w$ or $w' \succ_2 w$. But h is a sensible decoration, thus $h(w') >_1 h(w)$ or $h(w') >_2 h(w)$. Thus, by definition, $h(w) > h(w')$, and the first implication is proved. As for the second implication, suppose that $w \prec w'$. Thus $w' \succ w$. Thus, making use of the first implication, $h(w') > h(w)$. Thus, by Lemma 4.8 clause 4, $h(w) < h(w')$, the required result. □

Definition 4.14 (Models induced by decorations) *Let h be a decoration of $\mathbf{O}$ by $At(\Sigma)$. Let V be the valuation on $\mathbf{O}$ defined by $w \in V(p)$ iff $p \in h(w)$, for all nodes w in $\mathbf{O}$. The* model induced by *h is the pair $\langle \mathbf{O}, V \rangle$.* □

Lemma 4.15 (Truth lemma) *Let h be a sensible decoration of $\mathbf{O}$ by $At(\Sigma)$, and let $\mathbf{M}$ $(= \langle \mathbf{O}, V \rangle)$ be the model induced by h. Then, for all $\phi \in Cl(\Sigma)$, and all nodes w in $\mathbf{M}$: $\mathbf{M}, w \models \phi$ iff $\phi \in h(w)$.*

Proof: By induction on the structure of ϕ. For all atoms p we have that $\mathbf{M}, w \models p$ iff $w \in V(p)$ iff $p \in h(w)$ so the base case is clear. Assume the desired result holds for all wffs of degree less than n. Suppose that ϕ has degree n. If ϕ is of the form $\neg\psi$ or $\psi \vee \gamma$ then the desired result follows easily using Lemma 4.5, so let us consider the cases involving modalities.

Suppose ϕ has the form $\uparrow\psi$ and $\mathbf{M}, w \models \uparrow\psi$. Thus there is a w' such that $w \prec w'$ and $\mathbf{M}, w' \models \psi$. But as h is a sensible decoration, by Lemma 4.13 $h(w) < h(w')$, and by the inductive hypothesis $\psi \in h(w')$. So by Lemma 4.10 $\uparrow\psi \in h(w)$ as desired.

So assume $\mathbf{M}, w \not\models \uparrow\psi$. Then *either* $w = root$ *or* there is a w' such that $w \prec w'$ but $\mathbf{M}, w' \not\models \psi$. Now if $w = root$ then as h is a sensible decoration we have that $s \in h(w)$, thus by Lemma 4.9 we have $\uparrow\psi \notin h(w)$, the required result. On the other hand, suppose there is a w' such that $w \prec w'$ and $\mathbf{M}, w' \not\models \psi$. Thus $h(w) < h(w')$ (by Lemma 4.13) and by the inductive hypothesis $\psi \notin h(w')$. So $\uparrow\psi \notin h(w)$, the required result.

If ϕ is of the form $\downarrow^1\psi$ or $\downarrow^2\psi$, the argument is essentially the same as that just given, so let us consider the transitive closure operators.

Suppose $\mathbf{M}, w \models \downarrow^*\psi$. Then there is a node w' such that $w \succ_* w'$ and $\mathbf{M}, w' \models \psi$. That is, there is a finite sequence of nodes $w = w_1 \succ \cdots \succ w_k = w'$ such that $\mathbf{M}, w' \models \psi$. But as h is a sensible decoration we thus have (by Lemma 4.13) that $h(w) = h(w_1) > \cdots > h(w_k) = h(w')$, thus using Lemma 4.8 clause 5 we have $h(w) >_* h(w')$. Moreover, by the inductive hypothesis $\psi \in h(w')$. It thus follows from Lemma 4.10 that $\downarrow^*\psi \in h(w)$, as required.

So suppose $\mathbf{M}, w \not\models \downarrow^*\psi$. Then for all w' such that $w \succ_* w'$ we have $\mathbf{M}, w' \not\models \psi$. Thus, by the inductive hypothesis, for all w' such that $w \succ_* w'$ we have $\psi \notin h(w')$. Now if we suppose that $\downarrow^*\psi \in h(w)$, then by Lemma 4.4 we have that either $\psi \in h(w)$ or $\downarrow\downarrow^*\psi \in h(w)$. Taken together with the fact that $\mathbf{O}$ is a *finite* tree, this swiftly leads to contradiction (we leave the argument to the reader); we conclude that $\downarrow^*\psi \notin h(w)$ as required.

The arguments for $\uparrow^*\psi$ are similar. □

4.2 Building the model

With these preliminaries to hand we turn to the heart of the proof. Our task is to show that any consistent sentence ϕ is satisfiable. This is equivalent to showing that $s \wedge \downarrow^*\phi$ is satisfiable. Thus a natural strategy suggests itself.

Given a consistent sentence ϕ, form $At(\{s \wedge \downarrow^* \phi\})$, and inductively define a tree decorated by atoms from this set. First create a root node for the tree, and decorate it with any atom containing $s \wedge \downarrow^* \phi$. If there are no formulas of the form $\downarrow^1 \psi$ or $\downarrow^2 \psi$ in the decorating atom, stop. Otherwise, create the needed daughter nodes, extend the decoration in the obvious way, and so on.

Now this is the essence of what we'll do — but there is an obvious difficulty to be overcome. Our task is to make a *finite* decorated tree, but how can we guarantee that the inductive procedure just sketched produces only finitely many nodes? It is here that axiom F3 comes into play. F3 will allow us to assign each atom a unique 'level' measuring its distance from the atoms that contain the terminal nodes. Crucially, we will be able to show that if an atom A contains $\downarrow^1 \psi$ (or $\downarrow^2 \psi$) then there is an atom B of lower level than A containing ψ such that $A >_1 B$ (respectively, $A >_2 B$). This means that when inductively building our decorated tree we can always choose decorations of lower level, and doing this ensures that the construction halts after finitely many steps. The following sequence of lemmas shows that F3 really does allow us to impose such a level structure on the canonical graph.

Lemma 4.16 *If ϕ is consistent then $\phi \wedge \Downarrow\Downarrow^* \neg\phi$ is consistent.*

Proof: Suppose ϕ is consistent. As $\vdash \phi \rightarrow \downarrow^*(\phi \wedge \Downarrow\Downarrow^* \neg\phi)$ (this is F3), $\downarrow^*(\phi \wedge \Downarrow\Downarrow^* \neg\phi)$ is consistent, whence $\phi \wedge \Downarrow\Downarrow^* \neg\phi$ is consistent. □

Lemma 4.17 *Let $\mathcal{A}$ $(= \{A_1, \ldots, A_n\})$ and $\mathcal{B}$ $(= \{B_1, \ldots, B_m\})$ be non-empty subsets of $At(\Sigma)$ that partition $At(\Sigma)$. (That is, $\mathcal{A} \cap \mathcal{B} = \emptyset$, and $\mathcal{A} \cup \mathcal{B} = At(\Sigma)$.) Then for some A_i in $\mathcal{A}$, $A_i \wedge \Downarrow\Downarrow^*(\bigwedge B_1 \vee \cdots \vee \bigwedge B_m)$ is consistent.*

Proof: As all atoms are consistent, any disjunction of atoms is consistent. Thus $A_1 \vee \cdots \vee A_n$ is consistent and thus by the previous lemma

$$(\bigwedge A_1 \vee \cdots \vee \bigwedge A_n) \wedge \Downarrow\Downarrow^* \neg(\bigwedge A_1 \vee \cdots \vee \bigwedge A_n)$$

is consistent. Now by lemma 4.6 we have that

$$\vdash \bigwedge A_1 \vee \cdots \vee \bigwedge A_n \vee \bigwedge B_1 \vee \cdots \vee \bigwedge B_m,$$

or equivalently

$$\vdash \neg(\bigwedge A_1 \vee \cdots \vee \bigwedge A_n) \rightarrow \bigwedge B_1 \vee \ldots \vee \bigwedge B_m,$$

thus it follows that

$$(\bigwedge A_1 \vee \cdots \vee \bigwedge A_n) \wedge \Downarrow\Downarrow^*(\bigwedge B_1 \vee \cdots \vee \bigwedge B_m)$$

is consistent. But this in turn means that for some A_i in $\mathcal{A}$,

$$\bigwedge A_i \wedge \Downarrow\Downarrow^*(\bigwedge B_1 \vee \cdots \vee \bigwedge B_m)$$

is consistent, the required result. □

We now inductively define the levels on $At(\Sigma)$. The 0-th level, L_0, is defined to be $\{A \in At(\Sigma) : t \in A\}$. Next, suppose the i-th level, L_i, is defined. First we define:

$$S_i = \bigcup_{0 \leq j \leq i} L_j.$$

(That is, S_i is the 'sum' of all levels up to and including level i. Note that S_0 is just L_0.) Next, if $At(\Sigma)\backslash S_i$ is non-empty then the $i+1$-th level L_{i+1} exists and is defined as follows:

$$L_{i+1} = \{A \in At(\Sigma) : A \notin S_i \text{ and } A \wedge \Downarrow\Downarrow^* \bigvee_{B \in S_i} \bigwedge B \text{ is consistent}\}.$$

On the other hand, if $At(\Sigma)\backslash S_i$ is empty then there is no $i+1$-th level.

Lemma 4.18 *Every atom A in $At(\Sigma)$ belongs to exactly one level.*

Proof: It is clear from the inductive definition that every atom A belongs to at most one level, thus it remains to show that each atom belongs to some level.

We first show by induction that there are no empty levels. For the base case, note that as $\downarrow^* t$ is an axiom, t is consistent, thus there are atoms that contain t and so L_0 is non-empty. Further note that as S_0 is just L_0 we have that S_0 is also non-empty. For the inductive step we show that if S_i is non-empty and L_{i+1} exists, then L_{i+1} is non-empty. To see this note that for L_{i+1} to exist it must be the case that $At(\Sigma)\backslash S_i$ is non-empty. But then $At(\Sigma)\backslash S_i$ and S_i are a pair of non-empty sets that partition $At(\Sigma)$; thus applying lemma 4.17 we deduce that there is an atom A in $At(\Sigma)\backslash S_i$ such that $A \wedge \Downarrow\Downarrow^* \bigvee_{B \in S_i} \bigwedge B$ is consistent. But then A is in L_{i+1}, thus the $i+1$-th level is non-empty. It follows by induction that no level is empty.

But now it is easy to see that every atom A belongs to at least one level. For, as every level is non-empty, and as every atom belongs to at most one level, then as there are only finitely many atoms there must be a maximum level. Call this level L_{max}. Suppose for the sake of a contradiction that there is some atom A that does not belong to any level. This means, $A \notin S_{max}$. But this means that $At(\Sigma)\backslash S_{max}$ is non-empty, thus L_{max+1} exists and is non-empty: a contradiction. So every atom belongs to some level. □

Now for the vital lemma:

Lemma 4.19 *Suppose A is an atom belonging to L_i and that there is a wff of the form $\downarrow^1 \phi \in A$ (respectively, $\downarrow^2 \phi \in A$). Then there is an atom B belonging to L_m where $m < i$ (respectively, there is an atom C belonging to L_n where $n < i$) such that $A >_1 B$ (respectively, such that $A >_2 C$).*

Proof: Suppose A is an atom in L_i and that there is a wff of the form $\downarrow^1 \phi \in A$. Note that as $\downarrow^1 \phi \in A$ we have by lemma 4.9 that $t \notin A$, thus $i > 0$. This means that S_{i-1} exists and is non-empty. We will show that there is an atom B in S_{i-1} such that $A >_1 B$.

Suppose for the sake of a contradiction that this is not the case. That is, suppose that for all atoms B in S_{i-1} we have that $\bigwedge A \wedge \downarrow^1 \bigwedge B$ is inconsistent. This means that for all B in S_{i-1} we have $\vdash \bigwedge A \rightarrow \Downarrow^1 \neg \bigwedge B$. Enumerate all the atoms in S_{i-1} as $B_1, \ldots, B_n$. It follows by simple modal reasoning that

$$\vdash \bigwedge A \rightarrow \Downarrow^1 \neg (\bigwedge B_1 \vee \cdots \vee \bigwedge B_n).$$

Abbreviate $\bigwedge B_1 \vee \cdots \vee \bigwedge B_n$ to $\mathbf{B}$. Thus $\vdash \bigwedge A \rightarrow \Downarrow^1 \neg \mathbf{B}$.

As A is in L_i and $i > 1$, by our definition of levels we have that $\bigwedge A \wedge \Downarrow \mathbf{B}$ is consistent. Writing $\Downarrow$ explicitly in terms of $\Downarrow^1$ and $\Downarrow^2$, we thus have that $\bigwedge A \wedge \Downarrow^1 \mathbf{B} \wedge \Downarrow^2 \mathbf{B}$ is consistent. But we have $\vdash \bigwedge A \rightarrow \Downarrow^1 \neg \mathbf{B}$ by our previous argument, thus

$$\bigwedge A \wedge \Downarrow^1 \mathbf{B} \wedge \Downarrow^2 \mathbf{B} \wedge \Downarrow^1 \neg \mathbf{B}$$

is consistent. This means that $\bigwedge A \wedge \Downarrow^1 \mathbf{B} \wedge \Downarrow^1 \neg \mathbf{B}$ is consistent, which means that $\bigwedge A \wedge \Downarrow^1 \bot$ is consistent. As $\downarrow^1 \phi \in A$ we deduce that $\downarrow^1 \bot$ is consistent, which is impossible. So our original supposition was mistaken and we conclude that there is an atom B in S_{i-1} such that $A >_1 B$. This means that for some level L_m, where $m < i$, $B \in L_m$, the required result.

The result for the operator $\downarrow^2$ is proved analogously. □

We are now ready for the inductive construction of our desired model. Suppose ϕ is consistent. Form $At(\{s \wedge \downarrow^* \phi\})$. Let $\mathcal{W}$ be some denumerably infinite set; we shall use (finitely many) of its elements as the tree nodes.

Stage 0. Choose some $w_0 \in \mathcal{W}$, and some atom A_0 in $At(\{s \wedge \downarrow^* \phi\})$ such that $s \wedge \downarrow^* \phi \in A_0$. As ϕ is consistent, so is $s \wedge \downarrow^* \phi$, thus by Lemma 4.4 such an A_0 exists. Define W_0 to be $\{w_0\}$; $\succ_1^0$ to be $\emptyset$; $\succ_2^0$ to be $\emptyset$; and h_0 to be $\{\langle w_0, A_0 \rangle\}$.

Stage $n+1$. Suppose n stages of the inductive construction have been performed. We call a pair $\langle w, k \rangle$ (where $w \in W_n$ and $k \in \{1, 2\}$) an *unsatisfied demand* iff $\downarrow^k \phi \in h(w)$ but there is no $w' \in W_n$ such that $w \succ_k w'$.

If there are no unsatisfied demands the construction is complete.

Otherwise, choose an unsatisfied demand $\langle w, k \rangle$. Note that as $\downarrow^k \phi \in h(w)$, $h(w)$ cannot belong to level zero, for otherwise we would contradict Lemma 4.9. Let $w' \in \mathcal{W} \backslash W_n$. Let A_{n+1} be an atom such that $h(w) \succ_k A_{n+1}$ and A_{n+1} belongs to a *strictly lower* level than $h(w)$; Lemma 4.19 guarantees that such an A_{n+1} exists. Let j be 1 if k is 2, and 2 if k is 1. Define:

$$
\begin{aligned}
W_{n+1} &= W_n \cup \{w'\} \\
\succ_k^{n+1} &= \succ_k \cup \{\langle w, w' \rangle\} \\
\succ_j^{n+1} &= \succ_j \\
h_{n+1} &= h_n \cup \{\langle w', A_{n+1} \rangle\}.
\end{aligned}
$$

While adjoining a new node w' to w as described in the inductive step may result in new unsatisfied demands $\langle w', k \rangle$, where $k \in \{1, 2\}$, we were careful to choose $h(w')$ from a strictly lower level than $h(w)$. This means that in the course of the construction we will be forced to map the newly adjoined node w' to an atom of level zero; but doing so cannot give rise to an unsatisfied demand. Thus the construction process terminates after finitely many steps.

Let m be the stage at which the construction is completed. Define h to be h_m. Define W to W_m; define $\succ_1$ to be $\succ_1^m$; define $\succ_2$ to be $\succ_2^m$; define *root* to be w_0; and define Θ to be $\{w' : w' \not\succ_1 w \text{ and } w' \not\succ_1 w, \text{ for all } w \in W\}$. Define **O** to be

$$\langle W, \succ_1, \succ_2, \succ, \prec, node, \Theta \rangle.$$

Lemma 4.20 **O** *is the presentation of a finite tree. Moreover, h is a sensible decoration of* **O**.

Proof: That **O** is the presentation of a finite tree follows straightforwardly from its inductive definition. There is only one subtlety: if $w \succ_2 w'$ then how do we know that there is a w'' such that $w \succ_1 w''$ (and $w' \neq w''$)? In fact, by using our last remaining unused axiom, namely B10, we can guarantee this. For suppose $w \succ_2 w'$. The inductive construction would only have adjoined w' if at some stage $\langle w, 2 \rangle$ was an unsatisfied demand. But this means that some formula $\downarrow^2 \phi \in h(w)$. But then, by B10, $\downarrow^1 \top \in h(w)$. So if there is no w'' in W such that $w \succ_1 w''$ then $\langle w, 1 \rangle$ is an unsatisfied demand — but there are no unsatisfied demands.

Showing that h is a sensible decoration of **O** is straightforward. □

Theorem 4.21 (Completeness) *Every consistent sentence has a model.*

Proof: Given a consistent sentence ϕ, we use the inductive construction to build a tree decorated by $At(\{s \wedge \downarrow^* \phi\})$. By the previous lemma the decoration so constructed is sensible, thus by the Truth Lemma the induced model satisfies $s \wedge \downarrow^* \phi$ at its root node, hence ϕ is true somewhere in this model. □

5 Consequences of completeness

The most important consequence of completeness is that validity is decidable.

Theorem 5.1 *The set of valid wffs is recursive.*

Proof: The set of wffs that are *not* valid is recursively enumerable (r.e.). For, if a wff ϕ is not valid then it is falsifiable on some model. As all our models are based on *finite* ordered binary trees it is possible to write a procedure that systematically generates models and tests for the validity of ϕ on the models so produced. Any such procedure will eventually tell us that ϕ is not valid.

On the other hand, the set of valid wffs is also r.e.. For the set of **Lot** proofs is obviously an r.e. set (systematically generate finite sequences of wffs, discarding those that are not **Lot** proofs) and by the Completeness Theorem a wff is valid iff it is provable. As both the set of validities and its complement are r.e., the set of validities is recursive. □

Next, recall that in section 2 we defined a language $\mathcal{L}(\mathcal{L}^F)$, the result of layering $\mathcal{L}$ over $\mathcal{L}^F$. By making use of results proved by Finger and Gabbay (1992) we will be able to see that the set of valid $\mathcal{L}(\mathcal{L}^F)$ formulas is both recursively axiomatisable and decidable.

Finger and Gabbay's results are fairly general. Essentially they show how completeness and decidability results enjoyed by both languages participating in the layering process can be combined to obtain completeness and decidability results for the layered language. Now, we have just established such results for $\mathcal{L}$, so the next step is to ascertain that similar results hold for $\mathcal{L}^F$. As these results are well known (and rather simple) we only sketch the required proofs.

Suppose we have fixed some signature $\langle \mathcal{F}, \mathcal{A} \rangle$. The following axiomatisation suffices to capture the $\mathcal{L}^F$ wffs (over this signature) that are valid on all finite, point generated feature structures. As axioms take all the wffs of $\mathcal{L}^F$ (over this signature) that are instances of the following two schemas:

Feature 1 $[f](\phi \rightarrow \psi) \rightarrow ([f]\phi \rightarrow [f]\psi)$

Feature 2 $\langle f \rangle \phi \rightarrow [f]\phi$.

As rules of inference take modus ponens and the rule of necessitation for each 'box' modality (that is, if ϕ is provable, so is $[f]\phi$, for all $f \in \mathcal{F}$). A formal proof in this system is a sequence of wffs regulated by the axioms and rules of inference in the usual way, and $\vdash_F \phi$ means that ϕ is formally provable.

To prove completeness proceed as follows. Given a consistent $\mathcal{L}^F$ wff ϕ, use Lindenbaum's Lemma to form a maximal consistent set of sentences Φ that contains ϕ. The **Feature 2** axioms guarantee that the relations in the canonical model $\mathbf{M}^H$ for this system are partial functions, and by the usual argument $\mathbf{M}, \Phi \models \phi$; thus $\mathbf{M}^H$ is a feature structure that satisfies ϕ. Thus we have a (strong) completeness result for the axiomatisation with respect to the class of

all feature structures. The next step is to transform $\mathbf{M}^H$ into a finite point-generated feature structure that satisfies ϕ and thus prove the relevant (weak) completeness result. This is routine. Take the submodel of $\mathbf{M}^F$ generated by Φ, but only generate out m steps, where m is the maximal depth of nesting of modalities in ϕ, and only generate on the relations corresponding to modalities actually occurring in ϕ. This establishes the result. It also tells us a little more. First, as the axiomatisation is complete with respect to both the class of all feature structures and the class of all finite point-generated feature structures, the two definitions of $\mathcal{L}^F$ validity given in section 2 must coincide. A second consequence is the decidability of $\mathcal{L}^F$ validity, for the generation process yields an upper bound on the size of (finite) point generated feature structures that need to be inspected to determine ϕ's validity.

We now have all the information about $\mathcal{L}^F$ and $\mathcal{L}$ that we need to apply the results of Finger and Gabbay to $\mathcal{L}(\mathcal{L}^F)$. Following their account, here is an axiomatisation of the $\mathcal{L}(\mathcal{L}^F)$ validities:

Tree 1 All $\mathcal{L}(\mathcal{L}^F)$ instances of the **Lot** axioms.

Tree 2 All the **Lot** rules of inference.

Preserve For all wffs ϕ in $\mathcal{L}^F$, if $\vdash_F \phi$ then ϕ is provable.

It's worth being explicit about what this means. A proof in this system is any finite sequence of $\mathcal{L}(\mathcal{L}^F)$ wffs such that for any wff ϕ in the sequence *either* ϕ is an instance in $\mathcal{L}(\mathcal{L}^F)$ of our tree axioms or follows from earlier items in the sequence by making use of the rules of inference in our proof system for finite trees (this is the content of the **Tree 1** and **Tree 2** clauses); *or* ϕ is an $\mathcal{L}^F$ wff such that $\vdash_F \phi$ (this is the content of the **Preserve** rule of inference). If ϕ is the last item in some such sequence then we say that ϕ is provable. Thus the axiomatisation reflects the layered nature of $\mathcal{L}(\mathcal{L}^F)$ very clearly. The **Preserve** rule tells us that in addition we can 'lift' formulas provable in the feature logic to the layered system.

This axiomatisation is complete for the intended semantics of $\mathcal{L}(\mathcal{L}^F)$, that is, finite trees trees fibred over finite point-generated feature structures. We won't prove this here; it is a straightforward application of Finger and Gabbay's methods requiring no new ideas. Instead we will prove a simple corollary of completeness: our layered logic is decidable. First a lemma:

Lemma 5.2 *The set of validities of* $\mathcal{L}(\mathcal{L}^F)$ *is recursively enumerable.*

Proof: Clearly we can write a procedure that systematically generates the finite sequences of $\mathcal{L}(\mathcal{L}^F)$ wffs; but can we 'weed out' those sequences that are *not* formal $\mathcal{L}(\mathcal{L}^F)$ proofs? It is clear that we can write a procedure for determining whether or not a wff in a putative proof sequence is licenced by the **Tree 1** or **Tree 2** clauses, thus it only remains to check that we can write a

procedure for determining whether a wff in a putative proof sequence is licenced by the **Preserve** rule. But, by the decidability result for $\mathcal{L}^F$, this must be the case. In short, we can recursively enumerate $\mathcal{L}(\mathcal{L}^F)$ proof sequences. As this axiomatisation is complete, validity and formal provability coincide, and we have the result. □

By systematically generating finite trees fibred over finite point-generated feature structures we can recursively enumerate *non*-valid $\mathcal{L}(\mathcal{L})^F$ wffs. This fact together with the previous lemma yields:

Corollary 5.3 *The set of valid $\mathcal{L}(\mathcal{L}^F)$ wffs is recursive.* □

To close this section we briefly discuss how to generalise these results to cover the case of models constructed over finite trees when there is a fixed upper bound on the branch factor greater than 2. Recall that in section 1 we introduced suitable languages for such models: for any fixed $n > 2$ enrich $\mathcal{L}$ by adding operators $\downarrow^3, \downarrow^4, \ldots, \downarrow^n$ that look for information at 3rd, 4th, ..., n-th daughters respectively.

A complete axiomatisation of the validities in any such language is obtained from our axiomatisation for binary branching models as follows. First, we need a new defining axiom for the $\downarrow$ modality:

D3n $\downarrow\phi \leftrightarrow \downarrow^1 \vee \cdots \vee \downarrow^n \phi$.

Second, we generalise axiom B10 by adding, all instances of the following schema, where $2 \leq i \leq n$:

B10n $\downarrow^i \top \rightarrow \downarrow^{i-1} \top$.

It should be clear that the required completeness proof involves only trivial modification to that given for $\mathcal{L}$. With this completeness result established, the decidability result for the pure tree language follows by the argument used in Theorem 5.1, and the Finger and Gabbay methods yield completeness and decidability results for the layered languages in the manner just discussed.

6 Concluding remarks

In this paper we established completeness and decidability results for a simple modal language $\mathcal{L}$ interpreted over finite trees. As we saw, $\mathcal{L}$ is strong enough to be linguistically interesting, but it is natural to enquire whether similar results can be proved for stronger, or different, systems. To close this paper we mention some extensions and variations it seems worthwhile investigating, and discuss the long term goals of this research.

First, in the literature on temporal logics of programs, it is common to dispense with $\downarrow^*$ in favour of stronger operators, namely various generalisations

of Kamp's *Until* operator; see Goldblatt (1992) for definitions and discussion. We think it would be interesting to strengthen $\mathcal{L}$ in this fashion, and in addition to replace $\uparrow^*$ the stronger *Since* operator.

A second direction worth exploring is what happens when the *unbounded semantics* is employed, that is, when no finite upper bound is placed on the branch factor of the allowed models. This is important if one wants to give an 'unbounded' GPSG style analysis of co-ordination.

An obvious language for dealing with the unbounded semantics is the following. For every natural number n add a unary operator $\downarrow^n$ for looking for information at the n-th daughter. Actually, this extension isn't quite good enough. Because there is no upper bound on the branch factor, $\downarrow$ is no longer a definable operator. If we want to be able to assert that information holds at some unspecified daughter (and in most linguistic applications we probably do) we need to add $\downarrow$ as a primitive unary modality. Let us call the language with the operators $\uparrow$, $\downarrow$, $\uparrow^*$, $\downarrow^*$ and $\downarrow^n$ (for all natural numbers n) $\mathcal{L}^\infty$. Note that $\mathcal{L}^\infty$ has a 'horizontal' failure of compactness in addition to the 'vertical' failure already noted for $\mathcal{L}$. For while every finite subset of

$$\{\downarrow p, \neg\downarrow^1 p, \neg\downarrow^2 p, \neg\downarrow^3 p, \neg\downarrow^4 p, \ldots\}$$

is satisfiable, the whole set cannot be satisfied at any single node in any model.

But to conclude the paper, let's change tack slightly. While we feel it is important to investigate extensions and variations of the type just mentioned, in certain respects they give a misleading picture of the long term goals of this work. Ultimately we hope to give a logical analysis of the leading grammar formalisms, but many aspects of this investigation are not covered by the extensions just mentioned. To close the paper we will indicate why this is so and which questions we think are worth pursuing.

As was already mentioned, the languages discussed in this paper make use of fairly orthodox aspects of tree structure. The operators they employ quantify across relations on trees (such as mother, daughter and their transitive reflexive closures) that are familiar from applications in mathematics and computer science. In linguistics, when trees are used together with a notion of feature structure, these may well be the only relations that need to be considered: this seems to be the case as far as GPSG is concerned. However it is probably not the case in general. In GB, for example, the feature component is minimal. Instead, various more complex relations on trees are employed. The command relations discussed earlier are one example of the relations of interest, but these are merely the tip of the iceberg. In order to do justice to GB ideas on X-bar syntax and bounding, for example, it seems that one must explore a variety of other, less familiar, relations on trees and their interactions.

We feel that a systematic investigation of these richer structures could be rewarding. For a start, there are the more-or-less obvious logical questions: what languages are appropriate for describing these richer structures and what are

their properties? But, more importantly, a precise account of (say) the GB ontology would make it possible to address more substantial issues. What, precisely, are the trade-offs between imposing more structure on trees and adopting various notions of feature structure? Can the 'dynamic' notion of movement in GB be reduced to a 'static' one? Is there a canonical way of turning a GB account into (say) an LFG account and vice-versa? Different linguistic theories usually start from different pre-theoretic notions, and model these notions using very different mathematical ontologies. Nonetheless, mainstream syntactic theories are usually concerned with more or less the same data, and in spite of the (often vitriolic) inter-theoretic disputes, common themes are discernible. We believe that model theoretic investigations may provide a perspective from which the commonalities (and differences) emerge *clearly*.

It hardly needs stressing that these are difficult issues, and much of the groundwork remains to be done.[6] Nonetheless, we hope we have given the reader a taste of why we feel optimistic about the model theoretic approach to linguistic formalisms.

Acknowledgements We are very grateful to Claire Gardent, Jan van Eijck and Yde Venema for their comments on earlier drafts, and to Maarten de Rijke for editorial help over and above the call of duty. We would also like to thank the *IGPL Bulletin* referees for their comments on the penultimate draft. Patrick Blackburn would like to acknowledge the financial support of the Netherlands Organisation for the Advancement of Research (project NF 102/62-356, 'Structural and Semantic Parallels in Natural Languages and Programming Languages').

References

[1] Van Benthem J., and Meyer-Viol, W.: to appear, *Logical Semantics of Programming*.

[2] Blackburn, P., Gardent, C., and Meyer-Viol, W.: 1993, Talking about Trees. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics*, Utrecht, pp. 21–29.

[3] Blackburn, P. and Spaan, E.: 1993, A Modal Perspective on the Computational Complexity of Attribute Value Grammar. *Journal of Logic, Language and Information*, **2**, pp. 129–169

[6] Interesting work which addresses such general issues has appeared recently. Kracht (1992, 1993a, 1993b) gives a precise formulation of many of important ideas in GB, and relates them to the ideas underlying GPSG.

[4] Carpenter, Bob.: 1992, *The Logic of Typed Feature Structures*, Cambridge Tracts in Theoretical Computer Science, **32**, Cambridge University Press, Cambridge.

[5] Barker, C. and Pullum, G.: 1990, A Theory of Command Relations. *Linguistics and Philosophy*, **13**, pp. 1–34.

[6] Chomsky, N.: 1981, *Lectures on Government and Binding*, Foris, Dordrecht.

[7] Evans, R.: 1987, Towards a formal specification for defaults in GPSG. In Klein E. and van Benthem J. (ed.) *Categories, Polymorphism and Unification*, Edinburgh: Centre for Cognitive Science and Amsterdam: ITLI.

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

[9] Fisher, M. and Ladner. R.: 1979, Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, **18**, 194–211.

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

[11] Goldblatt, R.: 1992, *Logics of Time and Computation*, 2nd edition. CSLI Lecture notes, **7**, Center for the Study of Language and Information, Stanford.

[12] Kaplan, R. and Bresnan, J.: 1982, Lexical-Functional Grammar: A Formal System for Grammatical Representation. In Bresnan J. (ed.) *The Mental Representation of Grammatical relations*, pp. 173–281, MIT Press, Cambridge, Massachusetts.

[13] Kracht, M.: 1992, The Theory of Syntactic Domains. Logic Group Preprint Series **75**, Department of Philosophy, Rijksuniversiteit Utrecht, The Netherlands.

[14] Kracht, M.: 1993a, Nearness and Syntactic Influence Spheres, manuscript, II. Mathematisches Institut, Freie Universität Berlin.

[15] Kracht, M.: 1993b, Mathematical Aspects of Command Relations. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics*, Utrecht, pp. 240–249.

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

[17] Pollard, C. and Sag, I.: 1987, *Information-Based Syntax and Semantics: Volume 1, Fundamentals*, CSLI Lecture notes, **13**, Center for the Study of Language and Information, Stanford.

[18] Reinhart, T.: 1981, Definite np-anaphora and c-command domains. *Linguistic Inquiry*, **12**, pp. 605–635.

[19] Rogers, J. and Vijay-Shankar, K.: 1992, Reasoning with Descriptions of Trees. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pp. 72–80.

[20] Vijay-Shankar, K.: 1992, Using Descriptions of Trees in a Tree Adjoining Grammar. *Computational Linguistics*, **18**, pp. 481–517.