

---

## Inessential Features and Expressive Power of Descriptive Metalanguages

GEOFFREY K. PULLUM AND HANS-JÖRG TIEDE

### Abstract

Linguists employ a variety of features, ranging from traditional such morphosyntactic features like those encoding person or number to more recent inventions encoding bar level and gap locations. Linguists feel intuitively that there is a distinction between (i) real features reflecting genuine properties of languages and (ii) formal tricks exploiting the feature machinery. Our thesis in this chapter is that this issue is trickier and more subtle than might be thought. Notions like ‘spurious feature distinction’ or ‘artifact of the descriptive machinery’ are not really well-defined. There is a very close relationship between expressiveness of the formal metalanguage and necessity of particular features: in a fairly precise sense captured by a theorem, the more expressive the descriptive metalanguage employed, the smaller the number of features that need to be posited.

**Keywords** MODEL THEORETIC SYNTAX, FEATURES, METHODOLOGY OF LINGUISTICS

### 8.1 Introduction

It is natural enough for linguists to think that the features they posit in descriptions of natural languages are genuine, not spurious — that they reflect aspects of the subject matter rather than aspects of the machinery invoked in devising the description or the linguistic theory.

Having seen features like CASE, GENDER, NUMBER, PERSON, and TENSE used repeatedly in describing hundreds of languages, we feel that they have some inherent connection with the way human languages work, rather than with the way human linguists work. And anyone who has attempted to describe English syntax would probably feel that a feature AUX (distinguishing the verbs that take *-n't* from those that don't) and WH (distinguishing the rel-

ative and interrogative pronouns from the others) also draw real rather than artifactual distinctions.

But linguists don't always feel this way about all of the rich array of features posited in current or past work on syntax, and certainly not about devices such as the DOOM feature used by Postal (1970) to mark noun phrases targeted for erasure later in the derivation, or the '[±F]' annotations that have often been used to draw ad hoc distinctions among constituents with differing behaviours.

What is the basis of the feeling that we can tell a spurious feature from a genuine one? Generalised phrase structure grammar (GPSG) and head-driven phrase structure grammar (HPSG), for example, posit features such as SLASH, marking constituents containing 'gaps'; BAR, indicating the 'bar level' of phrasal constituents; SUBCAT, coding the subcategorisation of lexical heads according to the complements they select; and so on. These do not necessarily strike linguists as having the same kind of status as more traditional features. How can we tell when we are looking at spurious, artifactual, or superfluous features, features that should not be counted among the genuine ones reflecting attributes natural languages really have.

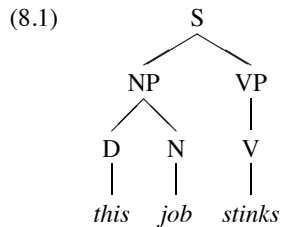
Our claim here is that this issue is trickier and more subtle than might have been thought. We argue that notions like 'spurious feature distinction' or 'artifact of the descriptive machinery' are not really well-defined. This means that linguists' feelings of distaste or acceptance for particular features may be based in nothing more solid than personal prejudice.

## 8.2 Model-theoretic syntax

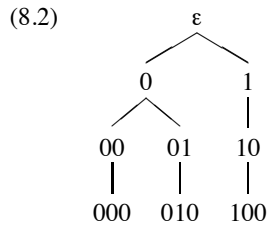
We make our argument via techniques from model-theoretic syntax. We will use trees formalised as relational structures. We briefly review the basics in the following section.

### 8.2.1 Logics on trees

To illustrate how we formalise trees as relational structures, we consider the phrase structure tree in (8.1).



The domain has 9 nodes. It is convenient to take them to be not just atomic elements but rather addresses of nodes — descriptions of positions relative to the root — represented by strings. The root will have the empty string ( $\epsilon$ ) as its address; ‘0’ will be the address of the left child of the root and ‘1’ the address of the right child, and we go on down: ‘01’ for the right child of the left child of the root, ‘010’ for the left child of that, and so on.



Each node in a binary branching tree will be a string over  $\{0, 1\}$ ; and since no string ever appears at more than one node — each string over  $\{0, 1\}$  corresponds to a unique address — we can equate trees with sets of such strings.

Of course, not every set of strings over  $\{0, 1\}$  represents a tree; but the two conditions we need to impose to guarantee that such a set does correspond to a tree are remarkably simple. Simplifying by assuming (with much recent literature) that all branching is binary (see Rogers 1998:19 for a more general statement, which also is very simple), the conditions defining binary tree domains are these:

- (8.3) A *binary tree domain* is a set  $T \subseteq \{0, 1\}^*$ , such that
- (a) if  $uv$  is in  $T$ , then so is  $u$ , and
  - (b) if  $u1$  is in  $T$ , then so is  $u0$ .

Part (a) requires any node to have all of its ancestors in the domain, and part (b) requires all right branches to have left siblings. This defines binary tree domains in terms quite independent of both the diagrams with which we depict trees and the generative grammars that linguists often assume for generating them, though thus far the result looks a bit unfamiliar.

We now define a **binary tree structure** as a triple  $(T, R_{\downarrow}, R_{\leftarrow})$ , where  $T$  is a binary tree domain,  $R_{\downarrow}$  is the child-of relation (we use androgynous kinship terminology for relations between nodes: ‘parent’, ‘child’, and ‘sibling’); i.e.,  $(n, m) \in R_{\downarrow}$  iff  $m = n0$  or  $m = n1$ , and  $R_{\leftarrow}$  is the left-sibling-of relation, i.e.  $(m, n) \in R_{\leftarrow}$  iff  $m = s0$  and  $n = s1$  for some  $s$ .

So the tree in (8.2) would be formalised as a tree structure triple,  $(T, R_{\downarrow}, R_{\leftarrow})$ , where  $T$  is the set  $\{\epsilon, 0, 1, 00, 01, 10, 000, 010, 100\}$  and, for example, the pair  $(10, 100)$  stands in the  $R_{\downarrow}$  relation and the pair  $(00, 01)$  stands in the  $R_{\leftarrow}$  relation.

### 8.2.2 Modal logic metalanguages

To talk about trees we will use **propositional modal logics**, an idea that goes back to Gazdar et al. (1988); see Blackburn et al. (1993) for a much more thoroughgoing development of the idea, and Moss and Tiede (2006) for a detailed and up-to-date survey. Modal logics provide a very tightly restricted way of describing the structure of trees, and their relationships with and translations into other formalisms are beginning to be very well understood. Though they originate in philosophical efforts to understand the notions of necessity and possibility, they are best seen much more generally, as languages for describing relational structures (such as trees) from a local and internal perspective. Blackburn et al. (2001:xii) observe that “the reader who pictures a modal formula as a little automaton standing at some state in a relational structure, and only permitted to explore the structure by making journeys to neighbouring states, will have grasped one of the key intuitions of modal model theory.” The linguist can read ‘state’ as ‘node’ and ‘relational structure’ as ‘tree’.

The key idea is to represent the labels attached to nodes in trees by atomic propositional formulae of the logic. In other interpretations of modal logic these would be propositions true at particular worlds in a set of worlds under an accessibility relation; here they simply represent labelings present at certain points in the model.

Before we formalise labelled trees in these terms, we need to fix the syntax of the logics that we will be considering. We can use a very simple and basic kind of modal logic  $\mathcal{L}_B$  as a reference point.  $\mathcal{L}_B$  has a set of atomic formulae corresponding to syntactic categories (we can assume this is finite), and just two modalities,  $\langle \rightarrow \rangle$  and  $\langle \downarrow \rangle$ . The semantics is such that a formula  $\langle \downarrow \rangle \varphi$  is true at a node  $v$  iff  $\varphi$  is true at a child of  $v$  (so it means ‘ $v$  has a  $\varphi$ -labelled child’) and  $\langle \rightarrow \rangle \varphi$  is true at a node  $v$  iff  $\varphi$  is true at a right sibling of  $v$ .

$\mathcal{L}_B$  can be used to say some of the things about trees that could be guaranteed by a context-free grammar. Consider the two formulae in (8.4):

- (8.4) a.  $\mathbf{PP} \Rightarrow \langle \downarrow \rangle \mathbf{P}$   
 b.  $\langle \downarrow \rangle \mathbf{during} \Rightarrow (\mathbf{P} \wedge \langle \rightarrow \rangle \mathbf{NP})$

We use ‘ $\Rightarrow$ ’ for material implication, defined  $\varphi \Rightarrow \psi \equiv \neg(\varphi \wedge \neg(\psi))$ . So what (8.4a) says is that a node where  $\mathbf{PP}$  is true has a child where  $\mathbf{P}$  is true; that is, it expresses the claim of X-bar theory that every PP node must immediately dominate a Preposition node. And what (8.4b) says is that a node that has a child labelled **during** is labelled P and has a right sibling labelled NP — a subcategorisation statement for the preposition *during*.

However, in general  $\mathcal{L}_B$  is too weak to permit much interesting progress toward the description of human languages, so we need to consider stronger logics.

Three modal logics of increasing expressive strength have been a particularly focus of attention in the context of model-theoretic syntax. They are known as  $\mathcal{L}_{core}$ ,  $\mathcal{L}_{cp}$  and  $PDL_{tree}$ , respectively. All of them are less expressive than  $\mathcal{WMSO}$ .

For our purposes here, it will be sufficient to concentrate on  $\mathcal{L}_{core}$  and  $\mathcal{L}_{cp}$ . (For a detailed discussion of  $PDL_{tree}$ , which is increasingly important for reasons relating to the rise of XML as a data representation language, see Afanasiev et al. 2005.) The syntax of formulae for the  $\mathcal{L}_{core}$  and  $\mathcal{L}_{cp}$  languages is defined as follows:

- (8.5) Basic syntax for  $\mathcal{L}_{core}$  and  $\mathcal{L}_{cp}$
- a. any atomic proposition is a formula;
  - b.  $\neg(\varphi)$  is a formula if  $\varphi$  is;
  - c.  $\varphi \wedge \psi$  is a formula if  $\varphi$  and  $\psi$  are;
  - d. a formula prefixed by a modal operator is a formula.

In addition we will use ‘ $\top$ ’ for a dummy proposition that is always true (it could be defined as  $\neg(\mathbf{a} \wedge \neg(\mathbf{a}))$  or in some similar way; its utility will become clear below).

The logics  $\mathcal{L}_{core}$  and  $\mathcal{L}_{cp}$  differ only with respect to the modal operators they employ. These operators are logically akin to the diamond operators that represent possibility in the alethic modal logics familiar from formal semantics. Each is written in the form  $\langle \pi \rangle$  (the angle brackets ‘ $\langle \rangle$ ’ are intended to convey a visual suggestion of the diamond ‘ $\diamond$ ’). The ‘box’ modalities are used as well, and as usual they are defined in terms of the diamond ones:  $[\pi]\varphi$  (with the square brackets visually suggesting the box  $\square$ ) abbreviates  $\neg\langle \pi \rangle\neg\varphi$ .

$\mathcal{L}_{core}$  is a logic with eight modal operators, the two in  $\mathcal{L}_B$  plus their inverses, and operators corresponding to the ancestrals of all four:

- (8.6)  $\mathcal{L}_{core}$  modal operators:  $\rightarrow$   $\leftarrow$   $\uparrow$   $\downarrow$   $\rightarrow^*$   $\leftarrow^*$   $\uparrow^*$   $\downarrow^*$

$\mathcal{L}_{core}$  permits not only statements like  $\langle \downarrow \rangle \varphi$ , meaning that at one dominance step down there is a node where  $\varphi$  holds, but also  $\langle \downarrow^* \rangle \varphi$ , which corresponds to the ancestral of the relation that  $\downarrow$  corresponds to: it means that there is some finite number  $k$  such that  $k$  dominance steps down there is a node where  $\varphi$  holds. Thus  $\langle \downarrow^* \rangle \varphi$  means that either  $\varphi$  holds ( $k = 0$ ), or  $\langle \downarrow \rangle \varphi$  holds ( $k = 1$ ), or  $\langle \downarrow \rangle \langle \downarrow \rangle \varphi$  holds ( $k = 2$ ), and so on for all  $k \geq 0$ .

The logic  $\mathcal{L}_{cp}$  has an infinite set of modalities. They are defined recursively. All modalities from  $\mathcal{L}_{core}$  are available in  $\mathcal{L}_{cp}$ , but it has additional modalities that cannot be defined using those modalities. The following four simple operators are included (as in  $\mathcal{L}_{core}$ ):

- (8.7)  $\mathcal{L}_{cp}$  basic modal operators:  $\rightarrow$   $\leftarrow$   $\uparrow$   $\downarrow$

But in addition, for any modal operator  $\pi$  and any formula  $\varphi$ , the following are both modal operators in  $\mathcal{L}_{cp}$ :

(8.8) Recursively defined modal operators of  $\mathcal{L}_{CP}$ :

- $\pi^*$  for each modal operator  $\pi$   
 $\pi; \varphi?$  for each modal operator  $\pi$  and formula  $\varphi$

As in the case of the  $\langle \downarrow^* \rangle \varphi$  modality of  $\mathcal{L}_{core}$ , the  $\pi^*$  operators afford access to the ancestral of the accessibility relation for  $\pi$ : the formula  $\pi^* \varphi$  is satisfied at a node  $u$  iff  $\varphi$  holds at a node  $v$  that you can get to from  $u$  via a sequence of zero or more steps mediated by the  $R_\pi$  relation.

The interpretation of  $\langle \pi; \varphi? \rangle$  needs a little more explanation. Intuitively, evaluating  $\langle \pi; \varphi? \rangle \psi$  involves checking that  $\psi$  holds at a node that we can get to using  $\pi$  and at which  $\varphi$  holds. The two examples in (8.9) will help.

- (8.9) a.  $\langle \downarrow; \varphi? \rangle \psi$   
 b.  $\langle (\downarrow; \varphi?)^* \rangle \psi$

Formula (8.9a) can be read as ‘at some child of this node where  $\varphi$  is true,  $\psi$  is true’. This is equivalent to  $\langle \downarrow \rangle (\varphi \wedge \psi)$ , and thus would also be expressible in  $\mathcal{L}_{core}$ . But the same is not true of (8.9b). For (8.9b) to be satisfied at a node  $u$ , there has to be a node  $v$ , dominated by  $u$ , at which  $\psi$  is true, and additionally  $\varphi$  has to be true at all nodes on the path from  $u$  to  $v$ . Notice that the asterisk is on ‘ $\langle \downarrow; \varphi? \rangle$ ’, so what is repeated is both the step down from parent to child and the check on whether  $\varphi$  holds. There has to be a parent-child chain in which at every node the check to see if  $\varphi$  holds is successful, and it has to lead down to a node where  $\psi$  holds. This relation is inexpressible in  $\mathcal{L}_{core}$ .

Notice the potential applications of a formula like (8.9b) in describing syntactic facts in human languages. It has been suggested for various syntactic phenomena that they are permitted only within the region between the top and bottom of an unbounded dependency (see Zaenen 1983). Such phenomena could be described with a statement using  $\langle (\downarrow; \varphi?)^* \rangle \psi$ , where  $\psi$  is the property of being a trace and  $\varphi$ , holding at all the nodes on the spine leading down to the node where  $\psi$  holds, is the property that determines the relevant phenomena.

### 8.2.3 Trees as models for modal logics

We can identify a labelled tree with a tree model, in the following sense. A **binary tree model** is a pair  $\mathcal{M} = \langle T, \text{Val} \rangle$  where  $T$  is a tree structure and  $\text{Val}$  is a **valuation** function — a function from formulae to node sets which assigns to each atomic formula the set of all and only those nodes in the tree at which it holds.

So, to complete our example, assume that we have atomic formulae **S**, **NP**, **VP**, **D**, ..., and thus the binary tree model corresponding to the example in (8.1) would contain a valuation  $\text{Val}$  such that  $\text{Val}(\mathbf{NP}) = \{0\}$ ,  $\text{Val}(\mathbf{V}) = \{10\}$ , etc.

The remaining thing we need is a definition of the satisfaction relation. We write ‘ $\mathcal{M}, v \models \varphi$ ’ for ‘the model  $\mathcal{M}$ , at the node  $v$ , **satisfies** (or, **is a model of**) the formula  $\varphi$ .’ We define the relation  $\models$  in the following standard way.

- (8.10) For a model  $\mathcal{M}$ , a node  $v$  of  $\mathcal{M}$ , and a formula  $\varphi$ :
- a.  $\mathcal{M}, v \models p \iff v \in \text{Val}(p)$   
( $v$  satisfies atomic formula  $p$  iff  $v$  is in the set  $\text{Val}$  assigns to  $p$ )
  - b.  $\mathcal{M}, v \models \varphi \wedge \psi \iff \mathcal{M}, v \models \varphi \wedge \mathcal{M}, v \models \psi$   
( $v$  satisfies a conjunction iff it satisfies both conjuncts)
  - c.  $\mathcal{M}, v \models \neg\varphi \iff \mathcal{M}, v \not\models \varphi$   
( $v$  satisfies the negation of any formula that it doesn’t satisfy)

As is familiar from alethic modal logic, evaluating a formula containing a modality always involves an accessibility relation that defines permitted access from one state or node to another in the model. Given that both  $\mathcal{L}_{core}$  and  $\mathcal{L}_{cp}$  have multiple modalities, each modality  $\langle \pi \rangle$  will have a corresponding accessibility relation  $R_\pi$ :

- (8.11)  $\mathcal{M}, v \models \langle \pi \rangle \varphi \iff \exists u[(v, u) \in R_\pi \wedge \mathcal{M}, u \models \varphi]$   
( $v$  satisfies  $\langle \pi \rangle \varphi$  iff it bears the  $\pi$  relation to a node  $u$  that satisfies  $\varphi$ )

Given our discussion of  $R_\perp$  and  $R_\rightarrow$  in section 8.2.1 above, it is fairly straightforward to get a sense of the accessibility relations for the modalities in  $\mathcal{L}_{core}$ . The accessibility relations for the modalities in  $\mathcal{L}_{cp}$  are more complex, and will be omitted here (but the details are in Moss and Tiede 2006).

#### 8.2.4 Definability

In order to relate the model-theoretic approach to the generative approach, we need a model-theoretic notion that corresponds to the set of derived structures in the former approach. We will restrict a finite set of atomic formulae denoted by  $F$ . We will denote the set of trees that are labelled only with the features from the set  $F$  by  $\mathcal{T}^F$ , and for the set of formulae in the logic  $\mathcal{L}$  that only use the atomic formulae from the set  $F$  we will write  $\mathcal{L}^F$ .

We say that a subset of  $\mathcal{T}^F$  is **definable** in  $\mathcal{L}$  if there is a formula in  $\mathcal{L}^F$  such that the subset in question is exactly the set of all those trees which at their root nodes satisfy the formula. What it means to say that some set  $\mathcal{T} \subseteq \mathcal{T}^F$  is definable in  $\mathcal{L}$  is simply that there is some  $\varphi \in \mathcal{L}$  such that  $\mathcal{T} = \{\tau \mid \tau, \varepsilon \models \varphi\}$  (that is,  $\mathcal{T}$  is the set of all and only those trees which at the root node satisfy  $\varphi$ ).

As an example of how grammars of certain specific types can be formalised in certain specific logics, it is worth noting — with an informally sketched proof — that despite its very limited expressive power,  $\mathcal{L}_{core}$  is capable of defining the set of all the parse trees obtainable from an arbitrary

context-free phrase structure grammar (CF-PSG).

(8.12) **Theorem** For any context-free phrase structure grammar  $G$  there is a formula  $\varphi_G$  in  $\mathcal{L}_{core}$  that defines the set of all parse trees of  $G$ .

**Proof** Let the terminals, nonterminals, rules, and start symbol of  $G$  be respectively  $V_T$ ,  $V_N$ ,  $P$ , and  $S$ . Since we are only considering binary branching trees (it is not hard to generalise the result to  $n$ -ary branching), every rule in  $P$  is of the form  $A \rightarrow BC$  or  $A \rightarrow a$ , with  $A, B, C \in V_N$  and  $a \in V_T$ . (Here and below we reserve ‘ $\rightarrow$ ’ for the rewriting operation of CF-PSG rules.) The effects of such rules can be encoded directly in  $\mathcal{L}_{core}$  as follows.

The set of formulae covering the binary branching rules contains for each symbol  $A$  appearing on the left hand side of a branching rule a statement ‘ $\mathbf{A} \Rightarrow \Psi$ ’, where  $\Psi$  is a disjunction that for each rule  $A \rightarrow BC$  contains a disjunct of this form:

$$(8.13) \quad \langle \downarrow \rangle (\mathbf{B} \wedge \langle \rightarrow \rangle \mathbf{C})$$

So if the only rules with PP on the left of the arrow were (i)  $VP \rightarrow V_1$ , (ii)  $VP \rightarrow V_2 NP$ , and (iii)  $VP \rightarrow V_3 \text{ Clause}$ , the corresponding logical statement would contain a statement that in effect says this: ‘If **VP** holds at a node then either (i) **V<sub>1</sub>** holds at a child node that has no right sibling, or (ii) **V<sub>2</sub>** holds at a child node that has a right sibling where **NP** holds, or (iii) **V<sub>3</sub>** holds at a child node that has a right sibling where **Clause** holds.’

To this we add, for each  $A$  that appears on the left hand side of a unary rule, a statement  $\mathbf{A} \Rightarrow \Psi$ , where  $\Psi$  is a disjunction with disjuncts of the form  $\langle \downarrow \rangle \mathbf{a}$ , one for each  $a$  such that  $A \rightarrow a$ .

This much ensures that the models of  $\varphi_G$  comply with the restrictions that the rules impose on parse trees of  $G$ . The rest of what we need to do is to ensure that *only* parse trees of  $G$  are models of  $\varphi_G$  (from what has been said so far, there could be other models of  $\varphi_G$  with all sorts of labellings about which the rules say nothing, and they could vacuously satisfy the statements summarised above). This we accomplish by adding four further statements.

First, we affirm that node labels are unique — at each node exactly one propositional symbol is true, by stating that at every node some proposition holds:

$$(8.14) \quad [\downarrow^*](\mathbf{A}_1 \vee \mathbf{A}_2 \vee \cdots \vee \mathbf{A}_n) \quad \text{where } V_T \cup V_N = \{A_1, A_2, \dots, A_n\}$$

and we state that for all pairs of distinct propositions the negation of one of them holds:

$$(8.15) \quad [\downarrow^*](\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_k)$$

where  $\varphi_1, \dots, \varphi_k$  is the list of all statements of the form ‘ $\neg(\alpha) \vee \neg(\beta)$ ’, for  $\alpha, \beta \in V_T \cup V_N$  and  $\alpha \neq \beta$

Second, we assert that the start symbol  $S$  is true at the root — that is,  $\mathbf{S}$  must hold at any node where not even the dummy tautology  $\top$  holds at the immediately dominating node:

$$(8.16) \quad [\uparrow]\neg(\top) \Rightarrow \mathbf{S}$$

Third, we stipulate that the terminal symbols are true only at leaves — that wherever a terminal symbol holds, not even the dummy tautology holds at any immediately dominated node thereof (which means there cannot be one):

$$(8.17) \quad [\downarrow^*]\Phi, \text{ where } \Phi \text{ is the conjunction} \\ (\mathbf{a}_1 \Rightarrow \neg(\downarrow)\top) \wedge (\mathbf{a}_2 \Rightarrow \neg(\downarrow)\top) \wedge \cdots \wedge (\mathbf{a}_k \Rightarrow \neg(\downarrow)\top) \\ \text{for all } a_i \in V_T.$$

Fourth, we assert that non-terminal symbols are true only at internal nodes — that wherever a nonterminal holds, the dummy tautology holds at the immediately dominated node (which means there must be one).

$$(8.18) \quad [\downarrow^*]\Phi, \text{ where } \Phi \text{ is the conjunction} \\ (\mathbf{A}_1 \Rightarrow \langle \downarrow \rangle \top) \wedge (\mathbf{A}_2 \Rightarrow \langle \downarrow \rangle \top) \wedge \cdots \wedge (\mathbf{A}_k \Rightarrow \langle \downarrow \rangle \top) \\ \text{for all } A_i \in V_N.$$

This guarantees that any model of the complex formula we have constructed will be a parse tree of  $G$ , which completes the proof.  $\blacksquare$

### 8.3 Features

The trees considered so far are labelled with atomic category labels, represented in the description logic as atomic formulae with the property that each node satisfies exactly one of them. If we want to label trees with features, we have to extend the approach presented above. One easy way to include features is to allow each node to satisfy *multiple* atomic formulae. That way, each atomic formula corresponds to a binary valued feature: if some feature  $\mathbf{F}$  holds at a node, that node is [+F], and if it is false, the node is [-F].

We can use the approach to represent non-binary features too, as long as it is combined with some formulae limiting their co-occurrence. Thus we could represent the BAR feature in a two-bar system by means of three features, call them BAR0, BAR1, and BAR2. It is easy to assert what the GPSG literature calls a feature co-occurrence restriction — a statement of the type exemplified in (8.15), saying that one and only one of these features is true at each node.

To give an example of the use of  $\mathcal{L}_{cp}$ , consider the following formalisation of projection from heads, based on Palm (1999). We first introduce an abbreviation meaning ‘the feature  $f$  belongs to a node that is a head’, where (for this purpose) we treat being a head as simply a matter of bearing an atomic feature, corresponding to the atomic proposition **head**, with the statement  $H\varphi \equiv \varphi \wedge \mathbf{head}$ .

Then we define what it is for a feature  $\varphi$  to be projected from a leaf:

$$(8.19) \quad \text{Proj}\varphi \equiv \langle (\downarrow; (H\varphi))^* \rangle (H\varphi \wedge \text{lexical})$$

Here **lexical** is just an abbreviation for  $\langle \downarrow \rangle \neg (\langle \downarrow \rangle \top)$ .

Finally, we can require every node to be a projection: given a finite set of lexical features  $\text{Lex}$ , we assert  $[\downarrow^*]\Phi$ , where  $\Phi$  is the disjunction of all the statements  $\text{Proj}\varphi$  such that  $\varphi$  is in  $\text{Lex}$ .

The feature indicating that a node is the head would be needed in cases where two siblings shared the same lexical feature. Furthermore, there are certain regularities that this head feature has to observe, such as that (if we set aside the multiple-heads treatment of coordination argued for in some GPSG work) no two siblings may both be heads, a condition that we could state thus:

$$(8.20) \quad [\downarrow^*](\text{head} \Rightarrow \neg(\langle \leftarrow \rangle \text{head} \vee \langle \rightarrow \rangle \text{head}))$$

### 8.3.1 Eliminable Features

The clearest sense in which a feature can be considered intuitively superfluous is when one can eliminate it from the grammar without any loss to the description. Given a tree  $\tau \in \mathcal{T}^F$  and a subset of features  $G \subseteq F$ , there is a corresponding tree  $\tau' \in \mathcal{T}^G$  that is the result of removing the features in  $F - G$  from  $\tau$ . We will denote the corresponding function by  $\hat{\pi}$ ; thus  $\hat{\pi}(\tau) = \tau'$ , and define  $\hat{\pi}(\mathcal{T})$  as  $\{\hat{\pi}(\tau) \mid \tau \in \mathcal{T}\}$ .

The notion of a feature being superfluous in the sense that it can be eliminated without loss from the distinction can now be formalised by means of the following definition:

$$(8.21) \quad \text{Let } F \text{ be a finite set of features, } G \subseteq F, \mathcal{T} \subseteq \mathcal{T}^F, \text{ and } \mathcal{L} \text{ a logic. Suppose that } \mathcal{T} \text{ is definable in } \mathcal{L}^F. \text{ We say that } G \text{ is } \textit{eliminable in } \mathcal{L} \textit{ for } \mathcal{T} \textit{ iff } \hat{\pi}(\mathcal{T}) \textit{ is definable in } \mathcal{L}^{F-G}.$$

Notice that this notion of eliminability is relative to a given logic: the features in  $G$  are eliminable in some language  $\mathcal{L}$  with respect to some set of trees  $\mathcal{T}$  if and only if the function that gets rid of the  $G$  features is definable in  $\mathcal{L}$  without using any  $G$  features. This might hold for some particular  $\mathcal{L}$  but not in another metalanguage. In other words, questions of whether some feature is truly needed cannot be addressed in isolation, but only in the context of a particular descriptive metalanguage in which the feature is used. This observation is made more precise in the following theorem:

$$(8.22) \quad \textbf{Theorem (Tiede, 2008)} \quad \text{Any tree language that is not definable in } \mathcal{L}_{\text{core}} \text{ but is definable in } \mathcal{L}_{\text{cp}} \text{ can be defined with additional features in } \mathcal{L}_{\text{core}} \text{ that are not eliminable in } \mathcal{L}_{\text{core}}.$$

This theorem could actually be strengthened, as its proof (for which see Tiede 2008) does not depend on any of the logics in particular. It applies to

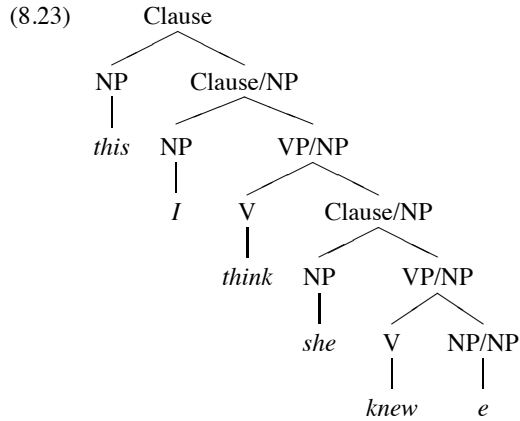
any case of two different formal ways of defining sets of trees, each capable of defining all local sets (those that a CF-PSG can define), and one defining a proper subset of the tree-sets definable by the other, provided they are not more powerful than wMSO. For any set  $\mathcal{T}$  of trees definable in the more powerful formalism but not in the less powerful one,  $\mathcal{T}$  will be definable in the less powerful formalism if we are permitted to decorate its nodes with additional features.

These results can be read in two different ways. First, they state that any language that cannot be defined in a weaker formalism but can in a stronger one can be defined in the weaker one if additional features are added. Conversely, they state that the only difference between the different formalisms mentioned above, as well as a variety of other formalisms, is which features are required to define languages: the more expressive the formalism, the fewer features are required for defining languages.

When we move to the most powerful of the logics commonly used in model theoretic syntax, wMSO, a single feature suffices. This follows from the fact that wMSO characterises the tree-sets that are recognisable by finite-state tree automata (Doner, 1970). These tree-sets are known as the **regular tree-sets** (or ‘regular tree languages’). The set of all regular tree-sets is known to be closed under linear tree homomorphisms, which means that any systematic symbol-for-symbol relabeling of all the nodes in all the trees of a regular tree-set will always yield a regular tree-set.

An example will make this point clearer. Consider a finite-state tree automaton recognising some set of trees in which the nodes are labelled with distinct labels  $A$  and  $B$ , distinguished by some crucial syntactic feature, and suppose we wanted to relabel the  $B$  nodes as  $A$  nodes without losing track of which ones they were. This can be accomplished by modifying the automaton so that it has two different states for admitting  $A$  nodes: one corresponding to the original  $A$  nodes, and one corresponding to the notion ‘ $A$ -labelled node that is really one of the relabelled  $B$  nodes’. Since any finite-state tree automaton is equivalent to a wMSO logical description (Doner, 1970), there is a wMSO theory that corresponds to the new automaton.

So consider in this light the question of whether the SLASH feature of GPSG and HPSG is a genuine substantive element of the grammars of human languages. A node dominated by a category  $\alpha$  is marked with a feature specification [SLASH: $\beta$ ] in GPSG in order to identify it as containing an extraction site of category  $\beta$  that is present somewhere within  $\alpha$ . This eliminates any need for movement transformations in the description of unbounded dependencies (Gazdar, 1981), as seen in (8.23), where we simplify visually by notating  $\alpha$ [SLASH :  $\beta$ ] in the form  $\alpha/\beta$ .



It might be charged that this simply substitutes another formal device for the formal device of movement: instead of the NP *this* being moved from an NP position to another NP position as sibling of a Clause node, it is a sister of a Clause/NP node that dominates a chain of  $\alpha$ /NP nodes that leads to an NP/NP node at what would have been the pre-movement location. The chain of slash categories marks the path from the root of the landing-site constituent down to the extraction site. So is the feature SLASH artifactual, rather than corresponding to a genuine syntactic property of constituents?

Our thesis is that the answer is neither yes nor no. The question is a pseudo-question, insufficiently well-defined to receive an answer.

### 8.3.2 Inessential Features

Given that the question whether a feature is eliminable depends on the formalism employed, it is only natural to try to give a purely structural definition of uselessness applying to features. Marcus Kracht (1997) has proposed such a definition. Kracht called a feature *inessential* “if its distribution is fixed by the other features,” and he proposed the following formal definition.

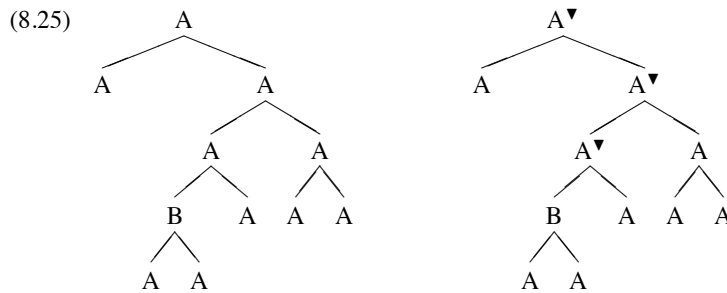
(8.24) Let  $F$  be a finite set of features; let  $G$  be a subset of  $F$ ; and let  $\mathcal{T}$  be a set of trees labelled with the features in  $F$ . The features in  $G$  are **inessential for  $\mathcal{T}$**  if the function that eliminates the features in  $G$  is one-to-one.

The reason for identifying superfluous features with those that can be eliminated by a one-to-one (injective) function is that no two trees can be distinguished only with these features. If they could, the function that eliminates them would map them to the same tree, hence it would not be one-to-one.

The features referred to in the theorem in (8.22) are inessential in exactly Kracht’s sense. And this might seem to conform to the intuition that when a

feature is added solely to make a non-definable set of structures definable, it has an ad-hoc nature. When there is a distinction in tree structure that we are unable to capture using a given logic, and we add a special feature to certain nodes in certain trees just to enable the distinction to be captured using that logic, erasing the features we added would just give us back our original trees. They would not be identical with any other trees in the set, because if they were we would not have needed to add the feature annotations in the first place.

An example due to Thatcher and used by Rogers (1998:60) will be useful in making the point. Consider the set of all finite binary trees in which all nodes are labelled *A* except that in each tree exactly one node is labelled *B*. This set of trees is not definable in  $\mathcal{L}_{core}$ , or by any CF-PSG. But we can make it describable if we add a feature. We will assume that its presence or absence can be explicitly referenced at any node, and we will indicate its presence by ‘▼’. We annotate a tree like (8.25a) as shown in (8.25b).



The ‘▼’ feature is attached to every *A* node that dominates the unique *B*, and to no other node. This allows us to describe the set with a CF-PSG, using  $A^\blacktriangledown$  as the start symbol:

$$(8.26) \quad \begin{array}{llll} A^\blacktriangledown \longrightarrow A A^\blacktriangledown & A^\blacktriangledown \longrightarrow AB & A \longrightarrow AA \\ A^\blacktriangledown \longrightarrow A^\blacktriangledown A & A^\blacktriangledown \longrightarrow BA & B \longrightarrow AA \end{array}$$

By the theorem in (8.12) we know that the set of trees this grammar generates is describable in  $\mathcal{L}_{core}$ . However, if we erase the ▼ feature from every tree, we will get exactly the set mentioned above: in every tree there will be one *B* and all other nodes will be labelled *A*. Yet no two distinct trees will ever be collapsed into one under this ▼-erasure operation. Therefore the ▼ feature is inessential in Kracht’s technical sense.

Both the SLASH feature of GSPG and the BAR feature familiar from X-bar syntax are inessential in exactly the same way. The feature SLASH works in a way almost exactly analogous to ‘▼’ above: a constituent containing a gap is marked by placing a specification of a value for SLASH on the root of the

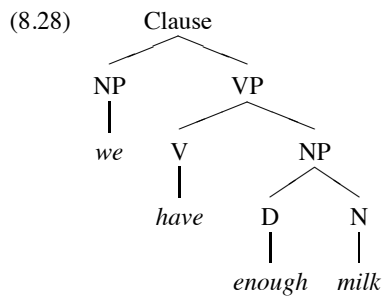
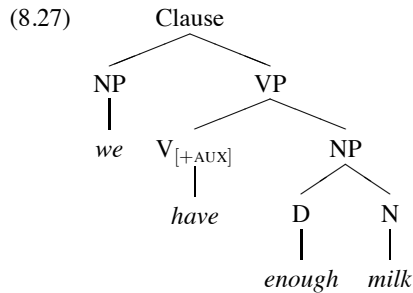
constituent and on each node in the continuous sequence of nodes from the root downwards, that dominate the gap.

The BAR feature is also (as Kracht notes) inessential. An easy way to see this intuitively is to imagine being presented with any of the trees in, say, Jackendoff (1977) with all of the bar-level indications (Jackendoff uses primes) removed. It would be easy to put them all back without error, given nothing more than the content of the X-bar principles that Kornai and Pullum (1990) call Lexicality, Uniformity, Succession, and Weak Maximality, plus the observation that Jackendoff simplifies his diagrams in certain respects (specifically, a branch with  $X'''$ ,  $X''$ ,  $X'$ ,  $X$ , and some terminal symbol  $\sigma$  will generally be shown with just  $X'''$  and  $\sigma$ ). Stripping all the primes from Jackendoff's trees will never collapse one legal tree into the prime-stripped version of another distinct tree. (Many versions of X-bar theory are much less principled than Jackendoff's, of course, and it may be that in some of those the feature BAR might turn out to be essential.)

What would be most desirable would be for the notion 'inessential' to be diagnostic for features that are spurious in the sense of being linguist's artifacts. Unfortunately things do not work out this way. Many features that linguists probably wouldn't want to eliminate are inessential under this definition, and many features of which they would be suspicious are not inessential.

Consider, for example, the CASE feature for pronouns in English, taking Nom, Acc, DepGen, and IndGen as its values. No two trees in most varieties of Standard English can be distinguished on the basis of this feature, as its distribution is fixed by other aspects of the trees. On a pronoun functioning as Subject in a Clause having a tensed Verb, the Nom value (e.g., *they*) is mandated; on a pronoun functioning as Determiner in an NP the DepGen value (e.g., *their*) is required. The ordinary morphosyntactic feature CASE is plainly inessential.

At first it might seem that any feature appearing on lexical category labels would be inessential in Kracht's sense, but this is not quite so. Counter-intuitively, features are essential when they are optional on a certain node. Consider the feature AUX in English. Any tree showing an 'inverted' auxiliary (clause-initial before a subject) will be one that predictably has AUX on its initial verb, and likewise any tree with a verb form ending in the suffix *-n't*, which only appears on auxiliary verbs. But take a dialect in which both *We haven't enough milk* and *We don't have enough milk* are grammatical. In such dialects, possession *have* may be treated either as an auxiliary verb or a lexical verb, optionally. So both of the following trees would be well-formed (we intend the V in the second one to be taken as [-AUX]):



These trees will be collapsed if the  $[\pm\text{AUX}]$  markings are erased, and solely because of this, AUX counts as an essential feature! Yet of course, its presence is intuitively quite unimportant: because the verb is not in one of the auxiliary-only *n't* forms, and is not before the subject, it simply doesn't matter whether it bears the marking  $[\text{+AUX}]$  or not. Though essential in the technical sense, it is entirely superfluous in the intuitive descriptive sense.

In short, Kracht's notion of inessentiality does not correspond at all to the descriptive linguist's notion of being an essential or significant component of a description.

#### 8.4 Conclusions

Our summary and conclusions can be given briefly. The argument of this chapter has been that it is unlikely that any formal reconstruction can be given of the notion of a feature that is a technical artifact rather than a genuine element of natural language structure that we should expect to turn up in some guise in any reasonable description. There is a crucial tradeoff between eliminability of features and expressive power of the descriptive metalanguage; the two issues cannot be separated.

Thus, just like the question of when a feature used in the description of one language should be equated with a feature used in the description of another, the issue of when a feature is a technical trick and when it is a properly mo-

tivated distinguishing property of linguistic expressions will not, we suspect, be reducible to any formal criterion. It may perhaps be approached informally through an understanding of what natural languages are typically like, but it will not submit to an authoritative mathematical adjudication.

## References

- Afanasiev, Loredana, Patrick Blackburn, Ioanna Dimitriou, Bertrand Gaiffe, Evan Goris, Maarten Marx, and Maarten de Rijke. 2005. PDL for ordered trees. *Journal of Applied Non-Classical Logic* 15(2):115–135.
- Blackburn, Patrick, Maarten de Rijke, and Yde Venema. 2001. *Modal Logic*. Cambridge: Cambridge University Press.
- Blackburn, Patrick, Claire Gardent, and Wilfried Meyer-Viol. 1993. Talking about trees. In *Sixth Conference of the European Chapter of the Association for Computational Linguistics: Proceedings of the Conference*, pages 21–29. Morristown, NJ: European Association for Computational Linguistics.
- Doner, John. 1970. Tree acceptors and some of their applications. *Journal of Computer and System Sciences* 4:406–451.
- Gazdar, Gerald. 1981. Unbounded dependencies and coordinate structure. *Linguistic Inquiry* 12:155–184.
- Gazdar, Gerald, Geoffrey K. Pullum, Bob Carpenter, Ewan Klein, Thomas E. Hukari, and Robert D. Levine. 1988. Category structures. *Computational Linguistics* 14:1–19.
- Jackendoff, Ray S. 1977. *X̄ Syntax*. Cambridge, MA: MIT Press.
- Kornai, András and Geoffrey K. Pullum. 1990. The X-bar theory of phrase structure. *Language* 66:24–50.
- Kracht, Marcus. 1997. Inessential features. In C. Retoré, ed., *Logical Aspects of Computational Linguistics: First International Conference, LACL '96 (Selected Papers)*, no. 1328 in Lecture Notes in Artificial Intelligence, pages 43–62. Berlin and New York: Springer.
- Moss, Lawrence S. and Hans-Jörg Tiede. 2006. Applications of modal logic in linguistics. In P. Blackburn, J. van Benthem, and F. Wolter, eds., *Handbook of Modal Logic*. Amsterdam: Elsevier.
- Palm, Adi. 1999. Propositional tense logic for finite trees. Presented at the Sixth Meeting on Mathematics of Language, University of Central Florida, Orlando, Florida; <http://www.phil.uni-passau.de/linguistik/palm/papers/mol99.pdf>.
- Postal, Paul M. 1970. On coreferential complement subject deletion. *Linguistic Inquiry* 1:439–500.

- Rogers, James. 1998. *A Descriptive Approach to Language-Theoretic Complexity*. Stanford, CA: CSLI Publications.
- Tiede, Hans-Jörg. 2008. Inessential features, ineliminable features, and modal logics for model theoretic syntax. *Journal of Logic, Language and Information* 17(2):217–227.
- Zaenen, Annie. 1983. On syntactic binding. *Linguistic Inquiry* 14:469–504.

