Comp. by: PG2720 Stage : Revises1 ChapterID: 0001139978 Date:23/4/10
Time:12:01:40 Filepath:d:/womat-filecopy/0001139978.3D
OUP UNCORRECTED PROOF – REVISES, 23/4/2010, SPi

11

Inessential features and expressive power of descriptive metalanguages

Geoffrey K. Pullum and Hans-Jörg Tiede

11.1 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, linguists tend to feel that such features 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 features like AUX (distinguishing the verbs that can be clause-initial in closed interrogative independent clauses) or WH (distinguishing the relative and interrogative pronouns from other pronouns) also draw real rather than artefactual distinctions.

But linguists do not always feel this way about all of the rich array of features posited in current or past work on syntax. Few feel the same way 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? Generalized 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 subcategorization 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.

Why (to put it rather flippantly) does one feel so confident that the Cambridge Textbooks in Linguistics devoted to particular features – Corbett (1991) on *Gender*, Blake (1994) on *Case*, Corbett (2000) on *Number*, and so on – will never be joined by future books in the series called *Slash*, or *Bar*, or *Doom*?

Linguists are so used to focusing on microproblems and ignoring irrelevant surrounding complexities that they seldom try to pull together even a partial list of the syntactic or morphosyntactic features that they are likely to need to recognize in, say, an analysis of Standard English. But we can get a rough idea by reviewing the (non-semantic) feature distinctions implicitly or explicitly appealed to by a comprehensive grammar of the language such as Huddleston et al. (2002) (henceforth *CGEL*). Some of these are shown in Table 11.1. (Here and from now on we give feature names in small caps.)

The terminology for features in Table 11.1 is either traditional (as with CASE or GENDER), or fairly transparent (PHRASAL, INFLECTABLE), or used in works such as Gazdar et al. (1985) (PFORM, VFORM), or adapted from the informal presentation in *CGEL* (CLAUSETYPE, HOLLOW). Examples of constituents that would bear particular values for the features are given in parentheses in the right-hand column. None of these features seems likely to be dispensable in any fully explicit grammar of English. But are they all genuine properties of natural language constituents, or are some of them just artefacts of choices made in theory construction? This is the question we ultimately address in this chapter.

For our purposes, we can ignore the fact that Table 11.1 takes features to be n-ary, for various $n \geq 2$. It should be obvious to any formally sophisticated linguist that issues about the arity of features are extremely unlikely to be interpretable as having empirical implications. Given any description using a feature with x values (for $x \geq 3$), an exactly equivalent description could be achieved using x unary (privative) features, or a set of n-ary features for any choice of n (provided $2 \leq n \leq x$), possibly with some feature co-occurrence restrictions. (To be exact, the minimum number of n-ary features needed to keep x distinct subcategories apart is the smallest integer equal to or greater than $\log_n(x)$.)

Take CLAUSETYPE, which we can take to have the values Declar (declarative), Imper (imperative), ClosInt (closed interrogative), OpenInt (open interrogative), and Exclam (exclamative). We could recode it in terms of binary features, possibly in a way that had some kind of intuitive semantic rationale: [±INTERROG] to separate interrogatives from the rest, with [±OPEN] (limited to [+INTERROG] clauses) subclassifying question-expressing clauses as having open or closed answer sets; [±ASSERTIVE] to separate the proposition-expressing declaratives and exclamatives from the imperatives; and [±EMOTIVE] (limited to [+ASSERTIVE]) to separate exclamatives off from declaratives.

Table 11.1 Some features used in English syntax

Feature name	Value range (with examples)
ADJ-FUNC	Normal, AttribOnly, NeverAttrib, PostPosOnly
AUX	+ (*may*), − (*make*)
AUXINIT	+ (*May I go*), − (*I may go*)
CASE	Nom (*I*), Acc (*me*), DepGen (*my*), IndGen (*mine*)
CATEGORY	Noun, Verb, Adj, Adv, P, D, Sbr, Cdr, Intj
CLAUSESTRUC	Main, Content, Relative, Comparative, Verbless, ...
CLAUSETYPE	Declar, Imper, ClosInt, OpenInt, Exclam
COUNT	+ (*cup*), − (*crockery*)
DEFINITE	+ (*the*), − (*some*)
EVER	+ (*whoever*), − (*who*)
FINITENESS	+ (*that it go*), − (*for it to go*)
DET-HEAD	+ (*this*), − (*the*)
GENDER	Masc (*he*), Fem (*she*), Neut (*it*)
GRADE	Plain, Compar, Superl
HOLLOW	+ (*to look at _____*), − (*to look at it*)
HUMAN	+ (*who*), − (*which*)
INFLECTABLE	+ (*big*), − (*enormous*)
NEGATIVE	+ (*no*), − (*all*)
NFORM	Ordinary, Pron, Dummy-*it*, Dummy-*there*
NUMBER	Sing (*woman*), Plur (*women*)
NUMTYPE	Cardinal (*two*), Ordinal (*second*)
PERSON	1st (*we*), 2nd (*you*), 3rd (*they*)
PFORM	By, To, Of, On, ...
PHRASAL	+ (*see it*), − (*see*)
PROPER	+ (*Microsoft*), − (*microscope*)
VFORM	Pret, 3sgPres, Pres, Plain, Psp, Ger
WH	+ (*who*), − (*he*)
WH-TYPE	Wh-Interrogative, Wh-Relative

And there would of course be many other ways to do it, some involving only three features (since three binary features that cross-classify fully yield $2^3 = 8$ definable subsets of constituents).

Ruminations on which set of features to use, and whether they should be binary, were common in the era of generative phonology. Halle (1957: 67) treats the proposition that all phonetic features are binary as a scientific hypothesis, and seeks to support it on the grounds that (i) it does not impair coverage, (ii) it permits some simplifications, and (iii) it permits an 'evaluation procedure' to be devised. But it is logically impossible for reducing *n*-ary features to binary ones to impair coverage of any set of facts; and it is notoriously difficult to bring claims about descriptive simplicity or evaluation metrics to bear on comparison of theories couched in different theoretical metalanguages. It seems quite implausible that anything factual could be discovered about languages that would settle a question about whether there exists a syntactic feature having more than two values.

We note in passing a point about cross-linguistic identification of features (discussed in Section 2.3.2 of Corbett, this volume, as 'the correspondence problem'). Since features are structurally defined – it is the system of contrasts between values that defines them, not their names – they are most unlikely to be cross-linguistically identifiable, except in the rather loose and approximate semantically based way discussed in *CGEL*, pp. 31–33. For example, we cannot determine that some particular feature in a formal grammar of French is to be equated with some particular feature in a formal grammar of German. Even if they have the same arity, we do not have a formally precise way of determining which one to equate with which using formal criteria. And they may not have the same arity even when they are intuitively similar in semantic terms.

Thus, we have no grounds for formally equating the notion of 'feminine' gender in French with what we call 'feminine' in German: in French the feature we call gender is binary (*le* ∼ *la*) and in German it is ternary (*der* ∼ *die* ∼ *das*). At a detailed level, meaning fails to clarify anything: the French translation of 'the table' is *la table*, the same gender as *la personne* 'the person' and *la jeune fille* 'the girl'; but the German translation of 'the table' is *der Tisch*, a different value from *die Person* 'the person', and different again from *das Mädchen* 'the girl'. So is it *der*, *die*, or *das* that corresponds to French *la*, and why, exactly?

The best we can do is to say that the class of French nouns co-occurring with the *la* form of the French definite article includes a large number of the core nouns denoting female humans (as well as thousands of other referents), and the same is true of the class of German nouns co-occurring with the *die* form of the definite article, and in that sense there is a common-sense semantic rationale for calling both the French *la* class and the German *die* class 'feminine nouns' (see Huddleston et al. 2002: 31–33, for some related remarks).

But there is no hope of rigorous necessary and sufficient conditions being given for calling some feature value the 'feminine' value for an arbitrary language. And there is no hope of answering questions about whether some feature in a Minimalist account is to be equated with a feature employed in an HPSG grammar, or even whether a certain feature in one Minimalist grammarian's analysis is the same one as some other Minimalist is employing.

The specific thesis of this chapter is that distinguishing between the 'real' features and the spurious, artefactual, or superfluous features is even more difficult than might have been thought. We argue that notions like 'spurious feature distinction' or 'artefact 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.

Our main point is a mathematical one. In making it we shall employ the methods of what is becoming known as model-theoretic syntax.

11.2 Model-theoretic syntax

Model-theoretic syntax, henceforth MTS, is represented by several different theoretical frameworks, but only a small minority of theoreticians. The Arc Pair Grammar of Johnson and Postal (1980) was the first full presentation of the MTS idea within linguistics. HPSG in its recent versions is another fairly clear example.

The leading idea is that linguistic expressions – syllables, words, phrases, etc. – should be represented for theoretical purposes as *structures* in the sense familiar from model theory. A structure is simply a set with various relations and functions defined on it and certain individual constants identified within it. And an MTS grammar is simply a finite set of statements, formulated in a logical metalanguage adapted to the statement of descriptions of natural language structural properties.

Model theory is typically familiar to linguists only through its application in semantics, but the application we are talking about here is very different. In semantics, model-theoretic structures are used to make sentence meaning precise in terms of denotation in a model: the meaning of an expression is explicated in terms of the set of structures (e.g. possible worlds, or situations) that are its models. In effect, we fix a particular natural language sentence in some semantic representational form and consider which structures satisfy it. The way structures are used in MTS is in a sense the opposite of this: instead of fixing a sentence and giving it a meaning by specifying what models it has, we *fix a certain structure and consider what statements it satisfies* – the relevant statements, of course, being the metalanguage statements that make up (or are entailed by) the grammar.

To be more specific, we use a logical metalanguage interpreted on structures of a certain sort. These structures are taken to be idealized representations of syntactic form for particular expressions. For any such structure we can ask whether all of the statements in the grammar are true in it – i.e. whether it is a model of the grammar. We define a structure as well formed in the natural language being described iff it satisfies all the statements that make up the grammar.

Notice that there is no single MTS framework. MTS is an approach to formalization. It is possible, in fact, to take a generative grammatical framework and re-formalize its claims in MTS terms. The insights that can be gained from such re-formalization are explored in such works as Blackburn and Gardent (1995) (on LFG), Blackburn et al. (1993) and Rogers (1997) (on GPSG), Rogers (1998) (on GB), etc. Certain wider theoretical consequences of the MTS approach are discussed by Pullum and Scholz (2001) and Pullum and Scholz (2005). Here, however, we are not attempting to compare or relate MTS to generative frameworks. We merely use certain techniques that have emerged within MTS to make explicit a point that holds for all theories of syntax of whatever kind.

The structures we use below will be of the very simple type known as relational structures. In these there are no functions or individual constants. A relational structure is simply a set of points (the domain) with certain relations defined on it. Properties are just the special case of unary (one-place) relations.

Strings, trees, and graphs of all sorts clearly fall under this definition. Strings represent a very simple special case. A single word like *aardvark* in its ordinary spelling could be represented as a relational structure with the domain {1,2,3,4,5,6,7,8} (the eight positions in the string of letters), with six relations defined on it. There are five unary relations (properties) that we can write with **a**, **d**, **k**, **r**, and **v**, each intuitively corresponding to the property of being (a position labelled with) a certain letter. In addition there is one binary relation intuitively corresponding to the arithmetical successor relation on the integers in the domain. Writing such a structure down in the usual way can be seen as compressing the information it contains in two ways: the actual elements of the domain are represented by the unary relation letters, and the successor relation is represented by immediate right-adjacency on the page.

To say that a grammar Γ defines a certain structure $\mathcal{A}$ as well formed is just to say that $\mathcal{A}$ *satisfies* Γ (or *is a model of* Γ), notated $\mathcal{A} \vDash \Gamma$. The structure which represents *aardvark*, for instance, satisfies the first-order formula $\exists x \exists y[\mathbf{a}(x) \wedge \mathbf{a}(y) \wedge x \prec y]$, meaning 'there is an *a* that has an *a* immediately following it'. It does not satisfy $\forall x[\mathbf{a}(x) \Rightarrow \exists y[\mathbf{r}(y) \wedge x \prec y]]$, meaning 'every *a* has an *r* immediately following it' (position 1, for instance, is a counterexample to this).

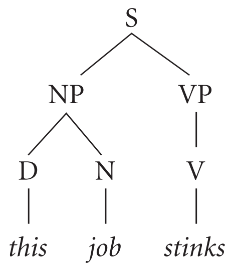
However, most of the interest within model-theoretic syntax has centered on the study of logics interpreted on relational structures much more complex than strings. Logics interpreted on strings can be remarkably weak in expressive power. For example, the example just cited used a first-order metalanguage with a predicate interpreted by 'immediately precedes', and this is an extremely weak formalism. It cannot even describe all the regular (finite-state) sets of strings. It can describe only the 'locally threshold testable' stringsets (Straubing 1994: 46–49), a proper subclass of the star-free or 'counter-free' stringsets studied by McNaughton and Papert (1971), which are in turn a proper subclass of the finite-state. Even if we switch to the most powerful kind of logic that has played a major role in model-theoretic syntax, namely the logics known as weak monadic second-order (wMSO) on string models, we get no more than the regular stringsets, so our descriptions are limited to the descriptive power of finite-state machines. (A wMSO logic is like a first-order language with an extra set of variables for quantifying over finite sets of nodes. See Straubing 1994 or Libkin 2004 for detailed exposition of results on first-order and wMSO logics on strings.)

The most important structures employed in natural language syntax are labelled constituent-structure trees. Our next task is to present a brief introduction to the study of logics on trees because it will be crucial for our main argument.

11.2.1 *Logics on trees*

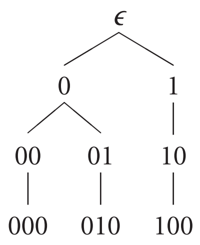
To illustrate how trees can be represented mathematically, we consider the phrase structure tree in (1).

(1)



This tree has nine nodes. These will be the elements of the domain of the structure. It does not really matter what we take them to be – it is the system of relations defined on them that is crucial – but it is convenient to take them to be not just atomic elements like integers but rather *addresses of nodes* – descriptions of positions relative to the root. These addresses can be represented by strings of integers. Starting from the root, which has the empty string (ϵ) as its address, we add '0' for the left child of the root and '1' for 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, thus:

(2)



This means we can identify each node in a binary branching tree with 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.

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:

(3) A *binary tree domain* is a set $T \subseteq \{0, 1\}^*$, such that (where x and y are strings over {0, 1}),

(a) if xy is in T, then so is x, and
(b) if $x1$ is in T, then so is $x0$.

Part (a) of this definition requires any node to have all of its ancestors in the domain (because prefixes of digit strings correspond to addresses higher up on the path to the root), and part (b) requires all right branches to have left siblings (because ending in 1 corresponds to being a leaf on a right branch, and for every right branch there has to be a left branch).

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 $\langle T, R_{\downarrow}, R_{\rightarrow} \rangle$, 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 $R_{\rightarrow}$ is the left-sibling-of relation, i.e. $(m, n) \in R_{\rightarrow}$ iff $m = s0$ and $n = s1$ for some s.

So the tree in (2) would be formalized as a tree structure, $(T, R_{\downarrow}, R_{\rightarrow})$, 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 $R_{\rightarrow}$ relation.

Writing such a structure down in the way linguists usually do can be seen as compressing the information in three ways: writing the appropriate unary relation letters for the elements of the domain; representing the parent relation by sloping downward lines connecting parents to children; and representing precedence by left-right orientation of siblings on the page.

11.2.2 *Modal logic metalanguages*

To talk about trees in this chapter we will use *propositional modal logics*. This choice may need some explanation, though the idea of using modal logics to describe trees goes back at least as far as Gazdar et al. (1988) (see Blackburn et al. 1993 for a more thoroughgoing development of the idea, and Moss and Tiede 2007 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, de Rijke, and Venema (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 (here, at least) 'relational structure' as 'tree'.

The key idea is to represent the labels attached to nodes in trees by atomic propositional formulas 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 labellings present at certain points in the model.

Before we formalize 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 formulas 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 $\langle\downarrow\rangle\phi$ is true at a node v iff ϕ is true at a child of v (so it means 'v has a ϕ-labelled child') and $\langle\rightarrow\rangle\phi$ is true at a node v iff ϕ is true at a right sibling of v.

Comp. by: PG2720 Stage : Revises1 ChapterID: 0001139978 Date:23/4/10
Time:12:01:42 Filepath:d:/womat-filecopy/0001139978.3D

OUP UNCORRECTED PROOF – REVISES, 23/4/2010, SPi

$\mathcal{L}_B$ can be used to say many of the things about trees that could be guaranteed by a context-free grammar. Consider the two formulas in (4):

(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 '⇒' for material implication, defined by $\phi \Rightarrow \psi \equiv \neg(\phi \wedge \neg(\psi))$. So what (4a) says is that a node where **PP** is true has a child where **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 (4b) says is that a node that has a child labelled **during** is labelled P and has a right sibling labelled NP – a subcategorization 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 – it is far weaker than context-free grammars, for example. We need to consider stronger logics.

Three modal logics of increasing expressive strength have been a particular 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 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 formulas for the $\mathcal{L}_{core}$ and $\mathcal{L}_{cp}$ languages is defined as follows:

(5) Basic syntax for $\mathcal{L}_{core}$ and $\mathcal{L}_{cp}$

 a. any atomic proposition is a formula;
 b. $\neg(\phi)$ is a formula if ϕ is;
 c. $\phi \wedge \psi$ is a formula if ϕ and ψ 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 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: $[\pi]\phi$ (with the square brackets visually suggesting the box $\Box$) abbreviates $\neg\langle\pi\rangle\neg\phi$.

$\mathcal{L}_{core}$ 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:

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

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

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

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

(8) Recursively defined modal operators of $\mathcal{L}_{cp}$:

π^*	for each modal operator π
$\pi; \phi?$	

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

The interpretation of $\langle\pi;\ \phi?\rangle$ needs a little more explanation. Intuitively, evaluating $\langle\pi;\ \phi?\rangle\psi$ involves checking that ψ holds at a node that we can get to using π and at which ϕ holds. The two examples in (9) will help.

(9) a. $\langle\downarrow;\phi?\rangle\psi$
b. $\langle(\downarrow;\phi?)^*\rangle\psi$

Formula (9a) can be read as 'at some child of this node where ϕ is true, ψ is true'. This is equivalent to $\langle\downarrow\rangle(\phi \wedge \psi)$, and thus would also be expressible in $\mathcal{L}_{core}$. But the same is not true of (9b). For (9b) to be satisfied at a node u, there has to be a node v, dominated by u, at which ψ is true, and additionally ϕ has to be true at all nodes on the path from u to v. Notice that the asterisk is on '$(\downarrow;\phi?)$', so what is repeated is both the step down from parent to child and the check on whether ϕ holds. There has to be a parent–child chain in which at every node the check to see if ϕ holds is successful, and it has to lead down to a node where ψ holds. This relation is inexpressible in $\mathcal{L}_{core}$.

Notice the potential applications of a formula like (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 $\langle(\downarrow;\phi?)^*\rangle\psi$, where ψ is the property of being a trace and ϕ, holding at all the nodes on the spine leading down to the node where ψ holds, is the property that determines the relevant phenomena.

11.2.3 *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 Val is a *valuation* function – a function from formulas 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 formulas **S**, **NP**, **VP**, **D**, . . . , and thus the binary tree model corresponding to the example in (1) would contain a valuation Val such that $\text{Val}(\textbf{NP}) = \{0\}$, $\text{Val}(\textbf{V}) = \{10\}$, etc.

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

(10) For a model $\mathcal{M}$, a node v of $\mathcal{M}$, and a formula ϕ:

a. $\mathcal{M}, v \vDash p \quad \Leftrightarrow \quad v \in \text{Val}(p)$
(v satisfies atomic formula p iff v is in the set Val assigns to p)

b. $\mathcal{M}, v \vDash \phi \wedge \psi \quad \Leftrightarrow \quad \mathcal{M}, v \vDash \phi \wedge \mathcal{M}, v \vDash \psi$
(v satisfies a conjunction iff it satisfies both conjuncts)

c. $\mathcal{M}, v \vDash \neg\phi \quad \Leftrightarrow \quad \mathcal{M}, v \nvDash \phi$
(v satisfies the negation of any formula that it does not 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_π:

(11) $\mathcal{M}, v \vDash \langle\pi\rangle\phi \quad \Leftrightarrow \quad \exists u[(v,u) \in R_\pi \wedge \mathcal{M}, u \vDash \phi]$
(v satisfies $\langle\pi\rangle\phi$ iff it bears the π relation to a node u that satisfies ϕ)

Comp. by: PG2720 Stage : Revises1 ChapterID: 0001139978 Date:23/4/10
Time:12:01:43 Filepath:d:/womat-filecopy/0001139978.3D
OUP UNCORRECTED PROOF – REVISES, 23/4/2010, SPi

Given our discussion of $R_{\downarrow}$ and $R_{\rightarrow}$ in Section 11.2.1 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 2007).

11.2.4 *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 the set of atomic formulas, denoted by F, to be finite. Atomic formulas will be used to represent features. 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 formulas in the logic $\mathcal{L}$ that only use the atomic formulas 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 $\phi \in \mathcal{L}$ such that $\mathcal{T} = \{\tau \mid \tau, \varepsilon \models \phi\}$ (that is, $\mathcal{T}$ is the set of all and only those trees which at the root node satisfy ϕ).

As an example of how grammars of certain specific types can be formalized 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).

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

Proof Let the terminals, non-terminals, 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 generalize the result to n-ary branching), every rule in P is of the form $A \longrightarrow BC$ or $A \longrightarrow a$, with $A, B, C \in V_N$ and $a \in V_T$. (Here and below we reserve '$\longrightarrow$' 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 formulas 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 Ψ is a disjunction that for each rule $A \longrightarrow BC$ contains a disjunct of this for

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

So if the only rules with VP on the left of the arrow were (i) VP $\longrightarrow$ V_1, (ii) VP $\longrightarrow$ V_2 NP, and (iii) VP $\longrightarrow$ V_3 Clause, the corresponding logical statement would contain a statement that in effect says this: 'If **VP** holds at a node then either (i) $\mathbf{V_1}$ holds at a child node that has no right sibling, or (ii) $\mathbf{V_2}$ holds at a child node that has a right sibling where **NP** holds, or (iii) $\mathbf{V_3}$ 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 Ψ 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 ϕ_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 ϕ_G (from what has been said so far, there could be other models of ϕ_G with all sorts of labellings about which the rules say nothing, and they could vacuously satisfy the statements summarized 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:

(14) $[\downarrow^*](\mathbf{A_1} \vee \mathbf{A_2} \vee \ldots \vee \mathbf{A_n})$ where $V_T \cup V_N = \{A_1, A_2, \ldots, A_n\}$

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

(15) $[\downarrow^*](\phi_1 \wedge \phi_2 \wedge \ldots \phi_k)$ where $\phi_1, \ldots, \phi_k$ 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, **S** must hold at any node where not even the dummy tautology $\top$ holds at the immediately dominating node:

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

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):

(17) $[\downarrow^*]\Phi$, where Φ is the conjunction
$(\mathbf{a}_1 \Rightarrow \neg\langle\downarrow\rangle\top) \wedge (\mathbf{a}_2 \Rightarrow \neg\langle\downarrow\rangle\top) \wedge \ \ldots \ \wedge (\mathbf{a}_k \Rightarrow \neg\langle\downarrow\rangle\top)$ for all $a_i \in V_T$.

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

(18) $[\downarrow^*]\Phi$, where Φ is the conjunction
$(\mathbf{A}_1 \Rightarrow \langle\downarrow\rangle\top) \wedge (\mathbf{A}_2 \Rightarrow \langle\downarrow\rangle\top) \wedge \ \ldots\ \wedge (\mathbf{A}_k \Rightarrow \langle\downarrow\rangle\top)$
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.

11.3 Features

The trees considered so far are labelled with atomic category labels, represented in the description logic as atomic formulas 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 formulas. That way, each atomic formula corresponds to a binary valued feature: if some feature 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 formulas limiting their co-ocurrence. 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 (15) – saying that one and only one of these features is true at each node ('$\mathbf{bar0} \Rightarrow (\neg\mathbf{bar1} \wedge \neg\mathbf{bar2})$', and so on).

To give an example of the use of $\mathcal{L}_{cp}$, consider the following formalization 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 $H\phi \equiv \phi \wedge \mathbf{head}$.

Then we define what it is for a feature ϕ to be projected from a leaf:

(19) $\text{Proj } \phi \equiv \langle(\downarrow; (H\phi))^*\rangle(H\phi \wedge \mathbf{lexical})$

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 Lex, we assert $[\downarrow^*]\Phi$, where Φ is the disjunction of all the statements Proj ϕ such that ϕ is in 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:

Comp. by: PG2720 Stage : Revises1 ChapterID: 0001139978 Date:23/4/10
Time:12:01:44 Filepath:d:/womat-filecopy/0001139978.3D

(20) $[\downarrow^*](\mathbf{head} \Rightarrow \neg(\langle\leftarrow\rangle\mathbf{head} \vee \langle\rightarrow\rangle\mathbf{head}))$

11.3.1 *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 τ. 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 to the description, can now be formalized by means of the following definition:

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

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

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.

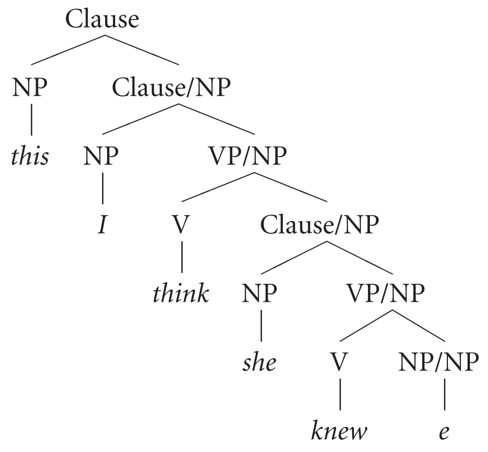
These results can be read in two different ways. First, they state that any language that cannot be defined in a weaker formalism but can in a stronger one can be defined in the weaker one if additional features are added. Conversely, they state that the only difference between the different formalisms 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 characterizes the tree-sets that are recognizable 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 relabelling of all the nodes in all the trees of a regular tree-set will always yield a regular tree-set.

To make this clearer, imagine a finite-state tree automaton that recognizes some set of trees with all nodes labelled by either *A* or *B*. Suppose we wanted to relabel the *B* nodes as *A* nodes without losing track of which were the original *A* nodes. We can simply modify the automaton so it has two different states for admitting *A* nodes: one in effect corresponding to 'original *A* node', and the other to 'relabelled *B* node'. 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 α is marked with a feature specification [SLASH:β] in GPSG in order to identify it as containing a β extraction site. This eliminates any need for movement transformations in the description of unbounded dependencies (Gazdar 1981), as seen in (23), where we simplify visually by notating α[SLASH:β] in the form α/β.

(23)



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 α/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 artefactual, 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.

11.3.2 *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.

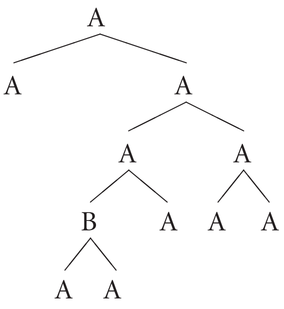
(24) Let F be a finite set of features; let G be a subset of F; and let $\mathcal{T}$ be a set of trees labelled with the features in F. The features in G are **inessential for** $\mathcal{T}$ if the function that eliminates the features in G is one-to-one.

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.

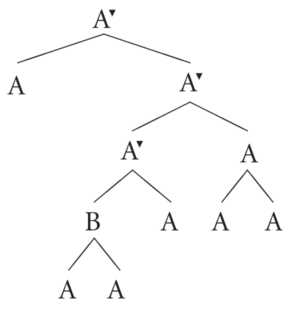
The features referred to in the theorem in (22) are inessential in exactly Kracht's sense. And this might seem to conform to the intuition that when a feature is added solely to make a non-definable set of structures definable, it has an ad hoc nature. When there is a distinction in tree structure that 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 feature 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 (25a) as shown in (25b).

(25) a.



b.



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:

(26) $A^{\blacktriangledown} \rightarrow AA^{\blacktriangledown}$ $\quad A^{\blacktriangledown} \rightarrow AB$ $\quad A \rightarrow AA$
$\mathrm{A}^{\blacktriangledown} \rightarrow \mathrm{A}^{\blacktriangledown}\mathrm{A}$ $\quad A^{\blacktriangledown} \rightarrow BA$ $\quad B \rightarrow AA$

By the theorem in (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.

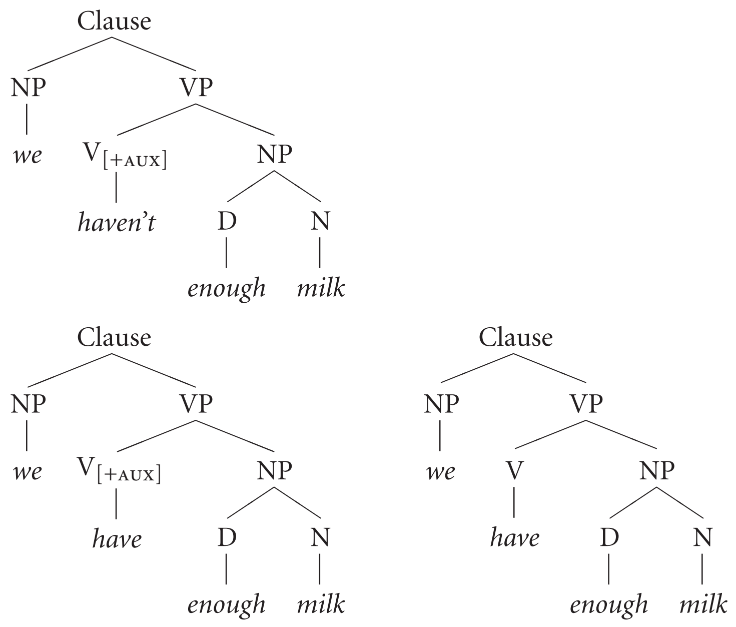
The BAR feature is also (as Kracht notes) inessential. To see this intuitively, imagine being presented with a tree from, say, Jackendoff (1977), with all of the bar-level indications removed. It would be easy to put them all back without error, given the content of the X-bar principles that Kornai and Pullum (1990) call Lexicality, Uniformity, Succession, and Weak Maximality, plus the observation that Jackendoff simplifies his diagrams in certain respects (specifically, a branch of the form X'''—X''—X'—X—σ will be shown as X'''—σ). Stripping the primes from Jackendoff's trees never collapses a legal tree into the prime-stripped version of another legal tree. (This might not be true for every version of X-bar theory.)

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

Take the feature CASE in a language with a simple Nom vs. Acc contrast, for example. In many such languages no two trees will be distinguished on the basis of this feature distinction, as its distribution will be fixed by other aspects of the trees: an NP functioning as Subject in a tensed Clause will take the Nom value and an NP functioning as Object will take Acc, and so on. In such a case the morphosyntactic feature CASE will be 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 so. Counter-intuitively, features are essential when they are *optional* on a node. Consider AUX in English. Any 'subject-aux inversion' clause will predictably have AUX on the initial verb. So will any tree with a verb form ending in the suffix *n't*. But take a dialect where both *We haven't enough milk* and *We don't have enough milk* are grammatical. In such dialects, possession **have** is optionally [+AUX]. So all three of these trees should be well formed (the V in the third is [−AUX]):

(27)



The second and third of these trees will be collapsed if the [±AUX] markings are erased. Solely because of this, AUX counts as an essential feature. Yet of course, on the second and third of the trees in (27) 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 does not matter whether it bears the marking [+AUX] or not. Though essential in the technical sense, it is entirely superfluous in the intuitive descriptive sense.

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

11.4 Conclusions

We have argued that it is highly unlikely that any formal reconstruction can be given of the intuitive notion of a feature that is a technical artefact 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 trade-off 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 motivated 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.

Philosophy of linguistics, like any other branch of the philosophy of science, is just not going to be that easy.