Computational Semantics—Linguistics and Processing*

John Nerbonne
Behavioral and Cognitive Neurosciences
Rijksuniversiteit Groningen
Oude Kijk in 't Jatstraat 41
NL 9700 AS Groningen, The Netherlands
nerbonne@let.rug.nl

1 Introduction

Computational linguistics assumes from linguistics a characterization of grammar—the relation between meaning and form—in order to focus on processing, e.g., the algorithms needed to compute meaning given form (parsing), etc. Computationalists naturally construe grammar as a set of constraints on the meaning-form relation, which suggests constraint-resolution as a processing strategy. This paper illustrates the constraint-based view of semantics and semantic processing and attempts to draw implications for semantic theory. This introductory section explains background assumptions, particularly about the division of labor between linguistics and computational linguistics and also about constraint-based linguistics theory. Section 2 illustrates how constraint-based theory reconstrues the syntax-semantics interface, and Section 3 illustrates how constraint resolution provides a welcome freedom for processing. Section 4 offers some modest conclusions about this work.

The present paper is NOT an attempt to provide a picture of the state of the art in computational semantics, which includes some sensible attention to Artificial Intelligence, and also to the needs of attempting to deal with applications. The papers in Rosner and Johnson 1992 provide a good overview of trends in the fields (see also the review in Nerbonne 1994), while Nerbonne et al. 1993 sketches one

*I had hoped to coauthor this with Per-Kristian Halvorsen of Xerox PARC, but we both became too busy for effective collaboration. He certainly influenced my ideas on these matters, as did the audience at the 1990 Annual Meeting of the Association for Computational Linguistics, where he and I presented related material as a tutorial. Thanks too to Ron Klopstra, who implemented a parser illustrating the ideas in Section 4, as part of his 1993 Groningen Master's Thesis.

approach to trying to accommodate applications even while making use of work in theoretical semantics. The present paper focuses on topics which lie closer to the core of interest in theoretical semantics—the syntax-semantics interface and the processing of semantic information.

1.1 What is Computational Semantics?

There is a natural division of THEORETICAL labor between the disciplines of linguistics and computational linguistics, namely that linguistics is responsible for the description of language and computational linguistics for the algorithms and architectures needed to compute with these. On this view the theoretical fields are related by their common focus on language, and moreover, computational linguistics is dependent on linguistics for the characterization of the relations it computes. Kaplan 1987 articulates this view further, and it is popular among computationalists.

Each of the fields has its more empirical and more theoretical aspects—the distinction at hand is orthogonal. Linguistics has its descriptive and theoretical perspectives, and so does computational linguistics. Computationalists DESCRIBE concrete algorithms and architectures (and report on their relative successes), but they also analyze these theoretically—in terms of their decidability, time/space complexity, the data structures they require, and, in the case of parallel algorithms, the communication protocols needed.

The division of theoretical labor suggested here is sometimes obscured by the many other purposes which computers serve in linguistic research, e.g., as vehicles for projects in applied linguistics (natural language interfaces, information retrieval, computer-assissted language learning, etc.); as visualization tools; as laboratories for linguistic experiments; as channels to immense data reserves in the form of corpora; as repositories for data organization, storage, and retrieval; etc. But clearly the use of computers cuts across the usual divisions of theory/application, theory/experiment and theory/data.

1.2 Feature-Based Theories

The extensive use of features and various concepts of feature matching gave rise in the 80's to "feature-based grammars" and eventually "feature logics" (Bresnan 1982, Gazdar et al. 1985, Carpenter 1992). Although these were initially developed by linguists, mathematical work on feature-based formalisms was also taken up by computational linguistics. The present paper is too limited in scope to provide an introduction to all this work; Shieber 1986 is the fundamental introduction and may be studied accompanied by PATR-II (Shieber et al. 1983), an implementation of the basic mechanisms. The uses we make of the work should be clear from the informal illustrations. The main aspects of feature grammars we wish to exploit here is their ability to encode PARTIAL INFORMATION, including information specified variably. We turn to this below after providing some illustration of the use of feature description languages in semantics.

1.3 A Simple Illustration

There are several alternative means of specifiying the constraints associated with a grammar (cf. Bresnan 1982, Gazdar et al. 1985, Carpenter 1992). The simplest formalism, PATR-II (Shieber 1986), sees linguistic objects as trees with feature-value decorations. We might have used this simplest theory to emphasize that no parochial assumptions bear on the points below, but for the sake of conciseness, we use an attribute-value representation. The relation to PATR-II is illustrated in a first version of a grammar, to which we now turn.

The fundamental idea is simply to use feature structures to represent semantics. If one wishes to compute the semantics of a sentence such as *Sam runs*, one first defines a primitive grammar which admits this. In PATR-II this can take the following form:

```
;; Lexicon (cont.)
;; =======

Word Sam:  <cat> = NP          Word runs: <cat> = VP
           <agr> = sg                     <agr> = sg
           <sem> = m.                     <sem pred> = run
                                          <sem arg>  = <subj sem>.
;; Grammar Rule
;; ============

Rule {Sentence}

     S -> NP VP   <NP agr> = <VP agr>
                  <sem> = <VP sem>
                  <NP sem> = <VP subj sem>.
```

The context-free notation in this example grammar can be read in the usual way, while the equations constrain properties of the grammatical objects. For example, the first equation associated with *Sam* specifies that its category is NP, while the first equation associated with the rule specifies that the agreement of NP and VP must coincide. Note that equations specifying syntactic as opposed to semantic properties have no special status. Accordingly, various strategies about the optimal order in which to process constraints are possible.

The feature SEMANTICS (`sem`) is included in this example in order to show how the mechanism can be used to encode semantics. The semantics feature is lexically provided for in the case of *Sam* and *runs*, and is specified for the example sentence on their basis. As primitive as this example is, it illustrates two techniques crucial to using features-based systems to specify semantics: first, specifications may be complex, as the lexical semantics for *runs* (involving `<sem pred>` and `<sem arg>`) illustrates. Second, variables may be employed esp. to specify the semantics of

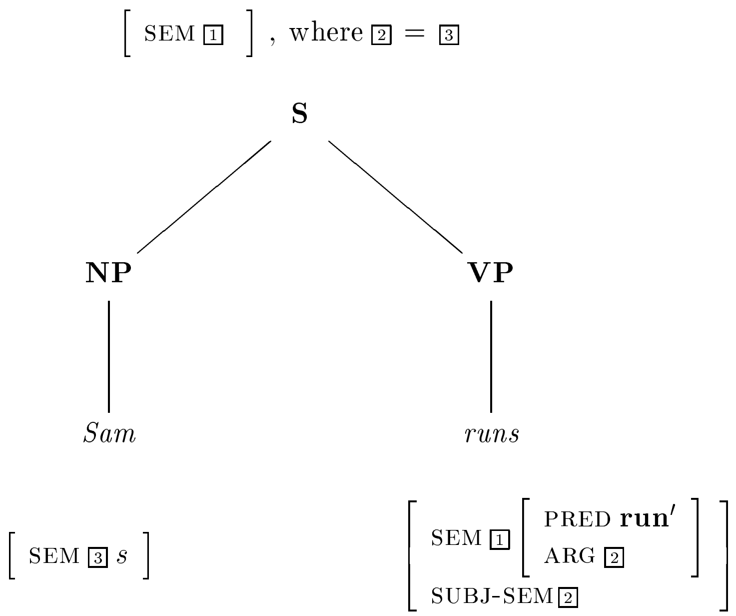


Figure 1: A sketch of the semantic derivation of *Sam runs*, **run**$'(s)$ as this would proceed using unification. Unification applies to syntactic and semantic representations alike, eliminating the need to compute these in distinct processes, and unification is employed to bind variables to argument positions, eliminating the need for (a great deal of) β-reduction as used in schemes derived from Montague and the lambda calculus. The reader may verify that the matrices of feature-value specifications are equivalent to those in the PATR-II grammar in the text, but the representation of shared structure via the boxed numbers allows grammars to be more concise.

complex expressions. Thus the semantics of sentences is specified via a variable (required to be equal to the VP semantics). Figure 1 illustrates this employing a more popular representation.

Syntax/Semantics interfaces using feature-based formalisms may be found in Shieber 1986, Pollard and Sag 1987, Fenstad et al. 1987, and Moore 1989. The motivation for these early attempts was certainly the success feature-based descriptions enjoyed in syntax. Treating semantics in the same way meant that syntactic and semantic processing (and indeed all other feature-based processing) can be as tightly coupled as one wishes—indeed, there needn't be any fundamental distinction between them at all. In feature-based formalisms, the structure shared among syntactic and semantic values constitutes the syntax/semantics interface.

Our earlier papers have explored several advantages of the constraint-based view of the syntax/semantics relation over standard views, including (i) the opportunity to incorporate nonsyntactic constraints on semantics, such as those arising from phonology or context (Nerbonne 1992a); (ii) the opportunity to formulate principles which generalize over syntax and semantics, such as those found in HEAD-DRIVEN PHRASE STRUCTURE GRAMMAR (Pollard and Sag 1987, Nerbonne 1992a). Halvorsen 1988, Nerbonne 1992a elaborate on the virtue of understanding feature-based semantics as specifying constraints on logical forms, not model structures. The virtues of viewing semantics processing as manipulation of logical form descriptions includes not only a characterization of semantic ambiguity, which in turn provides a framework in which to describe disambiguation, but also the opportunity to underspecify meanings in a way difficult to reconcile with other views. The latter point is illustrated with an application to the notorious scope ambiguity problem.

1.4 The Constraint-Based View

Montague's famous characterization of the relation between syntax and semantics as a homomorphism naturally gives way here to a CONSTRAINT-BASED view. The constraint-based view was originally motivated by the close harmony it provides with syntax, which is universally processed in a constraint-based fashion. Employing the same processing discipline in syntax and semantics allows that their processing (and indeed other processing) can be as tightly coupled as one wishes—indeed, there needn't be any fundamental distinction between them at all.

In this paper we shall focus on two consequences of the constraint-based view—one linguistic, one computational. Given our own interests it will not be surprising that the lion's share of attention is paid to the computational point. We discuss a linguistic consequence of the constraint-based view in order to give some of the flavor of the linguist-computationalist dialectic.

Given a constraint-based view, it is natural to formulate hypotheses about syntax/semantic interfaces as manipulations of *constraints* rather than as functions on the *semantic objects* directly. This leads immediately to a relaxing of compositionality requirement, i.e. the requirement that the meanings of phrases be the values of functions defined on the constituents' meanings. We illustrate this linguistic point briefly with an eye toward illuminating the computational perspective on linguistic theory.

The computational point is addressed at more length. In order to view semantics as a homomorphic mapping from syntax, Montague needed an artificially "disambiguated syntax"—the only plausible candidate to serve as basis for a *function* from syntax to semantics: $f : DS \mapsto S$. DS, the level of disambiguated syntax, was normally seen as a minor technical inconvenience in the Montague framework. After all, one may always eliminate the apparent unnaturalness by viewing the "real relation" as the image of the disambiguated mapping under a suitable notion of syntactic equivalence (perhaps string identity). $R : sR\sigma \leftrightarrow \exists s' \mathbf{disambig}(s', s) \wedge f(s') = \sigma$. It has always been clear that syntax/semantics constraints may not uniquely determine a semantics. Instead, the constraints UNDERSPECIFY the semantics (in general). We interpret this technically in the following way: the syntax/semantics interface is a relation between syntactic structures (decorated trees) and formulas in a meaning representation language. The relation is underspecified if a single syntactic structure is interpreted as two or more formulas (which are moreover not merely alphabetic variants). Of course the formulas may normally have a good deal in common. We then ignore the function f above and attempt to specify R directly.

This wider view of the syntax/semantics interface has a liberating effect: it becomes quite natural to exploit the set of constraints associated with any stretch of syntactic material—even if that material does not form a constituent. We shall exploit this property in order to illustrate how constraint-based semantics allows a particularly simple approach to the problem of computing semantics incrementally. In this case the partial constraints may combine to restrict syntactic processing hypotheses significantly. The advantage our approach has over others here lies in the fewer assumptions we make about syntactic structure.

There is finally a deeper perspective on our eschewing compositionality. Our motivation for exploring constraint-based formulations of grammar is the freedom this allows in processing. Concretely, this means that we wish to experiment with the order in which information is combined, which translates mathematically into the requirement that we base our semantic specifications on information-combining operations which are COMMUTATIVE—which composition most clearly is not: $f \circ g \neq g \circ f$. This also explains the preference for UNIFICATION as an information-combining operation in constraint-based theories, but, in fact, it is only one of several choices.

2 Noncompositional Constraints

In this section we illustrate the linguistic benefits of focusing on constraints. Into the primitive grammar above we successively introduce lexical ambiguity, phrasal ambiguity, and noncompositional constraints on interpretation.

The constraint-based view has a RELATIONAL take on ambiguity that is fundamentally different (from that of the homomorphic view). Consider further the example in Figure 1 by way of illustration. The verb *runs* is, like most natural language words, highly ambiguous. It can mean 'to go quickly by foot' *She's running in the Marathon*, but also 'to function' *The printer's not running*, 'to flow' *Your mascara is running*, 'to flee' *At the first sign of danger, they ran*, etc. (any large dictionary will list several more). Now, if we require a homomorphic relation between syntax and semantics, then we must define a mapping with this range. Since the mapping must be functional, we can only do this by assuming the ambiguity in the syntax, and then carrying it forward into the semantics. The constraint-based view suggests postulating a non-functional relation between syntax and semantics. There is then a single syntactic item corresponding to *run*, which is constrained to mean one of the things in its dictionary entry. This may be accomplished by changing the specification for *runs*'s semantics (we continue here and throughout with the notation of Figure 1):

$$\begin{bmatrix} \textsc{head}|\textsc{cat}\ \mathrm{VP} \\ \textsc{sem}|\textsc{pred}\ \{\mathbf{go\text{-}fast}', \mathbf{function}', \mathbf{flow}', \mathbf{flee}', \ldots\} \end{bmatrix}$$

This is a disjunctive specification with the content: the semantic predicate is exactly one of the values **go-fast′,...**.[1] It is quite simple, but it already breaks the compositional mold: a single syntactic entity is mapped to several semantic entities. Of course, it is a simple matter to "carry" the ambiguity back into the syntax, and so preserve compositionality, but there is no need to.[2] In fact on reflection it seems strained to postulate various lexical items for *run*—all sharing the same syntactic and morphological properties. The approach taken here can postulate a single lexical item with a variety of semantics interpretations.[3]

Before continuing to remark on issues of "compositionality", we need to clarify the sense in which we use this term. In linguistic semantics the term has normally been taken to require that the meaning of a phrase be the value of a function defined on the meanings of its subconstituents. Taken this way, compositionality is a hypothesis about the semantics of natural language—i.e., the hypothesis that the correct semantics for a natural languge is such that, for any construction (grammar) rule, there is a function which takes the meanings of its constituents as arguments, and yields the meaning of the composed phrase as value. Now, Zadrozny 1994 has shown that, if any semantic mapping exists, then it has a compositional reformulation, which shows the hypothesis to be unfalsifiable. Zadrozny adds that the functions in question are not guaranteed to be computable or even finitely specifiable, and that there may well be nonvacuous hypotheses about compositionality if limited to particular classes of functions, but this is not our concern here. The compositionality hypothesis has such widespread acceptance that in fact grammatical descriptions and description schemes normally simply assume it—and implicitly or explicitly require it. Our point is just that stepping back from this common assumption allows one to assume a different perspective on some problems of grammatical description and processing.

A relational treatment of lexical ambiguity is a harmless deviation from compositionality introduced here for the sake of suggesting that the semantic community has come to assume compositionality rather too automatically. A more interesting variation occurs when we begin treating semantics by manipulating constraints rather than writing functions. In compositional treatments the function yielding the semantics of a mother node views the semantics of daughter constituents as black boxes—as units whose internal make-up is ultimately irrelevant. We automatically shed such blinders when we manipulate constraints.

Consider the case of prepositional phrases used on the one hand as free adjuncts and on the other as optionally subcategorized arguments. These are quite common,

[1] In fact, this involves a mild (and well-studied) extension of the PATR-II formalism to allow disjunction. Kasper and Rounds 1986, Carpenter 1992 have details.

[2] More interesting (but also more difficult to illustrate briefly) are attempts to provide constraint-based theories of nonlexical ambiguity. Nerbonne 1993 provides the foundation for a constraint-based treatment of quantifier scope ambiguity, but it would take us too far afield to present it here.

[3] Of course, this is not the same as a single disjunctive semantics, either. See Nerbonne 1992a for discussion.

as a few examples easily suggest:

(1)
- a. The mill ran on Wednesday
- a′. The mill ran on methane
- b. Sam waited on Wednesday/Mary
- c. Sam waited for hours/Mary
- d. Sam decided on Wednesday (ambig., cf. on Mary)
- e. Sam decided about Wednesday/Mary
- f. Sam voted for Mary (ambig.)
- g. Sam invested in May/Texaco
- h. Sam went on about Christmas (ambig.)

Clearly some strings (e.g., (1d)) are ambiguous. In the adjunct reading, Wednesday was the time Sam's decision was made; in the argument reading it was (part of) the decision itself (in this case the sentence might be taken to mean that Sam decided on Wednesday as the day for a meeting, for example, but the decision might have been made on another day). The analytic question is not vexing: the ambiguity correlates with the argument/adjunct distinction.

But now consider just the prepositional phrase *on Wednesday* in isolation: what meaning should it be assigned in a compositional treatment? In case it appears as a temporal adjunct, it expresses a relation between an event and a time, but in case it's a subcategorized-for argument (as it might also be interpreted in (1d)), it seems merely to denote the day Wednesday. Thus the meaning of the PP phrase seems to depend, not just on the meanings of its parts, but rather on its syntactic context—the fact that it occurs in construction with a particular verb. The difference in the two kinds of PPs is particularly striking in some cases, e.g. (1c), where the subcategorized phrase is interpreted merely as standing in a particular relation to the rest of the arguments, while the adjunct is interpreted as scoping over the relation denoted by the verb (this is a standard analysis for duratives—cf. e.g., Dowty 1979).

In the adjunct (frame adverbial) reading the preposition makes an independent contribution to semantics, at the very least distinct from other adpositions which can head free adjuncts such as *with, on, in spite of, notwithstanding* etc., or the other temporal or locative prepositions such as *before* or *after*. In the argument (co-specifying) case the preposition is simply required so that no independent contribution to semantics is discernible (which is not to deny that the choice of preposition is "partially motivated", i.e., semantically rather better suited to function here than most alternatives). I believe that semanticists are agreed that very different treatments of these phrases are required. Although both types of phrase are optional, the adjuncts may occur multiply in a single clause, which requires recursive structure in representation. (Benefactives are admittedly strained if iterated, so perhaps they ought to be classed with optional arguments.) Arguments, on the other hand, occur once or not at all. It is most straightforward to simply reserve a position for them in a relation.[4]

[4]Of course this means that provision must be made for the case where arguments are

Just as in the case of lexical ambiguity, one can avoid noncompositional treatments in this phrasal case by postulating ambiguity—beginning with the preposition *on*, which can be translated as expressing a relation in the adjunct case, and as vacuous (an identity function) in the argument case. The ambiguity percolates naturally to the PP. Technically, there is nothing amiss with such a treatment, but it seems counterintuitive—in particular, in locating the ambiguity in the preposition, rather than the combination of verb plus prepositional phrase.

We now sketch an alternative, continuing to focus discussion on (1d), because its representation is logically simple. For the sake of concreteness, let's suggest representations: the meaning of the optional argument as part of the relation denoted by *decide*: $\mathbf{decide\text{-}on}'(e, s, w)$ holds iff e is an event of Sam deciding in Wednesday's favor. We'll then represent the temporal adjunct *on Wednesday* as a relation between the event denoted by the verb and Wednesday: $\sqsubseteq_t (e, w)$, i.e., a relation which holds just in case the event is temporally contained within Wednesday. More generally, any theme argument may be specified to stand in the containment relation with respect to the object of the preposition *on*, but we focus on the example (1d).

This is the puzzle for compositional treatments: how can the semantics of the PP construction be a function yielding m (from $\mathbf{on}'$ and m) (*Sam decided on Mary*), but $\mathbf{on}'(e, w)$ (from $\mathbf{on}'$ and w) (*Sam decided on Wednesday*)? One might hypothesize a type senstivity (distinguishing the PERSON m from the TIME w), but the ambiguous examples (e.g., (1d,f,h)) indicate that this does not generalize—more is at stake than polymorphic functions. We seem ultimately forced to postulate ambiguity in the semantics of *on* (and the phrases headed by it), which seems counterintuitive.

Proceeding from a constraint-based perspective, we have a slightly different response available—*viz.*, that there is a single set of constraints which entail that the semantics of the PP is *always* $\mathbf{on}'(e, \mathbf{NP}')$, but that the constraints specifying the semantics of the VP sometimes uses this entirely, and sometimes use only the semantics of the NP object—a clearly noncompositional step (see Davis 1995 for a more compositional suggestion in a constraint-based vein). Below, we extend the grammar presented in Section 1.

Before presenting the extended grammar, it is worth noting two things about this sort of effort. First, by virtue of its being concrete (and implemented), the example grammar is specific about some irrelevant points. This can be distracting, but it has the advantage of being more reliable and perhaps more easily understandable (for being concrete). Second, the example is formulated in an "HPSG" style (á la Pollard and Sag 1987, Pollard and Sag 1994), but this is strictly inessential to the points being made. The same demonstration could be given in LFG, Word Grammar, Definite Clause Grammar, or a logical formulation of GB or minimalism (e.g., that of Stabler 1992). HPSG is used here because I use it elsewhere, and its

missing. In fact there is a variety of interpretations, depending on the verb, but including at least the narrow scope existential (*He sang* iff *He sang something*), contextually definite (*He noticed* iff *He noticed what is contextually definite*), and reflexive (*wash*). We shall simply use the appropriate meaning for the case at hand, the narrow scope existential (*decide*).

attribute-value "boxes" may be easier to read.

We first need to encode the new lexical items. In a larger fragment, most of the specifications included here would be "inherited" from more abstract word class specifications (see Flickinger and Nerbonne 1992 for an extended presentation).

$$\textit{on} \qquad \left[\begin{array}{ll} \multicolumn{2}{l}{\text{HEAD}|\text{CAT } p} \\ \text{SUBCAT} & \left\langle \left[\begin{array}{l} \text{HEAD}|\text{CAT } np \\ \text{SEM } \boxed{2} \end{array}\right] \right\rangle \\ \text{SEM} & \left[\begin{array}{l} \text{PRED } \sqsubseteq_t \\ \text{THEME } \boxed{1} \\ \text{GOAL } \boxed{2} \end{array}\right] \end{array}\right]$$

$$\textit{decided} \qquad \left[\begin{array}{ll} \multicolumn{2}{l}{\text{HEAD}|\text{CAT } v} \\ \text{SUBCAT} & \left\langle \left(\left[\begin{array}{ll} \multicolumn{2}{l}{\text{HEAD}|\text{CAT } \textit{PP-on}} \\ \text{SEM} & \left[\text{GOAL } \boxed{2} \right] \end{array}\right] \right) \right\rangle \\ \multicolumn{2}{l}{\text{SUBJ}|\text{SEM } \boxed{1}} \\ \text{SEM} & \left[\begin{array}{l} \text{PRED } \mathbf{decide\text{-}on}' \\ \text{EVENT } \boxed{3} \\ \text{SOURCE } \boxed{1} \\ \text{THEME } \boxed{2} \end{array}\right] \end{array}\right]$$

Like all our specifications, these are partial. (We shall have occasion to flesh out the prepositional specifications directly.) In comparison with the earlier example, we have first complicated the verb semantics by adding the event argument (to allow a simple treatment of the time adverbial). Second, we make use of a SUBCATEGORIZATION feature to specify the complements which these words select. Its value is enclosed in angle brackets because it is a list (of potentially several items). The specification of the prepositional phrase selected by *decide* is further enclosed by parentheses to indicate that it is optional, as is standard. Finally, we assume that *Mary* is entered in the lexicon as *Sam* is.

A sketch of a (partial) analysis tree is provided in Figure 2. What is noncompositional here is the specification of the THEME semantics of *decide* as covarying NOT with the semantics of the selected complement, but rather with the semantics of its complement's complement. This is a natural noncompositional specification in the formalism here. The (single) rule licensing the different head-complement combinations in the two interior nodes has not been provided, but it is just the head-complement schema of HPSG (Pollard and Sag 1994, p.38), which is subject to the subcategorization principle (*ibid.*, p.34), requiring that specifications imposed by the head must unify with those on complement daughters. The example likewise assumes (uncontroversially) that semantic specifications follow head lines (in head-complement structures).

In order to contrast the adjunct reading to this we need to sketch a treatment of adjuncts. We assume, fairly standardly, that adjuncts selects the heads they combine with, rather than *vice versa*, and we encode selection in the feature MOD.

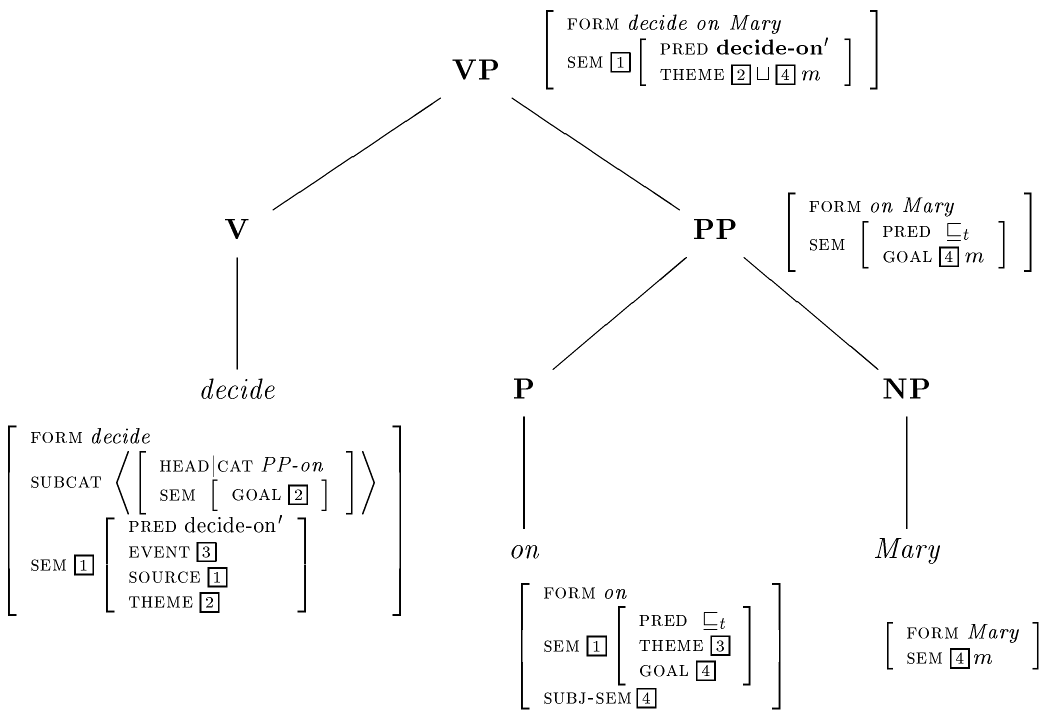


Figure 2: A sketch of the semantics of the VP *decide on Mary*, as it is derived from noncompositional specifications. The semantics of the verb *decide* is specified to bind its theme role to the semantics of the object (goal) of the preposition—without intermediate reference to the semantics of the prepositional phrase it stands in construction with.

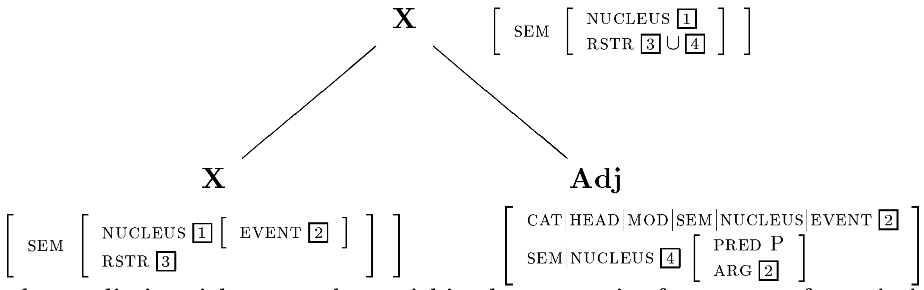
Like subcategorization information, this will unify with the information associated with structures admitted by the rule. The earlier lexical entry for the prepostion *on* suppressed this information, which we therefore now supplement:

$$
\textit{on} \quad \begin{bmatrix}
\text{HEAD} & \begin{bmatrix} \text{CAT } p \\ \text{MOD } \begin{bmatrix} \text{SEM } \boxed{4} \begin{bmatrix} \text{EVENT } \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\
\text{SUBCAT} & \left\langle \begin{bmatrix} \text{HEAD}|\text{CAT } np \\ \text{SEM } \boxed{2} \end{bmatrix} \right\rangle \\
\text{SUBJ}|\text{SEM } \boxed{1} & \\
\text{SEM} & \begin{bmatrix} \text{PRED } \sqsubseteq_t \\ \text{THEME } \boxed{1} \\ \text{GOAL } \boxed{2} \end{bmatrix}
\end{bmatrix}
$$

Note that the argument position bound to the variable '$\boxed{1}$' is now further specified as the EVENT (time) of the object modified. This will eventually account for the semantic effect of the frame adverbial—that of restricting the time at which the event is said to take place.

The received view of the semantic contribution of adjuncts is that they fall into two classes—one class, typified by conditional clauses, serves as an OPERATOR scoping over the head it is in construction with, and a second class, typified by locative and most temporal adverbials, serves as a RESTRICTOR of some argument position within the head. Kasper 1994 provides an elegant means of reducing the second class to the first, so that the apparent grammatical distinction may be reduced to a purely lexical one. Kasper's specifications are too sophisticated to be introduced and explained here in full (but see note). We shall assume the effect of his specifications: the semantics of a head-adjunct construction with a restrictor adjunct will be assumed from the head, and the adjunct semantics will serve to restrict some argument with the head semantics.[5] In order to implement Kasper's ideas on restrictive adjunction, we require a minor complication of the grammar: the feature SEMANTICS (abbreviated 'SEM' above), which has heretofore specified the semantic translations of words and phrases, will now be divided into NUCLEUS (abbr. NUCL), the core around which the full-fledged semantics is built, and RESTRICTIONS (abbr. RSTR), which describe constraints the arguments in the nucleus are subject to. The few examples treated thus far may all be construed as manipulating only the nucleus part of the semantics.

The specifications foresee the following schema for this restricting class of adjuncts:



The schema distinguishes a nucleus within the semantics from a set of restrictions on

[5]Kasper's treatment is particularly attractive in that it requires no idiosyncratic assumptions about syntactic structure; in particular, it is compatible with there being multiple adjuncts within a single (nonrecursive) VP node. The following contains (a simplification of) the relevant specifications (from Kasper 1994, p.63, which contains full explanation and justification):

$$
\textit{gestern} \quad \begin{bmatrix} \textsc{cat}|\textsc{head}|\textsc{mod}|\textsc{sem} & \begin{bmatrix} \textsc{nucl}\ \boxed{1} \\ \textsc{rstr}\ \boxed{3} \\ \textsc{context}|\textsc{ref-time}\ \boxed{1} \end{bmatrix} \\ \textsc{sem} & \begin{bmatrix} \textsc{nucl}\ \boxed{1} \\ \textsc{rstr}\ \boxed{3} \cup \left\{ \begin{bmatrix} \textsc{pred} & \sqsubseteq_t \\ \textsc{theme} & \boxed{1} \\ \textsc{goal} & (\boxed{4}|\mathbf{yest}'(\boxed{4},\boxed{5})) \end{bmatrix} \right\} \\ \textsc{context}|\textsc{speech-time}\ \boxed{5} \end{bmatrix} \end{bmatrix}
$$

The features SEM|NUCL "semantic nucleus" and HEAD|MOD|SEM|NUCL are coindexed to require that the semantics of the adjunct (which is to be passed on as the semantics of the adjunct-head phrase) be taken from the head it modifies. The effect is that the semantics of head-adjunct constructions involving this class of adverbials is determined by the head. The contribution of the adjunct itself is to contribute to the set of restrictions associated with this semantics. This may be found in SEM|RSTR, in the right side of the set-union operator, where it is specified that the time '$\boxed{1}$' (reference time of the head semantics) fall within the day before speech time. This sort of specification is itself a further example of the sort of noncompositionality possible in this approach.

its arguments. Within a head-adjunct phrase (involving the class of adjuncts under discussion), the head determines the semantic nucleus of the phrase and contributes whatever restrictions it has accumulated, while the adjunct adds its own restriction. Figure 3 illustrates how this scheme is applied to the case of the frame adverbial *on Wednesday.*

In the illustration in Figure 3, the head-adjunct phrase *decided on Wednesday* would have a head semantics consisting of a nucleus specifying an event of deciding, and a restriction arising from tense that the event be past. It would have an adjunct semantics specifying that the event occur on (a contextually determined) Wednesday. The phrasal semantics has the same nucleus as the head and contains all the daughters' restrictions.

The purpose of this section has been the illustration of the advantage of the added freedom which contraint-based semantics allows as compared to strictly compositional treatments of syntax-semantics interaction. For further applications of

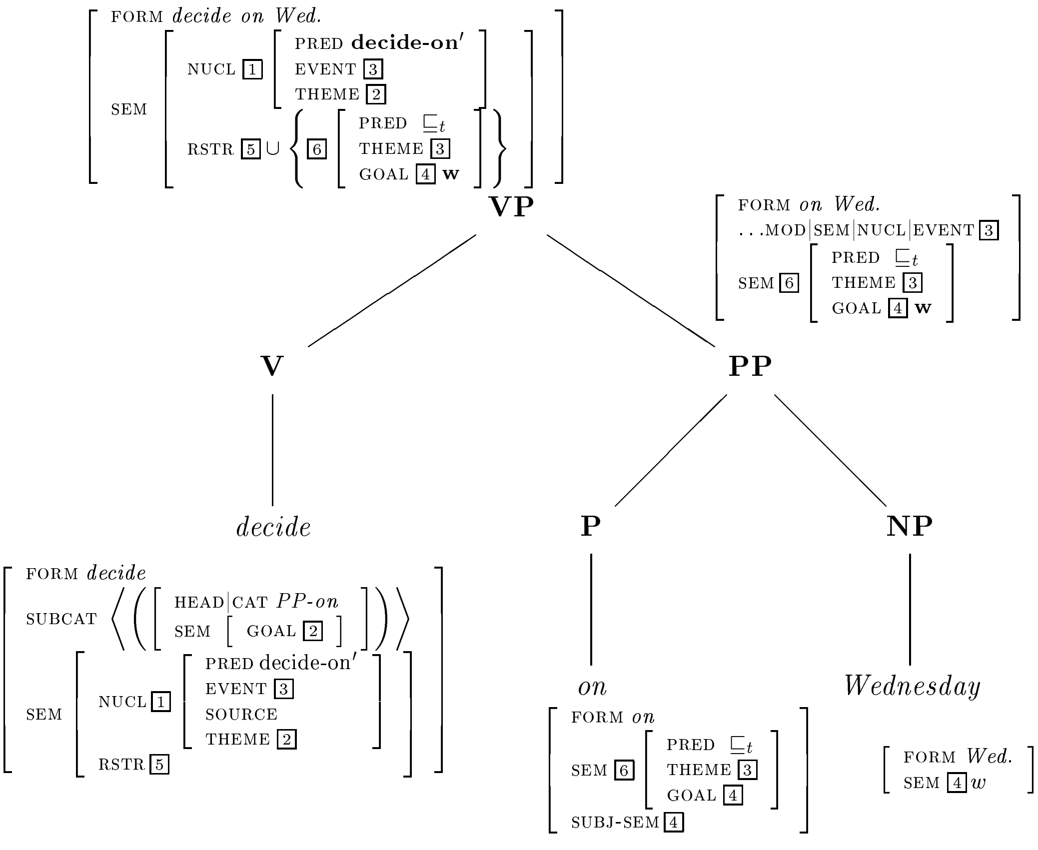


Figure 3: A sketch of the semantics of the VP *decide on Wednesday*, as it is derived from (incidentally) compositional specifications. The semantics of the verb *decide* includes a set of restrictions to which the adjunct semantics is added.

constraint-based semantics, the reader is referred to the references in this section, but also to Dalrymple et al. 1991, which contains an interesting treatment of ellipsis which makes essential use of constraint-based description.

3 "Noncompositional" Processing

In this section we interpret "selectional restrictions" semantically, exploiting the fact that they can be used to reduce the number of parses available.[6] Once one takes this step, it is natural to try to take advantage of such information early in processing—maximizing the efficiency benefits, and plausibly modeling human language users more faithfully in this respect as well. Taken to its extreme, this means that semantic processing must allow information flow along noncompositional lines. This does NOT reject the grammatical thesis that syntax-semantics dependence is ultimately compositional, only that processing is organized along the same lines.

COMPOSITIONALITY concerns the relation between syntax and semantics—i.e., (static) linguistic structure, not processing. Nonetheless, the compositional view often extends to a natural and popular procedural interpretation, that of bottom-up processing. This is a natural interpretation because compositional semantics specifies the semantics of phrases via functions on the semantics of their daughters. It is natural to evaluate arguments before attempting to apply functions to them (although partial and so-called "lazy" evaluation schemes certainly exist) (see, e.g., Kahn 1987). We will use the term BOTTOM-UP EVALUATION for semantic processing which proceeds bottom-up in the analysis tree, evaluating arguments and then functions applied to them along the lines suggested by compositional grammatical theory. We will therefore try to reserve the term COMPOSITIONAL for theses about grammatical structure (or the formulation of hypotheses about this)—the title of the present section notwithstanding. But the general subject will be processing, so if a bit of metonmymy creeps into the discussion, it should not be overly distracting. To forestall misunderstanding, let us reiterate immediately our position that processing considerations may not be conflated with linguistic ones. That fact that purely bottom-up processing is less than optimal is completely consistent with there being a compositional syntax-semantics interface.

For the purposes of the present section, we may view semantic interpretation as a form of PARSING, i.e., computing the (set of) structure(s) which a grammar associates with a string. In syntactic parsing the relevant structures are syntax trees, while in semantic processing the structures are semantic representations—normally, expressions in a logic designed for meaning representation. It is possible to separate syntax and semantic processing, in which case one views semantic processing as inputing, not strings, but syntactic analyses. This division of labor is normal, practical, and—depending on assumptions about the temporal organization of the putative processes—possibly tenable from computational and psychological

[6]This has been common practice in computational linguistics at least since Wilks 1975. But see Gale et al. 1992 for arguments that some such selection is not semantic.

viewpoints as well. This section will not simply assume this, however. The ideas advanced here are independent of whether there are distinct process for syntax and semantics.

Bottom-up processing is only one of many possiblities, not distinctly superior either by virtue of its technical properties or by its fidelity to the psycholinguistic facts.[7] In fact, pure bottom-up parsing is clearly poor both in efficiency and as a psychological model (see previous note). It is easy to see why this should be, since bottom-up processing restricts the amount of information which is accessible when forming and prioritizing hypotheses. Top-down information can be useful.

INCREMENTAL PROCESSING computes analyses while inputting strings one word at a time, in the order in which they're read (the left-right order in text). It enjoys wide acceptance in psycholinguistic research, and it is useful in many applications, since it exploits information as quickly as it is available. The intuitive notion of 'incremental procesing' requires some sharpening: after all, almost all parsing algorithms—including bottom-up ones—read input from left to right. Schabes 1990 suggests that procedures be regarded as incremental when they obey the VALID PREFIX PROPERTY:

> If the input tokens $a_1 \ldots a_k$ have been read then it is guaranteed that there is a string of tokens $b_1 \ldots b_m$ (b_i may not be part of the input) such that the string $a_1 \ldots a_k b_1 \ldots b_m$ is a valid string of the language. (Schabes 1990, p.54)

That is, an incremental procedure must reject a string as soon as there is no valid continuation. This seems like a good definition; in particular, purely bottom-up and purely top-down parsing algorithms certainly do not count as incremental according to this definition, which is just as it should be.

We turn now to an illustration of how the computation of partial constraints on semantics can be useful in incremental evaluation. This should also further serve to clarify the distinction between incremental and nonincremental processing. For this purpose, we provide a brief feature-based treatment of selectional restrictions (see Nerbonne 1992b for further detail).

The word *chair* is ambiguous, possibly referring to a piece of furniture but also to the head of an organization, as in *the chair of the committee.* In a sentence such as *The chair decided on Mary*, we spontaneously understand only the reading of *chair* as human—or, at least as a mental agent.[8] It is trivial to write feature specifications which enforce the requirement that subjects of *decide* be things capable of mental

[7]It would far exceed the bounds of this contribution to attempt to review the range of possibilities or issues in parsing. Some recent surveys are useful. Sikkel 1993 reviews the formal properties of a very wide range of parsing algorithms, Bouma and van Noord 1993 reviews the practical performance of a range of algorithms, controlling for grammar type, and Mitchell 1994 reviews the psycholinguistics of human parsing.

[8]We ignore here and henceforth the status of the problematic "reading" in which a decision is attributed to a piece of furniture. Nothing crucial seems to hinge on it. There is a tricky issue associated with the exact representation of the deviant "reading", however.

agency. We simply introduce a feature, e.g. M-AGT and then require that subjects of *decide* be compatible with this feature:

$$\begin{bmatrix} \text{FORM } \textit{decide} \\ \text{SEM}|\text{NUCL} \begin{bmatrix} \text{PRED } \mathbf{decide\text{-}on'} \\ \text{SOURCE}|\text{M-AGT} + \end{bmatrix} \end{bmatrix}$$

Similarly, we shall wish to differentiate the readings of *chair* using this same feature:

$$\begin{bmatrix} \text{FORM } \textit{chair} \\ \text{SEM}|\text{NUCL} \left\{ \begin{bmatrix} \text{PRED } \mathbf{furn\text{-}for\text{-}sitting'} \\ \text{THEME}|\text{M-AGT} - \end{bmatrix}, \begin{bmatrix} \text{PRED } \mathbf{org\text{-}head'} \\ \text{THEME}|\text{M-AGT} + \end{bmatrix} \right\} \end{bmatrix}$$

(We again employ set braces to denote a disjunction of potential readings, as in the first example in Section 2 above.) The SOURCE role of *decide* is thus incompatible with the THEME role of *chair* in the furniture reading—the two values do not unify and therefore cannot be identified. This is sufficient to guarantee that any attempt to use use *chair* as the subject of *decide* will force the reading in which the chair is a potential mental agent. The other reading is simply unavailable given these specifications (see previous note for discussion).[9]

We are deliberately vague about the details of the grammatical specifications that necessitate the identification of the subject's features and those of the verb's subject specification, since these vary rather a lot depending on one's grammatical assumptions. For the sake of concreteness we sketch how the identification would be effected in HPSG (Pollard and Sag 1994). In HPSG the subcategorization principle would ensure that the **VP** would assume the undischarged (subject) valence requirements of its lexical head **V**, and a grammatical rule (Schema 1) would allow the sentential node only where the **NP** could be identified with the **VP**'s subject specifications. The relation between the **NP** and its head **N** is analyzed variously, and surely not many researchers would simple identify the semantic specifications of the two (preferring perhaps to view the semantics of the **N** as a predicate, and that of the **NP** as a generalized quantifier). But that is not essential, only that the sortal information associated with the noun somehow continue on to the **NP**, and that this be required to unify with the verb's subject specifications. In the examples below we assume that such a requirement is enforced without providing the specifications on rules, etc., which would effect it.

Suppose it is a genuine reading, and therefore something which should be accounted for linguistically. Then it would seem wrong to provide a *linguistic* account of its ill-formedness, as we are about to do. We do not simply embrace this conclusion, because it seems equally plausible that what happens here is that the "normal" linguistic construal of furniture as non-agentive is put aside. A deeper reason for not caring whether this attribution is, strictly speaking, linguistic, is that we shall want to represent *all* the information need to support linguistic processing, and if this leads us into borderline areas, so be it.

[9]The treatment suggested above may be extended to be compatible with the existence of a complex hierarchy of sorts of the kind found in knowledge-representation systems. Some further details may be found in Nerbonne 1992b, but we shall omit them here.

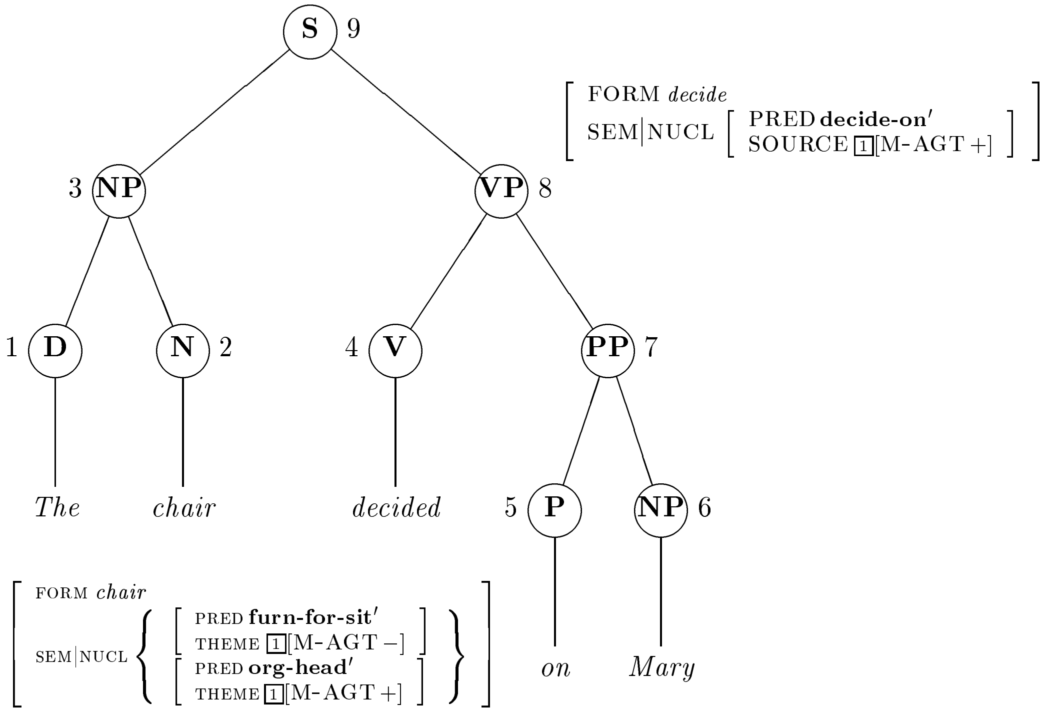


Figure 4: In (pure) bottom-up processing, no parent node is processed until all of its daughters are. The coindexing noted as '$\boxed{1}$' is compatible only with the second, lower construal of the word *chair*. In bottom-up processing the alternative, incompatible reading cannot be noted (and rejected) until the sentence level (given standard assumptions about constituent structure). The traversal indicated above by the numerical node annotations is the optimally incremental bottom-up traversal of the example sentence *The chair decided on Mary*. Since it is effectively the sentence node at which subject-verb agreement is enforced, including agreement of semantic selectional restrictions, there is no way to reject the senseless reading until the entire sentence is processed. Given the strong preference for right-branching structures in grammar, purely bottom-up processing can have no account of incremental understanding.

The purpose of developing this (perhaps overly brief) treatment of selectional restrictions is to provide an example to illustrate the consequences of some assumptions about semantic processing. The incompatibility of the furniture reading of *chair* with the mental agency required of subjects of *decide* must be enforced in any correct processing scheme—but in incremental schemes, it must be enforced at the point at which the word *decide* is encountered.

Figure 4 illustrates why purely bottom-up processing cannot be incremental (at least for standard grammars).

Since the meaning of a phrase depends on its daughters' meanings (compositionality), there cannot be a (final) computation of phrasal meaning before the all the daughters' meanings have been processed. One could imagine an argument for bottom-up evaluation proceeding from lingistic compositionality in this way. But several escape routes open before this conclusion is reached. The two most prominent ones are (i) currying and (ii) using partial specifications. If one curries an n-place function, one expresses it as a one-place function whose values are n-1-place functions. Thus, even if the value of the function cannot be determined in general, there will be circumstances in which the curried function can usefully be applied. In this paper we shall focus on the other possibility, however, that of using the partial specifications which are the trademark of feature-based theories. Thus, even though we cannot derive a complete semantics for a sentence before processing the entire VP, we shall be able to derive some properties of the sentential semantics early. In particular, we will be able to exclude some hypotheses about subject meanings based only on the verb—without waiting for the VP meaning to be computed. (We could in a parallel way illustrate how some hypotheses about verb meaning may be excluded on the basis of subject meaning, but this would add little.) Thus the programmatic point of this section is that computation with partial descriptions of semantics—as is common in feature-based theories—provides a basis for explaining the possibility of incremental understanding.

Steedman 1987 (and elsewhere) and Haddock 1988 have championed an approach to incremental interpretation which allows processing to be both incremental AND bottom-up. This approach has come to be known as FLEXIBLE CATEGORIAL GRAMMAR, (hence: flexible CG) and an illustration of its application to our example may be found in Figure 5.[10] The key to Steedman's solution lies in the wholesale abandonment of standard assumptions about constituent structure (see example), which Steedman takes to be rigorously left-branching in order to reconcile bottom-up and incremental processing. Since the syntax is completely left-branching, EVERY initial segment is a constituent. In fact, as Steedman 1987 argues, under the "strong competence hypothesis" of Bresnan 1982—the assumption of parallelism between linguistic structure and processing, the left-branching structure would appear to be necessary.

The flexible CG analyses have attracted attention because of their application to difficult problems of the grammatical analysis of coordination, including gapping and right-node raising (Steedman 1990). Here they have had some success, especially compared to other frameworks. But several aspects of the overall position remaining unconvincing. First, the "flexibility" promised by the approach cuts both ways—even if it seems appealing to posit alternative constituent structures for coordination (which motivated a great deal of the work in question), it seems difficult to accept the wholesale ambiguity in constituent structure which is the consequence of the flexibility. It is trivial to find sentences in which some initial segments would constitute ridiculous constituents (the node marked '?B' in Figure 5 is only suggestive—much less plausible examples are readily constructed). The postulation of such constituents is incompatible with most syntactic theory, which relies crucially on constituent structure to account for a great deal of syntax—e.g., word

[10]See Jacobson (this volume) for more on semantics and categorial grammar.

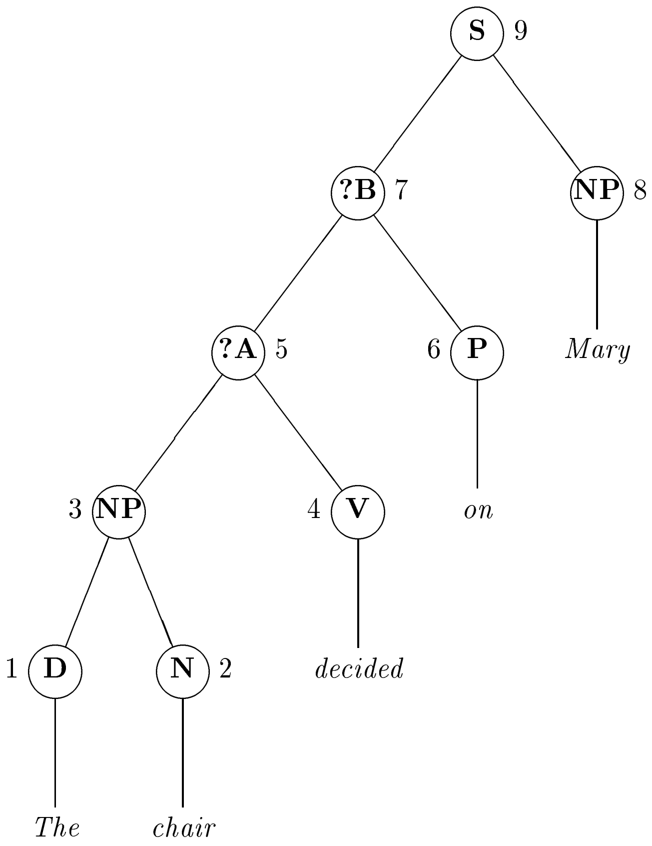


Figure 5: The flexible categorial grammar solution to incremental, bottom-up processing. Since no parent node is processed until all of its daughters are, the scheme indicated by the numerical traversal annotations is bottom-up. Since the grammar is left-associative, the processing can also be incremental. This solution is formally sound, but its linguistic assumptions are heterodox. It is safe to say that *no* evidence exists for most of the novel constituents posited on the left-associative view. If, on the other hand, the view is tenable, it indicates that constituent structure—the primary explanatory device in syntax—is relatively insignificant.

order, long-distance dependence, island constraints, restricitions on anaphoric relations ("binding theory"). For most syntacticians, constituent structure is the primary explanatory mechanism, so many syntactic explanations are lost if constituency is simply left-associative. Second, even within CG, the use of calucli flexible to this degree is controversial: Houtman 1994, pp.74-90 devotes most of a chapter to "Arguments against Flexibility" (focusing his attack, however, not on the combinatory grammars which Steedman advocates, but rather on Lambek-style CG). It is controversial not only because it fails to account for the coordination phenomena as generally as it was originally claimed to (Houtman 1994), but also because it leads to a counterintuitive collapse of categories in the analysis of adjuncts ("Dekker's paradox", discussed by Houtman, pp.85-89). Third, and perhaps less directly applicable to the point at hand, the flexibility in this approach is now overused. Joshi 1990 notes that "flexibility" is variously deployed, not only to enable incremental, bottom-up processing, but also to allow constituent structure to match intonational grouping, and to allow constituent structure to match coordination. These three requirements are certainly not simultaneously satisfiable. (This point is less directly applicable because it would be conceivable to focus only on the left-associative treatments, ignoring other applications of flexibility. But this would rob flexible CG of its most impressive success, in the analysis of coordination.)

Shieber and Johnson 1993 contains a rather different view of the work on incremental understanding in flexible CG, criticizing that its conclusions follow only under the implicit assumption of synchronous computation of semantics. They claim that incremental interpretation follows not from grammatical structure (the flexible CG position), nor even from the control structure of particular algorithms (the position to be illustrated below), but rather from the asynchronous nature of understanding. While we find this alternative interesting psychologically, we take it that an algorithmic solution remains of interest, at least technically. We turn to this now.

It seems best not to explain incremental understanding on the basis of grammatical structure, but rather to ask how might incremental understanding be possible if grammatical structure is roughly as we know it (in fact, mostly right-branching). Given our construal of understanding as parallel to parsing, we have a rich choice of incremental processing algorithms. To illustrate how the constraint-based view of semantics processing enables incremental semantic evaluation, we need only choose one. A popular choice for incremental parsing is (predictive) LEFT-CORNER (hence: LC) parsing. It is popular because it makes use of both bottom-up (lexical) and also top-down (grammatical) information.

An LC parser may be viewed as a process which constantly attempts to extend the analysis of an initial segment ("left corner") of a string.[11] The LC parser begins

[11]Pereira and Shieber 1987, pp.178-82 describes a nonpredictive LC parser. Cf. note 6 for general literature on parsing. The parser and grammar (slightly modified for presentation) described in this section were implemented by Ronald Klopstra in his Groningen Master's Thesis (Klopstra 1993).

with the assumption that the empty string has an analysis as the initial segment of **S** (the category to be found), and then reads words. After reading each word, it fills in as much of the analysis as is necessary to verify that there is a way of filling in the analysis tree from **S** to the segment, i.e. that the initial segment is possible in the language. The algorithm is easiest to understand as a mutual recursion between the following pair of routines:

```
proc parse(α:nonterminal-category);
    begin
        read next word W;
        CAT ← CAT(W);
        if CAT = α
          then SUCCESS
          else if ◇-lc(CAT,α)
                  then expand-lc(CAT,α)
                  else FAIL
               endif
        endif
    end parse

proc expand-lc(CAT,goal)
    begin
        for-each rule β → CAT γ
          if ◇-lc(CAT,goal)
               then parse(γ);
                  if end-of-input
                     then SUCCESS
                     else expand-lc(β,goal)
                  endif
          endif
        end-for-each
    end expand-lc
```

We omit the definition of the relation '◇-lc', which holds between a top category and a bottom category intuitively just in case there is path from the top to the bottom along the leftmost categories in (the right-hand) sides of grammar rules. Thus, in a grammar with **S** → **NP VP**, **NP** → **Det N**, the relation would hold of the pair (**S**,**Det**), and trivially of the pairs (**S**,**NP**) and (**NP**,**Det**).

To appreciate the workings of the LC parser, let's fill out the grammar just mentioned by adding the usual **VP** → **V NP** rule, and the lexical items *the* (**Det**), *saw* (**V**), and *child* and *toy* (**N**). Then the table below traces the execution of the LC parser on this grammar on the input string *The child saw the toy.* We seek an **S**, and therefore invoke parse(**S**). The table below provides a trace of the execution:

		parse			expand-lc		
step	reader	α	word	CAT	CAT	goal	rule
1	$\uparrow$ *The child saw the toy*	**S**	*The*	**Det**			
2	*The* $\uparrow$ *child saw the toy*				**Det**	**S**	**NP** $\rightarrow$ **Det N**
3	*The* $\uparrow$ *child saw the toy*	**N**	*child*	**N**			
4	*The child* $\uparrow$ *saw the toy*				**NP**	**S**	**S** $\rightarrow$ **NP VP**
5	*The child* $\uparrow$ *saw the toy*	**VP**	*saw*	**V**			
6	*The child saw* $\uparrow$ *the toy*				**V**	**VP**	**VP** $\rightarrow$ **V NP**
7	*The child saw* $\uparrow$ *the toy*	**NP**	*the*	**Det**			
8	*The child saw the* $\uparrow$ *toy*				**Det**	**NP**	**NP** $\rightarrow$ **Det N**
9	*The child saw the* $\uparrow$ *toy*	**N**	*toy*	**N**			

The table fills in cells only where the procedure is active. Thus, at Step 1, values are supplied for the local variables in the `parse` routine, but not in the routine `expand-lc`, which has not been called. At Step 2, `expand-lc` is executing so its variables are filled in, and `parse`'s variable are omitted.

The table provides snapshots of the execution of the routines. As `parse` is first called (with argument **S**), nothing has been read. We picture the sentence being processed as having a pointer indicating how much has been read. So initially, the pointer is before the first word:

$\uparrow$ *The child saw the toy*

This is reflected in the table. Since `parse` reads the first word, the pointer is advanced before the following word is read. For this reason, the reader in the second row of the table shows the pointer after the first word. And in general, whenever `parse` executes, there is an advance of the pointer.

Once the word has been read (*The*), and its category determined (**Det**), `parse` notes that there is no match between the category determined and the goal α (bound to **S**). Therefore, `expand-left-corner` is called (line 2 of the table), with parameters CAT and goal bound to **Det** and **S**. `expand-lc` hypothesizes about rules with LC **Det** (CAT), and we show the result of hypothesizing with the rule **NP** $\rightarrow$ **Det N**. Since **Det** is a possible left-corner of **S**, `parse` is recursively called (line 3), now with the goal parameter bound to **N**. In this case, the category matches, and control is passed back to `expand-lc`, which calls itself, this time with arguments **NP** and **S**. Given this much explanation, the rest of the trace should be fairly straightforward to follow. The rule column of the `expand-lc` part of the table is interested because it is there one can see how the structure for the initial segment is being built up as the parse progresses.

This procedure is indeed incremental in the relevant sense—if there is no possible continuation of an initial segment, processing will fail immediately, without continuing. This can be verified by attempting to execute the procedures on such input, for example, the string *The the child saw ...* This would procedure exactly as the execution in the table, but would fail at step 3—on determining that **Det** is not a possbile left corner of **N**.[12]

Figure 6 shows an application of a left-corner traversal to the sentence used as an example of the utility of incremental processing. To make the illustration watertight, the LC routines above should be modified to process not only the category information on nodes, but also the feature decorations associated with them. It is essential that these too be checked as the parse progress. If this is done, then the value of the subject's feature [SEM|NUCL|THEME|M-AGT ±] will be constrained to be identical both to the verb's subject specification, which in turn will be lexically constrained to be + in the case of *decide*, just as in Figure 4. This filters the furniture sense of *chair* from further consideration.

As long as the feature specifications associated with the rules are being checked as the rules are, then then objectionable analysis must be detected at the time the verb is read (more exactly, once its feature specifications are processed).

4 Conclusions

While linguistics characterizes grammar—the relation between meaning and form, computational linguistics focuses on processing, e.g., the algorithms and data structures needed to compute meaning given form (parsing), or form given meaning (generation), etc. Of course work in one field may have ramifications in the other.

Given this division of labor, it is natural to view grammar as a set of constraints on the meaning-form relation—whose computation is to be considered independently. The present paper focuses on consequences of this view for the syntax-semantics interface and the processing of semantic information. The interface topic is, properly speaking, linguistic, and was developed to illustrate the dialectic between grammar and processing which computational linguistics encourages. The processing topic is computational, and the present contribution advocates an approach which minimizes linguistic assumptions. In particular, it is argued that we

[12]The careful reader will not that this will end one execution path, using the hypothesis that the initial *The* is licensed by the **NP** $\rightarrow$ **Det N**, and that in principle all the rules may be tried. But of course none of the other rules have **Det** as a leftmost category on the righthand side. So this does indeed end the parse (in failure).

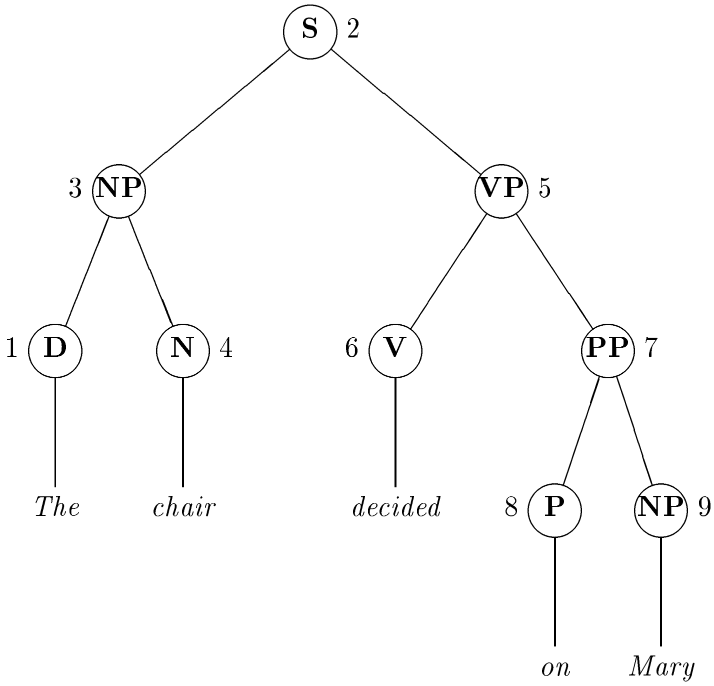


Figure 6: In left-corner processing, a parse is sought by continually seeking to extend the analysis of an initial segment of the string to be parsed (prefix). This leads to a complex traversal, allowing information to flow both bottom-up and top-down (see text for details). The processing is incremental in Schabes's sense, since impossible prefixes must be recognized as such. The traversal indicated above by the numerical node annotations is the record of the first visits to the node by the parsing traversal. Since subject properties are available for computation when the verb is visited, subject-verb agreement can be enforced at the earliest possible moment, including agreement of semantic selectional restrictions. Thus senseless readings may be rejected even before the entire sentence is processed. Given the strong preference for right-branching structures in grammar, (somewhat) elaborate processing models seem necessary.

need not assume that grammar is left-associative in order to account for incremental understanding. This can arise using standard processing techniques, as long as semantic processing consists of collecting and resolving constraints on semantic representation.

References

Bouma, G., and G. van Noord. 1993. Head-Driven Parsing for Lexicalist Grammars: Experimental Results. In *Proc. of 6th European ACL*, 71–80. Utrecht.

Bresnan, J. (ed.). 1982. *The Mental Representation of Grammatical Relations*. Cambridge, Mass.: MIT Press.

Carpenter, B. 1992. *The Logic of Typed Feature Structures*. No. 32Tracts in Theoretical Computer Science. Cambridge: Cambridge University Press.

Dalrymple, M., S. M. Shieber, and F. C.N.Periera. 1991. Ellipsis and Higher-Order Unification. *Linguistics and Philosophy* 14(4):399–452.

Davis, A. 1995. *Linking and the Hierarchical Lexicon*. PhD thesis, Stanford University.

Dowty, D. 1979. *Word Meaning and Montague Grammar*. Reidel: Reidel.

Fenstad, J. E., P.-K. Halvorsen, T. Langholm, and J. van Benthem. 1987. *Situations, Language, and Logic*. Dordrecht: Reidel.

Flickinger, D., and J. Nerbonne. 1992. Inheritance and Complementation: A Case Study of *Easy* Adjectives and Related Nouns. *Computational Linguistics* 19(3):269–309.

Gale, W., K. Church, and D. Yarowsky. 1992. A Method for Disambiguating Word Senses in a Large Corpus. *Computers and the Humanities* 26(5–6):415–440.

Gazdar, G., E. Klein, G. Pullum, and I. Sag. 1985. *Generalized Phrase Structure Grammar*. Harvard University Press.

Haddock, N. 1988. *Incremental Semantics and Interactive Syntactic Processing*. PhD thesis, Edinburgh.

Halvorsen, P.-K. 1988. Situation Semantics and Semantic Interpretation in Constraint-Based Grammars. In *Proc. of the International Conference on Fifth Generation Computer Systems*, Vol. 2, 471–478. Tokyo. Institute for New Generation Systems.

Houtman, J. 1994. *Coordination and Constituency: A Study in Categorial Grammar*. PhD thesis, Rijksuniversiteit Groningen.

Jacobson, P. n.d. The Syntax/Semantics Interface in Categorial Grammar. (this volume).

Joshi, A. K. 1990. Phrase structure and Intonational Phrases: Comments on the Chapters by Marcus and Steedman. In *Cognitive Models of Speech Processing*, ed. G. T. M. Altmann, 513–531. Cambridge: MIT Press.

Kahn, K. M. 1987. Partial Evaluation as an Example of the Relation between Programming Methodology and Artificial Intelligence. In *Artificial Intelligence Programming Environments*, ed. R. Hawley, 131–142. Chichester: Ellis Horwood.

Kaplan, R. 1987. Three Seductions of Computational Psycholinguistics. In *Linguistic Theory and Computer Applications*, ed. P. Whitelock, M. M. Wood, H. Somers, R. Johnson, and P. Bennett, 149–188. London: Academic Press.

Kasper, R. T. 1994. Adjuncts in the Mittelfeld. In *German Grammar in HPSG*, ed. J. Nerbonne, K. Netter, and C. Pollard, 39–69. Stanford. CSLI.

Kasper, R. T., and W. C. Rounds. 1986. A Logical Semantics for Feature Structures. In *Proc. of the 24th Annual Meeting of the Association for Computational Linguistics*, 257–266. Columbia University.

Klopstra, R. 1993. Incremental Interpretation. Master's thesis, Alfa-informatica, Rijksuniversiteit Groningen. also BCN poster, Theme Day 2/94.

Mitchell, D. C. 1994. Sentence Parsing. In *Handbook of Psycholinguistics*, ed. M. A. Gernsbacher, 375–409. Orlando: Academic Press.

Moore, R. C. 1989. Unification-Based Semantic Interpretation. In *Proc. of the 27th Annual Meeting of the Association for Computational Linguistics*, 33–41.

Nerbonne, J. 1992a. Constraint-Based Semantics. In *Proc. of the 8th Amsterdam Colloquium*, ed. P. Dekker and M. Stokhof, 425–444. Institute for Logic, Language and Computation. also DFKI RR-92-18.

Nerbonne, J. 1992b. Natural Language Disambiguation and Taxonomic Reasoning. In *DFKI Workshop on Taxonomic Reasoning (DFKI D-92-08)*, ed. J. Heinsohn and B. Hollunder, 40–47. Saarbrücken. DFKI.

Nerbonne, J. 1993. A Feature-Based Syntax/Semantics Interface. *Annals of Mathematics and Artificial Intelligence* 107–132. Proc. of the Second International Conference on the Mathematics of Language, ed. by A. Manaster-Ramer and W. Zadrozny.

Nerbonne, J. 1994. Review of M.Rosner and R.Johnson (eds.) *Computational Linguistics and Formal Semantics*. *Computational Linguistics* 20(1):131–136.

Nerbonne, J., J. Laubsch, A. K. Diagne, and S. Oepen. 1993. Software for Applied Semantics. In *Proc. of Pacific Asia Conference on Formal and Computational Linguistics*, ed. C.-R. Huang, C. H. hui Chang, K. jiann Chen, and C.-H. Liu, 35–56. Taipei. Academica Sinica. Also available as DFKI Research Report RR-92-55.

Pereira, F. C., and S. Shieber. 1987. *Prolog and Natural Language Analysis*. Stanford: CSLI.

Pollard, C., and I. Sag. 1987. *Information-Based Syntax and Semantics, Vol.I.* Stanford: CSLI.

Pollard, C., and I. Sag. 1994. *Head-Driven Phrase Structure Grammar.* Stanford: CSLI.

Rosner, M., and R. Johnson. 1992. *Computational Linguistics and Formal Semantics.* Cambridge: Cambridge University Press.

Schabes, Y. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars.* PhD thesis, University of Pennsylvania.

Shieber, S. 1986. *An Introduction to Unification-Based Approaches to Grammar.* Stanford University: Center for the Study of Language and Information.

Shieber, S., and M. Johnson. 1993. Variations on Incremental Interpretation. *Journal of Psycholinguistic Research* 22(2):287–318.

Shieber, S., H. Uszkoreit, F. Pereira, J. Robinson, and M. Tyson. 1983. The Formalism and Implementation of PATR-II. In *Research on Interactive Acquisition and Use of Knowledge.* Menlo Park, Calif.: Artificial Intelligence Center, SRI International.

Sikkel, K. 1993. *Parsing Schemata.* PhD thesis, University of Twente.

Stabler, E. P. 1992. *The Logical Approach to Syntax.* Cambridge: MIT Press.

Steedman, M. 1987. Combinatory Grammars and Human Language Processing. In *Modularity in Knowledge Representation and Natural-Language Understanding*, ed. J. Garfield, 187–205. Cambridge: MIT Press.

Steedman, M. 1990. Gapping as Constituent Coordination. *Linguistics and Philosophy* 13(2):207–263.

Wilks, Y. 1975. A preferential, pattern-seeking semantics for for natural language inference. *Artificial Intelligence* 6:53–74.

Zadrozny, W. 1994. From Compositional to Systematic Semantics. *Linguistics and Philosophy* 17(4):329–342.