6

Robust Chart Parsing with Mildly Inconsistent Feature Structures

Carl Vogel and Robin Cooper

Abstract

We introduce the formal underpinnings of our theory of non-classical feature structures. The resulting expanded universe of feature structures has direct implications for robust parsing for linguistic theories founded upon feature theory. We present an implementation of a robust chart parser for Head-driven Phrase Structure Grammar (HPSG). The problem of relaxed unification is in limiting it so that arbitrarily nongrammatical inputs needn't be accepted. In the worst case, excessively 'relaxed' parsers cannot provide meaningful interpretations to process. However, parsers which can guarantee minimally nongrammatical (inconsistent) interpretations provide an important tool for grammar development and online robust processing of natural language. Our parser prefers maximally consistent interpretations, and accommodates inconsistencies by discovering the minimal set of constraints that need to be relaxed for a parse to go through, without employing backtracking or post processing. The system is different from other related relaxational techniques for unification grammars which require advanced naming of features whose constraints are allowed to be relaxed. Yet it is compatible with those approaches in that there is a well defined location for preferences on sources of inconsistency to be named, as well as for resolution of inconsistent information. We use a simple approach to the problem of unknown words and suggest generalization of that for coping with missing and extra elements in an input.

1 Background

We apply a generalized theory of feature structures to robust processing of natural language. Feature structures and Typed feature structures are well understood (see Aït-Kaci, 1984; Johnson, 1987; Nebel & Smolka, 1989; Carpenter, 1992) mathematical objects related to frames in knowledge representation. Indeed, feature structures form the basis of languages for

Edinburgh Working Papers in Cognitive Science, Vol. 10: *Nonclassical Feature Systems*, pp. 197–216.
Andreas Schöter and Carl Vogel, eds.

representation and reasoning (Aït-Kaci & Nasr, 1986; Aït-Kaci & Lincoln, 1988; Carpenter, 1993) as well as underlying a dominant theoretical framework in contemporary linguistics. Our research program is to enlarge the class of objects available for formalizing analyses by incorporating limited inconsistency into feature structures. This has important implications for admitting a theory of grammatical degree into linguistic theory whose formal foundations rely on typed feature structures, and also for implemented systems based on this framework. Inconsistency in a feature structure and its description represents a point of nongrammaticality in the sentence under analysis. However, there is worry that once inconsistency is accepted in an analysis, any string will parse, and further that there will be no efficient way to separate consistent analyses from inconsistent ones. In this paper we demonstrate that inconsistency can be admitted in a controlled fashion, in a way that does not grind the parser to a halt. In fact, we achieve a novel and principled approach to robust parsing. We present a parser based on our theory of feature structures and give an overview of the theory of inconsistent feature structures as well.

A number of related taxonomies of ill-formed input to linguistic processors have been proposed (Hayes & Mouradian, 1981; Carbonell & Hayes, 1983; Douglas & Dale, 1992). For instance, Figure 1 gives the taxonomy and a subset of the examples used by Douglas and Dale (1992, p.469). Several extant systems address a range of potential ill-formedness (Weischedel & Black, 1980; Hayes & Mouradian, 1981; Weischedel & Sondheimer, 1983; Jensen, Heidorn, Miller, & Ravin, 1983; Mellish, 1989; Douglas & Dale, 1992); for example, in the terms shown in Figure 1, Douglas and Dale (1992) focus on constraint violation errors while Mellish (1989) concentrates on missing and extra elements.

(1)	**Constraint Violation Errors:**
	a. Subject-verb number disagreement: $*$*The dogs runs.*
	b. Premodifier-noun number disagreement: $*$*This dogs runs.*
	c. Subject-complement number disagreement: $*$*There is five dogs here.*
	d. Wrong pronoun case: $*$*This stays between you and I.*
	e. Wrong indefinite article: $*$*A apple and an rotten old pear.*
(2)	**Lexical Confusion.**
(3)	**Syntactic Awkwardness.**
(4)	**Missing or extra elements.**

Figure 1: Examples of Syntactic Errors

However, much of this research is tuned to specific (syntactic or semantic) grammars for natural language rather than at the general level of grammar formalisms; Weischedel and Black (1980), Hayes and Mouradian (1981), Carbonell and Hayes (1983), Weischedel and Sondheimer (1983), and Jensen et al. (1983) offer proposals that fall into the former category, while Mellish's (1989) system is in the latter. A number of systems in the former category are called *relaxational* for their system of relaxing constraints such as number or case agreement; however, the constraints that may be relaxed must be identified in the grammar itself, and

this process is often invoked only in backtracking and at the expense of repeated processing of exactly the same computations.

Mellish (1989) uses a two-step process based on an active chart parser for unaugmented context free grammar to process a variety of possible errors. He first uses a standard bottom-up parse and invokes error recovery only if no complete parse is available for the entire sentence. In that case his system invokes a modified top-down parser which identifies edges that can be tweaked in order to obtain an interpretation for the input. Mellish's system is very general; it copes with word deletion, word addition, and unknown words assuming any context free grammar without empty productions. An advantage of his system is that the fully grammatical parses are identified without additional cost, though the trade-off is significant processing (though none of it repeated, due to the advantage taken of the well-formed substring table) during the error recovery stage of processing.

Douglas and Dale (1992) provide a relaxational approach using PATR the grammar specification formalism for unification grammars. Their 'Robust PATR' includes specification of 'relaxation packages'—sets of constraints that are allowed to be relaxed to get a parse through (the 'package' means that some constraints are coupled to others such that one should not be relaxed without relaxing the rest). The remainder are taken to be strict. Relaxation packages are associated with each PATR grammar rule individually and refer to constraints mentioned in the scope of a rule. This system is thus attuned to specific grammars, although articulated within the unification based framework. Douglas and Dale rely on backtracking when no complete parse is available and recompute with relaxed constraints.

We offer an alternative chart-based relaxational approach to processing ill-formed input that does not rely on two-stage processing. Our parser is an extension of the `HPSG-PL` system (Popowich & Vogel, 1991; Kodrič, Popowich, & Vogel, 1992) which is a Prolog based chart parser for grammars written in the Head Driven Phrase Structure Grammar (HPSG; Pollard & Sag, 1987, 1994) framework. `HPSG-PL` has been used as a serious linguistics testing bed for HPSG coverage of complex NPs in English (Popowich, McFetridge, Fass, & Hall, 1992) and also for VP and general constituency structure in Slovene (Kodrič, 1994). In this paper we outline our approach to ill-formedness arising from constraint violation. A treatment for unknown words follows directly from the technique used by Kodrič et al. (1992)[1], and for coverage of extra-word, missing-word errors we can ultimately appeal to Mellish's techniques or to an alternative post-processing technique related to our proposal for unknown words. Our approach, like Mellish's is general to the grammar framework. Although we could incorporate a mechanism for stipulating which constraints *must* be satisfied, our system identifies the minimal constraints that need to be relaxed in order to obtain a complete parse *without declaring them in the grammar.* This work is part of our wider research program to define a class of generalized feature structures which, unlike traditional approaches (cf. Aït-Kaci, 1984; Johnson, 1987; Carpenter, 1992), admit inconsistencies and therefore offer a mathematical framework in which to articulate a theory of grammatical degree. First we detail the extensions to the parser that achieve the desired robustness and then we show the formal generalization of feature structures which guides our research.

[1]In this treatment, there are lexical entries with unspecified phonology, one with an empty subcategorization frame, one which subcategorizes for one other item, and so on. Other properties are left uninstantiated in the lexicon and are estimated from the known constraints that the unknown object combines with.

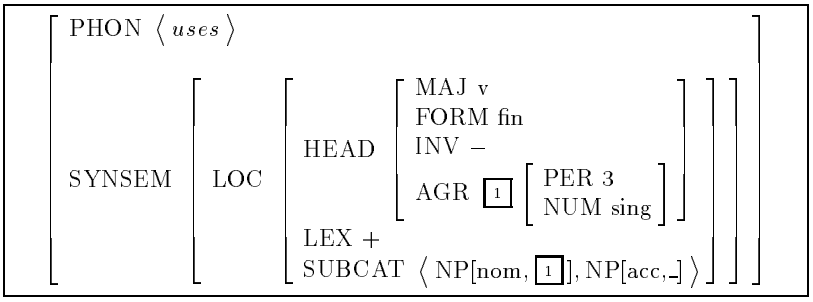

Figure 2: Part of an HPSG Sign for a Transitive Verb

2 A Robust Parser for Natural Language

2.1 Chart-Parsing Traditional Unification Grammars

We quickly review the key aspects of chart parsing methodology that are prerequisite to exposition of our modifications to the framework which achieve robust parsing. The idea is to maintain a *chart* of *edges* that represent well-formed substrings (edges have *category* and *span*) of a linguistic input and an *agenda* of edges to be processed with respect to the edges already in the chart and the grammar rules. Edges are either *active* or *inactive* depending on whether they have associated *expectations*. Expectations originate (in CFG terms) from the RHS of grammar rules appropriate for an object that is of that edge's *category*. A category corresponds to a terminal or nonterminal symbol in the grammar. The two main processing steps are *prediction*, which generates new active edges for the agenda from inactive edges in the chart by appealing to grammar rules, and *completion*, which generates active and inactive edges for the agenda by satisfying one of the expectations of an active edge that is found to be *adjacent* (adjacency may be given discontinuous interpretations) to an inactive edge of the expected category. An edge generated from completions spans the two constituent edges and is marked with the category of the active edge. Parsing is complete when an inactive edge spans the input. The methodology is indifferent to direction of processing; different modes of access to the agenda (e.g. FIFO, LIFO) achieve different directions of parsing.

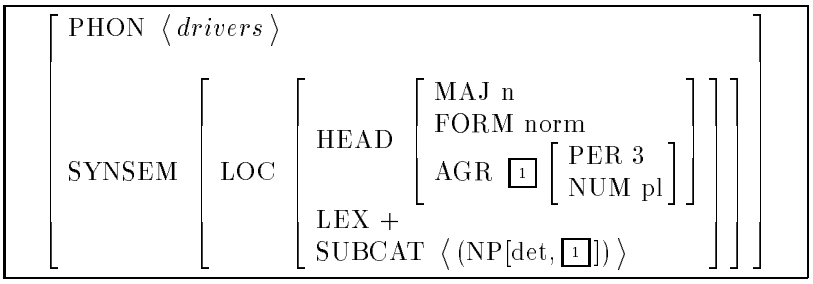

Figure 3: Part of an HPSG Sign for a Noun

Popowich and Vogel (1991) and Kodrič et al. (1992) present the `HPSG-PL` system which adapts the chart parsing methodology to unification grammar frameworks, particularly HPSG (Pollard & Sag, 1987, 1994). HPSG is a dominant framework in which formal research in

linguistics is conducted. It is a theory of syntax and semantics articulated within a mathematically well-understood feature structure formalism. The language of attribute-value matrices (AVMs) underneath HPSG can be understood in general terms as constraint satisfaction (cf. Carpenter, 1993). In `HPSG-PL`, edge categories are HPSG *signs*. Satisfaction of expectations is checked not with simple symbol matching but with unification. The properties of signs are those of total well-typed feature structure in Carpenter's (1992) sense. An example sign for the transitive verb *uses* is shown in Figure 2; co-indexing denotes structure sharing, and the items on the `SUBCAT` list (subcategorization requirements) are parameterized types that denote signs as well. The `HPSG-PL` algorithm for classical HPSG chart parsing is given below; it can terminate after the first spanning inactive edge is found or after all possible parses have been identified:

Data Structures.

- *Chart* of well-formed substrings consisting of *edges* with:
  - category – HPSG signs,
  - span,
  - potency – either *active* or *inactive* depending on *expectations.*
- *Agenda* of edges to be processed w.r.t. edges in the chart and grammar rules.

Processes.

- Category matching: sign unification.
- Prediction: generation of new active edges,
  - via appeal to appropriate rules (appropriateness calculated with category matching).
- Completion: generation of new active or inactive edges,
  - via combination of adjacent edges with compatible categories;
  - new edges span the constituent edges;
  - new edges are given the same category as the active edge in the complementary pair.
- Termination: quit when an inactive edge spans the input.

Chart Parsing for HPSG in `HPSG-PL`

The sign depicted in Figure 3, includes some of the information in lexical entries for plural nouns. In a full phrasal analysis of a sentence that includes both this lexical item and the lexical sign for *uses* from Figure 2, in which *uses* takes *drivers* as its subject (as in *the drivers uses the seatbelts*), the sign for *drivers* will be expected to unify with the nominative NP element of the verb's subcategorization list. The co-indexing indicates that the agreement features of the subject and the verb should be token identical. However, this is clearly not the case given those lexical entries: they have differing values for the `NUM` feature. This inconsistency, in classical treatments like Carpenter's (1992) means that unification fails. Thus, linguistic theory based on a classical formulation of feature structures is obliged to

analyze *all* nongrammaticalities as $\perp$ (read, "bottom"), the inconsistent feature structure.[2] This, of course, loses all the information, including the knowledge of which feature contained an inconsistent value in addition to the consistent information, that is actually available in a nongrammatical sentence. We prefer a treatment in which information can be represented and recovered from nongrammatical sentences like, *the drivers uses the seatbelts*, as depicted in Figure 4 with an HPSG phrasal sign with an inconsistency localized in the value of the root sign's NUM feature.

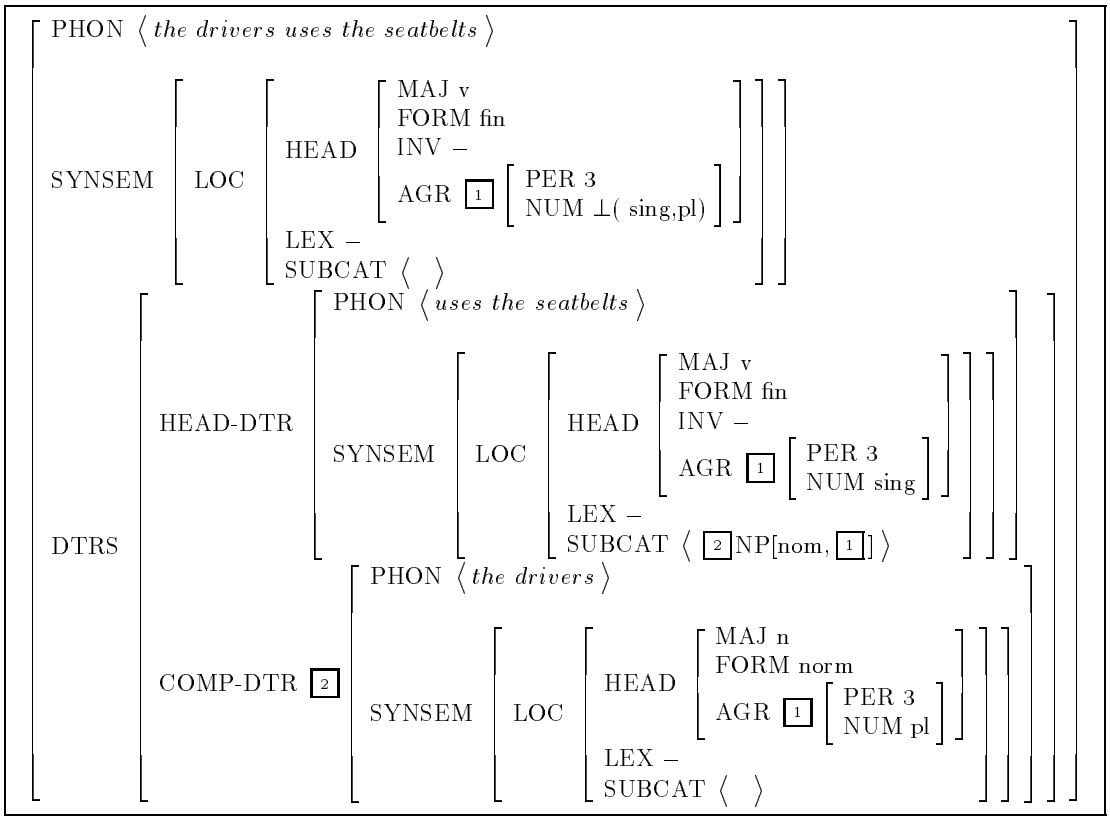

Figure 4: Part of a Mildly Inconsistent HPSG Phrasal Sign for a Sentence

2.2 Achieving Robust Processing

To obtain a more robust system from HPSG-PL we provide an additional, nonclassical unification algorithm for total feature structures which does not fail upon constant clash. Failure upon constant clash corresponds to *bottom smashing* in feature logics (cf. Carpenter, 1992). Instead, we maintain inconsistencies locally in resultant structures:

(5) $\quad nc\_unify([\texttt{NUM sing}],[\texttt{NUM pl}]) = [\texttt{NUM } \perp(\texttt{sing},\texttt{pl})]$

The symbol $\perp$ used as the name of a term which marks inconsistency. The unification of two feature structures may contain more than one inconsistency, and it is possible to

[2]Or, of course, as $\top$ ("top") depending on which way the subsumption hierarchy is specified!

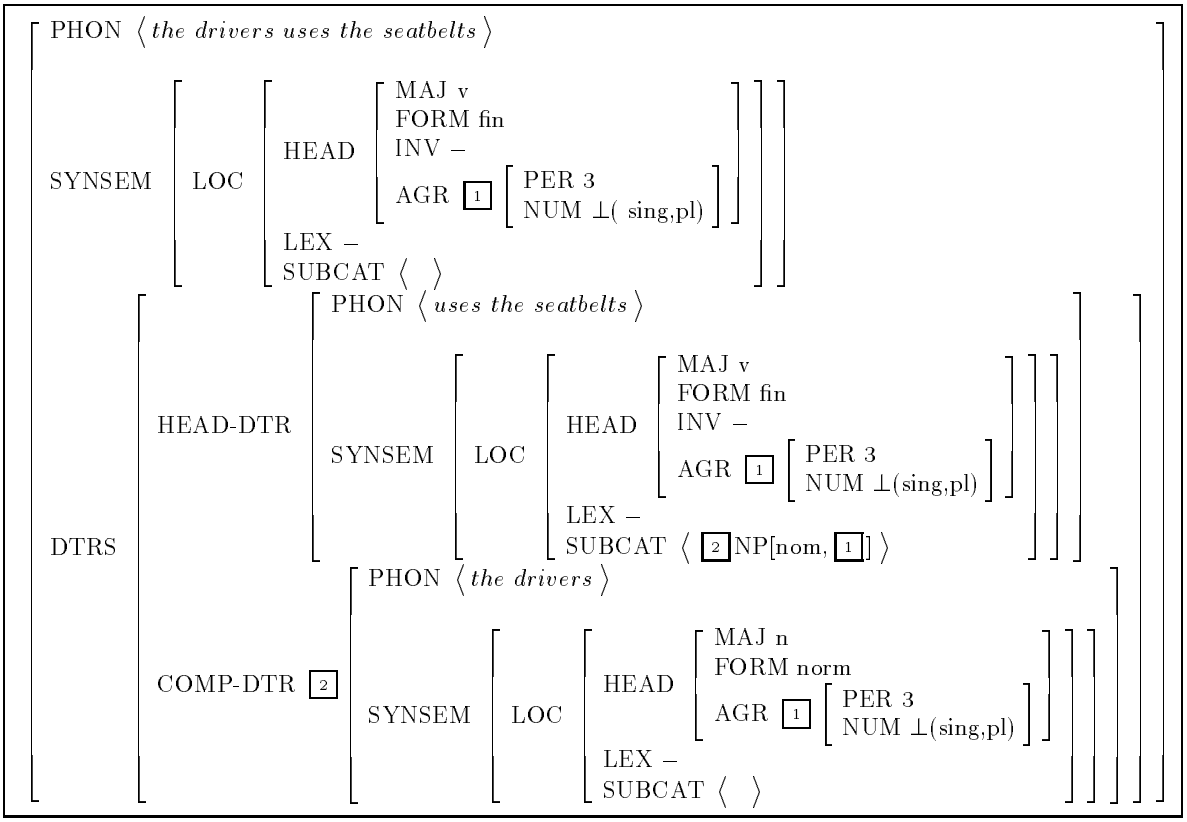

Figure 5: An Alternative Inconsistent HPSG Sign

define a hierarchy of consistency in terms of cardinality of bottom terms that they contain. This 'paraconsistent' unification is distinct from default unification which selects one of the conflicting values for the result (cf. Bouma, 1992; Grover, Brew, Manandhar, & Moens, 1994). Unlike many definitions of default unification (see Lascarides, Briscoe, Asher, and Copestake (1995) in this volume) this relaxed unification remains associative for nonreentrant structures; however, the class of nonreentrant feature structures is of limited enough practical interest for this to mean that relaxed unification is effectively nonassociative as well. Note that unlike Lascarides et al. (1995) we allow reentrancies to be overridden, in a sense, at the point where inconsistency enters. That is in the above example, although the `AGR` features on the full phrasal sign and the two daughters are coindexed, the daughters have distinct values that do not contain bottom terms: the agreement features on the VP are internally consistent, as are the agreement features on the NP, but both constituents indicate conflicting agreement features on the mother. At the same time, structure sharing is preserved among those values, albeit a hazy sharing of structure because of the different values. Clearly, this is a precarious position to assume since the logic of the resulting feature structures is not well understood. It is this handling of reentrancy that leads to nonassociativity, which may be an argument for always preserving reentrancy as Lascarides et al. (1995) do. However, our approach seems worthy of further investigation since it does preserve the consistency of information in the constituent structures, and thus separates the source of information from the landing site where inconsistency can occur. Understanding the full space of properties entailed by allowing

structure sharing to sustain different values is part of our future program. The second half of this paper presents the start of our work in formalizing these objects, including a formalization of the way inconsistent information is *indicated* by different sources (a weaker notion than inconsistency being supported by some individual situation). An alternative approach, using structure sharing in the well understood sense would propogate the inconsistency through all of the coindexed nodes, as depicted in Figure 5.

Our relaxed unification can fail to unify two feature structures: we still require graph morphism in the automata interpretation of feature structures. For instance, two list values of different lengths in otherwise identical feature structures will be sufficient to cause unification failure since that is not construable as a constant clash alone. In HPSG, this means that relaxed unification cannot alone accommodate missing or extra elements in the input as manifest in subcategorization list mismatches; we will discuss below two alternative methods of dealing in this framework with that type of ill-formedness. In any case, relaxed unification is accepting enough to make it prudent to restrict its use. Using relaxed unification in the predictor step would lead to spurious applicability of grammar rules,[3] so we restrict its use to the completion step where constraints on constituents and their complements are tested. This restriction guarantees termination in spite of the additional edges produced.

The revised system, when given an HPSG grammar and input sentence, will produce all of the edges that unaltered `HPSG-PL` would and additional edges reflecting the relaxed constraint verification that follows from our nonclassical unification. A revised specification of the parsing algorithm is given in Figure 6. We partition the agenda through which new edges flow into the *normal* agenda for consistent edges, and the *robust* agenda for those edges which result because of the relaxed unification. The later may be prioritized according to the degree of inconsistency in the sign that categorizes each edge (degree of inconsistency measured in terms of cardinality of $\perp$-terms in a feature structure). We appeal to edges in the robust agenda only when no consistent parse is available. This prevents inconsistency from propagating unnecessarily. We have also provided a parameter which may be set to retrieve all parses which are located in terms of increasing inconsistency. There is great potential for exploring heuristic strategies for sorting through the robust agenda, either with domain-informed priorities or linguistically motivated strategies, for example allowing preference for inconsistencies on agreement features over inconsistencies in major category. It is an advantage of our system that it does not require the advance naming of constraints that can be relaxed in order to obtain a parse, but it is consistent with those approaches (e.g. Douglas & Dale, 1992) in that preferences can be named and used as a prioritization metric on the robust agenda. However, we have so far explored only minimal inconsistency.

Douglas and Dale (1992) and Douglas (1995) utilize advance-naming of constraints to specify relaxation packages (of constraints that need to be relaxed together, if at all) as well as global constraints. Our approach can be extended to allow such constraints to be incorporated, adding some verification overhead, and probably obtaining an overall speed-up in processing. However in comparison to a strictly advanced-naming approach to constraint relaxation, our system is more easily integrated into a system designed to calculate priorities based on statistics like 'most often occurring inconsistency' obtained by processing novel sentences from

[3]Note though that as a lexicalized framework there are exceedingly few rules in an HPSG grammar—generally about 5 or 6—since the rules are basically just signs that declare immediate dominance restrictions with the main grammatical constraints expressed through universal principles and lexical entries.

corpora. An advance-naming system always has the potential to be faster than a system that discovers which constraints to relax, but since, as we just noted, it is possible to incorporate advance-naming into our system, and since there will always be novel sentences with previously unexperienced inconsistencies our approach is inherently more flexible.

Data Structures.

- *Chart* of edges with category, span, and potency, as before.
- *Agenda*, partioned:
  - *normal:* old agenda, consistent analyses.
  - *robust:* edges sanctioned by relaxed unification, potential for prioritization on the basis of degree of inconsistency, source of inconsistency, etc.
  - the idea: remove edges from the robust agenda for further processing only when the normal agenda is empty, and no spanning inactive edge is available.

Processes.

- Category matching: classical unification and relaxed unification.
- Prediction: generation of new active edges,
  - via appeal to appropriate rules (classical unification only).
- Completion: generation of new active or inactive edges.
  - via combination of adjacent edges with compatible categories (relaxed unification).
- Termination: quit when an inactive edge spans the input.

Figure 6: Robust Chart Parsing for HPSG.

In the grammar that we have been experimenting with, we obtain 2 parses for *The drivers uses the seatbelts*: the first takes the string as a sentence that has an inconsistency in number agreement. The second takes the string as a noun phrase in which the second *the* is seen as an inconsistent sort of NP complement to the verb *uses* and the 'sentence' *The drivers uses the* is taken as an inconsistent sort of determiner for *seatbelts*. For practical use of the system it is thus sensible to terminate with minimal inconsistency. However, alternative priority orderings on the robust agenda clearly would also have yielded similar results in this case. For instance, if inconsistencies on agreement are preferred over inconsistencies on major category, then the same minimally inconsistent parse would have been identified first. Minimal inconsistency is a general prioritizing strategy that will certainly have overlap with a sensible grammar-driven strategy. In fact, exploration of the overlap is an important task from the perspective of identifying grammar evaluation metrics.

Our system makes only one pass through the input without backtracking. This works because we use relaxed unification for category matching during the completion step, and relaxed unification includes classical unifications as a subset. Therefore, it is not necessary to compute classical unifications, and then invoke relaxed unification upon classical failure. Resort to inconsistency does not entail recomputing consistency checks because the paraconsistent edges are constructed and stored on the original pass through the sentence; it is only propagated inconsistency that waits until no consistent parse is identifiable, and this simply results in

additional edges being admitted to the chart and their consequences followed through before more edges are taken from the robust agenda. Thus we obtain robust relaxational processing of constraint violation errors. The cost, due to the more complex unification than in a non-robust system, is slower processing during parses that contain no constraint violations, but an inconsistent unification takes no longer to recognize than a consistent one.[4]

As mentioned earlier, we can adopt the treatment of unknown words from Kodrič et al. (1992), and we can also adopt Mellish's (1989) post-processing techniques to obtain robust coverage of missing and extra items. The treatment of unknown words offered by Kodrič et al. (1992) in HPSG-PL postulates lexical entries with unspecified phonology, one with an empty subcategorization frame, one which subcategorizes for one other item, one which subcategorizes for two items and one which subcategorizes for three items. These entries can exist up to arbitrary subcategorization; however, there are precious few English words that subcategorize for more than three constituents. When an input is processed that is not represented in the lexicon, the underspecified lexical entries are selected for processing and instantiated with the phonology of the unknown lexical item in the input string. Other properties on those underspecified items are also left uninstantiated in the lexicon, to be later estimated from the known constraints that the unknown object combines with.

It is possible to further generalize this lexicalized approach to unknown words to a treatment of the problem of missing elements as well. This would require an even more flexible unification operation which matches lists of different length to form a set of possible feature structures:

$$f\_unify(\langle \texttt{NP[nom]}, \texttt{NP[acc]} \rangle, \langle X \rangle) \quad = \quad \{\langle \texttt{NP[nom]} \rangle, \langle \texttt{NP[acc]} \rangle\}$$

Obviously such an operation would have to be extremely limited in its general application, but applied to missing and extra elements it could be used to unify lexical entries from the input with the underspecified elements in the lexicon invoked for unknown words to produce new structures with altered subcategorization requirements. These new signs can be constructed and used when edge based activity in the chart fails to produce a spanning inactive edge of the desired category of phrasal analysis for the input string. In this way, a transitive verb can be reprocessed as an intransitive verb, although semantic information reentrant with the element on the SUBCAT in the original lexical item will be retained. Thus, post-processing will still have information to process to conclude from interpretation of *The drivers use*, that there exists something which the drivers use. However, we have not implemented an extension like this in our system. It should be emphasized that this is a post-processing technique and lacks the sensitivity of Mellish's approach to correctly identifying potentially missing items. We propose the alternative mainly for its symmetry with our approach to constraint relaxation. It should also be noted that this bifurcation in ease of robust processing between constraint relaxation on constant-clashes and relaxation of subcategorization requirements is significant for a system that is to be seen as a potential model of psycholinguistic behaviors: as a psycholinguistic model, our system makes a concrete empirical prediction that number agreement errors should be 'easier' to process than subcategorization clashes.[5]

[4] Moreover, general recognition speedup is obtained by storing only the root sign information appropriate for the constituency represented by the edge as that edge's category: DTRS (daughter) information can be left implicit in the chart by also storing (recursively) pointers to the edges whose categories should be unified as constituents in the full sign for the parse.

[5] I am indebted to Holly Branigan for pointing this out to me, and for suggesting eyetracking experiments as a method of investigation.

3 Inconsistent Feature Structures

Here we sketch the formalization of nonclassical feature structures that guides our approach to the theory of robust linguistic processing. First consider the rooted directed graph representation of a feature structure; an example from Carpenter (1992, p.37) is given in Figure 7. For an example of formal definition of classical feature structures refer to Carpenter (1992, p.36) or Keller (1993, p.22). Subsumption in feature structures is defined in terms of graph morphisms, and unification is information conjunction. In a classical feature structures, unification of two feature structures gives the least upperbound if one exists. Figure 8 depicts how consistent feature structures can be composed into a paraconsistent one.

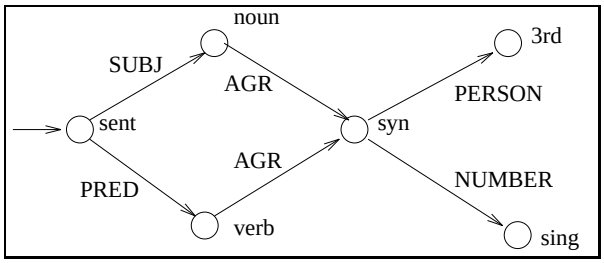

Figure 7: An Example Feature Structure

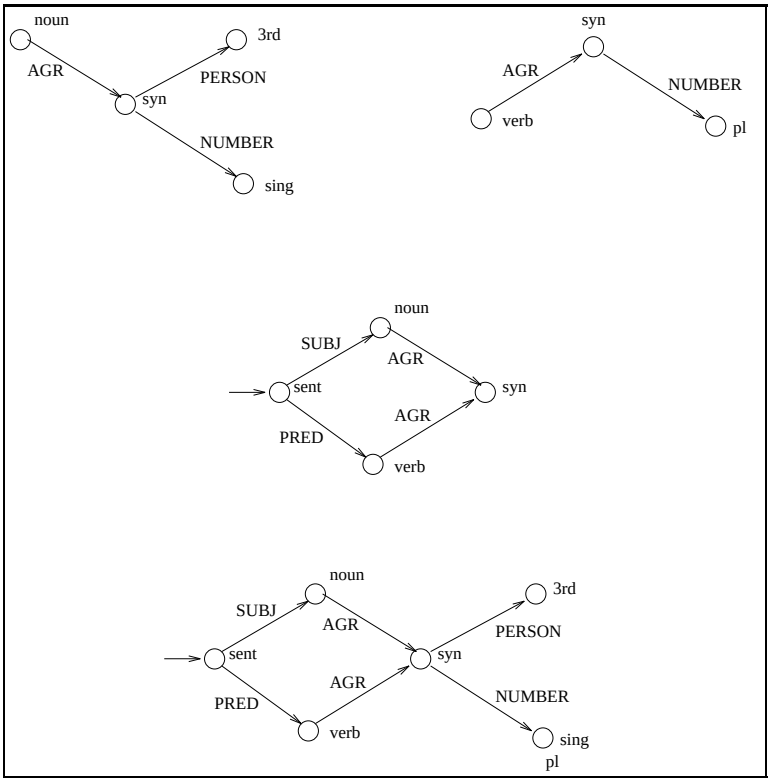

Figure 8: A Paraconsistent Feature Structure

We provide an alternative formulation using channel theoretic notions which make use of type/token distinctions in providing mathematical models for theories that invoke natural regularity (Barwise, 1991, 1993; Barwise & Seligman, 1992; Barwise, 1993). To relate our work to classical treatments, take nodes in feature graphs as tokens and node labels as types. Features are types supported by connections between nodes.

Definition 1 *A* classification *is a structure* $\langle S, T, : \rangle$*, where* S *is a set of tokens,* T *is a set of types, and* $:$ *is a relation on* $S \times T$ *that assigns types to tokens.*

Definition 2 *A* feature structure *over* `Type` *(a set of types) and* `Feat` *(a set of features) is a tuple* $F = \langle Q, \rightsquigarrow, \overline{q}, \mathcal{C}, T \rangle$ *where*

- Q *is a finite set of tokens*
- $\rightsquigarrow$ *is a binary relation (the connection relation) on* Q *and* $\rightsquigarrow = \bigcup\{\rightsquigarrow_c\}$ *(see below),*
- $\overline{q} \in Q$ *and for any* $q \in Q q \neq \overline{q}, \overline{q} \rightsquigarrow^* q$ *(*$\overline{q}$ *is the root)*[6]
- C *is a set of classifications of connections* $c = \langle \rightsquigarrow_c, \texttt{type}_c, :_c \rangle$ *where:*
  1. $\texttt{type}_c = \{f\} \cup \{\phi \Longrightarrow \psi\}$ *where* $f \in$ `Feat` *and* $\phi, \psi \in$ `Type`*, and*
  2. *if* $f' \in \texttt{type}_c$ *and* $f' \in$ `Feat` *then* $f' = f$*, and*
  3. $\forall q, q' \in Q, q \rightsquigarrow_c q'$ *iff* $\langle q, q' \rangle :_c f$ *and* $\langle q, q' \rangle :_c \phi \Longrightarrow \psi$
  4. $\forall q, q' \in Q$*, if* $q \rightsquigarrow_c q'$ *and* $q \rightsquigarrow_c q''$ *then* $q' = q''$
- $T \in$ `Type` *(the type of* F*), and*
- $\forall \tau \in$ `Type` *such that* $\tau \neq T, T \Longrightarrow^* \tau$ *is an element of the transitive closure of the set of indicating relations:* $\{\phi \Longrightarrow \psi | \exists c : \phi \Longrightarrow \psi \in \texttt{type}_c\}$

Links in the feature graph correspond to typed connections where the types are elements of `Feat` and `Type` $\times$ `Type`. A connection is thus classified by at least two types: the feature and the *indicating* relation ($\Longrightarrow$) on elements of `Type`. The connection relation ($\rightsquigarrow$) and set of classifications of elements of that relation (C) take the place of the state transition function δ in Carpenter's definition. As an example of a channel-theoretic model of a feature structure, consider (6) which depicts a nonclassical model corresponding to the structure in Figure 8, in which there is a clash on the `NUMBER` feature. Figure 9 focuses on the `NUM` feature and depicts the conflicting indications (tokens are represented as existing in the plane and types as points that float above or below the plane, classifying the points in the plane; the connection between s and s' is classified by both indicating relations, and the dashed lines specify the clasifications of the other tokens as defined below).

[6] The transitive closure of $\rightsquigarrow$ is denoted $\rightsquigarrow^*$.

(6)

Let:

$$\begin{aligned}\texttt{Type} &= \{\texttt{sent}, \texttt{noun}, \texttt{verb}, \texttt{syn}, \texttt{3rd}, \texttt{sing}, \texttt{pl}\}\\ \texttt{Feat} &= \{\texttt{SUBJ}, \texttt{PRED}, \texttt{AGR}, \texttt{PERSON}, \texttt{NUMBER}\}\end{aligned}$$

Then $F = \langle Q, \rightsquigarrow, \overline{q}, \mathcal{C}, T\rangle$ is a feature structure over Type and Feat where:

$$\begin{aligned}
Q &= \{q_1, q_2, q_3, q_4, q_5, q_6\}\\
\overline{q} &= q_1\\
\rightsquigarrow &= \rightsquigarrow_{c_\star} \cup \rightsquigarrow_{c_\diamond} \cup \rightsquigarrow_{c_\bullet} \cup \rightsquigarrow_{c_\triangle} \cup \rightsquigarrow_{c_*} \cup \rightsquigarrow_{c_\square}\\
\mathcal{C} &= \{c_\star, c_\diamond, c_\bullet, c_\triangle, c_*, c_\square\}\\
T &= \texttt{sent}\\
c_\star &= \langle\{\langle q_1, q_2\rangle\},\\
&\quad \{\texttt{SUBJ}, \texttt{sent} \Rightarrow \texttt{noun}\},\\
&\quad \{\langle\langle q_1, q_2\rangle, \texttt{SUBJ}\rangle, \langle\langle q_1, q_2\rangle, \texttt{sent} \Rightarrow \texttt{noun}\rangle\}\rangle\\
c_\diamond &= \langle\{\langle q_1, q_3\rangle\},\\
&\quad \{\texttt{PRED}, \texttt{sent} \Rightarrow \texttt{verb}\},\\
&\quad \{\langle\langle q_1, q_3\rangle, \texttt{PRED}\rangle, \langle\langle q_1, q_3\rangle, \texttt{sent} \Rightarrow \texttt{verb}\rangle\}\rangle\\
c_\bullet &= \langle\{\langle q_2, q_4\rangle\},\\
&\quad \{\texttt{AGR}, \texttt{noun} \Rightarrow \texttt{syn}\},\\
&\quad \{\langle\langle q_2, q_4\rangle, \texttt{AGR}\rangle, \langle\langle q_2, q_4\rangle, \texttt{noun} \Rightarrow \texttt{syn}\rangle\}\rangle\\
c_\triangle &= \langle\{\langle q_3, q_4\rangle\},\\
&\quad \{\texttt{AGR}, \texttt{verb} \Rightarrow \texttt{syn}\},\\
&\quad \{\langle\langle q_3, q_4\rangle, \texttt{AGR}\rangle, \langle\langle q_3, q_4\rangle, \texttt{verb} \Rightarrow \texttt{syn}\rangle\}\rangle\\
c_* &= \langle\{\langle q_4, q_5\rangle\},\\
&\quad \{\texttt{PERSON}, \texttt{syn} \Rightarrow \texttt{3rd}\},\\
&\quad \{\langle\langle q_4, q_5\rangle, \texttt{PERSON}\rangle, \langle\langle q_4, q_5\rangle, \texttt{syn} \Rightarrow \texttt{3rd}\rangle\}\rangle\\
c_\square &= \langle\{\langle q_4, q_6\rangle\},\\
&\quad \{\texttt{NUMBER}, \texttt{syn} \Rightarrow \texttt{pl}, \texttt{syn} \Rightarrow \texttt{sing}\},\\
&\quad \{\langle\langle q_4, q_6\rangle, \texttt{NUMBER}\rangle, \langle\langle q_4, q_6\rangle, \texttt{syn} \Rightarrow \texttt{pl}\rangle,\\
&\quad \langle\langle q_4, q_6\rangle, \texttt{syn} \Rightarrow \texttt{sing}\rangle\}\rangle
\end{aligned}$$

Our non-classical feature structures do not have a function assigning types to individual nodes in the feature graph. The determination of types associated with a node is derived from classifications of the feature structure containing the node. In particular, the type of a node is determined by the consequent types in indicating relations (e.g. $\texttt{syn} \Longrightarrow \texttt{sing}$ or $\texttt{syn} \Longrightarrow \texttt{plu}$) associated with connections in which the node participates (e.g. $q' \rightsquigarrow_{\texttt{NUMB}} q$ and $\langle q', q\rangle :_{\texttt{NUMB}} \texttt{syn} \Longrightarrow \texttt{sing}$ implies that q has the type sing). Non-classical feature structures emerge from the paraconsistent classification (Δ; see Dfn. 3), in which conflicting indications

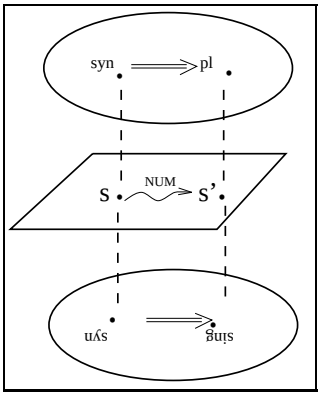

Figure 9: Conflicting Indications

cause inconsistent assignments of types to nodes.

Definition 3 *The* indication classification *based on F, denoted $\Delta(F)$, is a classification, $\langle Q, \mathtt{Type}, :_{\Delta(F)}\rangle$ where:*

- $\overline{q} :_{\Delta(F)} T$
- $\overline{q} :_{\Delta(F)} \phi$ *iff* $\exists \phi, \psi \in \mathtt{Type}$ *and* $\exists q \in Q$ *such that for some* c, $\overline{q} \rightsquigarrow_c q$ *and* $\langle \overline{q}, q\rangle :_c \phi \Longrightarrow \psi$
- $q :_{\Delta(F)} \psi$ *iff* $\exists q'$ *such that* $q' :_{\Delta(F)} \phi$ *and* $\exists c \in C$ *such that* $q' \rightsquigarrow_c q$ *and* $\phi \Longrightarrow \psi \in \mathtt{type}_c$

Definition 4 *Let $\theta_{\Delta(F)}$ be a mapping from $Q \rightarrow \mathtt{Type}$, such that for a feature structure $F = \langle Q, \rightsquigarrow, \overline{q}, \mathcal{C}, T\rangle$, $\theta_{\Delta(F)}(q) = \tau$ iff $q :_{\Delta(F)} \tau$*

As an example, Figure 10 shows the indication classification of the nodes from the feature structure depicted in (6), through the θ mapping. Note that conflicting types are indicated of the node q_6.

$$
\begin{array}{rcl}
\theta_{\Delta(F)}(q_1) & = & \mathtt{sent} \\
\theta_{\Delta(F)}(q_2) & = & \mathtt{noun} \\
\theta_{\Delta(F)}(q_3) & = & \mathtt{verb} \\
\theta_{\Delta(F)}(q_4) & = & \mathtt{syn} \\
\theta_{\Delta(F)}(q_5) & = & \mathtt{3rd} \\
\theta_{\Delta(F)}(q_6) & = & \mathtt{sing} \\
\theta_{\Delta(F)}(q_6) & = & \mathtt{pl}
\end{array}
$$

Figure 10: Types Indicated of the Nodes in the Feature Structure from the example (6)

Proposition 1 *Assume F is a feature structure:*

1. *$\theta_{\Delta(F)}$ is a total mapping;*
2. *$\theta_{\Delta(F)}$ is a function iff each connection $q \rightsquigarrow q'$ is classified by only one indicating relation, and $\forall q, r, s \in Q$ if $\langle q, s\rangle :_c \phi \Longrightarrow \psi$ and $\langle r, s\rangle :_c \tau \Longrightarrow \omega$ then $\omega = \psi$;*
3. *the equation $\theta_{\Delta(F)}(q) = \tau$ is computable in linear time relative to the size of the input feature structure;*
4. *if θ is a function $\theta_{\Delta(F)}(q)$ is computable in linear time relative to the size of the input feature structure;*
5. *Given an arbitrary feature structure, $\theta_{\Delta(F)}(q)$ is an NP-hard computation.*

Proofs of each of the parts of Proposition 1 follow. Essentially, the proposition shows that determination of the classification of a node in a wide class of feature structures remains an efficient computation.

Proof:

1.1 1. Recall that Q is connected to $\overline{q}$ by $\rightsquigarrow$ ($\forall q \neq \overline{q}, \overline{q} \rightsquigarrow^* q$) and all connections are classified: $q' \rightsquigarrow q \supset \exists c : q' \rightsquigarrow_c q$, which means that $\langle q', q\rangle :_c \phi \Longrightarrow \psi$ and $\phi \Longrightarrow \psi \in \texttt{type}_c$.
Also recall that `Type` is connected to T by $\Longrightarrow$: $\forall \tau \in$ `Type` such that $\tau \neq T, T \Longrightarrow^* \tau$ is an element of the transitive closure of $\{\phi \Longrightarrow \psi | \exists c : \phi \Longrightarrow \psi \in \texttt{type}_c\}$.
Since $\overline{q} :_{\Delta(F)} T$, $\forall q \neq \overline{q}$ such that $\overline{q} \rightsquigarrow q$, we also know that $\langle \overline{q}, q\rangle :_c T \Longrightarrow \tau$, and thus $q :_{\Delta(F)} \tau$.

2. Assume that $\forall q$ if $\exists q' \in Q : q' \rightsquigarrow^* q$, then $\exists \phi \in$ `Type` such that $\exists c : \langle q', q\rangle :_c \phi \Longrightarrow \psi$ and $q' :_{\Delta(F)} \phi$.
Since $\phi \Longrightarrow \psi \in \texttt{type}_c$ and $q' :_{\Delta(F)} \phi$, $q :_{\Delta(F)} \psi$.

1.2 This follows in the above proof by noting that as well as obtaining types from indicating relations that classify connections in which a token participates, under the assumptions in the proposition, each token can be classified by only one indicating type.

1.3 This assumes that indicating relations are composed of atomic types alone, in which case they can be seen as transitions in a finite state automaton. Verifying whether an arbitrary token q is Δ-classifiable by τ just requires being able to move from the start token to q along both the indicating and connection relations. The split between type and token levels offers no additional complexity.

1.4 The argument will build a construction similar to the above: θ's being a function guarantees only one indicating relation per connection relation, and only one type per token. Thus, determining the type assigned to q entails verifying that $\overline{q} \rightsquigarrow^* q$, and calculating corresponding type. (We also assume the same restricted calculus of types).

1.5 Informally, this is because it requires enumerating all the types that a feature structure indicates the token to be; this enumeration problem is exponential in the worst case where θ is not a function.

☺

Classical feature structures, particularly as defined by Carpenter (1992), are a restriction of paraconsistent classification in which arbitrary nodes either have a unique type or have no type at all. The *univocal classification* ($\mathtt{U}$; see Dfn. 5) formalizes this; Figure 11 shows the univocal classification of the nodes from the feature structure modeled in Figure 6. This definition can be modified to provide a total typing function on all nodes similar to Aït-Kaci's (1984) definition. His formulation admitted inconsistencies at nodes but then ruled them out via smashing to obtain $\perp$ for the whole feature structure. We leave the inconsistencies in place in the Δ classification and move to agnosticism with respect to the types of the same nodes in the $\mathtt{U}$ classification. Our treatment is similar to that which would be provided by a 4-valued logic for feature structures (Schöter, 1992, 1995).

Definition 5 *The* univocal classification *based on F, denoted $\mathtt{U}(F)$, is a classification, $\langle Q, \mathtt{Type}, :_{\mathtt{U}(F)}\rangle$ such that:*

- $\forall q, q :_{\mathtt{U}(F)} \phi$ *if* $\phi = \bigwedge\{\psi \mid q :_{\Delta(F)} \psi\}$ *and ϕ is consistent*
- *otherwise,* $\neg\exists\phi$ *such that* $q :_{\mathtt{U}(F)} \phi$

$$
\begin{array}{lll}
q_1 & :_{\mathtt{U}(F)} & \mathtt{sent} \\
q_2 & :_{\mathtt{U}(F)} & \mathtt{noun} \\
q_3 & :_{\mathtt{U}(F)} & \mathtt{verb} \\
q_4 & :_{\mathtt{U}(F)} & \mathtt{syn} \\
q_5 & :_{\mathtt{U}(F)} & \mathtt{3rd} \\
q_6 & :_{\mathtt{U}(F)} & \mathtt{UNDEFINED}
\end{array}
$$

Figure 11: Univocal Classification of Nodes in a Feature Structure

It is possible to consider just the restricted universe of consistent feature structures, but these are properly contained in a hierarchy of inconsistent structures. We define a relative inconsistency ordering based on cardinality of inconsistencies contained in a feature structure so that we can refer to the minimally inconsistent structures. To do this, we assume total typedness; thus we can compare the feature structure which has all features defined but no values specified at terminal paths in the structure with the maximally inconsistent structure which has a $\perp$-term as the value of each feature in the structure, and with all of the feature structures with intermediate degrees of instantiation and inconsistency.

The robust parser that we have presented captures the distinction between consistent and inconsistent elements in this hierarchy. It should be clear that the parser produces sound

analyses with respect to the space of feature structures we consider, but its description language is not complete. Our non-classical feature structures are richer than objects supplied by other theories of feature structures which lack the binary type-level relation that indicates the types of nodes. In addition to supplying information about the inconsistency of types supported by nodes, *indicating relationships supply the origins of type conflicts.* While the description language in our parser represents the existence of inconsistencies, we have not implemented a description language that expresses the source of inconsistency. Clearly, this additional information derived from linguistic regularities can offer critical advantage to a system designed to integrate domain-style reasoning with linguistic processing. We have not yet provided a way in either the parser or the theory of feature structures with which to articulate a preference ordering on features structures which would yield, for instance, preference for certain feature structures over others with equal cardinality of inconsistency.

4 Discussion

We have presented a robust chart parser for HPSG. The system uses a relaxed unification operation which is applied judiciously during the chart-parsing algorithm to discover the constraints which need to be relaxed in an arbitrary HPSG in order to obtain minimally inconsistent feature structures that correspond to analyses attributing minimal nongrammaticality to the inputs. The algorithm and techniques that we suggest are applicable to robust chart parsing for unification grammars in general.

We have a treatment for unknown words, and suggest a method for generalizing this treatment to contend with missing and extra elements in sentences to be parsed as well. Like Mellish's (1989) technique for the latter problem, our suggestion involves post-processing. However, the relaxational component of the system is not based on backtracking. The effect on time-complexity is a penalty on processing completely grammatical inputs, thus it would be interesting in the future to explore modifications to the system which invoke relaxed unification only when classical unification will not admit a parse. This, of course, makes the cost of parsing nongrammatical inputs significantly more dear than in our system, but it would be a useful exercise to determine the issues which should decide which conditions make either approach more attractive.

In terms of space complexity, Schöter (1995) has pointed out that the relaxed unification that we use in backtracking-free parsing has space complexity proportional to $O(mn)$ (where $m =$ the number of categories in the feature description, and $n =$ the maximum number of values for a feature) when applied to consistent feature descriptions, which is the same space complexity of classical unification. In the general case, the space complexity of relaxed unification is proportional to $O(2^m n)$. Thus, in terms of other approaches to paraconsistency in feature logics, our approach is on the low end of the complexity scale (since other paraconsistent systems can require exponential space for the consistent case as well (Schöter, 1995)).

We have argued that our approach which involves discovering the constraints which need to be relaxed in order to obtain an analysis of a sentence is more flexible than approaches like Douglas and Dale's (1992) which require the naming in advance of the features whose constraints can be relaxed. Advanced-naming approaches are more closely bound to specific grammars, and therefore offer potential computational advantage. We proposed to incorpor-

ate a facility for prioritizing the robust agenda based on preferences for sources of inconsistency (for instance, preferring analyses that have number clash over those which have inconsistencies on major category). Such an enhancement would offer the advantages of advanced-naming, but would let our system retain its greater robustness in coping with novel feature clashes.

It would be extremely interesting to develop a post-parsing interpretation module which examines the feature descriptions produced by our system and based on domain features or user models (e.g. *when dealing with native speakers of language X assume that the number information supplied by the phrasal head is the intended information, but for native speakers of language Y assume that the complement contains the intended information*). Our system can be used as a tool in the empirical determination of what those preferences should be by monitoring the frequency with which particular inconsistencies occur in relevant classes of corpora. However, we have not proposed an architecture for a module which attempts to resolve inconsistencies. For this, it would be useful to examine other approaches that integrate domain reasoning with linguistic representation (e.g. Iwańska, 1992).

Finally, we have presented the formalization of nonclassical feature structures that guides our research in paraconsistent feature logic. We are currently exploring the relationship between our system and other possible description languages for the expanded universe of feature structures that we propose. We have not yet determined the most appropriate logic for our needs and theoretical interests. We offer the present paper as proof of the concept that parsers for unification grammars based on feature theory can practically incorporate the representational advantages offered by a nonclassical feature theory that admits local inconsistency.

Acknowledgements

An earlier version of this paper was presented at the AAAI 1994 Fall Symposium: *Knowledge Representation for Natural Language Processing in Implemented Systems* and is available in the AAAI technical report that archives the meeting. Both Vogel and Cooper thank the anonymous reviewers from that meeting, as well as Paul King, Tomaž Erjavec, Andreas Schöter and the Language Technology Group within the Human Communications Research Centre for insightful comments and feedback. Vogel is grateful to the Marshall Aid Commemoration Commission for supporting his stay in Edinburgh.

References

Aït-Kaci, H. (1984). *A Lattice Theoretic Approach to Computation Based on a Calculus of Partially Ordered Types*. Ph.D. thesis, Computer and Information Sciences, University of Pennsylvania.

Aït-Kaci, H. & Lincoln, P. (1988). LIFE A Natural Language for Natural Language. Tech. rep. ACA-ST-074-88, Systems Technology Laboratory, Univ. of Austin.

Aït-Kaci, H. & Nasr, R. (1986). LOGIN: A Logic Programming Language with Built-in Inheritance. *Journal of Logic Programming*, *3*, 185–215.

Barwise, J. (1991). Information Links in Domain Theory. Tech. rep. IULG-91-7, Indiana University Logic Group.

Barwise, J. (1993). Constraints, Channels, and the Flow of Information. In *Situation Theory and its Applications, Volume 3*. CSLI Lecture Notes no. 37.

Barwise, J. & Seligman, J. (1992). The Rights and Wrongs of Natural Regularity. *Philosophical Perspectives*, *8 or 9*. to appear.

Bouma, G. (1992). Feature Structures and Nonmonotonicity. *Computational Linguistics*, *18*(2). Special Issue on Inheritance I.

Carbonell, J. G. & Hayes, P. J. (1983). Recovery Strategies for Parsing Extragrammatical Language. *American Journal of Computational Linguistics*, *9*(3-4). Special Issue on Ill-Formed Input.

Carpenter, B. (1992). *The Logic of Typed Feature Structures*. Cambridge Tracts in Theoretical Computer Science 32.

Carpenter, B. (1993). ALE. The Attribute Logic Engine User's Guide. Tech. rep. version β, Laboratory for Computational Linguistics, Carnegie Mellon University.

Douglas, S. (1995). Robust PATR for Error Detection and Correction. Centre for Cognitive Science, University of Edinburgh, Working Papers in Cognitive Science, Volume 10.

Douglas, S. & Dale, R. (1992). Towards robust PATR. In *Proceedings of COLING 92: 14th International Conference on Computational Linguistics*.

Grover, C., Brew, C., Manandhar, S., & Moens, M. (1994). Priority Union and Generalization in Discourse Grammars. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*.

Hayes, P. J. & Mouradian, G. V. (1981). Flexible Parsing. *American Journal of Computational Linguistics*, *7*(4).

Iwańska, Łucja. (1992). A General Semantic Model of Negation in Natural Language. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*. Cambridge, MA. October 25-29, 1992.

Jensen, K., Heidorn, G. E., Miller, L. A., & Ravin, Y. (1983). Parse Fitting and Prose Fixing: Getting a Hold on Ill-Formedness. *American Journal of Computational Linguistics*, *9*(3-4). Special Issue on Ill-Formed Input.

Johnson, M. (1987). *Attribute-Value Logic and the Theory of Grammar*. Ph.D. thesis, Department of Linguistics, Stanford University, CA.

Keller, B. (1993). *Feature-Logics, Infinitary Descriptions, and Grammar*. Centre for the Study of Language and Information, Stanford University, CA. CSLI Lecture Notes Number 44.

Kodrič, S. (1994). VP Constituency and Clause Structure in Slovene. *Slovene Studies*, *15*. to appear.

Kodrič, S., Popowich, F., & Vogel, C. (1992). The HPSG-PL System: Version 1.2. Tech. rep. CSS-IS TR 92-05, CMPT TR 92-08, School of Computing Science, Simon Fraser University, Burnaby, B.C.

Lascarides, A., Briscoe, T., Asher, N., & Copestake, A. (1995). Order Independent and Persistent Typed Default Unification. Centre for Cognitive Science, University of Edinburgh, Working Papers in Cognitive Science, Volume 10.

Mellish, C. S. (1989). Some Chart-Based Techniques for Parsing Ill-Formed Input. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*. Vancouver, BC.

Nebel, B. & Smolka, G. (1989). Representing and Reasoning with Attributive Descriptions. Tech. rep. IWBS Report 81, IBM Deutschland, Stuttgart, Germany.

Pollard, C. & Sag, I. (1987). *Information-Based Syntax and Semantics, Volume 1: Fundamentals*. Centre for the Study of Language and Information, Stanford University, CA.

Pollard, C. & Sag, I. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press and CSLI Publications.

Popowich, F., McFetridge, P., Fass, D., & Hall, G. (1992). Processing Complex Noun Phrases in a Natural Language Interface to a Statistical Database. In *Proceedings of COLING 92: 14th International Conference on Computational Linguistics*.

Popowich, F. & Vogel, C. (1991). A Logic Based Implementation of Head-Driven Phrase Structure Grammar. In Brown, C. & Koch, G. (Eds.), *Natural Language Understanding and Logic Programming, III*, pp. 227–246. Elsevier, North-Holland.

Schöter, A. (1992). Partial Logic, Logic Programming and Unification. MSc Thesis, Centre for Cognitive Science, University of Edinburgh.

Schöter, A. (1995). Paraconsistent Feature Logic. Centre for Cognitive Science, University of Edinburgh, Working Papers in Cognitive Science, Volume 10.

Weischedel, R. M. & Black, J. E. (1980). Responding Intelligently to Unparsable Inputs. *American Journal of Computational Linguistics*, *6*(2).

Weischedel, R. M. & Sondheimer, N. K. (1983). Meta-Rules as a Basis for Processing Ill-Formed Input. *American Journal of Computational Linguistics*, *9*(3-4). Special Issue on Ill-Formed Input.