Typed Feature Structures for Expressing and Computationally Implementing Feature Cooccurence Restrictions

Dale Gerdemann and Paul John King
Seminar für Sprachwissenschaft
Eberhard-Karls-Universität Tübingen
72074 Tübingen
Germany
Email: {dg,king}@sns.neuphilologie.uni-tuebingen.de
Tel: 07071 297313
Fax: 07071 550520

8th September, 1993

Abstract

It has long been part of the folklore that typed feature structures were introduced into linguistics in order to replace the perceived-to-be-messy FCRs of GPSG. This correspondence between FCRs and typed feature structures, however, has never been made explicit. In fact, with the notion of appropriateness conditions used in Carpenter (1992), it is initially difficult to see how any dependency could be expressed between the values of two features. We show in this paper that there is a way to express FCRs by expressing them on the maximally specific types. These FCRs are unfortunately ignored in the type inferencing procedure from Carpenter (1992), which is used in the ALE parser of Carpenter (1993). We show, however, that by using the idea of "type resolution", it is possible to maintain satisfiable feature structures in parsing. This idea is implemented in the "Troll Type Resolution System".

Typed Feature Structures for Expressing and Computationally Implementing Feature Cooccurence Restrictions

Early unification formalisms restricted neither the features a linguistic object could have, nor the values those feature could take. However, linguistic considerations demanded some mechanism for imposing such restrictions, and the FCRs (feature cooccurrence restrictions) of GPSG [3] were such a mechanism. This paper describes how FCRs can be encoded into a type hierarchy and efficiently implemented in the Troll system.

We define two subcases of FCRs. Call an FCR of the form

> if an object is of a certain kind
> then it deserves certain features with values of certain kinds

a *conjunctive* FCR. An FCR stating that a verb must have V and N features with values + and − respectively is an example of a conjunctive FCR. Call an FCR of the form

> if an object is of a certain kind
> then it deserves certain features with values of certain kinds,
> or it deserves certain (perhaps other) features with values of certain (perhaps other) kinds,
> or ...
> or it deserves certain (perhaps other) features with values of certain (perhaps other) kinds

a *disjunctive* FCR. An FCR stating that inverted verbs must be auxiliaries is an example of a disjunctive FCR: a verb must have the features INV and AUX with values + and +, − and +, or − and − respectively. Clearly, a conjunctive FCR is a simple disjunctive FCR. Though FCRs can be extremely complex, the great majority of FCRs are disjunctive.

However, FCRs were perceived to be overly procedural, and some recent formalisms have used appropriateness specifications in order to (among other things) encode FCRs declaratively. These formalisms employ a finite partial order $\langle \mathsf{Type}, \sqsubseteq \rangle$ of types under subsumption, a finite set Feat of features, and an appropriateness partial function $\mathsf{Approp}\colon \mathsf{Type} \times \mathsf{Feat} \rightharpoonup \mathsf{Type}$. Intuitively, the types formalise the hitherto informal kinds, and $\mathsf{Approp}(t, f) = t'$ iff each object of type t deserves feature f with a value of type t'. Call such formalisms *appropriateness formalisms*. Carpenter's ALE and Gerdemann and Götz's Troll are examples of implementations of appropriateness formalisms.

How an appropriateness formalism encodes a conjunctive FCR is obvious, but how it encodes a disjunctive FCR is less so. An example illustrates best how it is done. Suppose that FCR ρ states that objects of type t deserve features f and g, both with boolean values, but that the values of f and g must agree. ρ is the disjunctive FCR

> if an object is of type t
> then it deserves feature f with value $+$ and feature g with value $+$,
> or it deserves feature f with value $-$ and feature g with value $-$

To encode ρ, first introduce subtypes, t' and t'', of t, one subtype for each disjunct in the consequent of ρ. Then encode the feature/value conditions in the first disjunct by putting $\mathsf{Approp}(t', f) = +$ and $\mathsf{Approp}(t', g) = +$, and encode the feature/value conditions in the second disjunct by putting $\mathsf{Approp}(t'', f) = -$ and $\mathsf{Approp}(t'', g) = -$.

Call a type that has no subtypes a *variety*. Say that an appropriateness formalism meets the *partition condition* iff for each type t, if an object is of type t then the object is of exactly one variety subsumed by t. Say that an appropriateness formalism meets the *all-or-nothing condition* iff for each variety v and feature f, either every or no object of variety v deserves feature f. An appropriateness formalism that does not meet both conditions may not properly encode a disjunctive FCR. For example, consider disjunctive FCR ρ. An appropriateness formalism may not properly encode that t' and t'' represent all and only the disjuncts in the consequent of ρ without the partition condition. An appropriateness formalism may not properly encode the feature/value conditions demanded by each disjunct in the consequent of ρ without the all-or-nothing condition.

Though all appropriateness formalisms can encode disjunctive FCRs, very few of them deem well-formed only those feature structures that meet every encoded disjunctive FCR. Consider Carpenter's ALE for example. Carpenter calls a feature structure *well-typed* iff for each arc in the feature structure, if the source node is labelled with type t, the target node is labelled with type t' and the arc is labelled with feature f then $\mathsf{Approp}(t, f) \sqsubseteq t'$. Carpenter calls a feature structure *well-typable* iff the feature structure subsumes a well-typed feature structure, and Carpenter gives an effective algorithm for deciding if a feature structure is well-typable. Well-typability comes close to an effective criterion for the well-formedness of a feature structure with respect to encoded FCRs: certainly, a feature structure is well-typable iff it meets every encoded conjunctive FCR. However, consider our encoding of disjunctive FCR ρ and suppose that φ is the feature structure $\begin{bmatrix} t \\ f{:}+ \\ g{:}- \end{bmatrix}$. φ is well-typed, and hence trivially well-typable. Unfortunately, φ violates encoded disjunctive FCR ρ.

By contrast, the Troll system described in this paper has an effective algorithm for deciding the well-formedness of a feature structure with respect to encoded disjunctive FCRs. Call a well-typed feature structure in which all nodes are labelled with varieties a *resolved feature structure* and call a set of resolved feature structures that have the same underlying graph (that is, they differ only in their node labellings) a *disjunctive resolved feature structure*. We write $\mathcal{FS}$, $\mathcal{RFS}$ and $\mathcal{DRFS}$ for the collections of feature structures, resolved feature structures and disjunctive resolved feature structures respectively. Say that $F' \in \mathcal{RFS}$ is a resolvand of $F \in \mathcal{FS}$ iff F and F' have the same underlying graph and F subsumes F'. Let type resolution be the total function $\mathcal{R}{:}\,\mathcal{FS} \rightarrow \mathcal{DRFS}$ such that $\mathcal{R}(F)$ is the set of all resolvands of F.

Guided by the partition and all-or-nothing conditions, we have formulated a semantics of feature structures and developed a notion of a satisfiable feature structure such that a feature structure meets all encoded FCRs iff the feature structure is satisfiable. We have also shown that $F \in \mathcal{FS}$ is satisfiable iff $\mathcal{R}(F) \neq \emptyset$ (See King and Carpenter [9]). The Troll system, which is based on this idea, implements type resolution effectively.

Why does Troll succeed where ALE fails? Consider again the encoding of ρ and the feature structure φ. Loosely speaking, the appropriateness specifications for type t encode the part of ρ that states that an object of type t deserves features f and g, both with boolean values. However, the appropriateness specifications for the varietal subtypes t' and t'' of type t encode the part of ρ that states that these values must agree. Well-typability only considers varieties if forced to. In the case of φ, well-typability can be established by considering type t alone. Consequently, well-typability overlooks the part of ρ exclusively encoded by the appropriateness specifications for t' and t''. Type resolution, on the other hand, always considers varieties. Consequently, type resolving φ cannot overlook the part of ρ exclusively encoded by the appropriateness specifications for t' and t''.

A very important property of the disjunctive resolved feature structures is that they are closed under unification, i.e., if F and $F' \in \mathcal{DRFS}$ then $\mathcal{R}(F) \sqcup \mathcal{R}(F') \in \mathcal{DRFS}$.[1] Given this property, it would in principle be possible to use the disjunctive resolved feature structures in an implementation without any additional type inferencing procedure to maintain satisfiability. It would not, however, be efficient to work with such large disjunctions of feature structures. These disjunctions of feature structures, however, have a singular property: all of the disjuncts have the same shape. The disjuncts differ only in the types labeling the nodes. This property allows a disjunctive resolved feature structure to be represented more efficiently as a single untyped feature structure plus a set of dependent node labelings, which can be further compacted using named disjunction as in Gerdemann [4], Dörre & Eisele [8] or Maxwell & Kaplan [7].

For example, suppose we type resolve the feature structure $\begin{bmatrix} t \\ f\!: boolean \\ g\!: boolean \end{bmatrix}$ using our encoding of ρ. If we simply replace all types by varieties, there are eight possibilities:

$$\left\{ \begin{bmatrix} t' \\ f\!:+ \\ g\!:+ \end{bmatrix}, \begin{bmatrix} t' \\ f\!:+ \\ g\!:- \end{bmatrix}, \begin{bmatrix} t' \\ f\!:- \\ g\!:+ \end{bmatrix}, \begin{bmatrix} t' \\ f\!:- \\ g\!:- \end{bmatrix}, \begin{bmatrix} t'' \\ f\!:+ \\ g\!:+ \end{bmatrix}, \begin{bmatrix} t'' \\ f\!:+ \\ g\!:- \end{bmatrix}, \begin{bmatrix} t'' \\ f\!:- \\ g\!:+ \end{bmatrix}, \begin{bmatrix} t'' \\ f\!:- \\ g\!:- \end{bmatrix} \right\}$$

Of these eight, only two are well-typed.[2] These two can then be collapsed into one feature structure with named disjunction (since all the disjuncts interact, they are all given #1 as a name):

$$\left\{ \begin{bmatrix} t' \\ f\!:+ \\ g\!:+ \end{bmatrix}, \begin{bmatrix} t'' \\ f\!:- \\ g\!:- \end{bmatrix} \right\} \Rightarrow \begin{bmatrix} \langle 1\ t'\ t'' \rangle \\ f\!:\ \langle 1\ +\ - \rangle \\ g\!:\ \langle 1\ +\ - \rangle \end{bmatrix}$$

[1] In fact, it can be shown that $\mathcal{R}(F) \sqcup \mathcal{R}(F') = \mathcal{R}(F \sqcup F')$. Unification of sets of feature structures is defined here in the standard way: $S \sqcup S' = \{F \mid F' \in S \text{ and } F'' \in S' \text{ and } F = F' \sqcup F''\}$.

[2] We have described this process as a kind of generate and test procedure. It makes more sense, though to incorporate the appropriateness condition check into the type resolution procedure. One can also avoid full type resolution by allowing nodes to be labeled by equivalence classes of varieties such that all members of the class have exactly the same appropriateness conditions for all of the features that happen to be present.

We now have a reasonably compact representation in which the FCR has been translated into a distributed disjunction. However, one should note that this disjunction is only present because the features f and g happen to be present. These features would need to be present if we were enforcing Carpenter's [1] total well typing requirement, which says that features that are allowed must be present. But total well typing is, in fact, incompatible with type resolution, since there may well by an infinite set of totally well typed typed resolvants of a feature structure. For example, an underspecified list structure could be resolved to a list of length 0, a list of length 1, etc. Since total well typing is not required, we may as well go the other direction and actively unfill redundant features.[3] In this example, if the f and g features are removed, we are left with the simple disjunction $\{t', t''\}$, which is equivalent to the ordinary type t.[4] Thus, in this case, no disjuction at all is required to enforce the FCR. All that is required is the assumption that t will only be extended by unifiying it with another (compacted) member of $\mathcal{DRFS}$.

This, however, was a simple case in which all of the distributed disjunction could be removed. It would not have been possible to remove the features f and g if these features had been involved in reentrancies or if these features had had complex values. In general, however, our experience has been that even with very complex type hierarchies and feature structures for HPSG, very few distributed disjunctions are introduced. Thus, unification is generally no more expensive than unification with untyped feature structures. This, however, represents initial experience with the Troll system, which needs to be substantiated by further experience with other grammars. In future research, we will look at type hierarchies that have been used for HPSG-style grammars to see to what extent implicit FCRs are contained in these hierarchies and to what extend these FCRs introduce disjunctions into feature structures. Troll, as it stands, is correct though much may still be done to make it more efficient, in terms of both implementation improvements and guidelines to grammar writers for writing efficient grammars.

[3]Intuitively, features are redundant if their values are entirely predictable from the appropriateness specification. See Götz [6], Gerdemann [5] for a more precies formulation.

[4]In this case, it would also have been possible to unfill the original feature structure before resolving. Unfortunatly, however, this is not always the case, as can be seen in the following example: $\begin{bmatrix} t \\ f{:}+ \end{bmatrix} \Rightarrow \left\{ \begin{bmatrix} t' \\ f{:}+ \end{bmatrix} \right\} \Rightarrow t'$.

References

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

[2] Bob Carpenter. ALE *The Attribute Logic Engine, User's Guide*, 1993.

[3] Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. *Generalized Phrase Structure Grammar.* Harvard University Press, Cambridge, Mass, 1985.

[4] Dale Gerdemann. *Parsing and Generation of Unification Grammars.* PhD thesis, University of Illinois, 1991. Published as Beckman Institute Cognitive Science Technical Report CS-91-06.

[5] Dale Gerdemann. Troll: Type resolution system, user's guide. Technical report, University of Tübingen, 1993 (forthcoming).

[6] Thilo Götz. Forthcoming master's thesis. Master's thesis, Universität Tübingen, 1993.

[7] John T. Maxwell III and Ronald M Kaplan. An overview of disjunctive constraint satisfaction. In *Proceedings of International Workshop on Parsing Technologies*, pages 18–27, 1989.

[8] Jochen Dörre and Andreas Eisele. Feature logic with disjunctive unification. In *COLING-90 vol. 2*, pages 100–105, 1990.

[9] Paul John King and Bob Carpenter. Untitled manuscript, 1993.