Natural Language Engineering **1** (1): 1–4. Printed in the United Kingdom
© 2000 Cambridge University Press

Appendix: definitions of typed feature structures

ANN COPESTAKE
Center for the Study of Language and Information, Stanford University, Stanford, CA 94305, USA
e-mail: aac@csli.stanford.edu

(*Received 9 May 2001*)

1 Overview

The LinGO grammar consists of a specification of a type system and of various typed feature structures which are well-formed according to the type system. The typed feature structures function as grammar rules, lexical rules and lexical entries. There are several variant typed feature structure formalisms, with different computational properties, so in this appendix we very briefly specify the version assumed by the LinGO grammar.

This appendix is necessarily terse and is only intended to allow a reader who already has a knowledge of typed feature structures to understand the specific formalism used in the LinGO grammar. The definitions given below basically follow Carpenter (1992), with the notion of type constraint from Copestake (1992). For formal details of typed feature structures in general see Carpenter (1992). A detailed account of the specific assumptions made here is given in Copestake (1999) (see Chapter 4 for an introduction and Chapter 5 for a semi-formal account).

Note that the LinGO grammar uses a very restricted formalism. For instance, it does not utilize disjunctive feature structures, negation, implication, inequalities, defaults, set-valued features, extensionality or relational constraints. Constraint resolution does not require that every type be made maximally specific, and the type inference system is essentially non-recursive. The recursive power necessary in grammars is explicitly encoded via rules, which are expressed as typed feature structures, but interpreted as phrase structure rules.

2 Definitions

A type system consists of a type hierarchy plus a set of constraints which determine which typed feature structures are well-formed.

Definition 1 (Type hierarchy)
A type hierarchy is a finite bounded complete partial order $\langle \mathsf{Type}, \sqsubseteq \rangle$.

The type hierarchy specified by the LinGO grammar is not directly a BCPO, but is converted to a BCPO automatically by the introduction of additional types, referred to as *greatest lower bound* or *glb* types.

The following definition is for typed feature structures in general, without considering type constraints:

Definition 2 (Typed feature structures)
A typed feature structure is defined on a finite set of features Feat and a type hierarchy $\langle \mathsf{Type}, \sqsubseteq \rangle$. It is a tuple $\langle Q, r, \delta, \theta \rangle$, where:

- Q is a finite set of nodes,
- $r \in Q$ (r is the root node, see below)
- $\theta : Q \longrightarrow \mathsf{Type}$ is a partial typing function
- $\delta : Q \times \mathsf{Feat} \longrightarrow Q$ is a partial feature value function

subject to the following conditions:

1. r isn't a δ-descendant.
2. all members of Q except r are δ-descendants of r.

Some systems add an extra condition:

3 there is no node n or path π such that $\delta(n, \pi) = n$.

This stipulates that the structures are acyclic. The current LinGO grammar is neutral with respect to cyclicity (modulo bugs) — it does not construct cyclic structures and does not rely on cycles causing unification failure.

We will use $\mathcal{F}$ to denotes the collection of typed feature structures. We use the notation $\pi \equiv_F \pi'$ to mean that feature structure F contains path equivalence or reentrancy between the paths π and π' (i.e., $\delta(r, \pi) = \delta(r, \pi')$ where r is the root node of F); and $\mathcal{P}_F(\pi) = \sigma$ means that the type on the path π in F is σ (i.e., $\mathcal{P}_F(\pi) = \sigma$ if and only if $\theta(\delta(r, \pi)) = \sigma$, where r is the root node of F). Subsumption is then defined as follows:

Definition 3 (Subsumption)
F subsumes F', written $F' \sqsubseteq F$, if and only if:

- $\pi \equiv_F \pi'$ implies $\pi \equiv_{F'} \pi'$
- $\mathcal{P}_F(\pi) = t$ implies $\mathcal{P}_{F'}(\pi) = t'$ and $t' \sqsubseteq t$

The subsumption hierarchy is a BCPO. Unification can be defined as follows:

Definition 4 (Unification)
The unification $F \sqcap F'$ of two feature structures F and F' is the greatest lower bound of F and F' in the collection of feature structures ordered by subsumption.

In the LinGO grammar, types are associated with constraints expressed as typed feature structures. The constraint function is given by C: $\langle \mathsf{Type}, \sqsubseteq \rangle \rightarrow \mathcal{F}$. We describe the conditions on the constraint function below. For each type there is a set of features appropriate to that type: $Appfeat$: $\langle \mathsf{Type}, \sqsubseteq \rangle \rightarrow \mathsf{Feat}$. The appropriate features for a type are determined by its constraint:

Definition 5 (Appropriate features)

If $C(t) = \langle Q, q_0, \delta, \alpha \rangle$ then the appropriate features of t are defined as $Appfeat(t) = Feat(\langle F, q_0 \rangle)$ where $Feat(\langle F, q \rangle)$ is defined to be the set of features labeling transitions from the node q in some feature structure F i.e. $f \in Feat(\langle F, q \rangle)$ such that $\delta(f, q)$ is defined.

Given this, we can define well-formed feature structures as a subset of typed feature structures:

Definition 6 (Well-formed feature structures)
We say that a given typed feature structure $F = \langle Q, q_0, \delta, \alpha \rangle$ is a well-formed feature structure iff for all $q \in Q$, we have that $F' = \langle Q', q, \delta, \alpha \rangle \sqsubseteq C(\alpha(q))$ and $Feat(q) = Appfeat(\alpha(q))$.

Constraint feature structures cannot be specified arbitrarily, since the constraint function must meet certain conditions:

Definition 7 (Constraint function)
The constraint function $C\colon \langle \mathsf{Type}, \sqsubseteq \rangle \rightarrow \mathcal{F}$ obeys the following conditions:

Type For a given type t, if $C(t)$ is the feature structure $\langle Q, q_0, \delta, \alpha \rangle$ then $\alpha(q_0) = t$.
Monotonicity Given types t_1 and t_2 if $t_1 \sqsubseteq t_2$ then $C(t_1) \sqsubseteq C(t_2)$
Compatibility of constraints For all $q \in Q$ the feature structure $F' = \langle Q', q, \delta, \alpha \rangle \sqsubseteq C(\alpha(q))$ and $Feat(q) = Appfeat(\alpha(q))$.
Maximal introduction of features For every feature $f \in \mathsf{Feat}$ there is a unique type t such that $f \in Appfeat(t)$ and there is no type s such that $t \sqsubset s$ and $f \in Appfeat(s)$.

The first three conditions guarantee that the constraint of a type will itself be a well-formed feature structure, and that constraints are mutually consistent. The fourth condition is less obvious, but it guarantees that, given a feature structure, F, if there are any well-formed feature structures subsumed by F, there will always be a unique most general such structure.

The grammar specifies partial descriptions of type constraints, which are automatically expanded to give constraint structures which meet the conditions given. Similarly, once a type system is defined, specifications of rules and lexical entries are expanded so that they are well-formed according to the type system. The conditions on the constraint function guarantee that this will either yield a unique well-formed structure or fail (which is treated as an error in the specification). The essential operation during parsing and generation is then well-formed unification, as follows:

Definition 8 (Well-formed unification)
The well-formed unification $F \sqcap_{wf} F'$ of two feature structures F and F' is the greatest lower bound of F and F' in the collection of well-formed feature structures ordered by subsumption.

Well-formed unification may result in a structure which is more specific than that given by $\sqcap$ as defined above, since a well-formed feature structure must always be subsumed by the constraint on its type.

References

Carpenter, B. (1992). *The logic of typed feature structures.* Cambridge University Press.

Copestake, A. (1992). The ACQUILEX LKB: representation issues in semi-automatic acquisition of large lexicons. In *Proceedings of the 3rd conference on Applied Natural Language Processing* (pp. 88–96). Trento, Italy.

Copestake, A. (1999). *The (new) LKB system.* (CSLI, Stanford University: `http://www-csli.stanford.edu/~aac/doc5-2.pdf`)