

# First order axiomatization of typed feature structures

Richard Elling Moe\*

\*Department of information science and media studies  
University of Bergen

Richard.Moe@infomedia.uib.no

## Abstract

We axiomatize, in first order logic, the typed feature structures used in constraint based grammar formalism for natural language processing. This is based on a discussion of how type-hierarchies should be interpreted in terms of representation theory. We close with some observations about decidability of feature constraint satisfaction in parsing.

## 1 Introduction

The history of feature logics can be traced back to terminological logics which originally was meant for knowledge representation in AI but branched into different varieties such as feature logics and, more recently, description logics (Baader et al, 2003) which is used for semantic web ontologies (Baader et al, 2005).

Feature structures first appeared in the 80s for applications in computational linguistics. Specifically in constraint based grammar formalisms (Shieber, 1992) for natural language processing, where they proved useful when side-conditions were added to context-free grammar rules. In the grammar formalism HPSG (Pollard and Sag, 1994) feature structures were combined with types and inheritance hierarchies giving rise to *typed* feature structures (Carpenter, 1992). This approach is now well-established (NLE, 2000) and appears in more recent linguistic frameworks such as the LKB System (Copestake, 2002).

Johnson (1990, 1991) saw feature logics as a fragment of first order logic. In axiomatizing the untyped feature structures he provided a solid mathematical foundation which gave access to the rich body of theory connected to first order logic. He used this to study aspects of decidability. We aim to do the same for the typed variety of feature structures. Finally, in section 8 we briefly consider feature constraints that emerge while parsing. These correspond to a logical expression placing constraints on a single feature structure constructed during the process. The check of whether these constraints can be fulfilled by a feature structure may be reduced to a question of logical satisfiability of the axioms in conjunction with the constraints.

### 1.1 Preliminaries

Logical formulas are interpreted with relation to *structures*. If  $L$  is a first order language then an  $L$ -structure is a pair  $\langle D, \llbracket \cdot \rrbracket \rangle$  where  $D$ , the domain, is a nonempty set and  $\llbracket \cdot \rrbracket$  interprets the symbols of  $L$  by mapping each  $n$ -ary function symbol to a total function from  $D^n$  to  $D$  and each  $n$ -ary relation symbol to a relation on  $D^n$ , i.e a subset of  $D^n$ . The interpretation of a symbol  $s$  is denoted by  $\llbracket s \rrbracket$ . The logical symbols have the standard fixed interpretations, where  $\equiv$  is the identity relation on  $D$ .

We write  $\langle D, \llbracket \cdot \rrbracket \rangle \models_v \varphi$  to express that the formula  $\varphi$  is interpreted as true in  $\langle D, \llbracket \cdot \rrbracket \rangle$  when the free variables are assigned the values specified by a variable assignment function  $v$ . In that case  $\varphi$  is said to be *satisfiable*.  $\varphi$  is *valid* in  $\langle D, \llbracket \cdot \rrbracket \rangle$  iff  $\langle D, \llbracket \cdot \rrbracket \rangle \models_v \varphi$  holds for all  $v$ . We write  $\langle D, \llbracket \cdot \rrbracket \rangle \models \varphi$  to express this. We write  $\langle D, \llbracket \cdot \rrbracket \rangle \models \Gamma$  to express that each formula in a set  $\Gamma$  is valid in  $\langle D, \llbracket \cdot \rrbracket \rangle$ . We say that  $\langle D, \llbracket \cdot \rrbracket \rangle$  is a *model* for  $\Gamma$ .

A formula belongs to the Schönfinkel-Bernay class if it is equivalent to a prenex-formula without function symbols where all existential quantifiers precede the universal quantifiers. That is, formulas of the form  $\exists x_1 \dots \exists x_n \forall y_1 \dots \forall y_m \varphi$  where  $\varphi$  is a formula with no occurrences of function symbols or quantifiers. We refer to this class as SB. We know a few things about SB-formulas (Dreben and Goldfarb, 1979): Provided they have a model at all, they have a finite model. Satisfiability and validity is decidable and the class of SB-formulas is closed under conjunction.

An *axiomatization* of some phenomenon consists of a set of closed formulas such that its models are precisely the models that bear the characteristics of the phenomenon in question. Thus, an axiomatization

describes a class of models and a formula is consistent with the axiomatization iff it has a model in this class.

A *bounded complete partial order* (BCPO) is a partial order  $\langle A, \sqsubseteq \rangle$  in which every subset of  $A$  with upper bounds has a *least* upper bound. We refer to such sets as being *consistent*. In particular, the empty set is regarded as consistent, having a bottom element  $\perp$  as its least upper bound. The functions  $\sqcap$  (meet) and  $\sqcup$  (join) denote greatest lower bounds and least upper bounds, respectively. Note that  $\sqcup$  is partial. Sometimes BCPOs are referred to as *meet semi-lattices*.

## 2 Feature structures

We adapt Carpenter’s definitions (Carpenter, 1992; Copestake, 2000, 2002) concerning typed feature structures, leaving some of his requirements out but otherwise with only minor differences in notation and terminology.

**Definition 2.1** A type-hierarchy is a finite BCPO.

Suppose that  $\langle TYPE, \sqsubseteq \rangle$  is a type-hierarchy and  $\sigma, \tau \in TYPE$ . If  $\sigma \sqsubseteq \tau$  we say that  $\sigma$  *subsumes*  $\tau$ , or alternatively, that  $\sigma$  is *more general/less specific* than  $\tau$ .

Type-hierarchies are sometimes defined differently. In (Smolka and Ait-Kaci, 1989) and (Smolka et al, 1989), type-hierarchies are viewed as the least quasi-order obtained from a set of subsort declarations in a given signature. Such an order need not be a BCPO.

Types are incorporated into feature structures.

**Definition 2.2** Let  $H = \langle TYPE, \sqsubseteq \rangle$  be a type-hierarchy and let  $FEAT$  be a set disjoint with  $TYPE$ . A feature structure over  $H$  and  $FEAT$  is a tuple  $\langle Q, \bar{q}, \theta, \delta \rangle$  where:

- $Q$  is a set.
- $\bar{q} \in Q$  is the entry
- $\theta : Q \rightarrow TYPE$  is a total typing function
- $\delta : FEAT \times Q \rightarrow Q$  is a partial transition function

A feature structure  $\langle Q, \bar{q}, \theta, \delta \rangle$  can be viewed as a directed graph where the elements of  $Q$  are nodes, each labelled with the types specified by  $\theta$ , and where there is an arc, labelled  $f$ , from node  $q$  to node  $q'$  if  $\delta(f, q) = q'$ . This motivates using graph-theory terms such as ‘node’, ‘path’ and ‘root’ as well as depicting feature structures as graphs.

Carpenter’s definition of feature structures is more restrictive than ours in two respects. First, he allows only a finite number of nodes, and secondly, the entry node is required to be a root.

A well-known limitation of first order logic is the impossibility to axiomatize finiteness of a domain, except when explicitly stating the number of elements in it. Hence, we have no hope of axiomatizing feature structures in absolute accordance with Carpenter’s definition. However, it is not so obvious that feature structures should be finite. Others, for instance Johnson (1990), allow infinitely many nodes. Carpenter too includes a chapter on the generalisation to infinite feature structures.

Rootedness is defined as the existence of a node from which every other node is reachable. In our case the reachability-relation is the reflexive and transitive closure of the relation  $\{\langle d, d' \rangle \mid \delta(f, d) = d' \text{ for some } f \in FEAT\}$ . Another limitation of first order logic is that reflexive and transitive closures can not be axiomatized. Nor can we then axiomatize rootedness. At best, we could specify a relation that contains the closure. This approach is taken in the description logic SHIQ. However, it is intended for use as an ontology language for the semantic web (Baader et al, 2005). A main concern is then to strike a balance between expressiveness and the computational complexity of reasoning mechanisms and so it is acceptable to settle for ‘half-way’ solutions. This option is not open to us since our purpose is to provide a precise account rather than just trying to make the best of things. Again we escape the difficulties by observing that not all accounts agree on the question of rootedness. For instance, Johnson (1990) allows disconnected feature structures, while the PATR grammar formalism asserts the existence of a root (Shieber, 1992).

Thus, the framework of first order logic forces us to choose the more general approach, allowing infinite feature structures and dropping the requirement that the designated entry node should be a root.

An important use of types in feature structures lies in specifying that only certain nodes may possess a given feature.

**Definition 2.3** An appropriateness specification over the type-hierarchy  $\langle TYPE, \sqsubseteq \rangle$  and features  $FEAT$  is a partial function  $A : FEAT \times TYPE \rightarrow TYPE$  that meets the following conditions:

- For every feature  $f \in FEAT$ , there is a most general type  $Intro(f) \in TYPE$  such that  $A(f, Intro(f))$  is defined.
- upward closure and right monotonicity: If

$\mathcal{A}(f, \sigma)$  is defined and  $\sigma \sqsubseteq \tau$ , then  $\mathcal{A}(f, \tau)$  is also defined and  $\mathcal{A}(f, \sigma) \sqsubseteq \mathcal{A}(f, \tau)$ .

We may refer to  $\langle TYPE, \sqsubseteq \rangle$  and  $\mathcal{A}$  as a typing scheme (over FEAT).

Appropriateness-specification form a basis for coherence-conditions, such as the following (Carpenter, 1992).

**Definition 2.4** Let  $\langle TYPE, \sqsubseteq \rangle$  be a type-hierarchy and let  $\mathcal{A}$  be an appropriateness specification. A feature structure  $\langle Q, \bar{q}, \theta, \delta \rangle$  is said to be well-typed if whenever  $\delta(f, q)$  is defined,  $\mathcal{A}(f, \theta(q))$  is defined and  $\mathcal{A}(f, \theta(q)) \sqsubseteq \theta(\delta(f, q))$ .

Well-typedness is sometimes accompanied by the requirement that all appropriate transitions should be present.

**Definition 2.5** Let  $\langle TYPE, \sqsubseteq \rangle$  be a type-hierarchy and let  $\mathcal{A}$  be an appropriateness specification over FEAT. A feature structure  $\langle Q, \bar{q}, \theta, \delta \rangle$  is totally well-typed if and only if it is well-typed and if  $q \in Q$  and  $f \in FEAT$  are such that  $\mathcal{A}(f, \theta(q))$  is defined, then  $\delta(f, q)$  is defined.

### 3 Typed domains

In this section we address the question of how to embed type-hierarchies within the framework of first order logic. We want to view feature structures as certain first order structures with domains consisting of nodes. Then we must somehow impose the structure of the hierarchy onto the domain. We find the basis for our approach in *representation theory* (Davey and Priestley, 1991) where types are interpreted as sets in a natural way. This view has appeared in connection with a variety of ordering-relations, for instance in (Davey and Priestley, 1991; Smolka and Ait-Kaci, 1989; Smolka et al, 1989; Smolka, 1992). Carpenter (1992), however, does not explicate his type-hierarchies in these terms. So, we must address this question and need to resolve two issues: Should inconsistent types necessarily be associated with disjoint sets? Smolka (1992) requires that a top element  $\top$  should be interpreted as the empty set and that inconsistent types are interpreted as disjoint sets. On the other hand (Smolka and Ait-Kaci, 1989) and (Smolka et al, 1989) allow intersecting interpretations of inconsistent types. We shall discuss these alternatives. Initially we adopt the more general approach allowing intersecting interpretations of inconsistent types and nonempty interpretation of  $\top$ , if

present, before considering tightening the interpretation to consistent typing in the same vein as Smolka.

The second issue to address is whether the first order representation of types should be isomorphic to the original hierarchy. Let us begin by developing a straightforward representation theory for the type-hierarchies. At the core lies the basic definition of an interpretation of a type-hierarchy.

**Definition 3.1** If  $H = \langle TYPE, \sqsubseteq \rangle$  is a type-hierarchy and  $D$  is a set then a  $D$ -interpretation of  $H$  is a function  $f : TYPE \rightarrow 2^D$  assigning a subset of  $D$  to every type. Let  $D_\sigma^f$  denote the set assigned to a type  $\sigma$  by a  $D$ -interpretation  $f$ .

Sometimes it is convenient to illustrate interpretations of simple type-hierarchies by projecting the Euler-diagram of the interpreting sets onto the Hasse diagram representing the type-hierarchy, as shown in figure 1. However, we occasionally find it necessary to add a few comments to make such an illustration clear.

Definition 3.1 does not take into account the inheritance-information represented in the type-hierarchy. To obtain close correspondence between the type-hierarchy and its interpretation we need additional requirements. Given a  $D$ -interpretation  $f$  of a type-hierarchy  $H = \langle TYPE, \sqsubseteq \rangle$  we regard the partially ordered set  $\langle \{D_\sigma^f \mid \sigma \in TYPE\}, \supseteq \rangle$  as a *representation* of  $H$ . We shall look closer at two representation schemes with different degrees of correspondence with the type-hierarchy. Such representations will be referred to as *typed domains*. A typed domain is said to be *faithful* if it is isomorphic to the original hierarchy.

#### 3.1 Faithful representation

We shall interpret type-hierarchies as sets such that the subset-relation respects the partial order in the type-hierarchy. We do this by requiring that joins should correspond precisely to intersections of type-interpretations. Moreover, we want the interpretation of the bottom element to correspond to the universal type. That is, it should comprise ‘everything’ by demanding that a  $D$ -interpretation associates  $\perp$  with the entire set  $D$ .

**Definition 3.2** If  $D$  is a nonempty set,  $H = \langle TYPE, \sqsubseteq \rangle$  is a type-hierarchy and  $f$  a  $D$ -interpretation of  $H$  then  $f$  defines a faithfully  $H$ -typed domain iff the following conditions are satisfied:

- $D_\perp^f = D$

- $\sqcup\Sigma = \tau$  iff  $\bigcap_{\sigma \in \Sigma} D_\sigma^f = D_\tau^f$ , where  $\{\tau\} \cup \Sigma \subseteq TYPE$ .

Note that  $\sigma \sqsubseteq \tau$  iff  $\sqcup\{\sigma, \tau\} = \tau$ , so as a special case of definition 3.2 we get  $\sigma \sqsubseteq \tau$  iff  $D_\tau^f \subseteq D_\sigma^f$ .

Figure 1 illustrates an interpretation which satisfies definition 3.2.

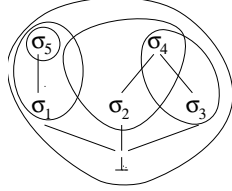


Figure 1: Interpretation satisfying definition 3.2. Note that the interpretation of  $\sigma_4$  is the intersection between the interpretations of  $\sigma_2$  and  $\sigma_3$ .

Inconsistency of types appears as a non-existing join in a type-hierarchy  $H$ , whereas in a faithfully  $H$ -typed domain defined by  $f$ , this inconsistency is reflected by their intersection not being assigned to a type by  $f$ . Note also that in order to satisfy definition 3.2,  $f$  can assign the empty set to no type other than a top element of  $H$ .

The representation defined by an interpretation satisfying definition 3.2 is in fact a consistently complete partial order. Specifically,  $\langle \{D_\sigma^f \mid \sigma \in TYPE\}, \supseteq \rangle$  is a (finite) BCPO, with  $\bigcap$  as join-function. The meet of a set  $X \subseteq \{D_\sigma^f \mid \sigma \in TYPE\}$  corresponds to

$$\min_{\sqsubseteq} \{Y \mid Y \in \{D_\sigma^f \mid \sigma \in TYPE\}, Y \supseteq \bigcup_{x \in X} x\}$$

Having established the particulars of the BCPO induced on a domain by an interpretation satisfying definition 3.2, we verify that it retains the original hierarchy.

**Theorem 3.1** *If  $H = \langle TYPE, \sqsubseteq \rangle$  is a type-hierarchy,  $f$  a  $D$ -interpretation of  $H$  and  $f$  defines a faithfully  $H$ -typed domain then  $\langle \{D_\sigma^f \mid \sigma \in TYPE\}, \supseteq \rangle$  and  $\langle TYPE, \sqsubseteq \rangle$  are isomorphic.*

The type-inheritance described by a type-hierarchy  $H$  and a faithfully  $H$ -typed domain correspond exactly to each other since they are isomorphic structures. Furthermore, the corresponding meet and join functions imitate each other in every detail. Thus, as far as subsumption, meet and join are concerned, faithfully typed domain provides perfect representation of type-hierarchies.

## 3.2 Degenerated representation

In this section we relax the requirement that typed domains should be faithful. There can be no isomorphic representation unless there are enough elements available to form different interpretations for all the types. Even if a representation contains a sufficient number of elements, the type-interpretations may be organised in a degenerated way.

In our context, type-representations shall be populated with nodes of feature structures. It may well be that a feature structure has too few nodes to obtain a faithful representation, in which case we may have to settle for a *degenerated* one. We use the term ‘typed domain’, dropping the qualifier ‘faithful’, when referring to such representation.

**Definition 3.3** *If  $H = \langle TYPE, \sqsubseteq \rangle$  is a type-hierarchy and  $f$  a  $D$ -interpretation of  $H$  then  $f$  defines an  $H$ -typed domain iff the following conditions are satisfied:*

- $D_\perp^f = D$
- $\sqcup\Sigma = \tau$  implies  $\bigcap_{\sigma \in \Sigma} D_\sigma^f = D_\tau^f$ , where  $\{\tau\} \cup \Sigma \subseteq TYPE$ .

As a special case of definition 3.3 we get that  $\sigma \sqsubseteq \tau$  implies  $D_\tau^f \subseteq D_\sigma^f$ . Figures 2 and 3 illustrate interpretations which satisfy definition 3.3 (but not definition 3.2).

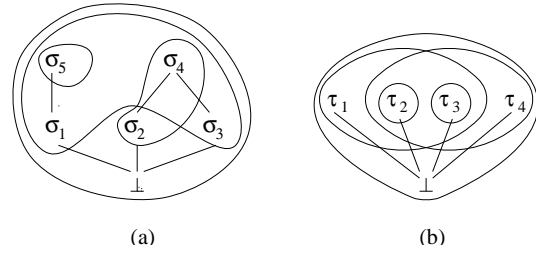


Figure 2: interpretations satisfying definition 3.3. Note that the interpretation of  $\sigma_4$  is the intersection between the interpretations of  $\sigma_2$  and  $\sigma_3$ .

On this approach, the ordering-relation is preserved but distinct types may be indistinguishable. Information about consistency may be lost and the representation need not even be a BCPO.

In figure 2(a) the types  $\sigma_1$  and  $\sigma_3$  are interpreted as the same set. Neither does definition 3.3 prohibit that the interpretation of a type  $\sigma$  contains the interpretation of a type incomparable with  $\sigma$ . Because of this, a typed domain may hold less information

about consistency than the type-hierarchy it represents. Furthermore, definition 3.3 does not guarantee that the corresponding representation is a BCPO. For instance, let  $f$  be the  $D$ -interpretation that defines the typed domain of figure 2(b). In the representation, both  $D_{\tau_2}^f$  and  $D_{\tau_3}^f$  are upper bounds for  $\{D_{\tau_1}^f, D_{\tau_4}^f\}$ , but there is no *least* upper bound for this set, so the BCPO-requirements are not met. Under these circumstances we can not expect a representation to be isomorphic to the type-hierarchy. On the other hand, the ordering relation will be preserved.

**Lemma 3.1** *If  $H = \langle TYPE, \sqsubseteq \rangle$  is a type-hierarchy and  $f$  a  $D$ -interpretation that defines an  $H$ -typed domain then  $f$  is a monotone map from  $\langle TYPE, \sqsubseteq \rangle$  to  $\langle \{D_\sigma^f \mid \sigma \in TYPE\}, \supseteq \rangle$*

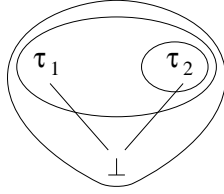


Figure 3: interpretation satisfying definition 3.3

The following illustrates that least upper bounds and greatest lower bounds are not preserved. Assume that  $f$  is the  $D$ -interpretation that defines the typed domain of figure 3, then

- The greatest lower bound of  $\{\tau_1, \tau_2\}$  is  $\perp$  but the greatest lower bound of  $\{D_{\tau_1}^f, D_{\tau_2}^f\}$  is  $D_{\tau_1}^f$
- The least upper bound of  $\{\tau_1, \tau_2\}$  is not defined while the least upper bound of  $\{D_{\tau_1}^f, D_{\tau_2}^f\}$  is  $D_{\tau_2}^f$

Having to live with the fact that a feature structure may contain too few nodes to provide a faithful representation, we may find comfort in that it is always possible to turn a degenerated representation into an isomorphic one while preserving the elements' memberships in type-interpretations. The transformation consists in simply adding 'junk'-elements to support the missing types.

In order to be able to talk about parts of a typed domain or the addition of elements, we introduce the concept of expansion/subinterpretation.

**Definition 3.4** *Let  $H = \langle TYPE, \sqsubseteq \rangle$  be a type-hierarchy and let  $f$  be a  $D$ -interpretation of  $H$ . An expansion of  $f$  is a  $D'$ -interpretation  $f'$  such that*

- $D \subseteq D'$

- $d \in D_\sigma^f$  iff  $d \in D'_\sigma^{f'}$  for all  $d \in D$  and  $\sigma \in TYPE$ .

We say that  $f$  defines a subinterpretation of  $f'$ .

If an expansion defines a faithfully typed domain it is said to be a *faithful expansion*.

**Theorem 3.2** *Let  $H$  be a type-hierarchy and let  $f$  be a  $D$ -interpretation of  $H$ .  $f$  defines an  $H$ -typed domain iff there is  $f'$  defining a faithfully  $H$ -typed domain such that  $f$  is a subinterpretation of  $f'$*

## 4 Typing functions

Recall our definition (2.2) of feature structures, where every node is associated with a type by means of a typing function. In this section we develop the corresponding idea for typed domains.

Obviously, an appropriate typing function should map a node to a type whose interpretation contains the node, but as we have seen in the previous sections, an element  $x$  of a typed domain may belong to the interpretation of several types. Some of these types may even be incomparable and therefore not well suited to be the designated type associated with  $x$ , unless some kind of overriding is employed. However, there is always a greatest type that offers a compromise solution to this difficulty. Specifically, the greatest type comparable with every other type whose interpretation contains  $x$ . Hence, for any element  $x$ , we can identify a unique designated type, referred to as *the* type of  $x$ . This provides a typing function.

**Definition 4.1** *Let  $H = \langle TYPE, \sqsubseteq \rangle$  be a type-hierarchy and let  $f$  be a  $D$ -interpretation that defines an  $H$ -typed domain. The typing function  $\theta$  is defined by*

$$\theta(x) = \max_{\sqsubseteq} \{ \sigma \mid x \in D_\sigma^f, \text{ for every type } \tau: \\ x \in D_\tau^f \text{ implies } \sigma \sqsubseteq \tau \text{ or } \tau \sqsubseteq \sigma \}$$

The typing function is illustrated in figure 4, where the interpretation of  $\sigma_5$  is the intersection between the interpretations of  $\sigma_2$  and  $\sigma_3$ . Note that  $d_2$  is contained in the interpretations of both  $\sigma_4$  and  $\sigma_5$ , neither of which is more suited than the other to be its designated type. As a 'compromise',  $\theta$  assigns  $d_2$  to  $\sigma_1$ .

Nodes that are not shared by interpretations of inconsistent types are easier to deal with. Then the type of the node is simply the join of the types whose interpretations the node belongs to. Moreover, this is the greatest such type.

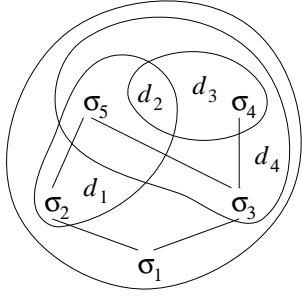


Figure 4:  $\theta(d_1) = \sigma_2$ ,  $\theta(d_2) = \sigma_1$ ,  $\theta(d_3) = \sigma_4$ ,  $\theta(d_4) = \sigma_3$

**Lemma 4.1** *Let  $H = \langle TYPE, \sqsubseteq \rangle$  be a type-hierarchy, let  $f$  be a  $D$ -interpretation defining an  $H$ -typed domain and let  $\theta$  be the corresponding typing function according to definition 4.1. Let  $d \in D$  and let  $\Sigma_d = \{\sigma \mid d \in D_\sigma^f\}$ . If  $\Sigma_d$  is consistent then  $\sqcup \Sigma_d = \max_{\sqsubseteq} \Sigma_d$  and  $\theta(d) = \max_{\sqsubseteq} \Sigma_d$*

We will get back to this observation in section 6.

## 5 First order typed feature structures

In this section we define typed feature structures in the framework of first order logic. The idea is to identify feature structures among the first order structures described in section 1.1. Domains will consist of nodes and we let the domains be typed in order to incorporate the type-hierarchy.

In section 3.2 we noted that it takes a certain number of elements to obtain an isomorphic representation of the type-hierarchy, and that a particular feature structure may have too few nodes. So, we can not expect the domain of a first order feature structure to be faithfully typed. This fact does not reflect any conflict between the feature structure and the type-hierarchy it is supposed to obey, only that there may be too few types involved to make use of the whole hierarchy. However, applying theorem 3.2 we find that any first order typed feature structure can be expanded to, and preserved within, a faithfully typed one.

**Theorem 5.1** *Let  $H = \langle TYPE, \sqsubseteq \rangle$  be a type-hierarchy and let  $\langle D, \llbracket \cdot \rrbracket \rangle$  be a structure over  $\widehat{TYPE} \cup \widehat{FEAT} \cup \{entry\}$ .  $\langle D, \llbracket \cdot \rrbracket \rangle$  is an  $H$ -typed feature structure iff it is a substructure of a faithfully  $H$ -typed feature structure  $\langle D', \llbracket \cdot \rrbracket' \rangle$*

In view of this we do not demand that the typed domain incorporated in the structure should be faithful.

First order structures are defined with respect to a language and, in our case, the language must contain symbols representing the types and features. Types will be treated as properties of nodes so our language must contain a unary relation-symbol  $\hat{\sigma}$  for every  $\sigma \in TYPE$ . Clearly, a feature can be seen as a partial function on the set of nodes. However, the function symbols of classical first order logic are required to be interpreted as *total* functions. We conform to this tradition by letting binary relation symbols, rather than unary function symbols, represent the features, at the expense of having to state their functional nature explicitly. Thus, for each feature  $f$  we include a binary function-symbol  $\hat{f}$ . We let  $\widehat{TYPE}$  and  $\widehat{FEAT}$  denote the sets of type- and feature-symbols. Finally we include in our language a constant *entry* to represent the entry node.

A first order structure  $\langle D, \llbracket \cdot \rrbracket \rangle$  interprets all non-logical symbols by means of  $\llbracket \cdot \rrbracket$ . Sometimes we wish to focus on the interpretation of types only. For this purpose we let  $\llbracket \cdot \rrbracket_T$  denote the restriction of  $\llbracket \cdot \rrbracket$  to  $\widehat{TYPE}$ .

Now we are ready to give a definition of typed feature structures in the framework of first order logic.

**Definition 5.1** *Let  $H = \langle TYPE, \sqsubseteq \rangle$  be a type-hierarchy,  $FEAT$  a set of feature and  $\langle D, \llbracket \cdot \rrbracket \rangle$  a first order structure over  $\widehat{TYPE} \cup \widehat{FEAT} \cup \{entry\}$ . Then  $\langle D, \llbracket \cdot \rrbracket \rangle$  is a first order typed feature structure (over  $H$  and  $FEAT$ ) iff the following conditions are satisfied*

- $\llbracket \cdot \rrbracket_T$  defines an  $H$ -typed domain.
- For all  $f \in FEAT$  and all  $d \in D$  there is at most one  $d' \in D$  such that  $\langle d, d' \rangle \in \llbracket f \rrbracket$

In order to specify when a first order typed feature structure is well-typed etc, we need to take a closer look at the connection between our two definitions of feature structures and see how we can view the new definition in light of the old one (2.2). For this purpose we translate first order typed feature structures into ordinary ones, i.e. according to definition 2.2.

**Definition 5.2** *Consider  $H$ ,  $FEAT$  and entry of definition 5.1 and let  $S = \langle D, \llbracket \cdot \rrbracket \rangle$  be a first order structure over  $\widehat{TYPE} \cup \widehat{FEAT} \cup \{entry\}$ . The translation of  $S$  is the tuple  $\langle D, \llbracket entry \rrbracket, \theta, \delta \rangle$  where  $\theta : D \rightarrow TYPE$  and  $\delta : FEAT \times D \rightarrow D$  are defined by*

$$\theta(x) = \max_{\sqsubseteq} \{\sigma \mid x \in \llbracket \hat{\sigma} \rrbracket_T, \text{ for every } \tau : x \in \llbracket \hat{\tau} \rrbracket_T \text{ implies } \sigma \sqsubseteq \tau \text{ or } \tau \sqsubseteq \sigma\}$$

and

$$\delta(f, d) = \begin{cases} d' & \text{if } \langle d, d' \rangle \in \llbracket \hat{f} \rrbracket \\ \text{undefined} & \text{otherwise} \end{cases}$$

The translation of a first order typed feature structure is a proper feature structure according to definition 2.2, but the converse does not hold. When applied to a structure that does not satisfy definition 5.1, the translation may well produce a feature structure.

The definitions of properties concerning our original feature structures can be transferred to the first order typed feature structures by means of the translation. I.e. a first order typed feature structure is well-typed iff its translation is well-typed, and similarly for other properties of feature structures.

## 5.1 A mismatch

We have just seen that every first order typed feature structure can be translated into an ordinary typed feature structure, so we may view a first order typed feature structure as a representation of an ordinary one. It should also be reasonably clear that every ordinary feature structure has such a representation. However, there is an unfortunate circumstance in that the translation is not one to one. An ordinary feature structure may be represented by several first order feature structures. Take for instance the feature structure  $\langle \{q_0, q_1, q_2\}, q_0, \theta, \delta \rangle$ , depicted in figure 5, over a type-hierarchy  $H$  and features  $\{f_1, f_2\}$ .

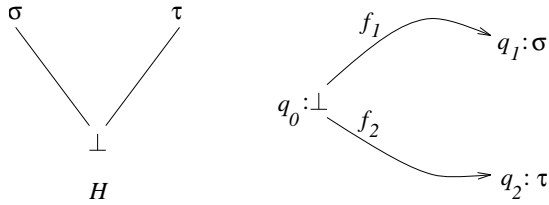


Figure 5: a feature structure over  $H$  with entry  $q_0$

Now, let  $\langle \{q_0, q_1, q_2\}, \llbracket \rrbracket \rangle$  and  $\langle \{q_0, q_1, q_2\}, \llbracket \rrbracket' \rangle$  be first order  $H$ -typed feature structures such that  $\llbracket \text{entry} \rrbracket = \llbracket \text{entry} \rrbracket' = q_0$ ,  $\llbracket f_1 \rrbracket = \llbracket f_1 \rrbracket' = \{\langle q_0, q_1 \rangle\}$  and  $\llbracket f_2 \rrbracket = \llbracket f_2 \rrbracket' = \{\langle q_0, q_2 \rangle\}$ . Also, let  $\llbracket \rrbracket_T$  define the  $H$ -typed domain depicted in figure 6(a) and let  $\llbracket \rrbracket'_T$  define the  $H$ -typed domain depicted in figure 6(b). In both cases the corresponding typing function maps  $q_0$  to  $\perp$ ,  $q_1$  to  $\sigma$  and  $q_2$  to  $\tau$ . Hence we have two different first order typed feature structures with the same translation. The cause of this effect will always be a nonempty intersection between interpretations of inconsistent types.

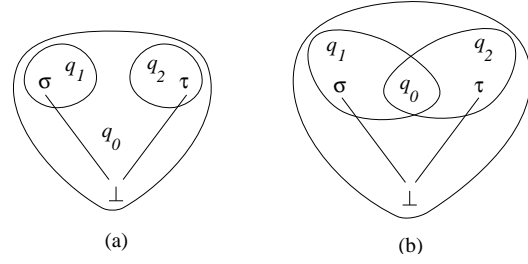


Figure 6:  $\{q_0, q_1, q_2\}$ -interpretations of  $H$  with equal typing functions

**Lemma 5.1** *Let  $H = \langle \text{TYPE}, \sqsubseteq \rangle$  be a type-hierarchy and  $\text{FEAT}$  a set of features. Let  $\langle D, \llbracket \rrbracket \rangle$  and  $\langle D, \llbracket \rrbracket' \rangle$  be distinct first order typed feature structures over  $\widehat{\text{TYPE}} \cup \widehat{\text{FEAT}} \cup \{\text{entry}\}$ . If they have the same translation then there is  $d \in D$  such that either  $\{\sigma \mid d \in \llbracket \hat{\sigma} \rrbracket\}$  or  $\{\sigma \mid d \in \llbracket \hat{\sigma} \rrbracket'\}$  is inconsistent.*

## 5.2 Well-typedness under inconsistency

First order typed feature structures are defined in terms of the ordinary feature structures such that properties are translated into the first order framework. However, this does not always work as well as we would like.

It turns out that in translation well-typedness becomes a stronger requirement than intended. The reason is that the typing function may have to pick a more general type than desired when a node belongs to the interpretations of inconsistent types. Consider the type-hierarchy in figure 4. Since the  $d_2$  is contained in the interpretations of both  $\sigma_4$  and  $\sigma_5$ , the typing function will associate it with the compromise type  $\sigma_1$ . But, there may be features specific for  $\sigma_4$  and  $\sigma_5$ , which are not appropriate for  $\sigma_1$ . This is unfortunate in the context of well-typedness, where a feature is required to be appropriate for *the* type of the node to which it belongs. Clearly, well-typedness of a first order typed feature structure may require that reasonable features for a node are omitted. There is a similar problem for the values of features.

These problems are rooted in the inconsistent typing scheme we adopted initially. Compromise types can not occur under consistent typing since then the type of a node corresponds to a unique maximal type. Cf lemma 4.1.

## 6 Consistently typed domains

We now consider tightening our typing scheme by demanding that the interpretations of inconsistent types should be disjoint. When we place these requirements on our typed domains we refer to them as being *consistently typed*.

**Definition 6.1** Let  $H = \langle TYPE, \sqsubseteq \rangle$  be a type-hierarchy and let  $f$  be a  $D$ -interpretation that defines an  $H$ -typed domain.  $f$  is said to define a consistently  $H$ -typed domain iff for every inconsistent  $\Sigma \subseteq TYPE$ ,  $\bigcap_{\sigma \in \Sigma} D_{\sigma}^f = \emptyset$

We have seen several indications that consistent typing has advantages in our setting. First, the typing function is simplified. It follows from lemma 4.1, that if  $f$  is a  $D$ -interpretation defining a consistently typed domain then  $\theta(d) = \max_{\sqsubseteq} \{\sigma \mid d \in D_{\sigma}^f\}$  for any  $d \in D$ .

This in turn removes the problems, reported in section 5.2, of well-typedness being a stronger condition than intended.

The HPSG grammar formalism is governed by the so-called *modelling convention* which involves the following, as Carpenter (1992) puts it: ‘According to this convention, the nodes of a feature structure are taken to represent objects, and we assume that every node is labelled with a type symbol which represents the most specific conceptual class to which the object is known to belong.’ This convention obviously corresponds precisely to  $\theta$  under consistent typing.

In section 5.1 we saw that the translation of first order feature structures is not one to one. When we restrict ourselves to consistently typed feature structures we obtain this property. That is, an ordinary typed feature structure is the translation of at most one first order typed feature structure.

**Lemma 6.1** Let  $H = \langle TYPE, \sqsubseteq \rangle$  be a type-hierarchy. Let  $\langle D, \llbracket \cdot \rrbracket \rangle$  and  $\langle D', \llbracket \cdot \rrbracket' \rangle$  be first order typed feature structures such that  $\llbracket \cdot \rrbracket_T$  and  $\llbracket \cdot \rrbracket'_T$  defines consistently  $H$ -typed domains and let  $F$  and  $F'$  be their translations, respectively. If  $F = F'$  then  $\langle D, \llbracket \cdot \rrbracket \rangle = \langle D', \llbracket \cdot \rrbracket' \rangle$ .

The translation is not only one to one, it is also onto. The construction of a consistently typed first order feature structure which translates to a given ordinary feature structure is a straightforward matter. Hence, translation defines a one to one correspondence.

Apparently, consistent typing is the solution to all the problems reported above. We conclude that first

order typed feature structures should have consistently typed domains and confine the remaining discussion accordingly.

## 7 Axiomatization

Having defined the first order typed feature structures and settled for a degenerated and consistent representation of the type-hierarchy, their axiomatization is relatively straightforward.

In the following, let  $H = \langle TYPE, \sqsubseteq \rangle$  be a type-hierarchy,  $\mathcal{A}$  an appropriateness specification over  $H$  and  $FEAT$ , and let  $L$  be a first order language over  $\widehat{TYPE} \cup \widehat{FEAT} \cup \{entry\}$ . Our goal is to develop axiomatizations in the form of sets of  $L$ -formulas.

Starting with the typed domains we define  $\Gamma_H$  to be the set

$$\{\forall x \hat{\perp}(x)\} \cup \{\forall x (\hat{\tau}(x) \leftrightarrow \bigwedge_{\sigma \in \Sigma} \hat{\sigma}(x)) \mid \sqcup \Sigma = \tau\}$$

which constitutes a set of axioms for typed domains

Axioms for consistent typing are now obtained by simply extending  $\Gamma_H$  with axioms that keep the interpretations of inconsistent types disjoint.

$$\Gamma_c = \{\forall x \neg (\bigwedge_{\sigma \in \Sigma} \hat{\sigma}(x)) \mid \text{inconsistent } \Sigma \subseteq TYPE\}$$

The models for  $\Gamma_c \cup \Gamma_H$  are precisely those where the interpretation of type symbols forms a consistently  $H$ -typed domain.

We now turn to the features. The interpretation of the relation symbols in  $\widehat{FEAT}$  must be restricted to correspond to partial functions. Define  $\Gamma_{feat}$  to be the set

$$\{\forall x \forall y \forall z ((\hat{f}(x, y) \wedge \hat{f}(x, z)) \rightarrow y \equiv z) \mid f \in FEAT\}$$

It should be clear that the models for  $\Gamma_H \cup \Gamma_c \cup \Gamma_{feat}$  are precisely the first order consistently typed feature structures.

We add axioms to capture well-typedness. Define  $\Gamma_W$  to be the set

$$\begin{aligned} & \Gamma_{feat} \cup \Gamma_H \cup \Gamma_c \cup \\ & \{\forall x ((\exists y \hat{f}(x, y)) \rightarrow \hat{i}(x)) \mid \iota = Intro(f)\} \cup \\ & \{\forall x \forall y ((\hat{f}(x, y) \wedge \hat{\tau}(x)) \rightarrow \hat{\tau}(y)) \mid \mathcal{A}(f, \tau) = \tau'\} \end{aligned}$$

$\Gamma_W$  is indeed an axiomatization of well-typed feature structures.

**Theorem 7.1** Let  $\langle D, \llbracket \cdot \rrbracket \rangle$  be a structure over  $\widehat{TYPE} \cup \widehat{FEAT} \cup \{entry\}$ .

$\langle D, \llbracket \cdot \rrbracket \rangle$  is a consistently well-typed feature structure iff  $\langle D, \llbracket \cdot \rrbracket \rangle \models \Gamma_W$

Finally we extend to *total* well-typedness. Define  $\Gamma_{TW}$  to be the set

$$\Gamma_W \cup \{\forall x(\hat{\sigma}(x) \rightarrow \exists y \hat{f}(x, y)) \mid \mathcal{A}(f, \sigma) \text{ defined}\}$$

which provides the axiomatization.

**Theorem 7.2** *Let  $\langle D, \mathbb{I} \rangle$  be a structure over  $\widehat{TYPE} \cup \widehat{FEAT} \cup \{entry\}$ .*

*$\langle D, \mathbb{I} \rangle$  is a consistently and totally well-typed feature structure iff  $\langle D, \mathbb{I} \rangle \models \Gamma_{TW}$*

Provided with axiomatizations we can easily cash in some results about typed feature structures. Let us take a look at their use in natural language parsing.

## 8 Feature constraints, satisfaction and decidability

When parsing, information about constituents is gathered through accumulation of the feature structures encountered during the process. The forming of a phrase from subconstituents imposes constraints on this accumulation such that the question of whether the phrase is grammatically well formed relies on whether the feature structures may be unified into one in a way that respects the constraints.

First order formulas are also suitable for feature description and in Johnson (1991) the parsing process accumulates information and requirements about phrase-constituents using formulas instead of feature structures. I.e. these formulas place constraints on the construction of a single feature structure and we refer to them as feature constraints. Then, unification corresponds roughly to a logical conjunction. A phrase is admitted as grammatically well formed if its accumulated feature constraint is satisfiable by a feature structure. That is, if the feature constraint in conjunction with the axiomatization of feature structures is satisfiable in the sense of first order logic.

From a computational point of view, it is of course important that the question of satisfiability is decidable. However, this is not always the case. Undecidability occurs in grammar-formalisms where functional uncertainty is incorporated in the constraint-language, in the form of regular expressions in path description. Then, if negation is also admitted, the language becomes too expressive, and the question of satisfiability is thus rendered undecidable. Consult Keller (1993) and Baader et al (1993).

In our case, the axioms for well-typed feature structures are completely within the SB-fragment of

first order logic, for which satisfiability is known to be decidable. I.e. the satisfiability of  $\Gamma_W \cup \{\varphi\}$  is decidable for any formula  $\varphi$  in SB. Hence, the feature constraint language may be SB entirely and still remain decidable. The axioms for totally well-typed feature structures goes beyond SB so we do not get decidability results for free. Actually, the feature constraints or the type-hierarchy must be restricted in order to retain decidability, but that is an other story.

## 9 Conclusion

Our goal has been to axiomatize Carpenter's typed feature structures within first order logic. The restrictions of the logical framework forced a confinement to a more general variety of feature structures, abandoning requirements for finiteness and rootedness.

We have observed that a particular feature structure may contain too few nodes to support the entire type hierarchy and we therefore relaxed the requirement that first order feature structures should contain an isomorphic representation of it.

We have discussed whether the typing scheme should allow the interpretations of inconsistent types to intersect. If so, the notion of well-typedness is apparently a stronger condition than Carpenter intended it to be. This suggests that consistent typing is preferable. We find another indication when we consider the modelling convention of HPSG in light of lemma 4.1. The labeling of each node with a type symbol which represents the most specific conceptual class to which the node is known to belong, corresponds precisely to the typing function in a consistently typed domain. Finally, consistent typing provides for a one to one correspondence between first order feature structures and the ordinary ones. We conclude that the consistent typing is the appropriate interpretation scheme of type hierarchies in our context.

We have presented axioms for well-typed and totally well-typed feature structures. The checking of constraints in constraint-based parsing can be reduced to checking the satisfiability of these axioms in conjunction with corresponding expressions in a feature constraint language. Since the axioms for well-typed feature structures are within the SB-fragment we have established that the satisfiability-check is decidable. Whereas the axioms for totally well-typed feature structures fall outside SB and we may not be allowed the full expressive power of SB for feature constraints.

## References

- Franz Baader, Hans-Jürgen Bürckert, Bernhard Nebel, Werner Nutt, Gert Smolka. On the Expressivity of Feature Logics with Negation, Functional Uncertainty, and Sort Equations. *Journal of Logic, Language and information* volume 2, No. 1, 1993
- Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, Peter Patel-Schneider (eds). *The description logic handbook*. Cambridge University Press, 2003
- Franz Baader, Ian Horrocs, Ulrike Sattler. Description logics as ontology languages for the semantic web. In D. Hutter, W. Stephan. *Mechanizing mathematical reasoning*. LNAI 2605, Springer Verlag 2005
- Bob Carpenter. *The Logic of Typed Feature Structures*. Cambridge University Press 1992
- Ann Copestake. Definitions of typed feature structures. *Natural Language Engineering* 6 (1), Cambridge University Press 2000
- Ann Copestake. *Implementing typed feature structures grammars*. CSLI Publications, 2002
- B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press 1991
- Burton Dreben, Warren D. Goldfarb. *The Decision Problem: Solvable Classes of Quantificational Formulas*. Addison-Wesley 1979
- Mark Johnson. Expressing Disjunctive and Negative Feature Constraints with Classical First-Order Logic. The Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics 1990
- Mark Johnson. Features and Formulae. *Computational Linguistics* 17(2) 1991
- Bill Keller. *Feature Logics, Infinitary Descriptions and Grammar*. CSLI Lecture Notes Number 44, 1993
- NLE Special issue on HPSG Parsing. *Natural Language Engineering* 6 (1), 2000
- Carl Pollard, Ivan Sag. *Head-driven phrase structure grammar*. Chicago University Press, 1994
- Stuart M. Shieber. *Constraint-Based Grammar Formalisms*. The MIT Press, 1992
- Gert Smolka. A Feature Logic with Subsorts. In J. Wedekind and C. Rohrer (eds.) *Unification in Grammar*. The MIT press, 1992
- Gert Smolka, Hassan Aït-Kaci. Inheritance Hierarchies: Semantics and Unification. *Journal of Symbolic Computation* 7, 1989
- Gert Smolka, Werner Nutt, Joseph A. Goguen, José Meseguer. Order-Sorted Equational Computation. In Aït-Kaci, Nivat. *Resolution of Equations in Algebraic Structures Volume 2*. Academic Press 1989