In: Proc. of 4. Internationaler Gi-Kongreß Wissensbasierte Systeme

Linking Typed Feature Formalisms and Terminological Knowledge Representation Languages in Natural Language Front-Ends

Rolf Backofen, Harald Trost, Hans Uszkoreit
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Stuhlsatzenhausweg 3
6600 Saarbrücken 11
e-mail: backofen@dfki.uni-sb.de

Abstract

In this paper we describe an interface between typed feature formalisms and terminological languages like KL-ONE. The definition of such an interface is motivated by the needs of natural language front-ends to AI-systems where information must be transmitted from the front-end to the back-end system and vice versa.

We show how some minor extensions to the feature formalism allow for a syntactic description of individual concepts in terms of typed feature structures. Namely, we propose to include intervals and a special kind of sets. Partial consistency checks can be made on these concept descriptions during the unification of feature terms. Type checking on these special types involves calling the classifier of the terminological language. The final consistency check is performed only when transferring these concept descriptions into structures of the A-Box of the terminological language.

1 Introduction

Typed feature formalisms[1] are currently the most successful means for representing linguistic knowledge. There is even a tendency to extend their use to linguistic levels like semantics (e.g. in HPSG [PS87]) and phonology (e.g. [Col90]) and [Wie90]) which were traditionally described in different notations.

A number of terminological languages like KL-ONE[2] have been implemented during the last few years. They are theoretically well-understood and are widely used for the representation of world and domain knowledge in various types of AI-systems.

For natural language front-ends which are used in dialog situations, a continous communication with the back-end system (e.g. a planning component) is required. Consequently, there is a need to transmit pieces of information between these two components. Since each component encodes information in its own representation language it must be possible to exchange information between these two languages.

One can think of two different approaches for managing this communication process. The first one is to create a unique formalism to represent both kinds of knowledge. But this seems not to be very promising because the resulting formalism becomes to powerful to allow for efficient processing. Moreover, the modularity of the different knowledge bases would not be preserved, i.e. the natural

[1] for an overview see, e.g., [Shi86] or [Smo88]

[2] for an overview see, e.g., [NS90]

language front end would impose too strict requirements on the representation formalism the back-end system makes use of.

Therefore we opt for the second strategy, namely linking both formalisms via a syntactic translation. To tyhis end we encode concept descriptions in a special subtype of feature structures which entails the concept type, the set of role-value-maps, and role descriptions. Such a role description will contain the information associated to one role, for example the number restriction and the value restriction. Furthermore we have to provide for the ability to invoke the KL-ONE realizer on the translation of such structures.

We have organized the paper in such a way that we start by giving a syntax and a semantics for both feature formalism and terminological language. We then discuss the differences between the two formalisms. We sketch the workings of our interface which in turn motivates some minor extensions to the feature formalism. These extensions are discussed in some detail. At this point we are in the situation to describe how concept descriptions from the terminological language can be represented in terms of typed feature structures in this extended formalism. Finally we show how a partial consistency check on these concept descriptions can be performed during unification.

2 Typed Feature Logic

Typed feature formalisms as they are used in todays language processing systems have evolved from directed acyclic graphs (e.g., the PATR system [Kar86]). To allow for a more adequate description of linguistic data that formalism was extended in several ways. One very important extension was to allow for the use of disjunction. Another extension was the integration of types or sorts into the formalism. Many other extensions have been proposed and implemented in various systems. In the following we will define a core feature formalism which is provided with all the features which are important with regard to the interface definition.

2.1 Syntax

For the basic definition of feature terms we assume a *signature* Σ, which consists of a set of *variables* $\mathcal{V}$ (written $x, y, \ldots$), *features* $\mathcal{F}$ (written $f, g, \ldots$) and *atoms* $\mathcal{A}$ (written $a, b, \ldots$). Additionally, the language allows for the use of type symbols $A, B, C, \ldots \in \mathcal{T}$. On $\mathcal{T}$ a partial order $\preceq$ is defined with $\top \in \mathcal{T}$ as the greatest and $\bot \in \mathcal{T}$ as the least element (usually called *top* and *bottom*, respectively). The operator $\prec$ induces a lower semilattice on $\mathcal{T}$ (that means for every $A, B \in \mathcal{T}$ the greatest lower bound $GLB(A, B)$ is in $\mathcal{T}$). All these sets are pairwise disjoint.

Although feature terms can be seen as data objects with some internal structure, it is formally handier to describe them as complex constraints built out of primitive ones using conjunction and disjunction. The set of all *feature terms* is then given by the following context-free production rules:

$$
\begin{array}{rcll}
s, t & \longrightarrow & A & \text{a sort} \\
 & | & x & \text{a variable} \\
 & | & a & \text{an atom} \\
 & | & f : x & \text{selection} \\
 & | & \neg x & \text{negated coreference} \\
 & | & s \sqcap t & \text{conjunction} \\
 & | & s \sqcup t & \text{disjunction}
\end{array}
$$

2.2 Semantics

There is a set theoretic semantics for feature terms, which is defined in terms of interpretations (see e.g. [Smo88]. An interpretation $\mathcal{I} = (D^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a *domain* $D^{\mathcal{I}}$ and an interpretation function such that the following conditions are satisfied:

1. $\top^{\mathcal{I}} = D^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$

2. for all sorts $A, B : GLB(A,B)^{\mathcal{I}} = A^{\mathcal{I}} \cap B^{\mathcal{I}}$

3. every feature f is map to a function $f^{\mathcal{I}} : D^{\mathcal{I}} \mapsto D^{\mathcal{I}}$

4. for every feature f and every atom $a : a \notin \mathbf{ran}(f^{\mathcal{I}})$

For assigning a meaning to an expression containing variables, it is necessary to introdoce *variable assignment*. An assignment α is a function $\alpha : \mathcal{V} \mapsto D^{\mathcal{I}}$ and maps every variable of $\mathcal{V}$ to an element of the interpretation domain. The *denotation* of a feature term s under a valuation α in $\mathcal{I}$ is a subset of $D^{\mathcal{I}}$, which is defined inductively as:

1. $[\![x]\!]^{\mathcal{I}}_{\alpha} := \{\alpha(x)\}$ for a variable x,

2. $[\![a]\!]^{\mathcal{I}}_{\alpha} := \{a^{\mathcal{I}}\}$ for an atom a,

3. $[\![f : t]\!]^{\mathcal{I}}_{\alpha} := \{d \in D^{\mathcal{I}} \mid f^{\mathcal{I}}(d) \in [\![t]\!]^{\mathcal{I}}_{\alpha}\}$

The denotation $[\![s]\!]^{\mathcal{I}}$ of s in $\mathcal{I}$ is defined as

$$\bigcup_{\alpha \text{ valuation on } \mathcal{I}} [\![s]\!]^{\mathcal{I}}_{\alpha}$$

A feature term s is *consistent*, if there is an interpretation $\mathcal{I}$ with $[\![s]\!]^{\mathcal{I}} \neq \emptyset$.

2.3 Subsumtion and unification

Computationally, the two main operations are *subsumtion* and *unification*. A feature term s is said to be subsumed by a feature term t (abrev. $s \sqsubseteq t$) iff in every interpretation $\mathcal{I}$ the denotation of s is a subset of $[\![t]\!]^{\mathcal{I}}$. The relation $\sqsubseteq$ induces a lower semilattice that can be viewed as the extension of the type hierachy to the set of feature terms.

The most commonly used operation is unification. Unification takes two different terms as arguments and decides whether the conjunction of both terms is consistent. This consistency check is performed by rewriting the conjunction into a so-called *solved normal form*. This form is also the result of the unification operation. If during rewriting a clash occurs, the conjunction is inconsistent and unification fails. For details on rewriting rules see e.g. [Smo88] or [Smo89]

3 Terminological languages

Knowledge representation systems of the KL-ONE family make a distinction between terminological and assertional knowledge. The first one is stored in the so-called *T-Box* and describes the world (or domain) knowledge. The latter one is gathered in the so-called *A-Box* and describes the actual state of the world (or domain).

Although there exist many different systems with different syntax, one can define an abstract KL-ONE system, the properties of which are shared by most of the existing systems. The terminological formalism consists of a concept description language in order to define concepts and relations between concepts. The relations are called roles and are always binary. Main parts of the following abstract definition are taken from [Hol90].

3.1 Syntax and semantics of our terminological language

We assume two disjoint alphabets of symbols, called *concepts* (written A, B) and *roles* (denoted by R, S). There are two special concept symbols $\top$ and $\bot$. Then *concept descriptions* (written C, D) are defined by the following production rules:

$$\begin{array}{rcll} C, D & \longleftrightarrow & A & \text{atomic concept} \\ & | & C \sqcap D & \text{conjunction} \\ & | & \forall R.C & \text{value restriction} \\ & | & (\geq n\ R) \mid (\leq n\ R) & \text{number restrictions, } n \in \mathbb{N} \end{array}$$

An interpretation $\mathcal{I} = (D^{\mathcal{I}}, \mathcal{I}[\cdot])$ of a concept description consists of a set $D^{\mathcal{I}}$ (the domain of $\mathcal{I}$) and an interpretation function $\mathcal{I}[\cdot]$ such that the following holds:

1. For every concept description C and for every role R: $\mathcal{I}[C] \subseteq D^{\mathcal{I}}$ and $\mathcal{I}[R] \subseteq D^{\mathcal{I}} \times D^{\mathcal{I}}$.

2. $\mathcal{I}[\bot]$ is the empty set and $\mathcal{I}[\top]$ the whole domain.

3. The conjunction $\sqcap$ is interpreted as set intersection.

4. The following equations are satisfied:

$$\begin{array}{rcl} \mathcal{I}[\forall R.C] & = & \{a \in D^{\mathcal{I}} \mid \forall (a,b) \in \mathcal{I}[R] : b \in \mathcal{I}[C]\} \\ \mathcal{I}[(\geq n\ R)] & = & \{a \in D^{\mathcal{I}} \mid |\{b \in D^{\mathcal{I}} \mid (a,b) \in \mathcal{I}[R]\}| \geq n\} \\ \mathcal{I}[(\leq n\ R)] & = & \{a \in D^{\mathcal{I}} \mid |\{b \in D^{\mathcal{I}} \mid (a,b) \in \mathcal{I}[R]\}| \leq n\} \end{array}$$

Consistency and subsumtion of concept descriptions are defined as in the feature logic formalism. With the notion of concept description we can now define the T-Box and the A-Box. The T-Box consists of a finite set of concept definitions. Each definition has either the form $A \doteq C$ or $A \sqsubseteq C$, where A is a concept and C is a description. An interpretation $\mathcal{I}$ is a model of a terminolgy (T-Box) iff every definition holds in $\mathcal{I}$. This is equivalent to the following conditions:

$\mathcal{I}[A] \subseteq \mathcal{I}[C]$ for every $A \sqsubseteq C \in$ T-Box
$\mathcal{I}[A] = \mathcal{I}[C]$ for every $A \doteq C \in$ T-Box.

Now let's turn to the assertional part (the A-Box). As previously mentioned,the A-Box describes the actual state of the world. This is done in terms of individuation, that means introducing individual objects. The A-Box contains a finite sets of propositions about these individuals. Each proposition states either that one individual is of some sort or that two individuals are connect by a role. The syntax for those propositions is given by

$a : A$ (sort subsumtion)
$(a, b) : R$ (role instantiation)

where a, b are individuals.

Formally, individuals are treated as constants and the interpetation function is extended to these constants. The notion of model is also extended to the A-Box in a straightforward way.

Supplementary, some systems contain another kind of constraints in the terminological formalism, namely *role value maps*. With role value maps one can enforce the equivalence of two sets of elements. Each set is obtained by successively following a chain of roles starting from the same element.

3.2 Computational services

There are two kinds of operations which are usually available in terminological formalisms, namely *classifier* and *realizer*. Classifying a T-Box means to calculate the subsumtion hierarchy of concepts. With the realizer one determines for a given individual the least concept the individual is subsumed by[3].

4 Differences between the two formalisms

Although feature logic formalisms and terminological languages have a similar semantics there are significant differences. The most important of these differences are:

[3] For a precise description of these features see [Hol90].

- Feature formalisms use functional roles while terminological languages allow for relational roles, where the cardinality of the filler set may be restricted by an integer interval called number restriction.
- To express the fact that certain roles must have an identical filler feature formalisms use the notion of coreference (there is also current research on integrating negated coreference). Most terminological languages employ the somewhat broader concept of role value maps where a number of operators besides equality may be used.
- Terminological languages distinguish between conceptual and assertional level (i.e. concept vs. instance) while feature formalisms do not make this distinction.
- Typed feature formalisms usually support general disjunction (and sometimes negation) while only a very limited notion of disjunction is available in terminological languages (at least at the level of terminological description).

In conclusion, one may say that terminological languages tend to be more expressive than typed feature formalisms. But for most tasks in natural language processing the expressiveness of typed feature formalisms is adequate.

5 Linking the two formalisms via a syntactic translation

As mentioned, we want to encode concept descriptions within feature structures, which can be translated into the KL-ONE system whenever necessary (these objects are called *syntactic concept descriptions*). The semantics of such an object will be the set of all instances in the KL-ONE systems, which satisfies the translation of the syntactic description. During unification we want to do partial consistency check. A full consistency check is done by evaluating the translation of the description within the KL-ONE system.

To this end we need to extend the expressive power of the feature logic by numeric intervals for encoding the number restriction and some sort of set values for describing filler sets. The extension should be easily integrable whithin an existing unification formalism. Although both extensions can be integrated independently they will influence each other in our context. This has some effects on the way such constraints are evaluated. Although one can think of a combine constraint, which partially describes the filler set *and* the lower and upper bound of its cardinality, we didn't choose this alternative. One reason for us to keep both distinct is that we do want to keep the extension of the feature logic as simple as possible. Another reason is that these constraints can be used for other purposes too (e.g. one could use the intervals to encode position features).

5.1 Extensions to feature logic for coping with concept descriptions

5.1.1 Intervals

The first extension are numeric intervals. The syntax of such intervals is given by

$$[i..j] \quad \text{with } i \in \mathbb{N}_0, j \in \mathbb{N} \cup \{\infty\} \text{ and } i < j$$

The semantics of the interval constraint is just the set of all (natural) numbers in the range of the interval.

The rewriting rules for intervals have the following form:

$(R1)$ $[i_1..j_1] \sqcap [i_2..j_2] \longrightarrow [\mathbf{max}(i_1, i_2), .., \mathbf{min}(j_1, j_2)]$

$(R2)$ $[i..j] \sqcap a \longrightarrow a$

$(R3)$ $[i..j] \sqcap f : s(\text{resp. } \sqcap A\ ) \longrightarrow \bot.$
This means that no features are defined on intervals and that they are not element of some type.

5.1.2 Set values

The second extension are set values which are necessary for the description of sets of possible fillers. One can distinguish different ways to treat the cardinality of set values. Cardinality can be restricted in two different ways: via abstraction or via enumeration. Abstraction means to use an additional constraint restricting the cardinality of the set value (e.g. the cardinality is between n and m, $n < m$). But this seems to make no sense in our context. Let $s_1, \ldots, s_n$ be some description of the elements of the set value. As the cardinality is only restricted by abstraction and not by enumeration, there could be additional elements not mentioned yet. Moreover, because feature terms are only partial descriptions, some of the s_i could denote the same element (the set may even shrink to a single element in extreme cases). Consequently, such a kind of set value is too vague in order to be useful.

For enumeration there are two different possibilities. Let again $s_1, \ldots, s_n$ be some description of the elements of the set value.In the first case every element of the interpretation of the set value must fit into some description of s_i. Again some s_i could collapse. Therefore n is only an upper bound for the cardinality (for an example of such set values see [PM89]). But in terminological languages filler sets may have no upper bound (i.e. the upper bound equals ∞). Such a situation cannot be modeled with this approach.

Therefore we have decided for a second possibility. Here $s_1, \ldots, s_n$ are an an enumeration of *distinct* elements of the set value. As a result n defines the lower bound for the cardinality of the set value. This means, that the unique name assumption has to be applied to $s_1, \ldots, s_n$.

Now let's turn to the definition of these set values. Set values can appear at every point within feature structures. The syntax is given by

$$\begin{array}{rcll} s, t & \longrightarrow & \ldots & \\ & | & \{s_1, \ldots, s_n\} & \text{sets} \end{array}$$

where $s_1, \ldots, s_n$ are feature terms.

For the set values we have the following semantics:

$$[\![\{s_1, \ldots, s_n\}]\!]^{\mathcal{I}}_{\alpha} := \{m \in \mathbf{2}^{D^{\mathcal{I}}} \mid \exists d_1..d_n : \bigwedge_{i \neq j} d_i \neq d_j \;\wedge\; \forall i : [d_i \in [\![s_i]\!]^{\mathcal{I}}_{\alpha} \wedge d_i \in m]\} \tag{1}$$

The first condition in (1) is exactly the before-mentioned unique name assumption for the objects w.r.t the set-value[4]. With this condition one ensures that set values will never shrink.

Our sets are lower bound by enumeration and one could not give an upper bound for the set value by enumeration at the same time[5]. Therefore the semantics of a set value is the set of all possibly extensions of $\{[\![s_1]\!]^{\mathcal{I}}_{\alpha}, \ldots, [\![s_n]\!]^{\mathcal{I}}_{\alpha}\}$ (c.f. the second condition in (1)).

Given this semantics for set values, the conjunction of two set values (unification) describes all sets which at least contain all the elements of both set values. Because these elements are only partially described some of the elements of each set value could collapse.

This leads to the following extensions of the rewriting rules:

($R4$) $x \sqcap t\langle\{\ldots, x \sqcap t, \ldots\}\rangle \longrightarrow \bot$ Here $t\langle s\rangle$ denotes a pure conjunctive term which has s as an subterm.

($R5$) $\{\ldots, x, \ldots, x \ldots\} \longrightarrow \bot$

---

[4] We think that it in the framework of linguistic processing a unique name assumption in general makes no sense. Such an assumption would state that in every interpretation the denotation consist of a singleton set. Although one can think of introducing such an assumption for some elements of the set value, this could not be applied to all members of set values, because then no elements of different set values could ever collapse.

[5] This would lead to sets with fixed arity. Besides the fact that this is not suitable for describing set values such set values would not describe real sets anymore. Such set values would best be interpreted as fixed arity terms.

$(R6)$ $\{z_1, \ldots, z_l, s_1 \ldots, s_n\} \sqcap \{z'_1, \ldots, z'_l, t_1 \ldots, t_m\} \longrightarrow$

$$\bigsqcup_{k=1}^{\min(n,m)} \bigsqcup_{\substack{I \subseteq \{1,\ldots,n\} \\ J \subseteq \{1,\ldots,m\} \\ |I| = |J| = k}} \bigsqcup_{\phi \in I^J} \{ \begin{array}{l} (z_1 \sqcap z'_1), \ldots, (z_l \sqcap z'_l), (s_{i_1} \sqcap t_{\phi(j_1)}), \ldots, (s_{i_k} \sqcap t_{\phi(j_k)}), \\ x_{i_{k+1}}, y_{j_{k+1}}, \ldots, x_{i_n}, y_{j_m} \} \end{array}$$

where $(z_1, z'_1), \ldots, (z_l, z'_l)$ is the set of element pairs which share the same variable, $i_1, \ldots, i_k$ resp. $j_1 \ldots, j_k$ some arbitrary but fixed enumeration of I resp. J and $i_{k+1}, \ldots, i_n$ resp. $j_{k+1} \ldots, j_m$ an enumeration of the remaining elements.

Here are some comments on this set of rules. Rule R4 guarantees that there are no cycles envolving set values. This guarantees that the given set rewrite rules will always terminate[6]. Rule R5 is the before-mentioned unique name assumtion. Instead of introducing this rule we could have stated this assumtion using negated coreferences.

Now let us turn to R6, the most complex rule. Applying R6 during every unification is neither intended nor would the resulting algorithm be tractable. In this rule the value of k is the number of elements that have to be identified in order to get a set value with lower bound $n + m - k$. The problem that occurs during unification of set values is that one has different descriptions for the elements of the generated set value. In the regular case many of them will be identified during further linguistic processing. This enlarges the set of common elements $z_1 \ldots z_l$ which will in turn make it easier to apply this rule. Therefore we delay evaluation of rule R6.

Moreover, applying this rule would not be necessary if there where no restriction on the cardinality of the set. But in our application this is the case, because the cardinality of the filler set is constrained by the number restriction. For checking the consistency it suffices to identify as many elements as necessary to satisfy the number restriction.

5.2 Encoding of concept description

For the encoding of concept descriptions we use a special class of feature terms, which have some given structure[7]. All such special feature terms are typed by a KL-ONE concept. The features which are defined on such terms correspond to role names of the KL-ONE concepts and have again some distinguished structure. We will call a subclass of feature terms *role descriptions*. Role descriptions consist of three different feature-value pairs. The first entry contains a numeric interval for the number restriction, the second stores the value restriction. The last entry contains a set value for the description of the filler set, whose members are again syntactic concept descriptions.

The encoding of the concept descriptions is organized such that the unification of two syntactic concept descriptions results in a syntactic concept description the translation of which is equivalent to the conjunction of the translation of the input structures. Furthermore, unification does some partial consistency check:

1. combining the concept types (by calling the classifier in order to find the glb of both types)
2. unifying role descriptions of roles shared by both input structures. Within this process the type of the value restriction is calculated (see 1) and the conjunction of the number restrictions is determined by interval intersection.

A full consistency check is made transferring such structures to the KL-ONE System. As we have shown, it is sensible to delay the unification of set values until such a full consistency check.

Now let's turn to the translation of such syntactic concept descriptions. A straight-forward semantics for the translation is to assign a fixed KL-ONE instance to each syntactic concept description.

[6]We assume that allowing cycles involving set values would lead to undecidability (because of the simliarity of coreference and role value-maps of length 1)

[7]Such structural information could be stated using type definitions and closed types

But the notion of an underspecified KL-ONE instance does not fit to the notion of underspecification used in feature logics. A KL-ONE instance is underspecified in the sense, that it denotes one specific object the properties of which are only partially known. Underspecification in feature logic on the other hand means that a description denotes the set of all objects satisfying the description. Different descriptions can be satisfied by the same object, whereas different instances can never be equal.

This leads to an interpretation of concept descriptions as the set of all instances in the terminological language, which satisfies the translation of the description. But, as mentioned, only a partial consistency check is made during unification, which means that the set is possibly empty. There are two ways to check whether there exists some element satisfying the translation of a description. The first one is to classify the translated description within the T-Box of the KR-system. But we assume that it is not useful to change the terminology during processing. Therefore we use the second approch via Skolemisation, namely to create a new instance satisfying the description.

For the translation of a syntactic concept description we have to translate the syntactic concept descriptions that are contained in the set values of the filler entries. To do this we have perform the delayed set value unifications for this entry. As mentioned in the discussion of rule R6, we have to identify only as many elements such that the resulting filler set fits into the number restriction.

6 Conclusion

In work on natural language front-ends there is the problem of exchanging information with the back-end system. A prerequisite for such an exchange is that the respective knowledge bases of front-end and back-end are compatible with each other.

In this paper we have described a method for the linking of typed feature formalisms and terminological languages. The basic idea is to describe structures from the terminological language syntactically in the feature formalism. Such a syntactic description is possible with only minor extensions to the feature formalism. Furthermore, we can perform a partial consistency check on these structures during unification which helps in reducing spurious ambiguities at an early stage of processing.

Such a method of linking the two formalisms via an explicit interface is preferable to creating a unique more powerful formalism for two reasons. First, the expressiveness of terminological languages is not necessary at most stages of linguistic processing. Second, the interface approach leads to a more modular systems architecture because the back-end system may keep its own distinct formalism.

References

[Col90] J. Coleman. Unifikation phonology. In H. Karlgren, editor, *COLING-90*, volume 3, pages 79–84. Helsinki, 1990.

[Hol90] Bernhard Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. Research Report RR-90-06, Deutsches Forschungszentrum für künstliche Intelligenz, Saarbrücken, May 1990.

[Kar86] Lauri Karttunen. D-PATR: a development environment for unification-based grammars. In *Proceedings of the COLING'86*, volume 1, pages 74–80. Budapest, 1986.

[NS90] Bernhard Nebel and Gert Smolka. Representation and reasoning with attributive descriptions. In K.H. Blaesius, U. Hedtstueck, and C.R. Rollinger, editors, *Sorts and Types in Artificial Intelligence*. Springer, Berlin, 1990.

[PM89] Carl J. Pollard and M. Drew Moshier. Unifying partial descriptions of sets. In P. Hansen, editor, *Information, Language and Cognition*, volume 1 of *Vancouver Studies in Cognitive Sience*. University of British Columbia Press, Vancouver, 1989.

[PS87] Carl Pollard and Ivan A. Sag. *Information-Based Syntax and Semantics. Vol. 1: Fundamentals*, volume 13 of *CSLI Lecture Notes*. Chicago Univ. Press, Chicago, 1987.

[Rou88] William Rounds. Set values for unification-based grammar formalisms and logic programming. Report CLSI-88-129, CSLI, Stanford (CA), June 1988.

[Shi86] Stuart M. Shieber. *An Introduction to Unification-Based Approaches to Grammar*, volume 4 of *CSLI Lecture Notes*. Stanford University, Stanford (CA), 1986.

[Smo88] Gert Smolka. A feature logic with subsorts. LILOG-Report 33, IBM Deutschland GmbH, Stuttgart, May 1988.

[Smo89] Gert Smolka. Feature constraint logics for unification grammars. LILOG-Report 93, IBM Deutschland GmbH, Stuttgart, November 1989.

[Wie90] R. Wiese. Towards an unifikation-based phonology. In H. Karlgren, editor, *COLING-90*, volume 3, pages 283–286. Helsinki, 1990.