Unification and Default Unification

Gosse Bouma
Computational Linguistics
Rijksuniversiteit Groningen
Postbus 716
9700 AS Groningen
The Netherlands
gosse@let.rug.nl

June 29, 2004

Abstract

Unification of (typed) feature structures is an important tool for capturing linguistic generalizations. It is the core of a family of linguistic formalisms, it has been applied in phonology, morphology, syntax, and semantics, and is dominant in most linguistically informed work in natural language processing. Unification monotonically merges information. Default unification provides a non-monotonic counterpart of this operation, which can be used to streamline the definition of grammars and lexica, as it supports concise encodings of generalizations as well as exceptions to generalizations.

1 Introduction

Writing a contex-free grammar for a (fragment of) natural language can be a rather tedious and linguistically non-gratifying job. Some phenomena, such as agreement, are hard to describe using the standard machinery of context-free grammar. Other phenomena, such as the grammar of WH-movement, may seem even impossible to implement.

A related problem comes from the development of (wide coverage) lexica for formal or computational grammars. Grammars for natural language tend to make quite detailed assumptions about the information available in lexical entries. While for theoretical purposes it may suffice to specify *which* type of information one considers to be lexical, a lexicon that needs to be used in practical applications actually *has to provide* this information, usually for large number of lexical items. Although the amount of information that has to be present in individual lexical items can be substantial, the differences between large groups of lexical items are often small and predictable. This suggests that lexica for computational grammars can and should employ mechanisms which ensure that information can be shared among lexical items.

For these reasons, the use of feature-based grammar formalisms, which employ unification as the sole mode for sharing and checking information, has become wide-spread, both in theoretical linguistics and in computational linguistics. Unification grammar allows for concise descriptions of agreement and other phenomena, including constructions which are beyond the descriptive power of context-free grammar. Lexical items can be associated with information-rich feature structures. Commonalities between these structures can be expressed using inheritance.

Linguists have always felt the need to use indices, variables, and features in their theoretical frameworks. Unification grammar provides the formal underpinning for these notational devices, which are often presented as straightforward extensions of context-free grammar. Unification is also the cornerstone of a family of linguistic theories, such as Lexical Functional Grammar (Bresnan, 1982), Generalized Phrase Structure Grammar (Gazdar et al., 1985), and Head-driven Phrase Structure Grammar (Pollard and Sag, 1994) and has been fruitfully applied to Categorial

$$\text{a.}\ \begin{bmatrix} \text{CAT} & \mathit{np} \\ \text{AGR} & \begin{bmatrix}\text{NUM} & \mathit{pl} \\ \text{PER} & \mathit{1st}\end{bmatrix}\end{bmatrix} \qquad \text{b.}\ \begin{bmatrix}\text{SUBJ} & \boxed{1}\ \mathit{kim} \\ \text{COMPL} & \begin{bmatrix}\text{SUBJ} & \boxed{1}\end{bmatrix}\end{bmatrix}$$

Figure 1: Feature structures containing a feature with another feature structure as value (a), and containing a reentrancy (b).

Grammar and Tree-adjoining Grammar. What all these framewoks have in common, is the fact that they encode linguistic knowledge (rules, principles, lexical entries, etc.) in the form of feature structures, and that unification is used as the operation to combine information. Unification can be implemented efficiently, and most parsing algorithms for context-free grammar can be extended easily to unification-based grammars. Unification-based grammars are therefore popular in natural language processing, especially in work which emphasizes linguistic aspects.

2 Basic notions

Feature structures, as shown in figure 1, consist of features and values. Values can be either atoms or feature structures themselves. The feature structure in figure 1a contains a feature CAT whose value is the atom *np* and a feature AGR whose value is a feature structure (specifying values for NUM and PER). The *path* ⟨AGR PER⟩ refers to the value of the feature PER within the feature structure that is the value of AGR. The value of ⟨AGR PER⟩ in the feature structure in figure 1a is the atom *1st*. Feature structures may also contain *reentrancies*, stating that the value of two or more features is shared. This is usually indicated by prefixing the shared values with a boxed integer. The actual value of the reentrant features is only mentioned once. The feature structure in figure 1b contains a reentrancy for ⟨SUBJ⟩ and ⟨COMPL SUBJ⟩. The value of both paths is the atom *kim*.

Feature structures may be partially ordered according to the amount of information they contain. If feature structure A is more general than B, that is, if all the information in A is also present in B, then A *subsumes* B.

Definition 1 *Feature structure A* **subsumes** *feature structure B iff*

- *For every feature F in A,*
  - *if the value of F in A is atomic, the value of F in A and B is identical,*
  - *if the value of F in A is a feature structure, the value of F in A subsumes the value of F in B.*
- *Every reentrancy in A is also present in B.*

Unification is the operation which merges two feature structures into a single feature structure, which contains all the information encoded in the two input structures, and nothing more. If the two input structures contain conflicting information, unification fails. Unification is defined in terms of subsumption:

Definition 2 *The* **unification** *of feature structures A and B is the most general feature structure which is subsumed by both A and B, if such a feature structure exists.*

3 Unification-Based Grammar

A unification-based grammar consists of a lexicon, in which each word is associated with one or more feature structures, and grammar rules, also defined in terms of feature structures. In the

$$\text{a.}\quad \begin{bmatrix}\text{CAT} & s\end{bmatrix} \rightarrow \begin{bmatrix}\text{CAT} & np \\ \text{AGR} & \boxed{1}\end{bmatrix}\begin{bmatrix}\text{CAT} & vp \\ \text{AGR} & \boxed{1}\end{bmatrix}$$

$$\text{b.}\quad \begin{bmatrix}\text{MOTHER} & \begin{bmatrix}\text{CAT} & s\end{bmatrix} \\ \text{DAUGHTERS} & \begin{bmatrix}\text{FIRST} & \begin{bmatrix}\text{CAT} & np \\ \text{AGR} & \boxed{1}\end{bmatrix} \\ \text{REST} & \begin{bmatrix}\text{FIRST} & \begin{bmatrix}\text{CAT} & vp \\ \text{AGR} & \boxed{1}\end{bmatrix} \\ \text{REST} & nil\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Figure 2: Rules in a unification-based extension of context-free grammar can be interpreted as feature structures, consisting of a MOTHER and a list of DAUGHTERS.

simplest case, grammar rules are comparable to those of context-free grammar, in the sense that they consist of a mother and an ordered list of (zero or more) daughters. That is, the rule in figure 2 will allow for the derivation of a constituent of category S (i.e. with feature structure [CAT *s*]) just in case there is a constituent that is unifiable with the first feature structure on the right-hand side of the arrow, followed by a constituent that is unifiable with the second feature structure on the right-hand side. The reentrancy for the feature AGR between the two daughters requires that the whole rule must in fact be understood as expressing a single feature structure. This is shown in the bottom part of figure 2. Note that the list of daughters is expressed as a (recursive) feature structure, where FIRST is used to denote the value of the first element of a list, and REST to denote the value of the tail of a list.

Unification-based grammars are often referred to as constraint-based grammars. A grammar defines the feature structures that are associated with lexical items and grammar rules. In doing so, one has to use a language for describing feature structures. Following Kasper and Rounds (1986) and others, a language for feature structures can be given a semantics by specifying which feature structures *satisfy* a given description. Descriptions can also be seen as imposing constraints on the feature structures that satisfy them. For this reason, unification-based formalisms are often referred to as *constraint-based grammar formalisms*. In the simplest feature description languages, one can only describe a feature structure as a conjunction of statements which speficy an atomic value for a path or a reentrancy between paths. In such languages, there is always a unique feature structure satisfying a description which subsumes all other features structures satisfying the description. More powerful languages allow for disjunction and negation. I.e. one may state that the value of CASE is *nom or acc*, or *not gen*. Even more powerful languages allow the use of quantification and relations. In such languages, one may state that all elements in a list-like feature structure have to meet a certain requirement, or to define the value of a feature as the *append* of two list-like feature structures. In these more powerful languages, there is no longer a unique most general feature structure satisfying a constraint, which may pose considerable challenges for implementation.

4 Agreement as unification

The advantages of using unification in grammar can be easily demonstrated with an example based on agreement. Dutch nouns either have *neuter* or *non-neuter* gender. The definite determiners *het* and *de* select for neuter and non-neuter nouns, respectively. The indefinite determiner *een* may be used with both neuter and non-neuter nouns. Pre-nominal adjectives are normally inflected, but an uninflected adjective occurs when the adjective modifies a (singular) neuter noun, and the NP as a whole is indefinite (i.e. introduced by *een*):

a. $$\begin{bmatrix}\text{CAT} & np\\ \text{AGR} & \boxed{1}\end{bmatrix} \rightarrow \begin{bmatrix}\text{CAT} & det\\ \text{AGR} & \boxed{1}\end{bmatrix} \begin{bmatrix}\text{CAT} & n\\ \text{AGR} & \boxed{1}\end{bmatrix}$$

$$\begin{bmatrix}\text{CAT} & n\\ \text{AGR} & \boxed{1}\end{bmatrix} \rightarrow \begin{bmatrix}\text{CAT} & a\\ \text{AGR} & \boxed{1}\end{bmatrix} \begin{bmatrix}\text{CAT} & n\\ \text{AGR} & \boxed{1}\end{bmatrix}$$

b. $$\text{de} \mapsto \begin{bmatrix}\text{CAT} & det\\ \text{AGR} & \begin{bmatrix}\text{DEF} & +\\ \text{GEN} & nneut\end{bmatrix}\end{bmatrix} \qquad \text{oude} \mapsto \begin{bmatrix}\text{CAT} & a\\ \text{AGR} & \begin{bmatrix}\text{GEN} & nneut\end{bmatrix}\end{bmatrix}$$

$$\text{het} \mapsto \begin{bmatrix}\text{CAT} & det\\ \text{AGR} & \begin{bmatrix}\text{DEF} & +\\ \text{GEN} & neut\end{bmatrix}\end{bmatrix} \qquad \text{oude} \mapsto \begin{bmatrix}\text{CAT} & a\\ \text{AGR} & \begin{bmatrix}\text{DEF} & +\\ \text{GEN} & neut\end{bmatrix}\end{bmatrix}$$

$$\text{een} \mapsto \begin{bmatrix}\text{CAT} & det\\ \text{AGR} & \begin{bmatrix}\text{DEF} & -\end{bmatrix}\end{bmatrix} \qquad \text{oud} \mapsto \begin{bmatrix}\text{CAT} & a\\ \text{AGR} & \begin{bmatrix}\text{DEF} & -\\ \text{GEN} & neut\end{bmatrix}\end{bmatrix}$$

$$\text{huis} \mapsto \begin{bmatrix}\text{CAT} & n\\ \text{AGR} & \begin{bmatrix}\text{GEN} & neut\end{bmatrix}\end{bmatrix} \qquad \text{kerk} \mapsto \begin{bmatrix}\text{CAT} & n\\ \text{AGR} & \begin{bmatrix}\text{GEN} & nneut\end{bmatrix}\end{bmatrix}$$

Figure 3: Rules and lexical entries for internal agreement of Dutch singular NPs.

(1) a. de oude kerk
the old church(*nneut*)

b. het oude huis
the old house(*neut*)

c. een oude kerk
an old church

d. een oud huis
an old house

This agreement pattern is captured by the grammar fragment in figure 3. Agreement is expressed by the feature AGR, which contains a feature GEN, used to capture the gender distinction, and DEF, used to distinguish between definite and indefinite NPs. The determiner, adjective, and noun all may impose constraints on AGR. By unifying the value for AGR between noun and adjective, and between the projection of a noun and the determiner, it is enforced that agreement information contributed by all elements must be compatible. The fragment can be easily extended to take plurals into account.

It should be noted that the unification-based account of this phenomenon has several advantages over an account using only the standard machinery of context-free grammar. As there are four possible combinations of gender and definiteness values, a context-free grammar with atomic category symbols only, would have to provide four variants for the two rules in figure 3a, and also two lexical entries for the indefinite determiner *een* (which combines with both neuter and non-neuter nouns). (Below, a technique is mentioned which allows the two lexical entries for *oude* to be reduced to one.) Note also that (undirectional) unification of information is essential in this case. An account based on upward or downward percolation of feature-values would have to ensure that information from the determiner can be passed on to the N-sister, and information from the head noun can be passed on to the mother node as well as the A-sister. A unification-based account can deal with this situation by simply assuming across-the-board sharing of the relevant

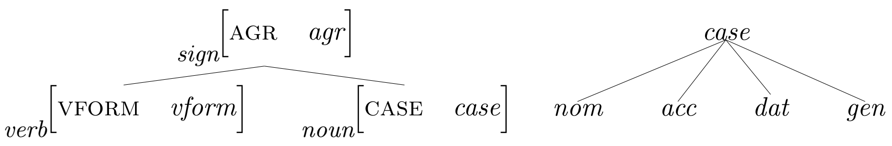

Figure 4: Fragment of a signature for a grammar with typed feature structures.

morphosyntactic information.

5 Typed Feature Structures

It is customary to impose restrictions on the values that a given feature may take on. For instance, the value of CASE for a given grammar can be restricted to *nom, acc, dat* or *gen*, while the value of NUM can be restricted to *pl* or *sg*. Similarly, one might want to impose constraints on which combinations of features can co-occur. For instance, the value of AGR could be restricted to feature structures which may contain the features PER and NUM, but not CAT or SUBJ.

Typed feature structures make it possible to express such constraints. A *typed feature structure* is only well-formed if it is in accordance with a *type signature*.

Definition 3 *A* **type signature** *consists of:*

- *A finite set of* types*,*
- *A partial order on types,*
- *Appropriateness conditions, specifying*
  - *the features that are appropriate for a given type,*
  - *the type of the values that each feature may take on.*

Figure 4 contains a fragment of a type signature. It defines the types *verb* and *noun* as subtypes of *sign*. For *sign* the attribute AGR is appropriate, and its value has to be of type *agr*. For nouns, an additional feature CASE is appropriate, whose value has to be of type *case*. The subtypes of *case* are given as well. Note that no features are appropriate for *case* and its subtypes.

The definition of subsumption for typed feature structures takes type subsumption into account:

Definition 4 *A typed feature structure A* **subsumes** *a typed-feature structure B iff*

- *The type of A subsumes the type of B,*
- *For every feature F in A it is the case that the value of F subsumes the value of F in B,*
- *Every reentrancy present in A is also present in B*

Note that, as all atomic values are types, the distinction between complex and atomic values is no longer relevant. The **unification** of two typed feature structures is the most general typed feature structure subsumed by both feature structures, if such a structure exists.

The use of types can make it easier to grasp a grammar. In implementations, it can be used to facilitate grammar maintenance (for instance by detecting errors at compile time) and to improve processing speed. Finally, types can be used to express generalizations which may be hard to express in an untyped system.

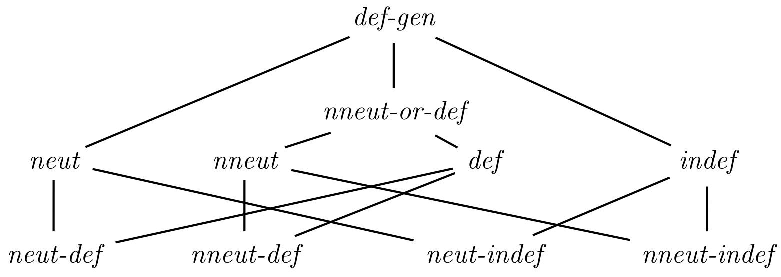

Figure 5: Type-signature for agreement in Dutch NPs.

For instance, to capture the agreement in Dutch singular NPs described in the previous section, one might eliminate the features GEN and DEF, and instead use the type hierarchy in figure 5 as potential values for a feature AGR. The most specific types in the hierarchy express combinations of the value for definiteness and gender. The definite determiners *de* and *het* would be [AGR *neut-def*] and and [AGR *nneut-def*], respectively. The intermediate types express a value for either definiteness or gender. The indefinite determiner is specified as [AGR *indef*], for instance, while nouns have are [AGR *neut*] or [AGR *nneut*]. The type *def-or-nneut*, finally, is added in order to be able to describe the agreement properties of an inflected adjective (*oude*). An uninflected adjective would be [AGR *neut-indef*]. Note that by providing a single lexical entry for inflected adjectives, a generalization is captured which is missed in the untyped fragment in figure 3.

6 Inheritance and Default Unification

Unification-based formalisms use unification to combine feature structures encoding linguistic properties. Unification is a monotonic operation, in the sense that it combines information and can never change or remove feature values. Monotonicity is in general a property to be appreciated: it supports declarative interpretations and implementations of grammar formalisms. Certain linguistic phenomena, however, are more easily described using default mechanisms of various sorts. This holds for lexical information in particular, but defaults have also een used in syntax.

Unification-based frameworks tend to organize the lexicon using some form of inheritance. All verbs have certain properties in common (i.e. they are of category *verb*, they may combine with a subject and zero or more complements, etc.), and these properties should preferably be stated only once. The idea of inheritance is to define a class or type (i.e. *verbal-lexeme*), which states that any element in this class has to be of category *verb*, etc. A lexical item which is defined to be of class *verbal-lexeme* inherits all the properties defined for this class. Classes may inherit from more general classes. Thus, one might define a class *transitive-verb*, which inherits from *verbal-lexeme*. A snapshot of a hypothetical lexical hierarchy is given in figure 6. Note that classes are similar to types in the sense that they are both partially ordered and impose constraints on feature structures. The distinction between types and classes is therefore often conflated.

The *verbal-lexeme* class in figure 6 states that a verb may combine with a subject whose NFORM is *norm*. This will rule out sentences where a verb combines with expletives such as *there* or *it*. While this is true for the majority of verbs, an intransitive verb such as *rain* is exceptional in that it selects for expletive *it*. If inheritance is a default mechanism, one can define *rains* as inheriting from *intransitive-verb*, with the exception of the value for NORM, which is explicitly assigned the value *it* in the lexical entry of *rain*. Similarly, there are verbs which take a clausal subject or object. Such verbs can share certain properties with either intransitive or transitive verbs. Thus, one might consider classifying them as either intransitive or transitive, but to ensure at the same

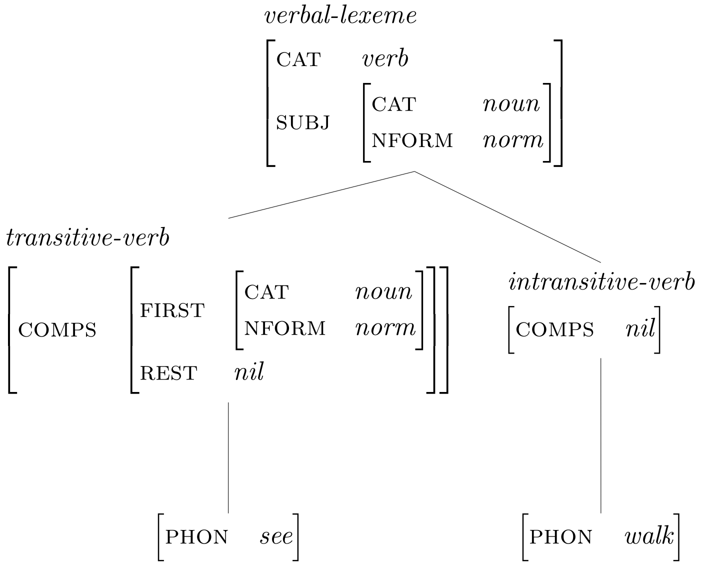

Figure 6: Fragment of a verbal lexical hierarchy.

time that the value of ⟨SUBJ CAT⟩ or ⟨COMPS FIRST CAT⟩ is *s* instead of *np*.

It should be noticed that the feature structures in figure 6 are artificially simple. Thus, while it may seem that in this case one could easily come up with an alternative hierarchy which is strictly monotonic, this will in general be harder for realistic examples. In that case, any attempt to encode a lexical hierarchy monotonically may lead to a considerable reduplication of information.

The use of inheritance to capture generalizations is not restricted to the lexicon. HPSG, for instance, was originally seen as a theory which puts strong emphasis on the lexicon, while minimizing the number of phrase structure schemata. In more recent versions of this theory (Sag, 1997; Ginzburg and Sag, 2000), however, a substantial number of construction specific phrase structure schemata is being used. Generalizations over various schemata are captured by means of inheritance.

A somewhat different reason for considering default mechanisms in syntax comes from observations concerning feature percolation. An important generalization about phrase structure rules is that the syntactic properties of a phrase are to a large extent determined by the syntactic properties of the head of that phrase. Authors differ, however, on the question whether this is only a default (i.e. this generalization holds for most syntactic properties, and in most syntactic constructions, but exceptions exist) or a requirement which can be imposed without exception on a certain class of syntactic properties. In GPSG for instance, the Head-Feature Convention states that head features are identical on mother and head daughter, *unless there is information in the rule which contradicts this*. Head features typically express morphosyntactic properties, such as agreement, case, verb form, etc. The Head-Feature Principle of HPSG as formulated in (Pollard and Sag, 1994) states that the head features of the mother and head daughter in a rule are reentrant. This excludes any possibility of exceptions. The Generalized Head Feature Principle of Ginzburg and Sag (2000), finally, is again a default principle. Furthermore, it is no longer restricted to morphosyntactic information, but applies to all syntactic and semantic information.

One might try to come up with specific operations on feature structures to implement the effects of default inheritance or the default interpretation of a head feature principle. It seems, however, that a more general solution can be found by extending the unification-based formalism with a nonmonotonic notion of unification. The specific mechanisms discussed above can ideally all be defined in terms of this general operation. Various proposals for a general *default unification* operation have been put forward. Bouma (1992) and Carpenter (1992b) both propose operations which nonmonotonically combine a feature structure D (contributing the default information) with a feature structure S (contributing strict information) in such a way that all information from S is preserved, as well as the information from D that is consistent with S. In terms of *subsumption*, one might define the *default unification* of D and S as the unification of D′ with S, where D′ is the maximally specific feature structure subsuming D and unifiable with S. Figure 7a illustrates default unification for a simple case not involving reentrancies, where we use $\sqcup!$ to denote the default unification operation.

Default unification of feature structures which do contain reentrancies is harder to define. Part of the problem is caused by the fact that there may be more than one way to generalize D to a feature structure D′, compatible with S. This is illustrated in figure 7b. D in this case might be generalized in two different ways (i.e. by removing [COMPL -] or by removing the reentrancy), leading to the result in figure 7c and 7d, respectively. Carpenter's *credulous default unification* defines the result of default unification in such cases as a disjunction of possibilities. His *skeptical default unification* defines the result as the maximal feature structure subsuming all the disjuncts obtained by *credulous default unification*. Bouma's default unification does define D′ as the maximal specific generalization of D, which does not contain paths with an atomic value or reentrant paths that are mentioned in S. In general, this surpresses more information from D than would be required by the consistency requirement. On the other hand, it does allow one to surpress information (i.e. to generalize a feature structure) by means of default unification. One application is the definition of auxiliary verbs in theories like HPSG. Auxiliaries may combine with any type of subject (i.e. normal and expletive NPs as well as sentential subjects), as long as this is in accordance with the selection requirements of the main verb heading the VP selected by the auxiliary. By providing strict information that the value of SUBJ is reentrant with the SUBJ value of the VP-complement, this dependency is expressed. Furthermore, the normal, default, value for SUBJ is surpressed.

Following earlier work by Young and Rounds (1993), Lascarides et al. (1996) and Lascarides and Copestake (1999) develop a notion of default unification for typed feature structures. It differs from other proposals in that it assumes that certain feature values are explicitly marked as default values. Default unification of two feature structures may never lead to conflicts in the non-default values, and, for features with a default value, preserves the value provided by the feature structure with the most specific type.

Default unification is usually seen as a means to streamline the definition of grammars. It allows one to express generalizations over classes of lexical items or grammar rules, while at the same time it does not exclude the possibility of exceptions to these generalizations. The interaction between default and strict information in grammar definitions can be computed off-line. That is, once a definition is complete, all information associated with a lexical item or phrase structure rule can be computed. During processing, however, the distinction between default and strict information is no longer relevant, and only monotonic unification is used. The proposal by Lascarides and Copestake is more radical, in that it assumes that the interaction between default and strict information is not necessarily restricted to the definitional stage, but may be preserved during processing.

7 Open Problems and Alternatives

Unification-based grammars, and especially unification-based approaches to morphosyntactic phenomena have impressive empirical coverage. This is not to say that there are no open problems, however. Ingria (1990) argues that a typical unification-based account of agreement and case

$$\text{a.}\quad D = \begin{bmatrix} \textsc{case} & \textit{nom} \\ \textsc{nform} & \textit{norm} \end{bmatrix} \qquad S = \begin{bmatrix} \textsc{nform} & \textit{it} \end{bmatrix}$$

$$D \sqcup! S = \begin{bmatrix} \textsc{case} & \textit{nom} \\ \textsc{nform} & \textit{it} \end{bmatrix}$$

$$\text{b.}\quad D = \begin{bmatrix} \textsc{head} & \boxed{1}\begin{bmatrix} \textsc{compl} & - \end{bmatrix} \\ \textsc{hd-dghtr} & \begin{bmatrix} \textsc{head} & \boxed{1} \end{bmatrix} \end{bmatrix} \qquad S = \begin{bmatrix} \textsc{head} & \begin{bmatrix} \textsc{compl} & + \end{bmatrix} \end{bmatrix}$$

$$\text{c.}\quad D \sqcup! S = \begin{bmatrix} \textsc{head} & \boxed{1}\begin{bmatrix} \textsc{compl} & + \end{bmatrix} \\ \textsc{hd-dghtr} & \begin{bmatrix} \textsc{head} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

$$\text{d.}\quad D \sqcup! S = \begin{bmatrix} \textsc{head} & \begin{bmatrix} \textsc{compl} & + \end{bmatrix} \\ \textsc{hd-dghtr} & \begin{bmatrix} \textsc{head} & \begin{bmatrix} \textsc{compl} & - \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 7: Default unification examples.

assignment makes the wrong prediction in cases such as (2).

(2) a. *Sie findet und hilft Männer/ Männern
She finds(ACC) and helps(DAT) men(ACC) men(DAT)
b. Er findet und hilft Frauen
He finds(ACC) and helps(DAT) women(DAT/ACC)

The ungrammaticality of example (2)a suggests that the case of the direct object must be compatible with the case requirements of both coordinated verbs. A unification-based account would predict this by simply unifying the case values of both verbs with that of the direct object. The grammaticality of (2)b suggests that this cannot be the full story, however. If the direct object neutralizes the distinction between ACC and DAT, the sentence is grammatical. This suggests that the realtion between the case requirements of the verbs and the case value of the object should not be modelled in terms of unification.

Coordination of unlike categories, as in (3) poses another challenge for unification-based approaches.

(3) Kim became wealthy and a Republican

Coordination of an AP and an NP is allowed only if the result is selected by a head which may combine with either an AP or NP. Bayer (1996) argues that unification-based accounts of this phenomenon make the wrong predictions in a number of cases, and argues for an account based on the proof system of Lambek Categorial Grammar. Johnson (1999) argues for a reformulation of Lexical Functional Grammar which is similar in spirit to the proposal of Bayer.

8 Conclusion

Unification-based approaches to syntax have given rise to a family of linguistic frameworks, which combine formal and descriptive precision and which have served as the basis of various large scale grammar implementation efforts.

Unification-based approaches to phonology and morphology have been developed as well (i.e. see Bird (1995) and Riehemann (1998)). The use of unification in semantics was initially seen as more problematic (Moore, 1989), but the recent interest in formalisms for underspecified semantics (starting with Reyle (1993) and Alshawi and Crouch (1992)) is heavily influenced by techniques from unification-based grammar.

More recently, computational linguistics has started to investigate combinations of unification-based grammars with stochastic models (Abney, 1997; Geman and Johnson, 2002). Such models typically use frequencies derived from large (annotated or unannotated) corpora to estimate probabilities for parses produced by a (wide-coverage) unification-based grammar.

References

Abney, Steven P. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23(4):597–618.

Alshawi, Hiyan and Richard Crouch. 1992. Monotonic semantic interpretation. In *30th Annual Meeting of the Association for Computational Linguistics*, pages 32–39, Newark, Delaware.

Bayer, Sam. 1996. The coordination of unlike categories. *Language*, 72(3):579–616.

Bird, Steven. 1995. *Computational phonology.* Cambridge: Cambridge University Press.

Bouma, Gosse. 1992. Feature structures and nonmonotonicity. *Computational Linguistics*, 18(2):183–204.

Bouma, Gosse, Frank van Eynde, and Dan Flickinger. 1997. Constraint-based lexicons. In Dafydd Gibbon, Frank van Eynde, and Ineke Schuurman, editors, *Lexicon Development for Speech and Language Processing.* Kluwer, Dordrecht.

Bresnan, Joan, editor. 1982. *The Mental Representation of Grammatical Relations.* MIT Press.

Carpenter, Bob. 1992a. *The Logic of Typed Feature Structures.* Cambridge: Cambridge University Press.

Carpenter, Bob. 1992b. Skeptical and credulous default unification with applications to templates and inheritance. In Ted Briscoe, Anne Copestake, and Valerie de Paiva, editors, *Default Inheritance within Unification-Based Approaches to the Lexicon.* Cambridge University Press, Cambridge.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullum, and Ivan Sag. 1985. *Generalized Phrase Structure Grammar.* Blackwell.

Geman, Stuart and Mark Johnson. 2002. Dynamic programming and estimation of stochastic unification-based grammars. In *40th Annual Meeting of the Association for Computational Linguistics*, pages 279–286, Philadelphia.

Ginzburg, Jonathan and Ivan Sag. 2000. *Interrogative Investigations.* Stanford, CA: CSLI Publications.

Ingria, Robert. 1990. The limits of unification. In *28th Annual Meeting of the Association for Computational Linguistics*, pages 194–204, Pittsburg.

Johnson, Mark. 1999. A resource-sensitive interpretation of lexical functional grammar. *Journal of Language, Logic, and Information*, 8(1):45–81.

Kasper, R.T. and W.C. Rounds. 1986. A logical semantics for feature structures. In *24th Annual Meeting of the Association for Computational Linguistics*, Columbia university, New York.

Lascarides, Alex, Ted Briscoe, Nicholas Asher, and Ann Copestake. 1996. Order independent and persistent typed default unification. *Linguistics and Philosophy*, 19(1):1–89.

Lascarides, Alex and Ann Copestake. 1999. Default representation in constraint-based frameworks. *Computational Linguistics*, 25(1):55–105.

Moore, Robert C. 1989. Unification-based semantic interpretation. In *27th Annual Meeting of the Association for Computational Linguistics*, pages 33–41, Vancouver.

Pollard, Carl and Ivan Sag. 1994. *Head-driven Phrase Structure Grammar*. Center for the Study of Language and Information Stanford.

Pulman, Steve. 1996. Unification encodings of grammatical notations. *Computational Linguistics*, 22(3):295–328.

Reyle, Uwe. 1993. Dealing with ambiguities by underspecification. *Journal of Semantics*, 10:123–179.

Riehemann, Suzanne. 1998. Type-based derivational morphology. *Journal of Comparative Germanic Linguistics*, pages 49–77.

Sag, Ivan. 1997. English relative clause constructions. *Journal of Linguistics*, 33:431–484.

Shieber, Stuart M. 1986. *Introduction to Unification-Based Approaches to Grammar*. Center for the Study of Language and Information Stanford.

Thomason, Richmond. 1997. Nonmonotonicity in linguistics. In *Handbook of Logic and Language*. North Holland, Dordrecht, pages 777–831.

Young, Mark and Bill Rounds. 1993. A logical semantics for nonmonotonic sorts. In *31th Annual Meeting of the Association for Computational Linguistics*, pages 209–215.