Feature Structures and Nonmonotonicity

Gosse Bouma*
Rijksuniversiteit Groningen

Unification-based grammar formalisms use feature structures to represent linguistic knowledge. The only operation defined on feature structures, unification, is information-combining and monotonic. Several authors have proposed nonmonotonic extensions of this formalism, as for a linguistically adequate description of certain natural language phenomena some kind of default reasoning seems essential. We argue that the effect of these proposals can be captured by means of one general, nonmonotonic, operation on feature structures, called default unification. *We provide a formal semantics of the operation and demonstrate how some of the phenomena used to motivate nonmonotonic extensions of unification-based formalisms can be handled.*

1. Introduction

While monotonicity is often desirable from a formal and computational perspective, it is at odds with a considerable body of linguistic work. Default principles, default rules, and default feature-values can be found in many linguistic formalisms and are used prominently in work on phonology, morphology, and syntax. In spite of their great expressive power and flexibility, unification-based grammar formalisms (see Shieber 1986a, for an introduction) are in general not very successful in modeling such devices. Unification is an information-combining, *monotonic*, operation on feature structures, whereas the implementation of default devices typically requires some form of nonmonotonicity. In this paper, we present a nonmonotonic operation on feature structures, which enables us to implement the effects of a number of default devices used in linguistics. As the operation is defined in terms of feature structures only, an important characteristic of unification-based formalisms, namely that linguistic knowledge is encoded in the form of feature structures, is preserved.

In the next section, we present an overview of linguistic phenomena that are best described using defaults. We also argue that previous proposals for handling these phenomena in a unification-based setting are unsatisfactory. Section 3 provides the formal background for the central part of the paper, Section 4, in which a definition of *default unification* is presented. Section 5 briefly presents some applications of this operation and the final section draws some conclusions concerning the role of nonmonotonicity in unification-based formalisms.

2. Previous Work

There are a number of phenomena that suggest that unification-based grammar formalisms might profit from the addition of some form of nonmonotonicity, and several authors have in fact suggested such extensions. In this section, we argue that these proposals suffer from a number of shortcomings. Most importantly, previous proposals have either been highly restricted in scope or have been presented in very informal

* Computational Linguistics Department, Postbus 716, 9700 AS Groningen, The Netherlands

terms, thus leaving a number of questions concerning the exact behavior of the proposed extensions unanswered.

An overview of the issues that call for the addition of non-monotonic reasoning and of some of the proposals in that direction is presented below.

- **Exceptional Rules.** Consider a language in which the vast majority of verbs cannot precede its subject, whereas a small number of exceptional verbs can. The rule accounting for inverted structures would probably require that verbs occurring in it be marked as +INV (i.e. $\langle \text{INV} \rangle : +$). As a consequence, all regular verbs must be marked explicitly as −INV (to prevent them from occurring in the inversion rule). Note that, in a unification-based grammar, there is no need to mark the exceptional verbs as +INV, which leads to the rather counterintuitive situation that regular verbs need to be marked extra, whereas the exceptional ones can remain underspecified. A more natural solution would be to assign all verbs the specification -INV by default (either by means of template inheritance or by means of lexical feature specification defaults as used in Generalized Phrase Structure Grammar [GPSG; Gazdar et al. 1985]) and to overwrite or block this specification in the exceptional cases. The possibility of incorporating an *overwrite* operation in a unification-based formalism is mentioned in Shieber (1986a, p. 60).

- **Feature Percolation Principles.** Both GPSG and Head-driven Phrase Structure Grammar (HPSG; Pollard and Sag 1987) adopt the so-called *Head Feature Convention* (HFC). In GPSG, the HFC is a default principle: head features will normally have identical values on mother and head, but specific rules may assign incompatible values to specific head features. In unification-based formalisms, it is impossible to express this principle directly. Adding the constraint $\langle X_0\ head \rangle = \langle X_i\ head \rangle$ to every rule of the form $X_0 \rightarrow X_1 \ldots X_n$ (with X_i $(1 < i < n)$ the head of the rule and assuming all head features to be collected under *head*) will not do, as it rules out the possibility of exceptions altogether. Shieber (1986b) therefore proposes to *add* this constraint *conservatively*, which means that, if the rule already contains conflicting information for some head feature f, the constraint is replaced by a set of constraints $\langle X_0\ head\ f' \rangle = \langle X_i\ head\ f' \rangle$, for all head features $f' \neq f$.

- **Structuring the Lexicon.** Flickinger, Pollard, and Wasow (1985), Flickinger (1987), De Smedt (1990), Daelemans (1988), and others, have argued that the encoding and maintenance of the detailed lexical descriptions typical for lexicalist grammar formalisms benefits greatly from the use of (nonmonotonic) inheritance. In Flickinger, Pollard, and Wasow (1985), for instance, lexical information is organized in the form of *frames*, which are comparable to the templates (i.e., feature structures that may be used as part of the definition of other feature structures) of PATR-II (Shieber 1986a). A frame or specific lexical entry may inherit from more general frames. Frames can be used to encode information economically and, perhaps more importantly, as a means to express linguistic generalizations. For instance, all properties typical of verbs are defined in the *VERB*-frame, and properties typical of auxiliaries are defined in the *AUX*-frame. The *AUX*-frame may inherit from the *VERB*-frame, thus capturing the fact that an auxiliary is a kind of verb.

In this approach, a mechanism that allows inheritance of information by default (i.e., a mechanism in which local information may exclude the inheritance of more general information) is of great importance. Without such a mechanism, a frame may contain only properties that hold without exception for all items that inherit from this frame. In practice, however, one often wants to define the properties that are *typical* for a given class in the form of a frame, without ruling out the possibility that exceptions might exist. In unification-based formalisms, templates can play the role of frames, but as unification is used to implement inheritance, nonmonotonic inheritance is impossible.

- **Inflectional Morphology**. In PATR-II the lexicon is a list of inflected word forms associated with feature structures. The only tools available for capturing lexical generalizations are templates (see above) and lexical rules. Lexical rules may transform the feature structure of a lexical entry. An example is the rule for agentless passive (Shieber 1986a, p. 62), which transforms the feature structure for transitive *past participles* into a feature structure for participles occurring in agentless passive constructions. Lexical rules can only change the feature structure of a lexical entry, not its word form, and thus, the scope of these rules is rather restricted. While the examples in Flickinger, Pollard, and Wasow (1985) and Evans and Gazdar (1989a,b) suggest that the latter restriction can be easily removed, it is not so obvious how a unification-based grammar formalism can cope with the combination of rules and exceptions typical for (inflectional) morphology. For instance, it is possible to formulate a rule that describes past tense formation in English, but it is not so easy to exclude the application of this rule to irregular verbs and to describe (nonredundantly) past tense formation of these irregular verbs. Evans and Gazdar (1989a,b) present the DATR-formalism, which, among other things, contains a nonmonotonic inference system that enables an elegant account of the *blocking*-phenomenon just described. The examples used throughout their presentation are all drawn from inflectional morphology and illustrate once more the importance of default reasoning in this area of linguistics.

- **Gapping**. In Kaplan (1987) it is observed that gapping constructions and other forms of nonconstituent conjunction can be analyzed in Lexical Functional Grammar (Bresnan and Kaplan, 1982) as the conjunction of two *functional-structures* (f-structures), one of which may be incomplete. The missing information in the incomplete f-structure can be filled in if it is merged with the complete f-structure, using an operation called *priority union*. Priority union of two f-structures A and B is defined as an operation that extends A with information from B that is not included (or filled in) in A. As not all information in B is present in the priority union of A and B, this operation introduces nonmonotonicity.

The proposals for incorporating the kind of default reasoning that is required for each of the phenomena above are both rather diverse and idiosyncratic and, furthermore, suffer from a number of shortcomings.

The *Head Feature Convention* and *Feature Specification Defaults* of GPSG, for instance, appear to be motivated with a very particular set of linguistic phenomena in mind and also are rather intimately connected to peculiarities of the GPSG-formalism. What

is particularly striking is the fact that two different conceptions of *default* appear to play a role: a head feature is exempt from the HFC only if this would otherwise lead to an *inconsistency*, whereas a feature is exempt from having the value specified in some feature specification default (among others) if this feature *covaries* with another feature.

Overwrite and *add conservatively* are also highly restricted operations. From the examples given in Shieber (1986a) it seems as if overwriting can only be used to add or substitute (nonmonotonically) one atomic feature value in a given (possibly complex) feature structure (which acts as default). Add conservatively, on the other hand, is only used to add one reentrancy (as far as possible) to a given feature structure (which acts as nondefault). An additional restriction is that add conservatively is well behaved only for the kind of feature structures used in GPSG (that is, feature structures in which limited use is made of *covariation* or reentrancy). Consider for instance the example in (1).[1] Adding the constraint $\langle X_0\ head\rangle = \langle X_1\ head\rangle$ to (1) conservatively could result in either 1a or 1b.

Example 1

$$\begin{bmatrix} X_0: \begin{bmatrix} head: \begin{bmatrix} f: \boxed{1} \\ g: \boxed{1} \end{bmatrix} \end{bmatrix} \\ X_1: \begin{bmatrix} head: \begin{bmatrix} f: a \\ g: b \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

a.
$$\begin{bmatrix} X_0: \begin{bmatrix} head: \begin{bmatrix} f: \boxed{1}\, a \\ g: \boxed{1}\, a \end{bmatrix} \end{bmatrix} \\ X_1: \begin{bmatrix} head: \begin{bmatrix} f: a \\ g: b \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

b.
$$\begin{bmatrix} X_0: \begin{bmatrix} head: \begin{bmatrix} f: \boxed{1}\, b \\ g: \boxed{1}\, b \end{bmatrix} \end{bmatrix} \\ X_1: \begin{bmatrix} head: \begin{bmatrix} f: a \\ g: b \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

As add conservatively and overwriting are, in a sense, mirror images of each other, it is tempting to generalize the definitions of these operations and to think of them as operations on arbitrary feature structures, whose effect is equivalent to that of *priority union*. Thus, given two feature structures FS_D (the default) and FS_{ND} (the nondefault), adding FS_D to FS_{ND} conservatively would be equivalent to overwriting FS_D with FS_{ND}, and to the priority union of FS_{ND} and FS_D (i.e. FS_{ND}/FS_D in the notation of Kaplan [1987]). However, in light of the example above, it should be clear that such a generalization is highly problematic. Other examples worth considering are 2 and 3.

1 Whether this kind of situation can occur in GPSG probably depends on whether one is willing to conclude from examples such as:

$$S[COMP\alpha] \rightarrow \{[SUBCAT\alpha]\}, H[COMP\ NIL]$$ (Gazdar et al. 1985, p. 248)

that covariation of arbitrary categories is in principle not excluded in this formalism.

Example 2

$$FS_D = \begin{bmatrix} f : a \\ g : b \end{bmatrix} \qquad FS_{ND} = \begin{bmatrix} f : \boxed{1} \\ g : \boxed{1} \end{bmatrix}$$

Example 3

$$FS_D = \begin{bmatrix} f : \boxed{1}\, a \\ g : \boxed{1}\, a \end{bmatrix} \qquad FS_{ND} = \begin{bmatrix} g : b \end{bmatrix}$$

Again, if we try to combine the two feature structures along the lines of any one of the operations mentioned above, there are at least two possible results (note that in Example 3, we could either preserve the information that features f and g are reentrant, or preserve the information that $f : a$), and there is no telling which one is correct.

Two conclusions can be drawn at this point. First of all, on the basis of the examples just given, it can be concluded that a nonmonotonic operation on feature structures that relies (only) on the fact that the result should be consistent must be very restricted indeed, as more generic versions will always run into the problem that there can be several mutually exclusive solutions to solving a given unification conflict. Second, claims that the operations *add conservatively*, *overwriting*, and *priority union* are equivalent are unwarranted, as no definitions of these operations are available that are sufficiently explicit to determine what their result would be in moderately complex examples such as 1–3.

The approach exemplified by Flickinger (1987) and others is to use a general-purpose knowledge representation formalism to represent linguistic information and model default inheritance. Feature structures are defined as *classes* of some sort, which may inherit from other, more generic, classes. The inheritance strategy used says that information in the generic class is to be included in the specific class as well, as long as the specific class does not contain local information that is in conflict with the information to be inherited. Such an inheritance strategy will run into problems, however, if reentrancies are generally allowed. For instance, think of the examples presented above as involving a generic class FS_D from which a specific class FS_{ND} inherits. The inheritance procedure in, for instance, Flickinger (1987, p. 59ff) does not say anything about which one of the possible results will be chosen.

The work of Evans and Gazdar (1989a,b), finally, is not easily incorporated in a unification-based formalism, as they use semantic nets instead of feature structures to represent linguistic information. That is, although the syntax of DATR is suggestively similar to that of, for instance, PATR-II, DATR descriptions do in fact denote graphs that differ rather substantially from the graphs used to represent feature structures (see Evans and Gazdar 1989b). The nonmonotonic reasoning facilities of DATR therefore are not directly applicable in a unification-based formalism either.

We conclude that a formally explicit definition of a nonmonotonic operation on feature structures is still missing. In particular, the interaction of reentrancy and nonmonotonicity is a subtle issue, which has not been given the attention it deserves. That there is a need for nonmonotonic devices is obvious from the fact that several authors have found it necessary to introduce partial solutions for dealing with nonmonotonicity in a unification-based setting. The intuitions underlying these proposals appear to be compatible, if not identical, and thus it seems attractive to consider an operation that subsumes the effects of the proposals so far. *Default Unification*, as defined below, is an attempt to provide such an operation.

3. Feature Structures and Unification

Feature structures are often depicted as matrices of attribute-value pairs where values are either atoms or feature structures themselves and, furthermore, values may be shared by different attributes in the feature structure. Feature structures can be defined using a description language, such as the one found in PATR-II (Shieber 1986a) or in Kasper and Rounds (1986; 1990). For instance, 4a is a description of 4b.

Example 4

$$a. \quad \begin{pmatrix} \langle f \rangle & = & a \\ \langle g\, f \rangle & = & a \\ \langle g\, f \rangle & = & \langle g\, g \rangle \end{pmatrix}$$

$$b. \quad \begin{bmatrix} f : a \\ g : \begin{bmatrix} f : \boxed{1}\, a \\ g : \boxed{1}\, a \end{bmatrix} \end{bmatrix}$$

Following the approach of Kasper and Rounds (1986; 1990), and others, we represent feature structures formally as finite (acyclic) automata (the definition below is taken from Dawar and Vijay-Shanker 1990):

Definition
A **finite acyclic automaton** A is a 7-tuple $\langle Q, \Sigma, \Gamma, \delta, q_0, F, \lambda \rangle$ where:

1. Q is a nonempty finite set of states,
2. Σ is a countable set (the alphabet),
3. Γ is a countable set (the output alphabet),
4. $\delta : Q \times \Sigma \rightharpoonup Q$ is a finite partial function (the transition function),
5. $q_0 \in Q$,
6. $F \subseteq Q$,
7. $\lambda : F \rightharpoonup \Gamma$ is a total function (the output function),
8. the directed graph (Q, E) is acyclic, where pEq iff for some $l \in \Sigma, \delta(p, l) = q$,
9. for every $q \in Q$, there exists a directed path from q_0 to q in (Q, E), and
10. for every $q \in F, \delta(q, l)$ is not defined for any l.

We will frequently write Q_A, Σ_A, etc. for the set of states of automaton A, the alphabet of A, etc.

The relationship between the matrix notation and the automaton concept should be obvious. The following automaton M is, for instance, equivalent to the matrix in 4b.

Example 5

$$
\begin{array}{llll}
Q_M & = \{q_0, q_1, q_2, q_3\} & \delta_M(q_2, g) & = q_3 \\
\Sigma_M & = \{f, g\} & \delta_M(q_2, f) & = q_3 \\
\Gamma_M & = \{a\} & F_M & = \{q_1, q_3\} \\
\delta_M(q_0, f) & = q_1 & \lambda_M(q_1) & = a \\
\delta_M(q_0, g) & = q_2 & \lambda_M(q_3) & = a
\end{array}
$$

Note that $\delta_M(q_0, gf) = \delta_M(q_0, gg) = q_3$,[2] which represents the fact that the two paths $\langle gf \rangle$ and $\langle gg \rangle$ are reentrant. Unification is defined in terms of *subsumption*, a relation that imposes a partial ordering on automata:

Definition
An automaton A **subsumes** an automaton B ($A \sqsubseteq B$) iff there is a homomorphism h from A to B such that:

1. $h(\delta_A(q, l) = \delta_B(h(q), l)$,
2. $\lambda_B(h(q)) = \lambda_A(q)$ for all $q \in F_A$, and
3. $h(q_{0A}) = q_{0B}$.

Intuitively, $A \sqsubseteq B$ if B extends the information in A. $A = B$ if $A \sqsubseteq B$ and $B \sqsubseteq A$. *Unification* of two automata A and B ($A \sqcup B$) is the *least upper bound* of these automata under subsumption. If no upper bound exists, unification fails.

The semantics of descriptions (sets of formulae of the description language) is given in terms of satisfaction:

Definition
An automaton $A = \langle Q, \Sigma, \Gamma, \delta, q_0, F, \lambda \rangle$ **satisfies** a description D ($A \models D$) or a formula ϕ ($A \models \phi$) in the following cases:

$$
\begin{array}{ll}
A \models D & \text{iff for all } \phi \in D : A \models \phi, \\
A \models a & \text{iff } Q = F = \{q_0\} \text{ and } \lambda(q_0) = a, \\
A \models \langle p \rangle = D & \text{iff } \delta(q_0, p) \text{ is defined and } q_0/p \models D, \\
A \models \langle p_1 \rangle = \langle p_2 \rangle & \text{iff } \delta(q_0, p_1) = \delta(q_0, p_2).
\end{array}
$$

q_0/p is the automaton obtained from A by making $\delta(q_0, p)$ the initial state and removing all inaccessible states. There is always a unique minimal element in the subsumption hierarchy that satisfies a description D. This element is the denotation of D.[3]

2 $\delta(q, pl)$ is defined for $pl \in \Sigma^*$ as $\delta(\delta(q, p), l)$.
3 Much of the formal work on feature structures is concerned with the semantics of feature structure descriptions involving disjunction and negation. Such descriptions do not denote a unique feature structure, but denote sets of feature structures. Such extensions are not taken into consideration here.

4. Default Unification

Default reasoning with feature structures requires the ability to modify feature structures nonmonotonically. Unification does not have this ability, as it can only replace a feature structure by more specific instances of that structure. Below, we define *default unification* as an operation that merges parts of one feature structure (the default argument) with another feature structure (the nondefault argument). We write $A \sqcup ! B$ for the default unification of the default feature structure A and the nondefault feature structure B. The operation has the following characteristics:

1. It has a declarative semantics and is procedurally neutral. That is, if $A = A'$ and $B = B'$, then $(A \sqcup ! B) = (A' \sqcup ! B')$.
2. It is monotonic only with respect to the nondefault argument. That is, $B \sqsubseteq (A \sqcup ! B)$ is always true, but in general $A \sqsubseteq (A \sqcup ! B)$ will not hold.
3. It never fails. If A is fully incompatible with B, $(A \sqcup ! B) = B$.
4. It gives a unique result.
5. Reentrancies in the nondefault argument may be replaced by a *weaker set of reentrancies* if necessary (this is the *add conservatively* operation of Shieber (1986b)).

Intuitions about default unification appear to be more clear in those cases where feature structures do not contain any reentrancies. Therefore, we will first define default unification for this case, moving to the general case in Section 4.2. Section 4.3. deals with the incorporation of *add conservatively*.

4.1 Default Unification without Reentrancies

Subsumption suggests a straightforward definition of an operation that has properties 1–4 above.

Definition

Default Unification (first version) $A \sqcup ! B = A' \sqcup B$, where A' is the maximal (i.e. most specific) element in the subsumption ordering such that $A' \sqsubseteq A$ and $A' \sqcup B$ is defined.

From this definition of $\sqcup !$, it follows immediately that properties 1–3 hold. The fact that default unification has a unique result follows from the fact that A' is unique (up to isomorphism).[4] Note furthermore that from the requirement that A' must be the maximal it follows that no information contained in A is left out in $A \sqcup ! B$ unnecessarily.

4 Unicity of A' is proved as follows: Assume that there is an A'' such that (1) $A'' \neq A'$, (2) $A' \sqsubseteq A$ and $A'' \sqsubseteq A$, (3) $A' \sqcup B$ and $A'' \sqcup B$ are defined, and (4) both A' and A'' are maximal. We show that these assumptions are inconsistent. From (2) it follows that $A' \sqcup A''$ is defined and $(A' \sqcup A'') \sqsubseteq A$. From (3) it follows that $(A' \sqcup A'') \sqcup B$ is defined (since, if there are no reentrancies, it holds in general that if $X \sqcup Y$, $Y \sqcup Z$, and $X \sqcup Z$ are defined, $X \sqcup Y \sqcup Z$ is defined). But then, if $(A' \sqcup A'') = A'$ or $(A' \sqcup A'') = A''$, either condition (1) or (4) is not met, or, if $A' \sqcup A'' \neq A' \neq A''$, condition (4) is not met. □

An example of default unification is presented below (where *nil* is used to represent the empty feature structure):

Example 6

$$A = \begin{bmatrix} f : a \\ g : \begin{bmatrix} f : [f : a] \\ g : \begin{bmatrix} f : a \\ g : a \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$B = \begin{bmatrix} f : a \\ g : \begin{bmatrix} f : a \\ g : [g : b] \end{bmatrix} \end{bmatrix}$$

$$A' = \begin{bmatrix} f : a \\ g : \begin{bmatrix} f : nil \\ g : \begin{bmatrix} f : a \\ g : nil \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$A' \sqcup B = \begin{bmatrix} f : a \\ g : \begin{bmatrix} f : a \\ g : \begin{bmatrix} f : a \\ g : b \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The definition of default unification above relies crucially on the fact that there is a unique maximal element A' unifiable with B. In Section 2, we argued that such an approach is only feasible for a limited domain. In particular, once reentrancies are introduced, A' is no longer guaranteed to be unique, and the definition above is therefore not easily generalized. Fortunately, it is also possible to define $A \sqcup\!! B$ without requiring unifiability of some element A' with B explicitly. This definition, which will be extended below, defines $A \sqcup\!! B$ in terms of the difference of the two arguments A and B.

Definition
Difference (first version) The difference of two automata A and B is the maximal element $A - B$ that meets the following conditions:

1. $A - B \sqsubseteq A$,
2. if $\delta_{A-B}(q_0, p)$ is defined, then there is no prefix p' of p such that $\delta_B(q_0, p') \in F_B$,
3. if $\delta_{A-B}(q_0, p) \in F_{A-B}$ then $\delta_B(q_0, p)$ is undefined.

Definition
Default Unification (second version)
$A \sqcup\!! B = (A - B) \sqcup B$.

It should be obvious that characteristics 1–3 continue to hold. Uniqueness follows in this case from the fact that the difference operation will give a unique result. ($A - B$ can be constructed from A by checking for each state in A whether it must be removed

or not and ensuring that the resulting automaton is connected.) For instance, assuming A and B to be defined as in Example 6, we find that $A - B$ is:

Example 7

$$A - B = \quad [\ g : [\ g : [\ f : a\ ]\ ]\ ]$$

Note that in $A - B$ all parts that are identical in A and B are removed, whereas this was not the case for A', as defined in Definition 3.1. The outcome of default unification, however, is identical in both cases. The reason for this restriction on $A - B$ will become apparent below.

While default unification monotonically extends the nondefault argument (i.e. $B \sqsubseteq A \sqcup ! B$) and nonmonotonically extends the default argument, the operation itself is monotonic in its default argument and nonmonotonic in its nondefault argument. The theorem below proves monotonicity for the default argument; that is, a more specific default argument will lead to a more specific outcome of default unification:

Theorem 1
For all feature structures A, B, and C, not containing reentrancies, if $A \sqsubseteq B$ then $(A \sqcup ! C) \sqsubseteq (B \sqcup ! C)$.

Proof
It suffices to show that $(A - C) \sqsubseteq (B - C)$, or in other words:

1. if $\lambda(\delta_{A-C}(q_0, p)) = a$ then $\lambda(\delta_{B-C}(q_0, p)) = a$, and
2. if $\delta_{A-C}(q_0, p)$ is defined then $\delta_{B-C}(q_0, p)$ is defined.

If these two conditions are met, there is a homomorphism from $A - C$ to $B - C$ as required by the definition of subsumption. (Remember that there are no reentrancies.)

Case (1): If $\lambda(\delta_{A-C}(q_0, p)) = a$, then (i) $\lambda(\delta_B(q_0, p)) = a$ (since $A - C \sqsubseteq A \sqsubseteq B$) and from the definition of $A - C$ it follows that (ii) there is no prefix p' of p such that $\delta_C(q_0, p') \in F_C$ nor is $\delta_C(q_0, p)$ defined. From (i) and (ii) it follows that $\delta_{B-C}(q_0, p)$ is defined and that $\lambda(\delta_{B-C}(q_0, p)) = a$.

Case (2): If $\delta_{A-C}(q_0, p)$ is defined and $\delta_{A-C}(q_0, p) \notin F_{A-C}$ (otherwise this case reduces to case (1)), it follows that (i) $\delta_B(q_0, p)$ is defined, and (ii) there is no prefix p' of p such that $\delta_C(q_0, p') \in F_C$. From (i) and (ii) it follows that $\delta_{B-C}(q_0, p)$ is defined. ■

Note, however, that addition of nondefault information does not necessarily lead to a more specific result. That is, the dual of Theorem 1. does not hold:

Example 8

$$\text{if } B \sqsubseteq C \text{ then } (A \sqcup ! B) \sqsubseteq (A \sqcup ! C)$$

The reason is that addition of nondefault information may lead to a larger amount of default information being removed, and thus, the resulting feature-structures $A \sqcup ! B$ and $A \sqcup ! C$ can be incompatible. An example that falsifies 8 is presented below.

Example 9

$$A = [f:a] \qquad B = [g:b] \qquad A \sqcup ! B = \begin{bmatrix} f:a \\ g:b \end{bmatrix}$$

$$C = \begin{bmatrix} f:b \\ g:b \end{bmatrix} \qquad A \sqcup ! C = \begin{bmatrix} f:b \\ g:b \end{bmatrix}$$

Finally, for feature structures without reentrancies, the following distribution law holds:

Theorem 2
For all feature structures A, B, and C, not containing any reentrancies and such that $A \sqcup B$ is defined, $(A \sqcup B) \sqcup ! C = (A \sqcup ! C) \sqcup (B \sqcup ! C)$

Proof
Since $(A \sqcup B) \sqcup ! C = ((A \sqcup B) - C) \sqcup C$ and $(A \sqcup ! C) \sqcup (B \sqcup ! C) = ((A - C) \sqcup C) \sqcup ((B - C) \sqcup C) = (A - C) \sqcup (B - C) \sqcup C$, it suffices to prove that $(A \sqcup B) - C = (A - C) \sqcup (B - C)$. Let $D = (A \sqcup B) - C$ and $E = (A - C) \sqcup (B - C)$. It must be shown that (1) $D \sqsubseteq E$ and (2) $E \sqsubseteq D$.

Case (1): If $\lambda(\delta_D(q_0, p)) = a$, then (i) $\lambda(\delta_{A \sqcup B}(q_0, p)) = a$ and thus $\lambda(\delta_A(q_0, p)) = a$ or $\lambda(\delta_B(q_0, p)) = a$ (since there are no reentrancies) and (ii) there is no prefix p' of p such that $\delta_C(q_0, p') \in F_C$, nor is $\delta_C(q_0, p)$ defined. From (i) and (ii) it follows that $\lambda(\delta_{A-C}(q_0, p)) = a$ or $\lambda(\delta_{B-C}(q_0, p)) = a$, and thus that $\lambda(\delta_E(q_0, p)) = a$. Similarly, if $\delta_D(q_0, p)$ is defined (but not an end state), it follows that $\delta_A(q_0, p)$ or $\delta_B(q_0, p)$ is defined and that there is no prefix p' of p such that $\delta_C(q_0, p) \in F_C$. Therefore, either $\delta_{A-C}(q_0, p)$ or $\delta_{B-C}(q_0, p)$ is defined, and thus $\delta_E(q_0, p)$ is defined. It follows that $D \sqsubseteq E$.

Case (2): If $\lambda(\delta_E(q_0, p)) = a$, then $\lambda(\delta_{A-C}(q_0, p)) = a$ or $\lambda(\delta_{B-C}(q_0, p)) = a$ (since there are no reentrancies). Therefore, $\lambda(\delta_{A \sqcup B}(q_0, p)) = a$ and also, there is no prefix p' of p such that $\delta_C(q_0, p') \in F_C$, nor is $\delta_C(q_0, p)$ defined. It follows that $\lambda(\delta_D(q_0, p)) = a$. Similarly if $\delta_E(q_0, p)$ is defined but not an end state, $\delta_D(q_0, p)$ is defined. It follows that $E \sqsubseteq D$. ■

As long as Theorem 2. holds, it is possible to define default unification by decomposing the default argument into simpler feature structures and adding these (nonmonotonically) to the nondefault argument. This approach appears to underlie some of the previous proposals, but is inadequate once reentrancies enter the picture.

4.2 Default Unification with Reentrancies

Taking reentrancies into account requires an extension of the difference operation. If we allow either default or nondefault information to refer to an extension of a nondefault or default reentrancy, respectively, there is in general no unique maximal element subsuming A and unifiable with B. A slight modification of Examples 2 and 3 will illustrate this.

Example 10

$$A = \begin{bmatrix} f:[f:a] \\ g:[f:b] \end{bmatrix} \qquad B = \begin{bmatrix} f:\boxed{1} \\ g:\boxed{1} \end{bmatrix}$$

Example 11

$$A = \begin{bmatrix} f : \boxed{1}[f : a] \\ g : \boxed{1}[f : a] \end{bmatrix} \quad B = \begin{bmatrix} g : [f : b] \end{bmatrix}$$

In Example 10, there is default information that refers to an extension of a nondefault reentrancy. $A - B$ could be constructed from A by removing either the fact that $\langle ff \rangle : a$ or $\langle gf \rangle : b$. In 11, nondefault information refers to an extension of a default reentrancy. In this case, we could either remove the reentrancy (and the fact that $\langle gf \rangle : a$) or remove the fact that $\langle ff \rangle : a$ and $\langle gf \rangle : a$ and preserve the reentrancy. Neither solution subsumes the other. To avoid such problems, it is best to avoid interaction between reentrancies and other information altogether and to treat reentrant nodes in a similar fashion as atomic nodes. That is, we remove default reentrancies if they refer to defined parts of the nondefault automaton, and default information in general is removed if it refers to extensions of nondefault reentrancies. Thus, the difference operation can be extended as follows:

Definition
Difference (final version) The difference of A and B is the maximal element $A - B$ in the subsumption ordering that meets the following conditions:

1. $A - B \sqsubseteq A$,
2. if $\delta_{A-B}(q_0, p)$ is defined, then there is no prefix p' of p such that $\delta_B(q_0, p') \in F_B$ or $\delta_B(q_0, p') = \delta_B(q_0, p'')(p' \neq p'')$,
3. if $\delta_{A-B}(q_0, p) \in F_{A-B}$ then $\delta_B(q_0, p)$ is undefined,
4. (4) if $\delta_{A-B}(q_0, p) = \delta_{A-B}(q_0, p')(p \neq p')$ then $\delta_B(q_0, p)$ and $\delta_B(q_0, p')$ are undefined.

The definition of default unification remains as before:

Definition
Default Unification (= second version)
$A \sqcup ! B = (A - B) \sqcup B$.

Again, characteristics 1–4 of default unification mentioned in the introduction of this section hold. Uniqueness of the result follows from the fact that $A - B$ is unique. ($A - B$ can be constructed in this case as follows: for all paths p, if $\delta_A(q_0, p) = \delta_A(q_0, p')$, and p is defined in B, introduce a new value for $\delta_A(q_0, p)$ such that the automata that have $\delta_A(q_0, p)$ and $\delta_A(q_0, p')$ as initial state are isomorphic. Next, check for all states in the modified automaton whether they must be removed and ensure that the resulting automaton is connected.)

The monotonicity properties of default unification also remain as before. The theorem below is the relevant generalization of Theorem 1.

Theorem 3
For all feature structures A, B, and C, if $A \sqsubseteq B$ then $(A \sqcup ! C) \sqsubseteq (B \sqcup ! C)$

Proof
It suffices to show that $A - C \sqsubseteq B - C$, or in other words:

1. if $\lambda(\delta_{A-C}(q_0, p)) = a$, then $\lambda(\delta_{B-C}(q_0, p)) = a$,

2. if $\delta_{A-C}(q_0,p) = \delta_{A-C}(q_0,p')$ then $\delta_{B-C}(q_0,p) = \delta_{B-C}(q_0,p')$, and

3. if $\delta_{A-C}(q_0,p)$ is defined, then $\delta_{B-C}(q_0,p)$ is defined.

If these three conditions are met, there is a homomorphism from $A - C$ to $B - C$ as required by the definition of subsumption.

Case (1): If $\lambda(\delta_{A-C}(q_0,p)) = a$, then (i) $\lambda(\delta_B(q_0,p)) = a$ (since $A - C \sqsubseteq A \sqsubseteq B$) and (ii) from the definition of $A - C$, it follows that there is no prefix p' of p such that $\delta_C(q_0,p') \in F_C$ or $\delta_C(q_0,p') = \delta_C(q_0,p'')$, nor is $\delta_C(q_0,p)$ defined. From (i) and (ii) it follows that $\delta_{B-C}(q_0,p)$ is defined and that $\lambda(\delta_{B-C}(q_0,p)) = a$.

Case (2): Similarly, if $\delta_{A-C}(q_0,p) = \delta_{A-C}(q_0,p')$, then (i) $\delta_B(q_0,p) = \delta_B(q_0,p')$, and (ii) there is no prefix p' of p such that $\delta_C(q_0,p') \in F_C$ or $\delta_C(q_0,p') = \delta_C(q_0,p'')$ nor is $\delta_C(q_0,p)$ defined. From (i) and (ii) it follows that $\delta_{B-C}(q_0,p) = \delta_{B-C}(q_0,p')$.

Case (3): If $\delta_{A-C}(q_0,p)$ is defined and $\delta_{A-C}(q_0,p) \notin F_{A-C}$ (otherwise this case reduces to case (1)) and $\delta_{A-C}(q_0,p)$ not reentrant (otherwise this case reduces to case (2)), it follows that (i) $\delta_B(q_0,p)$ is defined, and (ii) there is no prefix p' of p such that $\delta_C(q_0,p') \in F_C$ or $\delta_C(q_0,p') = \delta_C(q_0,p'')$. From (i) and (ii) it follows that $\delta_{B-C}(q_0,p)$ is defined. ■

The distribution law, however, continues to hold only in one direction:

Theorem 4
For all feature structures A, B, and C, such that $A \sqcup B$ is defined, $(A \sqcup !C) \sqcup (B \sqcup !C) \sqsubseteq (A \sqcup B) \sqcup !C$

Proof
As in the previous section, it suffices to prove that $(A - C) \sqcup (B - C) \sqsubseteq (A \sqcup B) - C$. From the fact that $X \sqsubseteq X'$ and $Y \sqsubseteq Y'$ implies $(X \sqcup Y) \sqsubseteq (X' \sqcup Y')$, it follows that $((A - C) \sqcup (B - C)) \sqsubseteq (A \sqcup B)$. Now, as in the previous proof, if some path p is atomic, reentrant, or merely defined in $(A-C) \sqcup (B-C)$, it follows that (i) p is atomic, reentrant, or defined in $A \sqcup B$ and (ii) there is no atomic or reentrant path p' in C that is a prefix of p, nor is p defined in C if p is atomic or reentrant in $(A - C) \sqcup (B - C)$. It follows that p is atomic, reentrant, or defined in $(A \sqcup B) - C$. ■

An illustration of this result is given below. Note that 12 also illustrates that the converse of Theorem 4. no longer holds.

Example 12

$$A = \begin{bmatrix} f : \boxed{1} \\ g : \boxed{1} \end{bmatrix}$$

$$B = \begin{bmatrix} g : a \end{bmatrix}$$

$$C = \begin{bmatrix} g : b \end{bmatrix}$$

$$(A \sqcup B) \sqcup !C = \begin{bmatrix} f : a \\ g : b \end{bmatrix}$$

$$(A \sqcup !C) \sqcup (B \sqcup !C) = \begin{bmatrix} f : nil \\ g : b \end{bmatrix}$$

4.3 Add Conservatively

Defining default unification as $(A - B) \sqcup B$ will fail to capture the idea of Shieber's (1986b) *add conservatively*, as the difference operation completely removes a default reentrancy if one of the paths leading to it is also defined in the nondefault argument. However, linguistic applications, such as an encoding of the *Head Feature Convention*, indicate that a more subtle approach should be taken. In particular, if a default structure contains the information that $\langle p \rangle = \langle p' \rangle$, whereas in the nondefault structure $\langle pl \rangle$ is defined for some feature l, we want to treat only l as an exception to the general rule that $\langle p \rangle = \langle p' \rangle$, and preserve the information that $\langle pl' \rangle = \langle p'l' \rangle$ (for $l' \neq l$).

We implement this idea using the following operation:

Definition

Let A and B be automata. The **extension** of A relative to B ($Ext(A, B)$) is the minimal (i.e. most general) element $Ext(A, B)$ such that

1. $A \sqsubseteq Ext(A, B)$,
2. if $\delta_A(q_0, p) = \delta_A(q_0, p')$ and $\delta_B(q_0, pql)$ is defined (for some $pql \in \Sigma^*$), then $\delta_{Ext(A,B)}(q_0, pql') = \delta_{Ext(A,B)}(q_0, p'ql')$ (wherever possible) for all $l' \in \Sigma$.

The automaton A is extended, sometimes somewhat redundantly, with reentrant paths that are extensions of paths already reentrant in A. $Ext(A, B)$ is nevertheless usually more informative than A itself, as the addition of a path pl blocks unification with feature structures in which p receives an atomic value. Note furthermore that path extensions are not always possible; that is, if $\delta_A(q_0, p) \in F_A$ and $\delta_B(q_0, pl)$ is defined, there is no extension of A in which pl is defined. (This explains the *wherever possible*). In order to get all relevant path-extensions, Σ will in general be the set of all features defined in the grammar, although in particular cases Σ can be restricted to a smaller set (the set of *head*-features, for instance).

We are now ready to give a definition of default unification that incorporates the effects of *add conservatively*. To avoid confusion, we use the operator $\sqcup_{ac}!$ for this extended version of default unification.

Definition
Default Unification (final version)
$A \sqcup_{ac}! B = (Ext(A, B) - B) \sqcup B$.

An example of default unification involving reentrancies is presented below. We assume that the set of features $\Sigma = \{f, g\}$.

Example 13

$$A = \begin{bmatrix} f : \boxed{1} \\ g : \boxed{1} \end{bmatrix}$$

$$B = [\, f : [\, f : a \,] \,]$$

$$Ext(A, B) = \begin{bmatrix} f : \boxed{1} \begin{bmatrix} f : \boxed{2} \\ g : \boxed{3} \end{bmatrix} \\ g : \boxed{1} \begin{bmatrix} f : \boxed{2} \\ g : \boxed{3} \end{bmatrix} \end{bmatrix}$$

$$Ext(A,B) - B = \begin{bmatrix} f : [\, g : \boxed{3} \,] \\ g : \begin{bmatrix} f : nil \\ g : \boxed{3} \end{bmatrix} \end{bmatrix}$$

$$(Ext(A,B) - B) \sqcup B = \begin{bmatrix} f : \begin{bmatrix} f : a \\ g : \boxed{3} \end{bmatrix} \\ g : \begin{bmatrix} f : nil \\ g : \boxed{3} \end{bmatrix} \end{bmatrix}$$

The example shows that default unification is slightly more restrictive than *add conservatively*, since the original reentrancy is removed even though A and B would have been unifiable. The reason is of course that this will guarantee uniqueness of the result of default unification, whereas this is not the case for *add conservatively*.

5. Linguistic Applications of Default Unification

In this section, we sketch how default unification can be incorporated in a grammar formalism and argue briefly that this can be an alternative for some of the extensions mentioned in Section 2.

5.1 Nonmonotonic Template Inheritance

In grammar formalisms such as PATR-II, feature structures are defined as sets of equations and templates. Each equation or template denotes a feature structure (i.e. the minimal feature structure that satisfies the equation or the equations that make up the template definition), and the denotation of a set of such elements is simply the unification of all their denotations. Incorporation of default unification requires that a distinction is made between default and nondefault information. In the notation used here, nondefault information is prefixed by a "!". The feature structure denoted by a definition that contains both default and nondefault information is arrived at by first unifying all default information and unifying all nondefault information. Next, the two feature structures are combined by means of default unification ($\sqcup_{ac}$!).

If templates are incorporated as default information, the feature structure denoted by the template is inherited nonmonontonically. (Monotonic inheritance is possible as well of course: this is achieved by prefixing a template with "!".) As an illustration, consider the following fragment, in which an attempt is made to encode some of the peculiarities of the English auxiliary system in a lexicalist grammar:

Example 14

NP : ($\langle cat \rangle = n$
$\langle nform \rangle = norm$).

$VERB$: ($\langle cat \rangle = v$
$\langle aux \rangle = -$
$\langle inv \rangle = -$).

VP : ($VERB$
$\langle subcat\ first \rangle = NP$
$\langle subcat\ rest \rangle = empty$).

AUX : ($VERB$
$!\langle aux \rangle = +$
$!\langle inv \rangle = +$
$\langle subcat\ first \rangle = VP$
$\langle subcat\ rest\ first \rangle = NP$
$\langle subcat\ rest\ rest \rangle = empty$
$!\langle subcat\ first\ subcat\ first\ nform \rangle =$
$\langle subcat\ rest\ first\ nform \rangle$).

Adding the equations $!\langle aux \rangle : +$ and $!\langle inv \rangle : +$[5] to the definition of *AUX* has an effect comparable to that of the *overwrite*-operation of (Shieber 1986a, p. 60). The *AUX* template inherits from *VERB* by default, but the equations just mentioned block inheritance of the values for $\langle inv \rangle$ and $\langle aux \rangle$. However, default unification allows us to do more. An auxiliary does not subcategorize for an ordinary NP subject, nor does it subcategorize for a complement VP that subcategorizes for an ordinary NP subject. Rather, the restrictions to be placed on the *nform* of the subject are inherited from the embedded VP:

Example 15

a. it will annoy Kim that she lost
b. *Sue will annoy Kim that she lost

This dependency between elements of the subcat list is encoded in the final equation, which also suppresses (or overwrites) the default value for $\langle nform \rangle$. The denotation of *AUX* is thus:

Example 16

$$\left[\begin{array}{l} cat : v \\ aux : + \\ inv : + \\ subcat : \left[\begin{array}{l} first : \left[\begin{array}{l} cat : v \\ aux : - \\ inv : - \\ subcat : \left[\begin{array}{l} first : \left[\begin{array}{l} cat : np \\ nform : \boxed{1} \end{array}\right] \\ rest : empty \end{array}\right] \end{array}\right] \\ rest : \left[\begin{array}{l} first : \left[\begin{array}{l} cat : np \\ nform : \boxed{1} \end{array}\right] \\ rest : empty \end{array}\right] \end{array}\right] \end{array}\right]$$

The nonmonotonic inheritance regime is flexible enough to allow for exceptions to exceptions. Gazdar et al. (1985, p. 65) observe that at least in some dialects of English, the auxiliary *might* cannot occur in inverted structures. This is expressed in the following lexical entry, in which *might* inherits nonmonotonically from *AUX*, which itself inherits nonmonotonically from *VERB*:

Example 17

$$might \quad : \left(\begin{array}{l} AUX \\ !\langle inv \rangle = - \end{array} \right).$$

There is an important difference between the approach to nonmonotonic inheritance sketched here and the majority of inheritance-based formalisms used for Knowledge Representation, which has to do with the way in which templates are evaluated. If a template is used as part of the definition of another feature structure, all we need to know to determine the denotation of this feature structure is the *denotation* of this

5 Note that the feature *INV* as used here indicates only whether a (lexical) item may occur in an inverted structure. It does not distinguish between inverted and noninverted clauses.

template (which is a feature structure). How this template was *defined* (as a set of equations or as a combination of (more general) templates, as a combination of default and nondefault information or not) is completely irrelevant to its meaning. Thus, the denotation of *AUX* would remain as before, if we defined it as:

Example 18

$$\begin{array}{lll} AUX: & (& \langle cat \rangle = v \\ & & \langle aux \rangle = + \\ & & \langle inv \rangle = + \\ & & \ldots \qquad). \end{array}$$

Consequently, the denotation of *might* is not affected by this change in definition either.

The role of *classes* (or *frames*) in inheritance-based systems, however, as described in, for instance, Touretzky (1986), is rather different. To determine the denotation of a class *might* that inherits from a class *AUX*, we not only need to know the contents of *AUX*, but also the classes from which *AUX* inherits. The latter is important for resolving multiple-inheritance conflicts. If the class *might* inherits from both *AUX* and *VERB*, for instance, and *AUX* in its turn inherits from *VERB* as well, information inherited from *AUX* must take precedence over information from *VERB*, as the former is more specific than the latter. In our nonmonotonic inheritance mechanism for templates, such reasoning is impossible. Adding the template *VERB* as default information to the definition of the template (or lexical entry) *might* would lead to a unification failure of the default information, and thus the definition as a whole would be considered as illegal.[6] This is as it should be, we believe, given the fact that the inheritance hierarchy as such should not play a role in determining the meaning of templates. The denotation of the template *AUX* is the feature structure in 16 (i.e., whether it is defined as in 14 or as in 18 is irrelevant), and from that it is impossible to conclude that *AUX* inherits from *VERB*, and thus the kind of reasoning used to justify the resolution of feature conflicts used in Touretzky (1986) is not applicable in our case.

5.2 Lexical Defaults

The definition of auxiliaries above is still unsatisfactory in that it predicts that auxiliaries subcategorize for verbal complements that are specified as $\langle aux \rangle = -$. Clearly, this requirement is too strong (although it is correct for the auxiliary *do*). One way to solve this problem is to redefine the *AUX*-template as:

Example 19

$$\begin{array}{lll} AUX & : (& \ldots \\ & & \langle subcat\ first\ cat \rangle = v \\ & & \langle subcat\ first\ subcat\ first \rangle = NP \\ & & \langle subcat\ first\ subcat\ rest \rangle = empty \\ & & \ldots \qquad). \end{array}$$

6 Of course, it is possible to combine incompatible default information if we impose the correct ordering explicitly. This can be done by using definitions (i.e. a set of equations in brackets) in definitions:

$$\begin{array}{lll} might & : (& (VERB \quad !AUX) \\ & & !\langle inv \rangle = - \qquad). \end{array}$$

This is equivalent to the definition of *might* given in 17, albeit more complex and possibly misleading.

This solution seems inelegant, however, as it reconstructs part of the *VP*-template in order to express the correct subcategorization requirements. Thus, the obvious generalization that an auxiliary subcategorizes for a *VP* is missed by this redefinition. The source of this inelegance is the fact that *VP* inherits from *VERB*, and that *VERB* contains default information about properties typical for verbs. However, while these properties hold for the vast majority of verbs, it is not the case that if an element subcategorizes for a verbal complement, the default properties need to hold for the complement as well. What is needed here is a distinction between properties that hold by default for all members of a class and default properties that can be assumed to hold if a lexical item subcategorizes for members of this class. While the latter can be expressed safely by means of templates, the former are more adequately expressed in the form of *lexical defaults*.

The extension of unification-based formalisms with lexical defaults can be implemented using default unification. The effect of lexical defaults is comparable to that of lexical *Feature Specification Defaults* in GPSG. A lexical default is a statement of the form *Name: Ant* $\Rightarrow$ *Cons*, where *Ant* and *Cons* are feature structure descriptions. The interpretation of lexical defaults is that the feature structure of each lexical entry that is subsumed by the antecedent of a lexical default is extended, by means of default unification, with the contents of the consequent. Lexical entries are thus compiled in two stages: first, the denotation of the feature structure description is computed and next, the lexical defaults are applied to this feature structure.

Consider for example the following lexical defaults:

Example 20

$$\begin{array}{llllll} FSD1 & :(& VERB &) \Rightarrow & (& \langle aux \rangle = - \ ). \\ FSD2 & :(& \langle aux \rangle = - &) \Rightarrow & (& \langle inv \rangle = - \ ). \end{array}$$

The fragment in 14 and 17 is assumed to be redefined as follows:

Example 21

$$\begin{array}{lll} VERB: & (& \langle cat \rangle = v \ ). \\ VP: & (& VERB \\ & & \langle subcat\ first \rangle = NP \\ & & \langle subcat\ rest \rangle = empty \ ). \\ AUX: & (& VERB \\ & & \langle aux \rangle = + \\ & & \langle subcat\ first \rangle = VP \\ & & \ldots \ ). \\ might: & (& AUX \\ & & \langle inv \rangle = - \ ). \end{array}$$

Each verbal lexical item will be extended with the information $\langle aux \rangle = -$, unless it is an auxiliary of course, since in that case, the lexical entry is already specified as $\langle aux \rangle = +$. Only nonauxiliary verbs are extended with the information $\langle inv \rangle = -$ (FSD2).[7] Auxiliaries remain unspecified for this feature, thus capturing the fact that

7 The evaluation of these two lexical defaults is thus order-sensitive. The same situation can in principle arise in GPSG as well, although the particular example given here is avoided in GKPS by

auxiliaries *can*, but not necessarily *do*, occur in inverted structures.[8] The exceptional character of *might* is expressed in this case by adding explicitly the information that it cannot invert.

The problem sketched at the beginning of this section is now resolved. An auxiliary subcategorizes for a *VP*, which in its turn inherits from the template *VERB*. However, since the latter template no longer contains default information that should hold for lexical entries only, an auxiliary no longer subcategorizes for verbal complements that are $\langle aux \rangle : -$. Auxiliaries that subcategorize for a restricted set of verbal complements, such as *do*, which requires a $\langle aux \rangle : -$ complement, can be encoded by adding the relevant constraint to their lexical entries.

5.3 Specialization of Reentrancies

Another important property of default unification is that it enables us to define exceptions to a reentrancy. Consider for instance the following GPSG rule (where H indicates the head of the rule):

Example 22

$$S \rightarrow X^2 \;\; H[-subj]$$

The symbol S can be analyzed as the feature structure in 23. Applying the *Head Feature Convention* to the rule in 22 amounts to adding to H all head features compatible with head features in S. Using default unification, this is implemented in 24 as a default reentrancy that equates the head features of S and H.

Example 23

$$\begin{array}{lll} S & : (& \langle head\ n \rangle = - \\ & & \langle head\ v \rangle = + \\ & & \langle head\ bar \rangle = 2 \\ & & \langle head\ subj \rangle = + \quad). \end{array}$$

Example 24

$$\begin{array}{lll} S\text{-}rule & : & X_0 \rightarrow X_1 \;\; X_2; \\ & (& \langle X_0 \rangle = S \\ & & \langle X_1\ head\ bar \rangle = 2 \\ & & !\langle X_2\ head\ subj \rangle = - \\ & & \langle X_0\ head \rangle = \langle X_2\ head \rangle \quad). \end{array}$$

The final equation in 24 both implements the *HFC* and defines X_2 as the head daughter. An exception to the reentrancy is the fact that $\langle X_2\ head\ subj \rangle = -$, which is therefore represented as nondefault information. In this approach, the *HFC* is part of the rules itself and thus, the effect of Shieber's (1986b) special-purpose compilations step, which *adds* the HFC *conservatively*, is achieved directly.

implementing the effect of FSD2 above as a feature coocurrence restriction.

8 Note that, as in GPSG, the feature *INV* plays a double role by indicating both an item's potential to occur in inverted structures as well as indicating whether a given structure is inverted or not.

6. Conclusions

We have shown in the preceding sections that it is possible to incorporate nonmonotonicity in a unification-based formalism, while at the same time preserving the idea that linguistic knowledge is represented in the form of feature structures.

In spite of their great flexibility, unification-based formalisms are in general not very well equipped to deal with linguistic rules or generalizations that have a default character and for which exceptions exist. In Sections 4 and 5 we hope to have demonstrated that a nonmonotonic operation on feature structures combined with straightforward extensions of the description languages used in unification-based formalisms enables a satisfactory account of the phenomena mentioned in the introduction. The applications illustrate that default unification can be used to give linguistically appealing implementations of certain natural language phenomena, not that it would be impossible to account for these facts using unification only. Thus, default unification serves to extend the expressive power of unification-based formalisms, but leaves the representation method of unification-based formalisms, in which linguistic objects are represented as feature structures, unchanged. Comparing default unification to earlier proposals, we believe that an advantage of our approach is that it is general, in the sense that one operation is used to achieve the effects of *overwriting, add conservatively, nonmonotonic template inheritance,* and *priority union.* Also, whereas previous proposals do not seem to be well behaved for feature structures containing reentrancies, default unification is defined for feature structures of arbitrary complexity.

Dörre et al. (1990) suggest that the use of nonmonotonic devices in unification-based formalisms will, for the time being, be limited to *off-line* extensions of these formalisms; that is, extensions whose effect can be computed at compile time and result in ordinary feature structures. They also note that while there may be linguistic arguments in favor of more dynamic notions of default reasoning, from a computational point of view the off-line approach is clearly preferred. Default unification, as used in the previous section, is an example of an off-line extension, as the effects of nonmonotonic template inheritance, lexical defaults, and the meaning of rule definitions in which default and non-default information is combined, can be computed at compile time. Again, this emphasizes the point that incorporation of default unification in principle only extends the expressive power of unification-based formalisms.

Acknowledgments

A syntactic approach to default unification is presented in Bouma (1990). The reactions on that paper made it clear to me that default unification should be defined not only for feature structure descriptions, but also for feature structures themselves. For helpful questions, suggestions, and comments on the material presented here, I would like to thank Bob Carpenter, John Nerbonne, audiences in Tilburg, Groningen, Tübingen, and Düsseldorf, and three anonymous CL reviewers.

References

Bouma, Gosse (1990). "Defaults in unification grammar." In *Proceedings, 28th Annual Meeting of the Association for Computational Linguistics,* Pittsburgh, PA, 165–172.

Bresnan, Joan, and Kaplan, Ronald (1982). "Lexical functional grammar: A formal system for grammatical representation." In *The Mental Representation of Grammatical Relations,* edited by J. Bresnan, 173–281. Cambridge, MA: The MIT Press.

Dawar, Anuj, and Vijay-Shanker, K. (1990). "An interpretation of negation in feature structure descriptions." *Computational Linguistics,* 16(1), 11–21.

Daelemans, Walter (1988). "A model of Dutch morphophonology and its applications." *AI Communications,* 1(2), 18–25.

Dörre, Jochen; Eisele, Andreas; Wedekind, Jürgen; Calder, Jo; and Reape, Mike (1990). *A Survey of Linguistically Motivated Extensions to Unification-Based Formalisms.* DYANA Deliverable R3.1.A., Centre for

Cognitive Science, University of Edinburgh.

Evans, Roger, and Gazdar, Gerald (1989a). "Inference in DATR." In *Proceedings, Fourth Conference of the European Chapter of the ACL*," University of Manchester, 66–71.

Evans, Roger, and Gazdar, Gerald (1989b). "The semantics of DATR." In *Proceedings, Seventh Conference of the Society for the Study of Artificial Intelligence and the Simulation of Behaviour*, edited by A. Cohn, 79–87. London: Pitman Publ.

Flickinger, Daniel (1987). *Lexical Rules in the Hierarchical Lexicon*. Doctoral dissertation, Stanford University, Stanford, CA.

Flickinger, Daniel; Pollard, Carl; and Wasow, Thomas (1985). "Structure-sharing in lexical representation." In *Proceedings, 23rd Annual Meeting of the Association for Computational Linguistics*. Chicago, Illinois, 262–267.

Gazdar, Gerald; Klein, Ewan; Pullum, Geoffrey; and Sag, Ivan (1985). *Generalized Phrase Structure Grammar*. London: Blackwell.

Kaplan, Ronald (1987). "Three seductions of computational psycholinguistics." In *Linguistic Theory and Computer Applications*, edited by P. Whitelock, H. Somers, P. Bennett, R. Johnson, and M. McGee Wood, 149–188. London: Academic Press.

Kasper, Robert, and Rounds, William (1986). "A logical semantics for feature structures." In *Proceedings, 26th Annual Meeting of the Association for Computational Linguistics*. New York, NY, 257–266.

Kasper, Robert, and Rounds, William (1990). "The logic of unification in grammar." *Linguistics and Philosophy*, 13(1), 35–58.

Pollard, Carl, and Sag, Ivan (1987). *Information-Based Syntax and Semantics, Volume 1: Fundamentals*. CSLI Lecture Notes 13. Chicago: University of Chicago Press.

Shieber, Stuart (1986a). *An Introduction to Unification-Based Approaches to Grammar*. CSLI Lecture Notes 4. Chicago: University of Chicago Press.

Shieber, Stuart (1986b). "A simple reconstruction of GPSG." In *Proceedings, COLING 1986*. Bonn, Germany, 211–215.

De Smedt, Koenraad (1990). *Incremental Sentence Generation*. Doctoral dissertation, Katholieke Universiteit Nijmegen, Nijmegen, The Netherlands.

Touretzky, David (1986). *The Mathematics of Inheritance Systems*. Los Altos, CA: Morgan Kaufmann.