DEFAULTS IN UNIFICATION GRAMMAR

Gosse Bouma
Research Institute for Knowledge Systems
Postbus 463, 6200 AL Maastricht, The Netherlands
e-mail : gosse@riksnl.uucp

ABSTRACT

Incorporation of defaults in grammar formalisms is important for reasons of linguistic adequacy and grammar organization. In this paper we present an algorithm for handling default information in unification grammar. The algorithm specifies a logical operation on feature structures, merging with the non-default structure only those parts of the default feature structure which are not constrained by the non-default structure. We present various linguistic applications of default unification.

1. INTRODUCTION

MOTIVATION. There a two, not quite unrelated, reasons for incorporating defaults mechanisms into a linguistic formalism. First, linguists have often argued that certain phenomena are described most naturally with the use of rules or other formal devices that make use of a notion of default (see, for instance, Gazdar 1987). The second reason is that the use of defaults simplifies the development of large and complex grammars, in particular, the development of lexicons for such grammars (Evans & Gazdar 1988). The latter suggests that the use of defaults is of particular relevance for those brands of Unification Grammar (UG) that are lexicalist, that is, in which the lexicon is the main source of grammatical information (such as Categorial Unification Grammar (Uskoreit 1986, Calder et al. 1988) and Head-driven Phrase Structure Grammar (Pollard & Sag 1987)).

We propose a method for incorporating defaults into UG, in such a way that it both extends the linguistic adequacy of UG and supports the formulation of rules, templates and lexical entries for many unification-based theories. In the next section, we define *default unification*, a logical operation on feature structures. It is defined for a language, *FML**, which is in many respects identical to the language *FML* as defined in Kasper & Rounds (1986). Next, we come to linguistic applications of default unification. A linguistic notation is introduced, which can be used to describe a number of linguistically interesting phenomena, such as feature percolation, coordination, and many aspects of inflectional morphology. Furthermore, it can be used in the sense of Flickinger et al. (1985) to define exceptions to rules, non-monotonic specialization of templates or irregular lexical entries.

BACKGROUND. There are several proposals which hint at the possibility of adding default mechanisms to the linguistic formalisms and theories just mentioned. The fact that GPSG (Gazdar et al., 1985) makes heavy use of defaults, has led to some research concerning the compatibility of GPSG with a formalism such PATR-II (Shieber 1986a) and concerning the logical nature of the mechanisms used in GPSG (Evans 1987). Shieber (1986a) proposes an operation *add conservatively*, which adds information of a feature structure A to a feature structure B, in as far as this information is not in conflict with information in B. Suggestions for similar operations can be found in Shieber (1986b:59-61) (the *overwrite* option of PATR-II) and Kaplan (1987) (*priority union*). Flickinger et al. (1985) argue for the incorporation of default inheritance mechanisms in UG as an alternative for the template system of PATR-II.

A major problem with attempts to define an operation such as default unification for complex feature structures, is that there are at least two ways to think about this operation. It can be defined as an operation which is like ordinary unification, with the exception that in case of a unification failure, the value of the non-default feature structure takes precedence (Kaplan 1987, Shieber 1986a). Another option is not to rely on unification failure, but to remove default information about a feature f already if the non-default feature structure constrains the contents of f in some way. This view underlies most of the default mechanisms used in GPSG[1]. The

[1] Actually, in GPSG both notions of default unification are used. In Shieber's (1986a) formulation of the of the *Foot Feature Principle*, for example, the operation *add conservatively* (which normally relies on unification failure) is restricted to features that are *free* (i.e. uninstantiated and not covarying with some other feature).

distinction between the two approaches is especially relevant for reentrant feature values.

The definition presented in the next section is defined as an operation on arbitrary feature structures, and thus it is more general than the operations *add conservatively* or *overwrite*, in which only one sentence at a time (say, *<X_0 head> = <X_1 head>* or *<subject case> = nominative*) is added to a feature description. An obvious advantage of our approach is that overwriting a structure F with F' is equivalent to adding F as default information to F'. Default unification, as defined below, follows the approach in which default information is removed if it is constrained in the non-default structure. This decision is to a certain extent linguistically motivated (see section 3), but perhaps more important is the fact that we wanted to avoid the following problem. For arbitrary feature structures, there is not always a unique way to resolve a unification conflict, nor is it necessarily the case that one solution subsumes other solutions. Consider for instance the examples in (1).

(1)	*default*	*non-default*
a.	<f> = a	<f> = <g>
	<g> = b	
b.	<f> = <g>	<f> = a
	<g> = b	

To resolve the conflict, in (a), either one of the equations could be removed. In (b), either the fact that *<g> = b* or the reentrancy could be removed (in both cases, this would remove the inplicit fact that *<f> = b*). An approach which only tries to remove the sources of a unification conflict, will thus be forced to make arbitrary decisions about the outcome of the default unification procedure. At least for the purposes of grammar development, this seems to be an undesirable situation[1].

2. DESCRIPTION OF THE ALGORITHM

*THE LANGUAGE FML**. Default unification is defined in terms of a formal language for feature structures, based on Kasper & Rounds' (1986) language FML. FML* does not contain disjunction, however, and furthermore, equations of the form *l: f* (where ϕ is an arbitrary formula) are replaced by equations of the form <p> : α (where α is atomic or NIL or TOP).

(2) **SYNTAX OF FML***

NIL	
TOP	
a	a $\in$ A (the set of atoms)
<p> : α	p $\in$ L* (L the set of labels) and $\alpha \in$ A $\cup$ {TOP,NIL}
[<p_1>,...,<p_n>]	each $p_i \in$ L*
$\phi \wedge \psi$	$\phi, \psi \in$ FML*

We assume that feature structures are represented as directed acyclic graphs (*dags*). The denotation D(ϕ) of a formula ϕ is the minimal element w.r.t. subsumption[2] in the set of dags that satisfy it. The conditions under which a dag D satisfies a formula of FML* (where D/<p> is the dag that is found if we follow the path p through the dag D) are as follows :

(3) **SEMANTICS OF FML***

a.	D $\models$ NIL	always
b.	D $\models$ TOP	never
c.	D $\models$ a	if D = a
d.	D $\models$ <p>: α	if D/<p> is defined[3] and D/<p> $\models \alpha$.
e.	D $\models \phi \wedge \chi$	if D $\models \phi$ and D $\models \chi$
f.	D $\models$ [<p_1>,...,<p_n>]	if the values of all p_i ($1 \le i \le n$) are equivalent.

NORMAL FORM REQUIREMENTS. Default unification should be a semantically well-behaved operation, that is, the result of this operation should depend only on the denotation of the formula's involved. Since default unification is a non-monotonic operation, however, in which parts of the default information may disappear, and since there are in general many formulae denoting the same dag, establishing this is not completely trivial. In particular, we must make sure that the formula which provides the default information is in the following normal form:

1 However, in Evans' (1987) version of *Feature Specification Defaults*, it is simply allowed that a category description has more than one 'stable expansion'.

2 A dag D subsumes a dag D' if the set of formulae satisfying D' contains the set of formulae satisfying D (Eisele & Dörre, 1988: 287).

3 D/<l> is defined iff l $\in$ Dom(D). D/<lp> is defined iff D/<l> and D'/<p> are defined, where D'= D/<l>.

(4) **FML* Normal Form**

A formula ϕ is in FML* NF iff:

a. $\forall E$ in ϕ, $\langle p_1 p_2\rangle : \alpha$ in ϕ:

$\langle p_1\rangle \in E \rightarrow \forall p_3 \in E : \langle p_3 p_2\rangle : \alpha$ in ϕ

b. $\forall E_1, E_2$ in ϕ:

$\langle p_1 p_2\rangle \in E2, \langle p_1\rangle \in E_1 \rightarrow$

$\forall p_3 \in E1 : \langle p_3 p_2\rangle \in E_2$

c. $\forall E$ in ϕ, there is no $\langle p\rangle \in E$, such that $\langle pl\rangle$ is realized in ϕ.

d. $\forall E$ in ϕ, there is no $\langle p\rangle \in E$ such that $\langle p\rangle : a$ $(a \in A)$ is in ϕ.

(5) **REALIZED**

A path $\langle pl\rangle$ is realized in ϕ iff $\langle pl'\rangle$ is defined in $D(\phi)$ $(l, l' \in L)$ (cf. Eisele & Dörre, 1988 : 288).

For every formula ϕ in FML*, there is a formula ϕ' in FML* NF, which is equivalent to it w.r.t unification, that is, for which the following holds:

(6) $\forall \chi \in$ FML*: $\phi \wedge \chi \neq \mathrm{TOP} \Leftrightarrow \phi' \wedge \chi \neq \mathrm{TOP}$

Note that this does not imply that ϕ and ϕ' have the same denotation. The two formulae below, for example, are equivalent w.r.t. unification, yet denote different dags :

(7) a. $\langle f\rangle : a \wedge [\langle f\rangle, \langle g\rangle]$
b. $\langle f\rangle : a \wedge \langle g\rangle : a$

For conditions (4a,b), it is easy to see that (6) holds (it follows, for instance, from the equivalence laws (21) and (22) in Kasper & Rounds, 1986: 261). Condition (4c) can be met by replacing every occurence of an equivalence class $[\langle p_1\rangle, \ldots, \langle p_n\rangle]$ in a formula ϕ by a conjunction of equivalences $[\langle p_1 l\rangle, \ldots, \langle p_n l\rangle]$ for every $\langle p_i l\rangle$ $(1 \leq i \leq n)$ realized in $D(\phi)$. For example, if $L = \{f, g\}$, (8b) is the NF of (8a).

(8) a. $[\langle f\rangle, \langle g\rangle] \wedge \langle ff\rangle : \mathrm{NIL}$
b. $[\langle ff\rangle, \langle gf\rangle] \wedge [\langle fg\rangle, \langle gg\rangle] \wedge \langle ff\rangle : \mathrm{NIL}$

Condition (4d) can be met by eliminating equivalence classes of paths leading to an atomic value. Thus, (7b) is the NF of (7a). Note that the effect of (4c,d) is that the value of every path which is member of some equivalence class is NIL.

A default formula has to be in FML* NF for two reasons. First, all information which is implicit in a formula, should be represented explicitly, so we can check easily which parts of a formula need to be removed to avoid potential unification conflicts with the non-default formula. . This is guaranteed by (4a,b). Second, all reentrant paths should have NIL as value. This is guaranteed by (4c,d) and makes it possible to replace an equivalence class by a weaker set of equations, in which arbitrary long extensions of the old pathnames may occur (if some path would have a value other than NIL, certain extensions could lead to inconsistent results).

Laws for Default Unification. Default unification is an operation which takes two formulas as arguments, representing default and non-default information respectively. The dag denoted by the resultant formula is subsumed by that of the non-default argument, but not necessarily by that of the default argument.

The laws for default unification (defined as *Default* $\oplus$ *Non-default* = *Result*, where *Default* is in FML*-NF) are listed below.

(9) **Default Unification :**

a. $\phi \oplus \mathrm{NIL} = \phi$
$\phi \oplus \mathrm{TOP} = \mathrm{TOP}$
$\mathrm{NIL} \oplus \phi = \phi$
$\mathrm{TOP} \oplus \phi = \phi$

b. $a \oplus \phi = \phi$
$\phi \oplus a = a$

c. $\langle p\rangle : \alpha \oplus \phi = \phi$, if $D(\phi) \models \langle p'\rangle : a$, p' a prefix of p, $a \in A$.
$= \phi$, if $D(\phi) \models \langle pp'\rangle : \alpha$,
$= \phi$, if $\exists p' \in E$: $D(\phi) \models E$ and p' is a prefix of p.
$= \langle p\rangle : \alpha \wedge \phi$, otherwise.

d. $E \oplus \phi = E\text{-}E'//Z$
where E' is $\{\langle p\rangle \in E \mid D(\phi) \models E'$ and $p' \in E'\}$ $\cup \{\langle p\rangle \in E \mid D(\phi) \models \langle p'\rangle : a\}$ (p' a prefix of p, $a \in A$) and Z is $\{\langle p'\rangle \mid D(\phi) \models \langle pp'\rangle : \alpha$, and $p \in E\}$.

e. $(\psi \wedge \chi) \oplus \phi = \phi$, if $\psi \wedge \chi = \mathrm{TOP}$,
$= (\psi \oplus \phi) \wedge (\chi \oplus \phi)$, otherwise.

This definition of default unification removes all default information which might lead to a unification conflict. Furthermore, it is designed in such a way that the order in which information is removed is irrelevant (note that otherwise the second case in (9e) would be invalid). The first two cases of (9c) are needed to remove all sentences $\langle p \rangle : \alpha$, which refer to a path which is blocked or which cannot receive an atomic value in ϕ. The third case in (9c) is needed for situations such as (10).

(10) $(\langle f\, g \rangle : a \wedge \langle h\, g \rangle : b) \oplus [\langle f \rangle, \langle h \rangle]$

In (9d), we first remove from an equivalence class all paths which have a prefix that is already in an equivalence class or which has an atomic value. The result of this step is E-E'. Next, we modify the equivalence class, so that it allows exceptions (i.e. the possibility of non-unifiable values) for all paths which are extensions of paths in E-E' and are defined in ϕ. We can think of modified equivalence classes as abbreviations for a set of (unmodified) equivalence classes:

(11) $[\langle p_1 \rangle,\ldots,\langle p_n \rangle]//Z = \phi$, where ϕ is the conjunction of all equivalence classes $[\langle p_1 pl \rangle,\ldots,\langle p_n pl \rangle]$, such that pl is not defined in Z, but pl' is in Z, for some $l, l' \in L$

An example should make this clearer:

(12) $[\langle f \rangle,\langle g \rangle,\langle h \rangle] \oplus (\langle g \rangle : a \wedge \langle f\, g \rangle : b) =$

$[\langle f \rangle,\langle h \rangle]//\{\langle g \rangle\} \wedge (\langle f \rangle : a \wedge \langle f\, g \rangle : b)$.

The result of default unification in this case is that one element ($\langle g \rangle$) is removed from the default equivalence class since it is constrained in by the non-default information. Furthermore, the equivalence is modified, so that it allows for exceptions for the paths $\langle f\, g \rangle$ and $\langle h\, g \rangle$. Applying the rule in (11), and assuming that $L = \{f,g,h\}$, we conclude that

(13) $[\langle f \rangle,\langle h \rangle]//\{\langle g \rangle\} =$

$[\langle f\, f \rangle, \langle h\, f \rangle] \wedge [\langle f\, h \rangle, \langle h\, h \rangle]$.

Note that the replacement of modified equivalence classes by ordinary equivalence classes is always possible, and thus the result of (9d) is equivalent to a formula in FML*.

Finally, (9e) says that, given a consistent default formula, the order in which default information is added to the non-default formula is unimportant.[1] (This does not hold for inconsistent default formulae, however, since default unification with the individual conjuncts might filter out enough information to make the resultant formula a consistent extension of the non-default formula, whereas $TOP \oplus \phi = \phi$).

The monotonicity properties of default unification are listed below (where $\leq$ is subsumption):

(14) a. $\phi \leq \chi \wedge \phi$

(but not $\chi \leq \chi \wedge \phi$)

b. $\chi \leq \chi' \Rightarrow (\chi \wedge \phi) \leq (\chi' \wedge \phi)$

(but not $\phi \leq \phi' \Rightarrow (\chi \wedge \phi) \leq (\chi \wedge \phi')$)

(14a) says that default unification is montonic addition of information to the non-default information. (14b) says that the function as a whole is monotonic only w.r.t. the default argument: adding more default information leads to extensions of the result. Adding non-default information is non-monotonic, however, as this might cause more of the default information to get removed or overwritten.

The laws in (9) prove that formulae containing the $\oplus$-operator can always be reduced to standard formulae of FML*. This implies that formulae using the $\oplus$-operator can still be interpreted as denoting dags. Furthermore, it follows that addition of default unification to a unification-based formalism should be seen only as a way to increase the expressive power of tools used in defining the grammar (and thus, according to Dörre et al. (1990) default unification would be an 'off line' extension of the formalism, that is, its effects can be computed at compile time).

A Note on Implementation. We have implemented default unification in Prolog. Feature structures are represented by open ended lists (containing elements of the form *label=Value*), atoms and variables to represent complex feature structures, atomic values and reentrancies respectively (see Gazdar & Mellish, 1989). This implementation has the advantage that it is corresponds to FML* NF.

1 This should not be confused with the (invalid) statement that $\psi \oplus (\chi \oplus \phi) = \chi \oplus (\psi \oplus \phi)$.

```
(15)  a.   [f=X, g=X|_Y]

      b.   [f=a,g=a|_Y]

      c.   [f=[h=a|X1],g=[h=a|X1]|_Y]

      d.   [f=[h=a|X1,g=[h=_Z|X1]|_Y]
```

If we unify (15a) with *[f=a|_Y1]*, we get (15b), in which the value of g has been updated as well Thus, the requirements of (4a,b) are always met, and furthermore, the reentrancy as such between *f* and *g* is no longer visible (condition 4c). If we unify (15a) with *[f=[h=a|_X2]|_Y3]*, we get (15c), in which the variable *X* has been replaced by *X1*, which can be interpreted as ranging over all paths that are realized but not defined under *f* (condition (4d)). Note also that this representation has the advantage that we can define a reentrancy for all realized features, without having to specify the set of possible features or expanding the value of *f* into a list containing all these features. If we default unify (15a) with *[f=[h=a|_X2]|_X3]* as non-default information, for instance, the result is representable as (15d). The reentrancy for all undefined features under f is represented by *X1*. The constant NIL of FML* is represented as a Prolog variable (*_Z* in this case). Thus, the seemingly space consuming procedure of bringing a formula into FML* NF and transforming the output of (9d) into FML* is avoided completely. The actual default unification procedure is a modified version of the merge operation defined in Dörre & Eisele (1986).

3. LINGUISTIC APPLICATIONS

Default unification can be used to extend the standard PATR-II (Shieber et al., 1983) methods for defining feature structures. In the examples, we freely combine default and non-default information (prefixed by '!') in template definitions.

```
(16)  a. DET : (  !<cat arg>        = N
                  !<cat val>        = NP
                  <cat dir>         = right
                  <cat arg>         = <cat val>
                  <cat val num>     = sg
                  <cat val case>    = nom ).

      b. NP : (   <cat>    = noun
                  <bar>    = 2        ).

      c. N  : (   <cat>    = noun
                  <bar>    = 1        ).
```

(16) describes a fragment of Categorial Unification Grammar (Uszkoreit, 1986, Calder et al. 1988, Bouma, 1988). The corresponding feature structure for a definition such as (16a) is determined as follows: first, all default information and all non-default information is unified separately, which results in two feature-structures (17a,b). The resulting two feature structures are merged by means of default unification (17c).

(17)

a. $$\begin{bmatrix} \text{val} = \langle 1 \rangle \begin{bmatrix} \text{num} = \text{sg} \\ \text{case} = \text{nom} \end{bmatrix} \\ \text{dir} = \text{right} \\ \text{arg} = \langle 1 \rangle \end{bmatrix}$$

b. $$\begin{bmatrix} \text{cat} = \begin{bmatrix} \text{val} = \begin{bmatrix} \text{cat} & = & \text{n} \\ \text{bar} & = & 2 \end{bmatrix} \\ \text{arg} = \begin{bmatrix} \text{cat} & = & \text{n} \\ \text{bar} & = & 1 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

c. $$\begin{bmatrix} \text{cat} = \begin{bmatrix} \text{val} = \{1\} \begin{bmatrix} \boldsymbol{\mathit{cat}} = \text{n} \\ \boldsymbol{\mathit{bar}} = 2 \\ \text{num} = \text{sg} \\ \text{case} = \text{nom} \end{bmatrix} \\ \text{dir} = \text{right} \\ \text{arg} = \{1\} \begin{bmatrix} \boldsymbol{\mathit{cat}} = \text{n} \\ \boldsymbol{\mathit{bar}} = 1 \\ \text{num} = \text{sg} \\ \text{case} = \text{nom} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

In (17c) the equivalence *<cat val>* = *<cat arg>* had to be replaced by a weaker set of equivalences, which holds for all features under val or arg, except cat and bar. We represent this by using {}-bracketed indices, instead of <> and by marking the attributes which are exceptions in ***bold italic***. .

Two things are worth noticing. First of all, the unification of non-default information prior to merging it with the non-default information, guarantees that all default information must be unifiable, and thus it eliminates the possibility of inheritance conflicts inside template definitions. Second, the distinction between default and non-default information is relevant only in definitions, not in the corresponding feature structures. This makes the use of the '!'-operator completely local: if a definition contains a template, we can replace this template by the corresponding feature structure and we do not need to worry about the fact that this template might contain the '!'-operator.

The notation just introduced increases the expressive power of standard methods for the description of feature structures and can be used for an elegant treatment of several linguistic phenomena.

Non-monotonic inheritance of information in templates. The use of default unification enables us to use templates even in those cases where not all the information in the template is compatible with the information already present in the definition.

German transitive verbs normally take an accusative NP as argument, but there are some verbs which take a dative or genitive NP as argument. This is easily accounted for by defining the case of the argument of these verbs and inheriting all other information from the template *TV*.

```
(18)  a.  TV: ( <cat val> = VP
                <cat arg> = NP
                <cat arg case> = acc ).

      b.  helfen  (to help)   :
            (  TV
               ! <cat arg case> = dat ).

          gedenken  (to commemorate)    :
            (  TV
               ! <cat arg case> = gen ).
```

Specialization of reentrancies. An important function of default unification is that it allows us to define exceptions to the fact that two reentrant feature structures always have to denote exactly the same feature structures. There is a wide class of linguistic constructions which seems to require such mechanisms.

Specifiers in CUG can be defined as functors which take a constituent of category *C* as argument, and return a constituent of category *C*, with the exception that one or more specific feature values are changed (see Bach, 1983, Bouma, 1988). Examples of such categories are determiners (see (16a)), complementizers and auxiliaries.

```
(19)  a. that : ( <cat val> = <cat arg>
                  <cat arg> = S
                  <cat arg vform> = fin
                  !<cat arg comp> = none
                  !<cat val comp> = that   ).

      b. will : ( <cat val> = <cat arg>
                  <cat arg> = VP
                  <cat val> = VP
                  !<cat arg vform> = bse
                  !<cat val vform> = fin   ).
```

Note that the equation *<cat val> = <cat arg>* will cause all additional features on the argument which are not explicitly mentioned in the non-default part of the definition to percolate up to the value.

Next, consider coordination of NPs.

(20) X_0 --> X_1 X_2 X_3

```
      (  <X2 cat> = conj
         <X0> = <X1>
         <X0> = <X3>
         <X0 cat> = np
         <X2 wform> = and
         !<X0 num> = plu
         !<X1 num> =NIL
         ! <X2 num> = NIL ).
```

(20) could be used as a rule for conjunction of NPs in UG. It requires identity between the mother and the two coordinated elements. However, requiring that the three nodes be unifiable would be to strict. The number of a conjoined NP is always plural and does not depend on the number of the coordinated NPs. Furthermore, the number of two coordinated elements need not be identical. The non-default information in (20) takes care of this. The effect of this statement is that adding the default information $<X_0> = <X_1>$ and $<X_0> = <X_3>$ will result in a feature structure in which X_0, X_1 and X_3 are unified, except for their values for *<num>*. We are not interested in the *num*-values of the conjuncts, so they are set to *NIL* (which should be interpreted as in section 2) . The *num* -value of the result is always *plu.*

Inflectional Morphology. When seen from a CUG perspective, the categories of inflectional affixes are comparable to those of specifiers. The plural suffix *-s* for forming plural nouns can, for instance, be encoded as a function from (regular) singular nouns into identical, but plural, nouns. Thus, we get the following categorization:

```
(21)   -s : (     <cat val> = <cat arg>
                  <cat arg cat> = noun
                  <cat arg class> = regular
                  !<cat arg num> = sg
                  !<cat val num> = plu ).
```

Again, all additional information present on the argument which is not mentioned in the non-default part of the definition, is percolated up to the value automatically.

Lexical Defaults. The lexical feature specification defaults of GPSG can also be incorporated. Certain information holds for most lexical items of a certain category, but not for phrases of this category. A unificiation-based grammar that includes a morphological component (see, for instance, Calder, 1989 and Evans & Gazdar, 1989), would probably list only (regular) root forms as lexical items. For regular nouns, for instance, only the singular form would be listed in the lexicon. Such information can be added to lexicon definitions by means of a lexical default rule:

```
(22)  a.  N ==> ( 3SG <class> = regular)

      b.  cow   =    N.
          sheep = (  N
                     <num>     =NIL
                     <class>   = irregular).
```

The interpretation of A ==> B is as follows: If the definition D of a lexical item is unifiable with A, than extend D to B $\oplus$ D. Thus, the lexical entry *cow* would be extended with all the information in the default rule above, whereas the lexical entry for *sheep* would only be extended with the information that *<person>* = *3*. Note that adding the default information to the template for N directly, and then overwriting it in the irregular cases is not a feasible alternative, as this would force us to distinguish between the template N if used to describe nouns and the template N if used in complex categories such as NP/N or N/N (i.e. for determiners or adjectives it is not typically the case that they combine only with regular and singular nouns).

4. CONCLUSIONS

We have presented a general definition for default unification. The fact that it does not focus one the resolution of feature conflicts alone, makes it possible to define default unification as an operation on feature structures, rather than as an operation adding one equation at a time to a given feature description. This generalization makes it possible to give a uniform treatment of such things as adding default information to a template, overwriting of feature values and lexical default rules. We believe that the examples in section 3 demonstrate that this is a useful extension of UG, as it supports the definition of exceptions, the formulation more adequate theories of feature percolation, and the extension of UG with a morphological component.

REFERENCES

Bach, Emmon 1983 Generalized Categorial Grammars and the English Auxiliary. In F.Heny and B.Richards (eds.) *Linguistic Categories*, Vol II, Dordrecht, Reidel.

Bouma, Gosse 1988 Modifiers and Specifiers in Categorial Unification Grammar, *Linguistics*, vol 26, 21-46.

Calder, Jonathan 1989 Paradigmatic Morphology. *Proceedings of the fourth Conference of the European Chapter of the ACL*, University of Manchester, Institute of Science and Technology, 58-65.

Calder, Jo; Klein, Ewan & Zeevat, Henk 1988 Unification Categorial Grammar: a concise, extendable grammar for natural language processing. *Proceedings of Coling 1988*, Hungarian Academy of Sciences, Budapest, 83-86.

Dörre, Jochen; Eisele, Andreas; Wedekind, Jürgen; Calder, Jo; Reape, Mike 1990 *A Survey of Linguistically Motivated extensions to Unification-Based Formalisms.* ESPRIT Basic Research Action 3175, Deliverable R3.1.A.

Eisele, Andreas & Dörre, Jochen1986 A Lexical-Functional Grammar System in Prolog. *Proceedings of COLING 86*, Institut für angewandte Kommunikations- und Sprachforschung, Bonn, 551-553.

Eisele, Andreas & Dörre, Jochen 1988 Unification of Disjunctive Feature Descriptions. *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, State University of New York, Buffalo, NY, 286-294.

Evans, Roger 1987 Towards a Formal specification of Defaults in GPSG. In E. Klein & J. van Benthem (eds.), *Categories, Polymorphism and Unification*. University of Edinburgh, Edinburgh/ University of Amsterdam, Amsterdam, 73-93.

Evans, Roger & Gazdar, Gerald 1989 Inference in DATR. *Proceedings of the fourth Conference of the European Chpater of the ACL*, University of Manchester, Institute of Science and Technology, 66-71.

Flickinger, Daniel; Pollard, Carl & Wasow, Thomas 1985 Structure-Sharing in Lexical Representation. *Proceedings of the 23rd Annual Meeting of the Association for*

Computational Linguistics, University of Chicago, Chicago, Illinois, 262-267.

Gazdar, Gerald 1987 Linguistic Applications of Default Inheritance Mechanisms. In P. Whitelock, H. Somers, P. Bennett, R. Johnson, and M. McGee Wood (eds.), *Linguistic Theory and Computer Applications*. Academic Press, London, 37-68.

Gazdar, Gerald; Klein, Ewan; Pullum, Geoffry; Sag, Ivan 1985 *Generalized Phrase Structure Grammar*. Blackwell, London.

Gazdar, Gerald & Mellish, Chris 1989 *Natural Language Processing in Prolog. An introduction to Computational Linguistics*. Addison-Wesley, Reading, MA.

Kaplan, Ronald 1987 Three seductions of Computational Psycholinguistics. In P. Whitelock, H. Somers, P. Bennett, R. Johnson, and M. McGee Wood (eds.), *Linguistic theory and Computer Applications*. Academic Press, London, 149-188.

Kasper, Robert & Rounds, William1986 A Logical Semantics for Feature Structures. *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, Columbia University, New York, NY, 257-266.

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

Shieber, Stuart; Uszkoreit, Hans; Pereira, Fernando; Robinson, Jane; & Tyson, Mabry 1983 The Formalism and Implementation of PATR-II. In B. Grosz & M. Stickel (eds.) *Research on Interactive Acquisition and Use of Knowledge*, SRI International, Menlo Park, Ca.

Shieber, Stuart 1986a A Simple Reconstruction of GPSG. *Proceedings of COLING 1986*. Institut für angewandte Kommunikations- und Sprachforschung, Bonn, 211-215.

Shieber, Stuart 1986b *An Introduction to Unification-based Approaches to Grammar*. CSLI Lecture Notes 4, University of Chicago Press, Chicago.

Uszkoreit, Hans 1986 Categorial Unification Grammars. *Proceedings of COLING 1986*. Institut für angewandte Kommunikations- und Sprachforschung, Bonn, 187-194.