

Feature Logic with Disjunctive Unification

Jochen Dörre, Andreas Eisele

Institut für maschinelle Sprachverarbeitung
Universität Stuttgart

Abstract

We introduce feature terms containing sorts, variables, negation and *named disjunction* for the specification of feature structures. We show that the possibility to label disjunctions with names has major advantages both for the *use* of feature logic in computational linguistics and its *implementation*. We give an open world semantics for feature terms, where the denotation of a term is determined in dependence on the *disjunctive context*, i.e. the choices taken for the disjunctions. We define *context-unique feature descriptions*, a relational, constraint-based representation language and give a normalization procedure that allows to test consistency of feature terms. This procedure does not only avoid expansion to disjunctive normal form but maintains also structure sharing between information contained in different disjuncts as much as possible. Context-unique feature descriptions can be easily implemented in environments that support ordinary unification (such as PROLOG).

1 Introduction

1.1 Ambiguity in Natural Language

Our use of language mirrors our intellectual capacities, which are as yet my no means understood. As long as we can not formally describe the processes involved in thinking and understanding, formal descriptions of human language have to be rough approximations. One particular instance of this general fact is the problem of disambiguation of human utterances. Since our use of words fits our capabilities of understanding their meaning, context and intent, systems that do not have such capabilities can, at best, produce sets of possible analyses. It is well known that such sets can be very large in practice.

Ambiguity in natural language is fed by a couple of sources, including lexical ambiguity, where differing analyses are possible for a given word concerning its part of speech, subcategorization for complements, morphological features, or any other information assigned to it, and structural ambiguity introduced by different possible groupings or interpretations of phrases or different interrelations between them with respect to subcategorization, meaning, pragmatics etc. On each level, a bunch of possibilities exist, which could potentially multiply to an enormous space of combinations. However, these possibilities interact and restrict each other in such a way, that — taking it all together — only a few (hopefully exactly one) interpretations remain.

1.2 Unification-Based Formalisms

For about a decade, many formal theories of natural language have tried to describe their subject in terms of so called feature structures, i.e. potentially nested bundles of features that are assigned to words and phrases. These structures are sometimes seen as abstract linguistic objects, which are described using a suitable description language, sometimes they are given a concrete shape in form of finite automata and regarded themselves as descriptions of the linguistic objects [Kasper/Rounds 86]. Despite such differences in interpretation, there is a consensus among the the-

ories that linguistic descriptions should provide constraints concerning feature structures and that a set of such constraints gives a partial description of the feature structures associated with a phrase. A set of constraints defines a minimal model, i.e. a minimal structure satisfying all constraints in the set. The union of two sets of constraints not contradicting each other leads to a minimal model which is the least common extension of the models of both sets. Such minimal common extensions can be constructed by unification of the given models, hence the term unification-based formalisms.

There is also a consensus among feature-based theories that ambiguity should be described with disjunctive formulas, and most formalisms offer ways to specify them. If disjunction is present, there is usually a finite number of minimal models instead of only one. However, until now, the way such disjunctive specifications have been processed computationally was not quite satisfactory. An enumeration of the possibilities using a backtracking scheme or a chart, which corresponds to an expansion to disjunctive normal form in the underlying logic, often leads to computational inefficiency.

Approaches to improve the situation both in terms of the logic and the implementation (see e.g. [Karttunen 84, Kasper 87, Eisele/Dörre 88, Maxwell/Kaplan 89]) can be subdivided in those assuming disjunctive values for features and those allowing for more general forms of disjunction. Roughly, we can state that formalisms and implementations that provide value disjunction can be implemented more easily and more efficiently, since they can exploit the fact that disjunctive information for a certain feature has no effect on other features (as long as disjunctive information does not interact with path equivalences, see [Eisele/Dörre 88]). But the restriction to value disjunction decreases the expressive power of the formalism, since disjunctions concerning different features must be stated on a higher level. Schemes providing for general disjunction allow for a more compact representation of such cases. But if disjunctive information is not local to single features, the interaction between different parts of the description is more difficult to handle (see e.g. [Kasper 87]).

The method we propose combines advantages of both approaches. It can be seen as a generalization of value disjunction, which allows for a concise description of disjunction concerning more than one feature or path. It can also be seen as an efficient implementation of general disjunction which allows to exploit the locality of disjunctive information whenever this is possible.

2 Feature Terms

2.1 Disjunction Names

The background of our approach is the simple observation that general disjunction affecting more than one feature can be reduced to value disjunction for those features, provided that the correspondence between such disjunctions can be expressed within the formalism. In order to state such correspondences, we will label disjunctions with a *disjunction name*. Take, for instance, the formula (1) that could be used

$s, t \longrightarrow A$	a sort	$\llbracket A \rrbracket_{\alpha, \kappa} := A^{\mathcal{I}}$
$ \quad x$	a variable	$\llbracket x \rrbracket_{\alpha, \kappa} := \{\alpha(x)\}$
$ \quad \neg A \quad \quad \neg x$	simple complements	$\llbracket \neg s \rrbracket_{\alpha, \kappa} := \mathcal{U} - \llbracket s \rrbracket_{\alpha, \kappa}$
$ \quad f:s$	selection	$\llbracket f:s \rrbracket_{\alpha, \kappa} := \{a \in \mathcal{U} \mid f^{\mathcal{I}}(a) \in \llbracket s \rrbracket_{\alpha, \kappa}\}$
$ \quad s \sqcap t$	conjunction (intersection)	$\llbracket s \sqcap t \rrbracket_{\alpha, \kappa} := \llbracket s \rrbracket_{\alpha, \kappa} \cap \llbracket t \rrbracket_{\alpha, \kappa}$
$ \quad s \sqcup_d t$	named disjunction (union)	$\llbracket s \sqcup_d t \rrbracket_{\alpha, \kappa} := \begin{cases} \llbracket s \rrbracket_{\alpha, \kappa} & \text{if } \kappa(d) = l \\ \llbracket t \rrbracket_{\alpha, \kappa} & \text{if } \kappa(d) = r \end{cases}$

Figure 1: Syntax and Semantics of Feature Terms

to express that the directional reading of the german preposition “in” (=into) corresponds to the accusative case of the following noun phrase, whereas the static reading (=in) corresponds to the dative case. This can also be expressed by (2), where the index d_1 at the disjunction sign indicates the mutual dependence of both disjunctions. Throughout this paper, we will assume that each disjunction is labelled with a name. Even in cases where a disjunction appears only once in the initial description, naming it will help us to treat the interaction between disjunction and path equivalence correctly.

- (1) $(\text{syn} : \text{arg} : \text{case} : \text{dat} \wedge \text{sem} : \text{rel} : \text{stat_in})$
 $\vee (\text{syn} : \text{arg} : \text{case} : \text{acc} \wedge \text{sem} : \text{rel} : \text{dir_in})$
- (2) $\text{syn} : \text{arg} : \text{case} : (\text{dat} \vee_{d_1} \text{acc})$
 $\wedge \text{sem} : \text{rel} : (\text{stat_in} \vee_{d_1} \text{dir_in})$

2.2 Syntax and Semantics of Feature Terms

We incorporate named disjunction into a language of so-called *feature terms* (similar to those in [Smolka 88]), where each feature term describes a set of possible feature structures. The language allows for the use of *sort symbols* $A, B, C \dots \in \mathbf{S}$, on which some partial order \preceq induces a lower semilattice (i.e. $\forall A, B \in \mathbf{S} : \text{GLB}(A, B) \in \mathbf{S}$). \top and \perp are the greatest and least element of \mathbf{S} . We also distinguish a set of *singleton sorts* $(a, b, c \dots \in \mathbf{Sg} \subset \mathbf{S})$, which include the special sort NONE. \perp is the only sort smaller than a singleton sort. The language provides a set \mathbf{F} of feature symbols (written f, g, h, \dots), an infinite set \mathbf{V} of variables (written x, y, z, x_1, y_1, \dots) to express path equivalence, and an infinite set \mathbf{D} of disjunction names (written d, d_1, d_2, \dots). $\mathbf{S}, \mathbf{F}, \mathbf{V}$ and \mathbf{D} are pairwise disjoint. Sort symbols and variables can be negated to express negative values and path equivalence (simple negation). The restriction of negation to sort symbols and variables is not essential, since the negation of any feature term can always be reduced to these forms in linear time [Smolka 88].

Definition 1 (Feature Terms) *We define the set \mathbf{FT} of feature terms with variables, simple negation and named disjunction by the context-free production rules given in Fig. 1. Letters s, t, t_1, \dots will always denote feature terms.*

The semantics of our terms is defined with respect to an interpretation, which is a pair $(\mathcal{U}, \cdot^{\mathcal{I}})$ of a universe of the interpretation and an interpretation function such that:

- $\top^{\mathcal{I}} = \mathcal{U}$ and $\perp^{\mathcal{I}} = \emptyset$
- for all sorts A, B : $\text{GLB}(A, B)^{\mathcal{I}} = A^{\mathcal{I}} \cap B^{\mathcal{I}}$
- singleton sorts are mapped onto singleton sets
- for every feature f : $f^{\mathcal{I}}$ is a function $\mathcal{U} \rightarrow \mathcal{U}$.
- if a is a singleton sort and f is a feature symbol, then $f^{\mathcal{I}}$ maps $a^{\mathcal{I}}$ into $\text{NONE}^{\mathcal{I}}$

When interpreting a feature term with variables and named disjunctions, we have to make sure that the same value is assigned to each occurrence of a variable and that the same branch is chosen for each occurrence of a named disjunction.

To achieve this, we introduce *variable assignments* that map variables to elements of the universe and *disjunctive contexts* that assign to each disjunction name the branch that has to be taken for this disjunction and hence specify a possible interpretation of a formula with named disjunction. Since we limit ourselves to binary disjunctions, a branch of a disjunction can be specified by one of the symbols l or r .

Definition 2 (\mathcal{U} -Assignment) *A \mathcal{U} -assignment α is an element of $\mathcal{U}^{\mathbf{V}}$, i.e. a function from \mathbf{V} to \mathcal{U} .*

Definition 3 (Context) *A context is an element of $\{l, r\}^{\mathbf{D}}$, i.e. a function from \mathbf{D} to the set $\{l, r\}$. The symbols κ, κ' , etc. will always denote contexts.*

For a given interpretation, we define the denotation of a feature term in a context $\kappa \in \{l, r\}^{\mathbf{D}}$ under an assignment $\alpha \in \mathcal{U}^{\mathbf{V}}$ as shown in Fig. 1. The denotation of a feature term as such is defined by:

$$\llbracket s \rrbracket := \bigcup_{\kappa \in \{l, r\}^{\mathbf{D}}} \bigcup_{\alpha \in \mathcal{U}^{\mathbf{V}}} \llbracket s \rrbracket_{\alpha, \kappa}$$

3 Context-Unique Feature Descriptions

To describe the computational mechanisms needed for an implementation, we will introduce a relational language to express constraints over variables. Unlike similar approaches (e.g. [Smolka 88]), our constraint language will also be used to express disjunctive information. For this language, we will define a normal form that exhibits inconsistencies, and simplification rules that allow to normalize a given specification. Our language will provide only two kinds of constraints, one that relates a variable to some feature term (written $x | t$) and one that expresses that certain contexts are excluded from consideration because the information known for them is inconsistent (written $\perp[k]$).

In order to refer to sets of contexts, we define

Definition 4 (Context Descriptions)

A context description is a propositional formula where the constant TRUE, variables written $d_i:l$ and $d_i:r$ with $d_i \in \mathbf{D}$, and the operators \wedge, \vee and \neg may be employed.

CD will denote the set of context descriptions. The symbols k, k_1, \dots will always denote members of **CD**.

The set of purely conjunctive context descriptions (not containing the operators \vee and \neg) is denoted by \mathbf{CD}_c .

Each context κ satisfies the context description TRUE (written $\kappa \models_c \text{TRUE}$), whereas $\kappa \models_c d : b$ for $b \in \{l, r\}$ only if $\kappa(d) = b$. The meaning of context descriptions involving \wedge, \vee and \neg is defined as in propositional logic.

If $\kappa \models_c k$, we will also say that k describes or covers κ or that κ lies in k .

A context description is called contradictory, if no context satisfies it.

Two context descriptions k, k' which are satisfied by exactly the same contexts are called equivalent (written $k \equiv k'$).

An important form of constraints for our approach are constraints like $x|x_1 \sqcup_{d_1} x_2$ which expresses that x and x_1 have to be equal in contexts where $\kappa(d_1) = l$ and so do x and x_2 in contexts where $\kappa(d_1) = r$. Such constraints are called *bifurcations* and x_1, x_2 are called (the $d_1:l$ - and $d_1:r$ -) *variants* of x . Assume an additional constraint $x_1|x_3 \sqcup_{d_2} x_4$, then x_3 will be called the $d_1:l \wedge d_2:l$ -variant of x and so on. Now, instead of accumulating constraints on the variable x which might be effective in different contexts and could interact in complicated ways, we can introduce new variables as variants of x and attach the information to them.

We will sometimes refer to a variant of a variable x without having a variable name for this variant. To this end, we will use a special notation x/k to denote the k -variant of x . Such expressions will be called *contexted variables*.

Definition 5 (Contexted Variables) A contexted variable is a pair x/k where $x \in \mathbf{V}$ and $k \in \mathbf{CD}_c$.

\mathbf{V}_c will denote the union of \mathbf{V} with the set of contexted variables. Elements of \mathbf{V}_c will be written with capital letters $X, Y, Z, X_1, Y_1 \dots$. To mark the distinction, we will sometimes call the members of \mathbf{V} *pure variables*.

During the normalization of feature descriptions we will sometimes need variable substitution. If a description contains e.g. $x|y$, where other constraints might express conflicting information about x and y , we want to concentrate this information on one variable (say x) by substituting all occurrences of y in other constraints by x . This could lead to problems when constraints attached to x and y are relevant in different contexts. One way to treat this situation correctly would be the introduction of conditional substitution (see [Eisele/Dörre 90] for details). The way we choose here is to restrict the use of variables in such a way that it is always safe to use conventional substitution.

Our trick will be to require that essentially all occurrences of a variable x are relevant to the same set of contexts. We call this condition (defined more precisely below) the *context-uniqueness of variables*. We will set up the normal form and the rewrite system in such a way, that context-uniqueness of a description is maintained during the simplification process. (See [Eisele/Dörre 90] for a more detailed motivation of context-uniqueness). The set of relevant contexts will be regarded as an inherent and invariant property of variables, and we will introduce a *context assignment*, i.e. a partial function $Con : \mathbf{V} \mapsto \mathbf{CD}_c$ that maps each variable in use to a purely conjunctive description of the contexts it is relevant to. We extend Con to contexted variables by defining $Con(x/k) := Con(x) \wedge k$.

In order to obtain context-unique descriptions, we generalize our feature terms so that they may also contain contexted variables.

Definition 6 (Contexted Feature Terms) A contexted feature term is built according to definition 1, but where both pure and contexted variables may occur. The set of contexted feature terms will be denoted by \mathbf{FT}_c . The symbols $s, t, t_1 \dots$ may henceforth also denote contexted feature terms.

The denotation of a contexted feature term in a context $\kappa \in \{l, r\}^{\mathbf{D}}$ under an assignment $\alpha \in \mathcal{U}^{\mathbf{V}}$ is defined as for usual feature terms by adding:

$$\llbracket x/k \rrbracket_{\alpha, \kappa} := \begin{cases} \{\alpha(x)\} & \text{if } \kappa \models_c k \\ \emptyset & \text{otherwise} \end{cases}$$

We can now define the context compatibility of a feature term. This definition is somewhat technical and the reader can skip it, since our algorithm will produce only context-unique descriptions, anyway.

Definition 7 (Context compatibility) Given a partial assignment $Con : \mathbf{V} \mapsto \mathbf{CD}_c$, a contexted feature term t is context-compatible to a context description k with respect to Con , written $t \sim_{Con} k$, according to the following conditions.

$$\begin{aligned} A &\sim_{Con} k && \text{for arbitrary } k \in \mathbf{CD}_c \\ X &\sim_{Con} k && \text{iff } Con(X) \equiv k \\ \neg t &\sim_{Con} k && \text{iff } t \sim_{Con} k \\ f:t &\sim_{Con} k && \text{iff } t \sim_{Con} k \\ s \sqcap t &\sim_{Con} k && \text{iff } s \sim_{Con} k \text{ and } t \sim_{Con} k \\ s \sqcup_d t &\sim_{Con} k && \text{iff } s \sim_{Con} k \wedge d:l \\ &&& \text{and } t \sim_{Con} k \wedge d:r \end{aligned}$$

Definition 8 (Context-unique feature descriptions)

A context-unique feature description (x_0, CUC, Con) is a triple such that:

- $x_0 \in \mathbf{V}$, called the root variable
- CUC is a set of context-unique constraints which either have the form $\perp[k]$, where $k \in \mathbf{CD}$ or $X|t$, where $X \in \mathbf{V}_c, t \in \mathbf{FT}_c$ and $t \sim_{Con} Con(X)$
- Con is a context assignment which is defined for all variables in CUC

The semantics of context-unique feature descriptions is given by the satisfaction relation \models_{Con} between variable assignments¹, contexts and constraints, which is parametrized with a context assignment.

$$\begin{aligned} \alpha, \kappa \models_{Con} X|t &\text{ iff } \kappa \not\models_c Con(X) \text{ or } \alpha(X) \in \llbracket t \rrbracket_{\alpha, \kappa} \\ \alpha, \kappa \models_{Con} \perp[k] &\text{ iff } \kappa \not\models_c k \end{aligned}$$

The denotation of a context-unique f -description is:

$$\llbracket (x_0, CUC, Con) \rrbracket := \{\alpha(x_0) \mid \alpha \in \mathcal{U}^{\mathbf{V}}, \kappa \in \{l, r\}^{\mathbf{D}} \text{ s.t. } \forall c \in CUC : \alpha, \kappa \models_{Con} c\}$$

Given a feature term t not containing the variable x_0 , we can find an equivalent context-unique feature description $(x_0, \{x_0|t'\}, Con)$ as follows. We initialize the context assignment Con so that x_0 and all variables contained in t are mapped to TRUE (they are regarded as relevant to all contexts). Then we obtain the contexted feature term t' by replacing all occurrences of variables in t which are embedded in disjunctions by their appropriate variants, such that $t' \sim_{Con} \text{TRUE}$ ².

Proposition: If t does not contain the variable x_0 , and if Con and t' are obtained from t as described above, then $\llbracket t \rrbracket = \llbracket (x_0, \{x_0|t'\}, Con) \rrbracket$. For a proof see [Eisele/Dörre 90].

4 Normal Feature Descriptions

One way to eliminate a contexted variable (take e.g. $x/d_1:l$) from a description is to introduce a bifurcation $(x|x_1 \sqcup_{d_1} x_2)$ and replace the variable by an appropriate variant (in this case x_1). Analogously, contexted variables with more complex context descriptions can be replaced by introducing several bifurcations. However, it turns out that our representation can be more compact if we allow for the use of contexted variables. But we have to prevent conflicting information from being attached to variants of a variable. Our normal form will therefore allow the use of contexted variables in certain places, but in some cases, a pure variable has to be used.

¹ α is extended to contexted variables by: $\alpha(x/k) := \alpha(x)$

²In the sequel we will also assume that inaccessible disjuncts resulting from nested disjunctions with identical names (e.g. t_2 in $t_1 \sqcup_d (t_2 \sqcup_d t_3)$) are removed.

A context-unique feature description (x_0, CUC, Con) is *normal* if it satisfies the following conditions:

A) All constraints in CUC have one of the forms:

- $\perp[k]$
- $x|x_1 \sqcup_d x_2$
- $x|\neg y$, where $x \neq y$
- $x|A$ or $x|\neg A$
- $x|f:Y$

where $k \in \mathbf{CD}$, $x_1, x_2, x, y \in \mathbf{V}$, $Y \in \mathbf{V}_c$, $d \in \mathbf{D}$ and $A \in \mathbf{S} \setminus \{\top, \perp\}$

B) The following restrictions apply:

1. If $\perp[k]$ and $x|t$ are in CUC , then $Con(x) \wedge \neg k$ is not contradictory
2. if $x|A$ and $x|B$ are in CUC , then $A = B$
3. if $x|a$ and $x|t$ are in CUC , then $t = a$
4. if $x|A$ and $x|\neg B$ are in CUC , then $A \not\leq B$ and $GLB(A, B) \neq \perp$
5. if $x|\neg A$ and $x|\neg B$ are in CUC , then $A \not\leq B$
6. if $x|f:Y$ and $x|f:Z$ are in CUC , then $Y = Z$
7. if $\perp[k]$ and $\perp[k']$ are in CUC , then $k = k'$
8. if $x|x_1 \sqcup_d x_2$ and $x|t$ are in CUC , then $t = x_1 \sqcup_d x_2$

4.1 Simplification Rules for Normalization

For normalization, we have to consider all ways a context-unique feature description could fail to be normal, and we have to find an equivalent description that is in (or closer to) normal form. To this end, we give simplification rules for each possible case. Since there are many different ways to violate normal form, we get a lot of different rules, but each of them is very simple and their correctness should be easy to see. The rules are parametrized with the root variable (which should not be substituted away) and with the context assignment, which will be extended to new variables during simplification. To facilitate notation, we use $c \ \& \ CUC$ to denote $\{c\} \cup CUC$ where CUC is supposed not to contain the constraint c , and $CUC_{x \rightarrow y}$ denotes CUC with all occurrences of x replaced by y . Also, if we write $d:b \wedge k'$, then k' is supposed not to contain $d:b$. The cases we have to handle are grouped in those that treat single non-normal constraints (S) and those that treat interactions between different constraints (M).

There are S-Rules for all forms of constraints which conflict with condition A), i.e. which are of one of the forms

1. $x/k|t$
2. $x|\neg y/k$
3. $x|Y$
4. $x|t$ or $x|\neg t$, where t has the form \top, \perp or x
5. $x|f:t_1$, where $t_1 \notin \mathbf{V}_c$
6. $x|t_1 \sqcap t_2$
7. $x|t_1 \sqcup_d t_2$, where $\{t_1, t_2\} \not\subseteq \mathbf{V}$

Among the situations in which a contexted variable x/k conflicts with normal form, we have to distinguish several cases. If $k \equiv \text{TRUE}$, then the context description is irrelevant and we can replace x/k by x (Rule S_{cu1b}). Otherwise, if there exists already a bifurcation $x|x_1 \sqcup_d x_r$, such that $k \equiv d:b \wedge k'$ for some $b \in \{l, r\}$ and $k' \in \mathbf{CD}_c$, where k' does not contain $d:b$, then we can replace x/k by the shorter term x_b/k' (Rule S_{cu1c}). If there is a bifurcation $x|x_1 \sqcup_d x_r$ where d does not appear in k , the constraint attached to x/k is distributed over the variables x_1 and x_r (Rule S_{cu1d}). In order to maintain context-uniqueness, the variables appearing in the constraint have to be replaced by their respective $d:l$ - and $d:r$ -variants. We use t/k as a shorthand for a contexted feature term, where each variable has been replaced

by its k -variant, i.e. z has been replaced by z/k and z'/k' by $z'/(k' \wedge k)$ (see also rule (M_{cu8c}) , below). Only if no bifurcation exists for x we have to introduce a new bifurcation (Rule S_{cu1e}). We select a disjunction name d from k such that $k \equiv d:b \wedge k'$ for some $b \in \{l, r\}$ and $k' \in \mathbf{CD}_c$, where k' does not contain $d:b$, we add a bifurcation $x|x_1 \sqcup_d x_r$ to CUC , where x_l and x_r are new variables, and we extend Con by mapping x_l to $Con(x) \wedge d:l$ and x_r to $Con(x) \wedge d:r$. Now we can replace x/k by x_b/k' .

The other rules handle equalities by substituting a variable by some other variable, eliminate redundant constraints, handle inconsistencies, or decompose constraints with complex feature terms into a set of simple constraints.

The cases where a pair of constraints violates some of the conditions B1–7 can be treated as for similar non-disjunctive rewrite systems (see [Smolka 88] or [Eisele/Dörre 90]). Rules $M_{cu1}–7$ handle those. When a bifurcation $x|x_1 \sqcup_d x_2$ occurs together with some other constraint on x , this could lead to a contradiction with information known about x_1 and x_2 . Here, we distinguish three cases. If the other constraint happens to be a bifurcation $x|y_1 \sqcup_d y_2$ with the same disjunction name d , we get equalities between both $d:l$ -variants and both $d:r$ -variants (Rule M_{cu8a}). If the other constraint is a bifurcation $x|y_1 \sqcup_{d'} y_2$ with a different disjunction name, then the two disjunctions interact and have to be multiplied out for the variable x (Rule M_{cu8b}). To this end, four new variables are introduced as variants of x and new bifurcations are installed that link the new variables to those already in use. Con is extended for the new variables. In any other case, the constraint attached to x is distributed over both variants, and context descriptions for variables on the right-hand side of the constraint are introduced or adapted as required by context-uniqueness (Rule M_{cu8c}).

4.2 Soundness, Completeness and Termination

We can show that our simplification rules constitute an algorithm for the consistency (or unification) problem, which is sound and complete and guaranteed to terminate. For detailed proofs the reader is referred to [Eisele/Dörre 90]. Below, we give the key intuitions or strategies for the proofs. Soundness can be seen by inspecting the rules. Each rule rewrites a clause to one with an equivalent denotation. To show that the algorithm always finds an answer, we first observe that to every context-unique feature description that is produced during translation or normalization and that is not normal at least one of the rules applies. When the result of simplification is the single constraint $\perp[k]$ where $k \equiv \text{TRUE}$, this means that the description failed to unify. In any other case we can construct models from the normal form result. The basic idea is to choose a context κ which is not covered by the context description of a constraint $\perp[k]$ in our formula and ‘project’ the formula into this context by regarding only those constraints which are relevant to this context, thereby degenerating bifurcations to nondisjunctive bindings $x|y$. This nondisjunctive set of constraints can be made into a model.

In order to prove termination we construct a complexity measure for descriptions (a natural number) which is decreased in every rewrite step (see [Eisele/Dörre 90]). Here we take advantage of the fact that although there are rules which increase the number of constraints and hence seem to add to complexity, these rules also can be seen as part of an inherently irreversible process, since they distribute information attached to a variable over variables in more specific contexts. But since the number of disjunction names

(<i>S_{cu}1a</i>)	$x/k x/k' \ \& \ CUC$	$\rightarrow_{x_0, Con} \ CUC \ (k \equiv k' \text{ due to context-uniqueness})$
(<i>S_{cu}1b</i>)	$x/k t \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x t \ \& \ CUC, \text{ if } k \equiv \text{TRUE}$
(<i>S_{cu}1c</i>)	$x/k t \ \& \ x x_l \sqcup_d x_r \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x_b/k' t \ \& \ x x_l \sqcup_d x_r \ \& \ CUC, \text{ if } k \equiv d:b \wedge k'$
(<i>S_{cu}1d</i>)	$x/k t \ \& \ x x_l \sqcup_d x_r \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x_l/k t/d:l \ \& \ x_r/k t/d:r \ \& \ x x_l \sqcup_d x_r \ \& \ CUC,$ if (<i>S_{cu}1c</i>) does not match
(<i>S_{cu}1e</i>)	$x/k t \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x_b/k' t \ \& \ x x_l \sqcup_d x_r \ \& \ CUC$ if (<i>S_{cu}1a, b, c, d</i>) do not match, $k \equiv d:b \wedge k'$, x_l, x_r are new, and $Con(x_b) := Con(x) \wedge d:b$
(<i>S_{cu}2</i>)	$x \neg y/k \ \& \ CUC$	$\rightarrow_{x_0, Con} \ y/k \neg x \ \& \ CUC$
(<i>S_{cu}3a</i>)	$x y/k \ \& \ CUC$	$\rightarrow_{x_0, Con} \ y/k x \ \& \ CUC$
(<i>S_{cu}3b</i>)	$x y \ \& \ CUC$	$\rightarrow_{x_0, Con} \ CUC_{x \rightarrow y} \ , \text{ if } x \neq x_0$
(<i>S_{cu}3c</i>)	$x_0 y \ \& \ CUC$	$\rightarrow_{x_0, Con} \ CUC_{y \rightarrow x_0}$
(<i>S_{cu}4a</i>)	$x t \ \& \ CUC$	$\rightarrow_{x_0, Con} \ \perp[Con(x)] \ \& \ CUC \ , \text{ if } t = \perp, t = \neg\top \text{ or } t = \neg x$
(<i>S_{cu}4b</i>)	$x t \ \& \ CUC$	$\rightarrow_{x_0, Con} \ CUC \ , \text{ if } t = \top, t = \neg\perp \text{ or } t = x$
(<i>S_{cu}5</i>)	$x f:t \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x f:y \ \& \ y t \ \& \ CUC \ , \text{ if } t \notin \mathbf{V}_c$ where y is new and $Con(y) := Con(x)$
(<i>S_{cu}6</i>)	$x t_1 \sqcap t_2 \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x t_1 \ \& \ x t_2 \ \& \ CUC$
(<i>S_{cu}7</i>)	$x t_l \sqcup_d t_r \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x x_l \sqcup_d x_r \ \& \ x_l t_l \ \& \ x_r t_r \ \& \ CUC$ where $\{t_1, t_2\} \not\subseteq \mathbf{V}$, x_b are new and $Con(x_b) := Con(x) \wedge d:b$
(<i>M_{cu}1</i>)	$\perp[k] \ \& \ x t \ \& \ CUC$	$\rightarrow_{x_0, Con} \ \perp[k] \ \& \ CUC, \text{ if } Con(x) \wedge \neg k \text{ is contradictory}$
(<i>M_{cu}2</i>)	$x A \ \& \ x B \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x GLB(A, B) \ \& \ CUC$
(<i>M_{cu}3a</i>)	$x a \ \& \ x \neg y \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x a \ \& \ y \neg a \ \& \ CUC$
(<i>M_{cu}3b</i>)	$x a \ \& \ x f:Y \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x a \ \& \ Y NONE \ \& \ CUC$
(<i>M_{cu}4a</i>)	$x A \ \& \ x \neg B \ \& \ CUC$	$\rightarrow_{x_0, Con} \ \perp[Con(x)] \ \& \ CUC, \text{ if } A \leq B$
(<i>M_{cu}4b</i>)	$x A \ \& \ x \neg B \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x A \ \& \ CUC, \text{ if } GLB(A, B) = \perp$
(<i>M_{cu}5</i>)	$x \neg A \ \& \ x \neg B \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x \neg B \ \& \ CUC, \text{ if } A < B$
(<i>M_{cu}6</i>)	$x f:Y \ \& \ x f:Z \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x f:Y \ \& \ Z Y \ \& \ CUC$
(<i>M_{cu}7</i>)	$\perp[k] \ \& \ \perp[k'] \ \& \ CUC$	$\rightarrow_{x_0, Con} \ \perp[k \vee k'] \ \& \ CUC$
(<i>M_{cu}8a</i>)	$x x_1 \sqcup_d x_2 \ \& \ x y_1 \sqcup_d y_2 \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x x_1 \sqcup_d x_2 \ \& \ (CUC_{y_1 \rightarrow x_1})_{y_2 \rightarrow x_2}$
(<i>M_{cu}8b</i>)	$x x_1 \sqcup_{d_1} x_2 \ \& \ x y \sqcup_{d_2} z \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x x_1 \sqcup_{d_1} x_2 \ \& \ x_1 y_1 \sqcup_{d_2} z_1 \ \& \ x_2 y_2 \sqcup_{d_2} z_2 \ \& \ y y_1 \sqcup_{d_1} y_2 \ \& \ z z_1 \sqcup_{d_1} z_2 \ \& \ CUC,$ where $d_1 \neq d_2$ and y_1, y_2, z_1, z_2 are new
(<i>M_{cu}8c</i>)	$x x_1 \sqcup_d x_2 \ \& \ x t \ \& \ CUC$	$\rightarrow_{x_0, Con} \ x x_1 \sqcup_d x_2 \ \& \ x_1 t/d:l \ \& \ x_2 t/d:r \ \& \ CUC$ where t is not a bifurcation

Figure 2: Simplification Rules

is limited, the contexts associated to variables can not be arbitrarily specific and hence, this process must terminate.

4.3 An Example

Due to lack of space, our example can not demonstrate all capabilities of the formalism, but will concentrate on the treatment of disjunction and the support of structure sharing between different disjuncts. Assume as initial feature term $f : (x \sqcap g : t_G) \sqcap h : ((x \sqcup_d y) \sqcap i : t_I)$ where t_G and t_I might be themselves complex. Translation to context-unique form will produce the description $(x_0, \{x_0|f : (x \sqcap g : t'_G) \sqcap h : ((x/d:l \sqcup_d y/d:r) \sqcap i : t'_I)\}, Con_1)$ where t'_G and t'_I might contain contexted variables if necessary. Partial normalization then produces

$$\left(x_0, \left\{ \begin{array}{l} x_0|f : x, \quad x|g : x_G, \\ x_0|h : z, \quad z|x/d:l \sqcup_d y/d:r, \quad x_G|t'_G, \\ z|i : x_I, \end{array} \right\}, Con_2 \right)$$

where the further decomposition of the constraints $x_G|t'_G$, $x_I|t'_I$ need not interest us. Since the bifurcation for z contains contexted variables, it is replaced by $z|z_l \sqcup_d z_r, z_l|x/d:l, z_r|y/d:r$, but the latter two constraints lead to the introduction of bifurcations also for x and y . Furthermore, the feature constraints on x and z are distributed

over their respective variants. We eventually get:

$$\left(x_0, \left\{ \begin{array}{l} x_0|f : x, \quad z_l|g : x_G/d:l, \\ x_0|h : z, \quad z_l|i : x_I/d:l, \quad x_G|t'_G, \\ x|z_l \sqcup_d x_r, \quad x_r|g : x_G/d:r, \quad x_I|t'_I \\ y|y_l \sqcup_d z_r, \quad z_r|i : x_I/d:r, \\ z|z_l \sqcup_d z_r, \end{array} \right\}, Con_3 \right)$$

Although the resulting description contains contexted variables which refer to variants of x_G and x_I , we do not have to introduce bifurcations for these variables. Hence the information contained in constraints on the variables x_G and x_I is not duplicated, although both variables are used within a disjunction. However, if there would be more information on the values of the g - or i -features of z_l , x_r , or z_r , for instance a constraint of the form $z_l|g : x'$, this would lead to the introduction of a bifurcation for x_G , and some parts of the structure embedded under x_G would have to be distributed over the variants of x_G . But the unfolding of the structure below x_G would be limited to the minimal necessary amount, since those parts of the structure that do not interact with information known about x' could make use of contexted variables.

Informally speaking, if we unify a structure with a disjunction, only those parts of the structure have to be copied that

interact with the information contained in the disjunction.

4.4 Algorithmic Considerations

One major advantage of our treatment is its similarity with conventional rewrite systems for feature logic. Since we perform only conventional substitution of variables (opposed to conditional substitution as in [Maxwell/Kaplan 89], see [Eisele/Dörre 90] for a discussion), our system can be easily implemented in environments providing term unification (PROLOG), or the almost linear solution of the union/find problem could be exploited (see e.g. [Ait-Kaci 84]). The only essential extension we need concerns the treatment of context descriptions. A context description contained in a contexted variable is always purely conjunctive. Hence the necessary operations (comparison with TRUE, locating, adding or deleting a simple conjunct) can each be implemented by one simple list operation. In the constraint expressing inconsistent contexts ($\perp[k]$), k is a disjunction of the inconsistencies found so far (which themselves are purely conjunctive). This could be also represented in a list of (purely conjunctive) contexts. However, the exclusion of irrelevant constraints $x|t$, where $Con(x)$ is covered by k in $\perp[k]$, and the (final) test if $k \equiv \text{TRUE}$ involves a bit more propositional calculation. Since these tests might occur more often than the detection of a new inconsistency, it might be worthwhile to use a representation that facilitates the test for entailment. In any case, the implementation can make use of fast bit-vector operations.

4.5 Maxwell and Kaplan's Approach

An approach which ours is especially interesting to compare with is the disjunctive constraint satisfaction procedure given in [Maxwell/Kaplan 89], because of the similar representations involved in the two approaches. They use also disjunction names and contexts to represent disjunctive constraints and propose a general transformation procedure which turns a rewrite system for non-disjunctive constraints into one which handles disjunction of constraints with the use of contexted constraints, having the implicational form ($k \rightarrow \phi$), where ϕ is some non-disjunctive constraint. This is done by replacing every rewrite rule by its "contexted version", e.g., $\phi_1 \wedge \phi_2 \rightarrow \phi_3$ is replaced by $(k_1 \rightarrow \phi_1) \wedge (k_2 \rightarrow \phi_2) \rightarrow (k_1 \wedge \neg k_2 \rightarrow \phi_1) \wedge (k_2 \wedge \neg k_1 \rightarrow \phi_2) \wedge (k_1 \wedge k_2 \rightarrow \phi_3)$, where k_1 and k_2 are variables for context descriptions. There are two severe efficiency-critical problems if we want to use the outcome of this translation without further optimization. First, any rule of the generated form should only apply to a pair of contexted constraints whose contexts are compatible, i.e. $k_1 \wedge k_2$ is not contradictory. But now, since context descriptions may include conjunction and negation at any level, this test itself is an \mathcal{NP} -complete problem, which has to be solved before every application of a rule. The second problem concerns substitution. Consider a rule like $x \doteq y \wedge \Phi \rightarrow \Phi_{y \rightarrow x}$. The translation produces a rule in which Φ is rewritten to both Φ and $\Phi_{y \rightarrow x}$, indexed with different context descriptions. Thus, we cannot simply perform a replacement, but instead, have to make a copy of Φ (or at least those parts of Φ containing y). Unfortunately, this prevents also the efficient union/find method to be employed for building equivalence classes for variables instead of actual substitution. All of these problems are avoided if we let the context description of a contexted constraint depend implicitly on the variables in it through the introduction of context-unique variables. From this point of view, our method can be seen as an op-

timized implementation of the translated rewrite system for unification in feature logic with sorts and negation.

5 Conclusion

To summarize, we have presented a new unification method for the full language of feature logic including variables, sorts and negation which avoids expansion to disjunctive normal form, if possible. The basic principle is to minimize unnecessary interaction of different disjunctions by keeping them local to those attributes which they specify different values for through the introduction of disjunction names. With this treatment we avoid exponential explosion in many practical cases. A precursor of this algorithm [Dörre/Eisele 89] has been implemented and is successfully used in a grammar development environment. Besides the obvious advantage of increased efficiency, our compact representation of disjunctive information also facilitates the comparison of alternative solutions with common parts, which has been proved to be a very valuable property in our application. Our algorithm is specified in a completely formalised way as a rewrite system for which a model-theoretic semantics is given. It may seem that there are a lot of rules, but this can be explained by the following facts: we include a complete reduction from feature terms (like in Kasper/Rounds logic) to feature descriptions (as used in LFG); we handle all different types of constraints, including sorts and negation in one framework; and our rules only involve few primitive operations for which simple and fast implementations exist.

References

- [Ait-Kaci 84] Ait-Kaci, H. (1984). A Lattice-Theoretic Approach to Computation Based on a Calculus of Partially-Ordered Type Structures. Ph.D. Thesis. University of Pennsylvania.
- [Dörre/Eisele 89] Dörre, J. and A. Eisele (1989). Determining consistency of feature terms with distributed disjunctions. In: D. Metzger (ed.) *GWAI-89, 13th German Workshop on Artificial Intelligence*, pp. 270–279. Informatik Fachberichte 216, Springer.
- [Eisele/Dörre 88] Eisele, A. and J. Dörre (1988). Unification of disjunctive feature descriptions. In: *Proc. of the 26rd Ann. Meeting of the ACL*, Buffalo, NY.
- [Eisele/Dörre 90] Eisele, A. and J. Dörre (1990). Disjunctive Unification. IWBS-Report, IWBS, IBM Germany, Postfach 80 08 80, 7000 Stuttgart 80, W. Germany. To app. in: *Proc. of the Workshop on Unification Formalisms — Syntax, Semantics and Implementation*, Titisee, MIT Press.
- [Karttunen 84] Karttunen, L. (1984). Features and Values. In: *Proceedings of COLING 1984*, Stanford, CA.
- [Kasper 87] Kasper, R.T. (1987). A Unification Method for Disjunctive Feature Descriptions. In: *Proc. of the 25th Ann. Meeting of the ACL*. Stanford, CA.
- [Kasper/Rounds 86] Kasper, R.T. and W. Rounds (1986). A Logical Semantics for Feature Structures. In: *Proc. of the 24th Ann. Meeting of the ACL*. Columbia University, New York, NY.
- [Maxwell/Kaplan 89] Maxwell, J. and R. Kaplan (1989). Disjunctive Constraint Satisfaction. In: *Proc. Int. WS on Parsing Technologies*, Carnegie Mellon, Pittsburgh, PA.
- [Smolka 88] Smolka, G. (1988). *A Feature Logic with Subsorts*. LILOG-Report 33. IBM Germany. To app. in: *J. of Automated Reasoning*.
- [Smolka 89] Smolka, G. (1989). *Feature Constraint Logics for Unification Grammars*. IWBS-Report 93. IWBS, IBM Germany. To app. in: *Proc. of the WS on Unification Formalisms — Syntax, Semantics and Implementation*, Titisee, MIT Press.