Structure Sharing Unification of Disjunctive Feature Descriptions*

Johannes Matiasek
Austrian Research Institute Artificial Intelligence
Schottengasse 3, A-1010 Vienna, Austria
john@ai.univie.ac.at

Abstract

A method is presented which allows for unification of disjunctive feature descriptions with a minimal amount of copying. This is accomplished by using a lazy incremental copy technique in combination with a representation of feature descriptions that allows for distributed disjunctions. The use of context descriptions keeps disjunctions as local as possible and prevents independent alternatives to interact unnessecarily, thus helping to avoid redundant copying. In that way structure sharing is possible between different feature descriptions as well as between disjuncts. Furthermore the unification algorithm need not consider nondisjunctive parts of a feature descriptions twice when dealing with alternatives as e.g., in implementations employing backtracking. This allows for an efficient implementation of feature based systems.

1 Introduction

Many recent linguistic theories use feature structures as their main formalism to represent linguistic objects and unification as the primary operation to combine them. Any implementation of these theories will thus have to face the problem of efficiently representing and unifying feature descriptions.

Constraints on linguistic objects cannot always be stated by purely conjunctive feature structures. This has led to various approaches to incorporate disjunction by distinguishing between the feature structures themself and a feature description language, in which disjunction can be expressed (cf. e.g. Johnson 1988, Smolka 1989, Kasper and Rounds 1990) and to several methods to cope with the exponential complexity inherent with disjunction. An important issue in efficiently unifying feature structures is to avoid unnecessary copying during unification, since the effort of copying would be wasted in case unification fails or substructures are not changed by unification.

In this paper a method is presented which allows for unification of disjunctive feature descriptions with a minimal amount of copying.

*This research has been sponsored by the Austrian *Fonds zur Förderung der wissenschaftlichen Forschung*, Grant No. P7986-PHY. I would like to thank Prof. R. Trappl for his continuing support.

2 Disjunctive Feature Descriptions

Feature structures represented by DAGs are inherently conjunctive, and thus alone not sufficient to describe linguistic data. Even grammar formalisms which lack explicit means to express disjunctions have to account for alternative ways to satisfy a given linguistic description (e.g. by means of alternative phrase structure rules). However, incorporating disjunction in the feature description formalism itself seems to be the appropriate way to express linguistic constraints most concisely (cf. Karttunen 1984).

A problem encountered with disjunctive feature discriptions is the exponential complexity of the satisfability problem, which in the worst case requires expanding to disjunctive normal form (DNF). Several Methods have been proposed to circumvent expanding to DNF in practical cases. Kasper (1987) uses a method of successive approximation which works best when incompatibilities of alternatives with nondisjunctive constraints arise, but otherwise tends to expanding to DNF; Eisele and Dörre (1988) try to keep disjunctive values as local as possible (as long they do not interact with other parts of the feature description, in which case the disjunction has to be lifted); Maxwell and Kaplan (1991) associate propositional variables with disjunctions, unsatisfiable combinations of alternatives are represented by propositional formulas of these variables; Eisele and Dörre (1990) use a similar approach of distributing disjunctions but use *conjunctive context descriptions* indicating to which combination of alternatives a variable belongs. This last method explicitly accounts for the possibility of structure sharing between different disjuncts and will therefore be used as the basis for the method presented here.

3 Approaches to Structure Sharing

Since node merging is a destructive operation, provisions have to be made not to affect the input structures during unification. The straightforward method of copying these before starting unification suffers from wasting time and memory in two cases: when unification fails or when substructures of the input structures are not affected by the unification operation and could possibly be shared with the resultant structure. Some methods have been proposed to avoid redundant copying (e.g. Pereira 1985, Kogure 1990). Emele (1991) compares these and other approaches and proposes a method (*Lazy Incremental Copying*) combining advantages of these approaches and avoiding redundant copying (at least for conjunctive feature structures).

Lazy Incremental Copying

The key ideas of this method to avoid redundant copying are: to copy lazily, i.e. a node is only copied if it is shared with another structure and has to be changed in the current unification; to record the updates to a node with the node itself and the use of a dereferencing scheme to find the actually active node. Each node is represented as a structure containing the type, the arcs of the node, its generation and a pointer to its copy. Any time

a new structure is created by unification, a global generation counter is incremented and all newly created copies belong to that generation. Dereferencing is performed with respect to the current environment represented by a sequence of generation counters (the current generation being the last) reflecting the history of how the structure under consideration has been created by unification. The representative node is the last node in the copy chain whose generation is a member of the environment. Unification of two nodes is simply performed by dereferencing both nodes, creating a copy (only if none of both nodes belongs to the current generation), linking the two original nodes with the result node via their copy slot, unifying their types and destructively merging the arcs of the arguments into the result node (possibly leading to further unifications recursively). This way redundant copying can completely be eliminated at the moderate cost of dereferencing.

However, the method has shortcomings in dealing with disjunctive feature descriptions. Although structure sharing between alternatives and the nondisjunctive part of a feature description is possible in principle it remains unclear how to represent local disjunctions (Emele's remark on generalizing environments to trees of generation counters is not sufficient). Furthermore the unification algorithm and the backtracking scheme proposed there would not be able to distinguish between the disjunctive and nondisjunctive parts of a feature description and will have to replicate unification of the nondisjunctive parts for every alternative.

4 The Unification Method

To remedy these disadvantages the method proposed here will make use of distributed disjunctions (named, i.e. numbered globally) to keep disjunctions as local as possible. If only structure sharing were the point, this could be done simply by associating each alternative with an environment. But then each alternative would have to be considered seperately and the unification algorithm cannot take advantage of the fact, that nondisjunctive parts of the feature description must be considered only once. To indicate that a node has different instances in different contexts disjunction nodes are introduced, splitting the node into two alternatives (since only binary disjunctions are considered, disjunctions with more alternatives are reduced automatically to binary disjunctions using the associative law). To keep track upon which alternative is currently being considered, each alternative is associated with its own environment and a context description indicating which combination of disjunction branches it represents. These context descriptions need only be stated conjunctively (cf. Eisele and Dörre 1990) and are represented by bitvectors allowing the frequently occurring test for context entailment to be performed by a single instruction.

The representation of environments now has to be somewhat enriched, containing: the generation counter of the environment; the context description; the list of generation counters representing the environments values may be inherited from; and, in case of environments in the universal context (parent environments) a hash table mapping subcontexts to subenvironments—in case of subenvironments a pointer to their parent environment.

Root nodes of feature descriptions are represented always in an universally con-

texted environment. Unification of two nodes then proceeds as above with two exceptions: node merging is only possible if both nodes belong to the same context (cf. Eisele and Dörre 1990) and a special treatment of disjunctive nodes is required.

The burden of guaranteeing that a node is represented in a certain context is put onto the dereferencing operation. Note, that copy links may only link nodes with the same context, splitting of context is only possible with disjunctive nodes. Dereferencing is now performed with respect to an environment and a target context. Following the copy links is done as before with the exception, that the accessibility of a contexted node is not checked with the generation count of the node itself but with the generation count of its parent environment (thus removing the need to calculate complex inheritance lists for subenvironments—which may share nodes with subenvironments of other feature descriptions). When a possible representative node is found, its context (i.e. the context corresponding to its generation counter) is checked against the target context. In case of mismatch, there are two possibilities: either the node found is a disjunctive node representing one of the disjunctions required in the target context, then the appropriate branch of the disjunction is taken and dereferencing proceeds; otherwise a disjunctive node has to be introduced, created within the subenvironment of the current environment which matches the present context(and possibly has to be created). The node-alternatives are copies of the original node created within the respective subcontexts. Node splitting then proceeds with the matching node variant until the target context is reached. This procedure guarantees the condition, that copy links may only connect nodes within the same context and also provides the possiblity of structure sharing between different alternatives, preserving the condition that node arcs may only point to nodes with the same or a higher context (which in turn is crucial for node unification employing the dereferencing procedure described to work).

Once node representatives with the required context have been established, node merging proceeds as usual unless one (or both) of the nodes are disjunction nodes. In that case each alternative of the disjunction node is unified with the other node recursively within the context of the alternative. The dereferencing and node splitting procedure performs then all the context splitting operation required. In case unification fails in some context, the corresponding subenvironment and all subenvironments corresponding to subcontexts of the failing contexts are marked as failing, so these have never to be considered again in further unification steps.

As can be seen, disjunctions independent from each other never interact, parts of alternatives not affected by other parts of the feature description may be shared, even structure sharing with alternatives of other feature descriptions is possible. Nondisjunctive parts of feature descriptions must be processed only once. With typical applications this results in savings of copying and processing time that overcompensate the effort of the more complex dereferencing procedure at large.

5 Summary

We have presented an unification algorithm that combines a lazy incremental copy technique with a representation of feature descriptions that allows for distributed disjunctions. The use of context descriptions allows to keep disjunctions as local as possible and prevents independent alternatives to interact unnessecarily, thus helping to avoid redundant copying. In that way structure sharing is possible between different feature descriptions as well as between disjuncts. Furthermore the unification algorithm need not consider nondisjunctive parts of a feature descriptions twice when dealing with alternatives as e.g. in implementations employing backtracking. This allows for an efficient implementation of feature based systems.

The algorithm has been implemented in CommonLisp and is used as the formal basis in the system VIE-$\mathcal{DU}$ (Buchberger *et al.* 1991)), a natural language consulting system being developed at the Austrian Research Institute for Artificial Intelligence.

References

Buchberger, E., E. Garner, W. Heinz, J. Matiasek, and B. Pfahringer. (1991) VIE-DU — Dialogue by Unification. In *Proceedings of the 7. Österreichische Artificial Intelligence Tagung.* Berlin: Springer

Eisele, A., and J. Dörre. (1988) Unification of Disjunctive Feature Descriptions. In *Proceedings of 26th Annual Meeting of the Association for Computational Linguistics*, 286–294. Buffalo, NY. State University of New York.

Eisele, A., and J. Dörre. (1990) Feature Logic with Disjunctive Unification. In *Proceedings of the 13th COLING*, Vol. 2, 100–105. Helsinki.

Emele, M. C. (1991) Unification with Lazy Non-Redundant Copying. In *Proceedings of 29th Annual Meeting of the Association for Computational Linguistics*, 323–330. Berkeley, CA. University of California.

Johnson, M. (1988) *Attribute–Value Logic and the Theory of Grammar.* CSLI Lecture Notes 16. Stanford, CA: CSLI.

Karttunen, L. (1984) Features and Values. In *Proceedings of the 10th COLING.* Stanford, CA.

Kasper, R. (1987) A Unification Method for Disjunctive Feature Descriptions. In *Proceedings of 25th Annual Meeting of the Association for Computational Linguistics*, 235–242. Stanford, CA. Stanford University.

Kasper, R., and W. Rounds. (1990) The Logic of Unification in Grammar. *Linguistics and Philosophy* 13:35–58.

Kogure, K. (1990) Strategic Lazy Incremental Copy Graph Unification. In *Proceedings of the 13th COLING*, 223–228. Helsinki.

Maxwell, J. T., and R. M. Kaplan. (1991) A Method for Disjunctive Constraint Satisfaction. In *Current Issues in Parsing Technology*, ed. M. Tomita. Dordrecht: Kluwer.

Pereira, F. C. N. (1985) A Structure Sharing Representation for Unification-based Grammar Formalisms. In *Proceedings of 23rd Annual Meeting of the Association for Computational Linguistics*, 137–144. Chicago, IL.

Smolka, G. (1989) A Feature Logic with Subsorts. Technical Report LILOG-Report 33, IBM-Germany, Stuttgart.