UNIFICATION AND CLASSIFICATION: AN EXPERIMENT IN INFORMATION-BASED PARSING

Robert T. Kasper
USC/Information Sciences Institute
Admiralty Way, Suite 1001
Marina del Rey, CA 90292

When dealing with a phenomenon as vast and complex as natural language, an experimental approach is often the best way to discover new computational methods and determine their usefulness. The experimental process includes designing and selecting new experiments, carrying out the experiments, and evaluating the experiments. Most conference presentations are about finished experiments, completed theoretical results, or the evaluation of systems already in use. In this workshop setting, I would like to depart from this tendency to discuss some experiments that we are beginning to perform, and the reasons for investigating a particular approach to parsing. This approach builds on recent work in unification-based parsing and classification-based knowledge representation, developing an architecture that brings together the capabilities of these related frameworks.

1. Background: Two General Frameworks for Representing Information

1.1. Unification-based Grammars

A variety of current approaches to parsing in computational linguistics emphasize declarative representations of grammar with logical constraints stated in terms of feature and category structures. These approaches have collectively become known as the "unification-based" grammars, because unification is commonly used as the primary operation for building and combining feature structures. Some of the simplest of these grammatical frameworks, as exemplified by the PATR-II system [Shieber 1984], state constraints on features entirely in terms of sets of unifications that must be simultaneously satisfied whenever a grammatical rule is used. In such systems all constraints on a rule or lexical item are interpreted conjunctively. Many of the more recent frameworks also use other general logical connectives, such as disjunction, negation and implication, in their representation of constraints. The usefulness of such logical constraints is abundantly illustrated by linguistic models, including Systemic Grammar (SG) [Halliday 1976] and Head Driven Phrase Structure Grammar (HPSG) [Pollard&Sag 1987], and by computational tools such as Functional Unification Grammar (FUG) [Kay 1985]. For example, SG and FUG even use disjunctive alternations of features, instead of structural rules, as the primary units of grammatical organization. While the intuitive interpretation of these logical constraints is rather straightforward, and they are quite natural for linguists to formulate, large-scale implementations of them have typically involved finding a delicate balance between expressive power and computational efficiency.

Some difficulties can be expected in developing a system for computing with disjunctive and negative feature constraints, because it has been established that common operations

on such descriptions, such as unification and subsumption, are NP-complete and require exponential time in the worst case. The most common and obvious way to deal with disjunctive constraints is to expand the grammatical description to disjunctive normal form (DNF) during a pre-processing step, thereby eliminating disjunction from the rules that are actually used by the parser. This method works reasonably well for small grammars, but it is clearly unsatisfactory for larger grammars, because it actually requires exponential space and time in all cases. For even modest amounts of disjunction, the parser is forced to operate on a huge description, even in many cases where no exponential expansion would be necessary.

It is possible to avoid exponential expansion for most practical grammars, and several unification algorithms for disjunctive feature descriptions have been developed in recent years. The first of these algorithms was developed by Karttunen [Karttunen 1984]. His method of representing disjunction allowed value disjunction (i.e. alternative values of a single feature), but it did not allow general disjunction (i.e. constraints involving multiple features). Although it is possible to transform any description that contains general disjunction into a formally equivalent description that contains only value disjunction, this transformation may sometimes result in loss of efficiency or lack of clarity in the structures produced by a parser.

Two more recent algorithms [Kasper 1987, Eisele&Doerre 1988] allow general disjunctive descriptions, and avoid expansion to DNF by exploiting logical equivalences between descriptions to produce normal forms that allow a more compact representation. Kasper's algorithm is based on a normal form that divides each description into definite and indefinite components. The definite component contains no disjunction, and the indefinite component contains a list of disjunctions that must be satisfied. The Eisele&Doerre algorithm uses a different normal form that guarantees the detection of any inconsistencies during the normalization process by selectively expanding disjunctions that might possibly interact with other information in the description. Although a precise characterization of the differences in performance between these algorithms involves many subtleties, the Eisele&Doerre algorithm usually handles value disjunction more efficiently, and the Kasper algorithm usually handles general disjunction more efficiently. The crucial technique shared by both algorithms is the use of a normal form that allows early elimination of alternatives when they are inconsistent with definite information.

The Kasper algorithm was first implemented as an extension to the unification algorithm of the PATR-II parser, and it has been further developed to handle conditional descriptions and a limited type of negation [Kasper 1988a]. These extensions to PATR-II have been used to construct an experimental parser for systemic grammars [Kasper 1988b], which has been tested with a large grammar of English.

Although these methods for processing complex feature constraints are generally much more efficient than expansion to DNF, they still have several significant sources of inefficiency:

1. a large amount of structure must be copied in order to guarantee correct unification;

2. consistency checks are required between components of a description that do not share any features in common, because unification cannot determine whether any dependencies exist between two structures without actually unifying them;

3. repeated computations are often required over sub-expressions of descriptions, because the results of prior consistency checks are not saved.

These sources of inefficiency are not unique to one method of parsing with disjunctive descriptions; similar shortcomings are commonly reported for most unification-based systems. For example, the Eisele&Doerre algorithm eliminates some redundant consistency checks, but it generally requires copying significant portions of a description to do so. The unification literature contains several techniques for reducing the amount of copying by structure sharing, but these techniques appear to solve only part of the problem. A more general approach to improving the efficiency of unification may be available by adopting methods that are used in classification-based systems.

1.2. Classification-based Knowledge Representation

The KL-ONE family of knowledge representation systems organize information about objects and the relations between them into conceptual hierarchies (a combination of semantic networks and frames) according to class membership, where *X* is below *Y* in the hierarchy if *X* is a subclass or instance of the class *Y*. For example, a hierarchy of English word classes would probably contain Verbs, Modal-Verbs as a subclass of Verbs, and the word "should" as an instance of Modal-Verbs. More formally, the hierarchy is a subsumption-ordered lattice based upon logical properties that can be deduced from the definitions of concepts and the facts known about particular objects. In these systems, *classification* is the operation that places a new class or object into the lattice according to the subsumption order. A primary benefit of classification is that it organizes large collections of knowledge in such a way that properties shared in common by many objects only need to be represented once, yet they can still be efficiently accessed.

KL-ONE and similar frameworks have been used for semantic interpretation in some natural language processing systems, but usually in a way that is quite separate from the grammatical parsing process. Recent research indicates that it may be advantageous to make use of a classification-based framework for processing grammatical knowledge as well. Many formal properties are shared by the feature descriptions used in unification-based grammars and the terminological definitions used in KL-ONE. Generally speaking, linguistic categories correspond to concepts, and their features (or attributes) correspond to binary relations in the knowledge representation system. The similarity between these two types of descriptions has been most clearly documented by Smolka [Smolka 1988] in his development of a logic that integrates a significant combination of their expressive capabilities. Smolka has also shown that the subsumption and unification problems for this logic can be reduced to each other in linear time. Thus, systems based on either term subsumption or unification can be expected to solve a similar range of problems, although differing levels of non-asymptotic time/space efficiency can be expected. Theoretical results have also been based on the observation that feature structures can be implicitly organized into a subsumption lattice of types according to their information content. In most unification-based systems the lattice is not explicitly constructed, but a classification-based system can be used to place the feature structures of a grammar and lexicon into a structure-sharing lattice, potentially improving both space and time efficiency.

Despite the underlying similarities between the KL-ONE framework and unification-based grammars, there are significant differences in the expressive capabilities that are usually provided. In particular, the knowledge representation systems typically have general constraints on relations with multiple values, whereas most unification-based systems do not provide a direct representation for features with set values. On the other hand, complex logical constraints involving disjunction and negation have been more extensively developed in unification-based systems than in classification-based systems. The LOOM system [MacGregor 1988], which has been developed at USC/ISI, appears to be the first in

the KL-ONE family to have included general disjunction and negation in its concept definition language. The implementation of classification for disjunctive concepts has been based on the same strategy that was originally developed for unification with disjunctive feature descriptions [Kasper 1987]. The implementation of classification for concepts defined by negation is still in progress. With these extensions, the LOOM system should be able to handle the full range of constraints that have been used in linguistic descriptions of feature structures.

2. An Experiment in Classification-based Parsing

In order to explore a strategy for parsing based on classification, our first experiment will be to emulate the unification component of our parser for a large systemic grammar of English [Kasper 1988b] within the framework of LOOM. It appears to be straightforward to convert the feature constraints of the grammar into a set of definitions that can be processed by LOOM, because of the underlying correspondences between LOOM's concept definitions and linguistic feature descriptions that we have already described. It is also straightforward to perform an operation that is equivalent to the unification of feature structures within LOOM. This is accomplished by forming an object which is defined as the conjunction of the objects corresponding to the feature structures.

Motivating this experiment are two primary goals:

1. to investigate the extent to which classification can be used to organize the knowledge contained in linguistic descriptions so that it can be more efficiently accessed during the parsing process;

2. to develop a suitable architecture for integrating semantic information into the parsing process, in a way that knowledge specific to application domains does not have to be re-organized for parsing.

2.1. Efficiency Considerations

The classification-based architecture used by LOOM solves a whole class of related efficiency problems by explicitly constructing and maintaining a subsumption-ordered lattice of terms with inheritance. In particular, it may provide substantial improvements for some of the above mentioned sources of inefficiency that have been observed with unification-based parsers.

2.1.1. Structure Sharing

The organization of objects into a lattice automatically provides a great amount of structure sharing. Pointers are copied instead of structures whenever objects are defined or modified.

In most unification-based parsers, it is necessary to make new copies of the feature structures that are associated with lexical items or grammatical rules whenever they are used in building a description of a sentence (or one of its constituents). In a classification-based system the entire structure does not need to be copied, because the description of a constituent can contain pointers to the classes of objects that it instantiates. This representation not only saves space, but it also allows the parser to make use of information that has already been precomputed (during the classification process) for classes of objects in the grammar and lexicon.

2.2. Integrating Semantic Information into the Parsing Process

In order for practical natural language parsers to be produced with less effort per application, it is desirable for the knowledge base of an application to also be usable by a general purpose parser. Existing systems often use semantic grammars that are specific to a particular application domain, or require substantial reorganization of the information used by an application so that it can be used by the parser. A more effective use of knowledge sources may be possible if linguistic features and information about an application's semantic domain are defined in the same general knowledge representation framework. Using a classification-based system, links can be established between terms of the semantic domain and terms of the linguistic knowledge base that correspond to them. This approach has already been explored in text generation research [Kasper 1989], where the links are established by stipulating that terms of the application domain specialize one or more terms of the linguistic model. This condition generally holds, because the linguistic model contains primarily abstract features.

Another potential benefit of using an integrated knowledge organization is early disambiguation according to features of the semantic domain. If objects of the semantic domain are directly linked in a knowledge base to lexical or grammatical features, the parser can use information about those objects without any special purpose machinery.

3. Summary

We are developing an experimental parser using the classification-based architecture of the LOOM knowledge representation system. The initial goal is to reproduce the functionality of an existing unification-based parser, using a large grammar of English. If successful, this experiment should enable a comparison of classification and unification as mechanisms for parsing. A classification scheme appears to provide a way of substantially reducing several of the most general sources of inefficiency that are observed in current unification-based parsers. However, this conjecture needs to be examined by performing experiments with several real grammars and applications. Because the classification mechanism is based on general logical properties of feature descriptions, it should be applicable to a broad class of grammars, just as unification-based parsers have been developed for grammars from a diverse range of linguistic theories and applications. In addition to providing an efficient engine for processing the constraints of linguistic feature descriptions, we also expect this type of information organization to provide a strong basis for integrating semantic knowledge and knowledge specific to particular applications into the parsing process.

2.1.2. Indexing Dependencies

The process of classification also keeps track of dependencies between different objects, eliminating the need for checking consistency between components of a description that have no features in common. In effect, an index is incrementally constructed from features to descriptions that contain them.

In most unification-based systems, feature structures are represented by directed graphs or terms. These representations effectively provide an index of features possessed by each object. This type of indexing is generally sufficient if only conjunctive constraints on features are used. When disjunctive constraints are also used, it becomes useful to keep track of dependencies between different parts of a complex description, in order to avoid repeated consistency checks between parts that share no features in common. A reverse index (from features to objects having those features) can be used to avoid these useless consistency checks. This second kind of index is created automatically when feature structures are classified into an explicit lattice.

2.1.3. Avoiding Redundant Computations

The first time that a component of a description is classified, it is placed into a lattice containing all other descriptions in the knowledge base. The lattice structure makes full consistency checks unnecessary between objects that are known to be in a subsumption relationship. The object-oriented representation of the lattice also makes it possible to store the results of consistency checks between components of a description, so that they do not need to be repeated.

2.1.4. Using Classification as a Grammar Compiler

The classification-based architecture is also able to impose a system of type constraints on feature structures. Constraints may be placed on the sets of features that are required or prohibited for particular types of objects, and on the types of objects that may occur as the values of particular features. Structures that violate one of these constraints are automatically marked as incoherent. In contrast, many of the unification methods used in computational linguistics have untyped feature structures. For applications of limited scale, an untyped unification-based system may provide acceptable results with somewhat less overhead than a classification-based approach. In particular, an untyped feature system allows greater flexibility in the early stages of developing a grammar. However, for applications that are necessarily knowledge-intensive, a classification-based system is likely to be preferable, because it organizes a large collection of linguistic knowledge (and related nonlinguistic knowledge) in such a way that it can be more efficiently processed.

From another perspective, the classification-based system can be seen as carrying out a compilation procedure on a linguistic knowledge base. The initial loading (or compilation) of a large grammar into the system may be computationally expensive, but the result is a parser that may be considerably more efficient at run-time than current unification-based systems. In the early stages of developing a grammar, when not many sentences are parsed with a particular version of the grammar before it is substantially revised, the benefits of compilation may not be appreciated. When the system is actually used in an application, or tested on a large body of text, it may significantly improve performance.

REFERENCES

Brachman, R. and Schmolze, J. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*, Vol. 9:2, 1985.

Eisele, Andreas and Doerre, Jochen. Unification of Disjunctive Feature Descriptions. *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, NY: June 7-10, 1988.

Halliday, Michael. *System and Function in Language*. Kress G., editor, Oxford University Press, London, England, 1976.

Karttunen, Lauri. Features and Values. *Proceedings of the Tenth International Conference on Computational Linguistics: COLING 84*, Stanford, CA: July 2-7, 1984.

Kasper, Robert. A Flexible Interface for Linking Applications to Penman's Sentence Generator. *Proceedings of the DARPA Workshop on Speech and Natural Language*, Philadelphia: February, 1989.

Kasper, Robert. Conditional Descriptions in Functional Unification Grammar. *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, NY: June 7-10, 1988a.

Kasper, Robert. An Experimental Parser for Systemic Grammars. *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest: August, 1988b.

Kasper, Robert. A Unification Method for Disjunctive Feature Descriptions. *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, Stanford, CA: July 6-9, 1987.

Kay, Martin. Parsing in Functional Unification Grammar. *Natural Language Parsing*, Dowty D., Karttunen L., and Zwicky A. (eds.), Cambridge University Press, Cambridge, England, 1985.

MacGregor, Robert. A Deductive Pattern Matcher. *Proceedings of AAAI-88, The National Conference on Artificial Intelligence*, St. Paul, MN: August, 1988.

Pollard, Carl and Sag, Ivan. *Information Based Syntax*. CSLI Lecture Notes Number 13, Univeristy of Chicago Press, 1987.

Shieber, Stuart. The Design of a Computer Language for Linguistic Information. *Proceedings of the Tenth International Conference on Computational Linguistics: COLING 84*, Stanford, CA: July 2-7, 1984.

Smolka, Gert. A Feature Logic with Subsorts. LILOG Report 33, IBM Deutschland, Stuttgart, West Germany, May 1988.