Performing Integrated Syntactic and Semantic Parsing Using Classification

Robert T. Kasper and Eduard H. Hovy

Information Sciences Institute of USC
4676 Admiralty Way
Marina del Rey, CA 90292-6695

Introduction

This paper describes a particular approach to parsing that utilizes recent advances in unification-based parsing and in classification-based knowledge representation. As unification-based grammatical frameworks are extended to handle richer descriptions of linguistic information, they begin to share many of the properties that have been developed in KL-ONE-like knowledge representation systems. This commonality suggests that some of the classification-based representation techniques can be applied to unification-based linguistic descriptions. This merging supports the integration of semantic and syntactic information into the same system, simultaneously subject to the same types of processes, in an efficient manner. The result is expected to be more efficient parsing due to the increased organization of knowledge.

The use of a KL-ONE style representation for parsing and semantic interpretation was first explored in the PSI-KLONE system [2], in which parsing is characterized as an inference process called *incremental description refinement*. The key idea underlying this process is that a description of an object can become increasingly more specific as additional features are learned from multiple knowledge sources, which is essentially the same idea that underlies most unification-based approaches. Bobrow and Webber identified four crucial capabilities that a representational system should have in order to support the process of incremental description refinement. These capabilities, not all available to Bobrow and Webber in 1980, have recently been developed in the Loom knowledge representation system [12] and hence enable the practical development of the new parsing method. They are:

1. What properties of a structured object provide sufficient information to guarantee the applicability of a description to (some portion of) that object — i.e., criteriality conditions. Loom provides a separation of definition (necessary and sufficient conditions) and constraints (implied features).

2. What mappings are possible between classes of relations — e.g., how functional relationships between syntactic constituents map onto semantic relationships. This is not part of Loom, but can be captured in the interrelationships between a syntax-oriented grammar and a semantics-oriented concept taxonomy.

3. Which pairs of descriptions are mutually incompatible — i.e., which cannot both apply to a single individual. Loom provides more complete inference of disjointness than previous systems in the KL-ONE family.

4. Which sub-categorizations of descriptions are exhaustive — i.e., at least one of the subcategories applies to anything to which the more general description applies. Loom provides inference with respect to coverings, implemented by disjunctive descriptions.

This paper outlines how a parser can be built using Loom's classifier as the primary inference operation. It first describes the process of unification, then points out similarities between unification and classification, then describes the process of parsing using the classifier with an example. The integration of semantic and syntactic information into the same system is discussed. Finally, the efficiency benefits of the new method are mentioned.

Constraints in 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 [16], 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 utility of such logical constraints is abundantly illustrated by linguistic models, including Systemic Grammar (SG) [5] and Head Driven Phrase Structure Grammar (HPSG) [14], and by computational tools such as Functional Unification Grammar (FUG) [11]. 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 balance between expressive power and computational efficiency, not an easy task.

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 [15]. 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. Though this method works reasonably well for small grammars, it turns out to be unsatisfactory for larger grammars.

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: [6, 10, 4]. The latter two algorithms 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 algorithm of Eisele and Doerre 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. 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 [8]. These extensions to PATR-II have been used to construct an experimental parser for systemic grammars [9], which has been tested with a large grammar of English called Nigel, which is part of the Penman language generation system [13].

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 subexpressions of descriptions, because the results of prior unifications (and compatibility tests) 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. 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.

Classification-based Knowledge Representation

Unfortunately, as a grammar (or knowledge base) grows in size and complexity, it becomes increasingly less efficient to simply unify partial descriptions of constituents with a description from the grammar (as in most unification-based frameworks). Instead, it becomes preferable to classify each partial description of a constituent with respect to the objects that are defined by the grammar, exploiting known relationships between components of the grammatical description. Since the components of the grammar are known before parsing commences, various relationships, such as subsumption and compatibility, can be used to construct a lattice of grammatical objects, eliminating the need to derive those relationships repeatedly at parse time.

The KL-ONE family of knowledge representation systems is based on an explicit logical formalization of many of the constructs that have been explored in semantic networks and frame-based representation systems. They organize information about objects and the relations between them into hierarchies according to specificity, with more specific objects placed below more general ones. For example, a hierarchy of English word classes would probably contain Verbs, Transitive-Verbs as a subclass of Verbs, and the word "like" as an instance of Transitive-Verbs. Each hierarchy is a subsumption-ordered lattice based upon logical properties that can be deduced from the definitions of objects and the facts known about them. 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 by many objects need only be represented once, yet they can still be efficiently accessed by inheritance.

KL-ONE and similar frameworks have been used for semantic interpretation in some natural language processing systems [18], but usually in a way that is quite separate from the grammatical parsing process (an exception is the aforementioned PSI-KLONE system). 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 [17] in his development of a logic that integrates a significant combination of their expressive capabilities. Many 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 [12], 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 several refinements of a strategy that was originally developed for unification with disjunctive feature descriptions [10]. The implementation of classification for concepts defined by negation is still in progress. With these extensions, the Loom system is able to handle a much fuller range of constraints that have been used in actual linguistic descriptions of feature structures.

An Experiment in Classification-based Parsing

In order to explore a strategy for parsing based on classification, we have to represent Penman's grammar in Loom and replace the existing unification component of our parser [9] with activations of Loom's classifier. Motivating this action 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.

It is 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 having a type that is defined as the conjunction of the types corresponding to the feature structures. Conjunction of types yields a type which captures the unification of descriptions in a non-destructive way. Loom also supports merging of instances, corresponding to destructive unification, which is necessary in order to satisfy feature equivalence constraints. When two instances are merged, the resulting instance has a type which is a conjunction of the types of the two original instances.

A disjunction is represented in Loom by an object that generalizes (i.e., classifies above) each of the disjuncts. It is important to note that a disjunctive description entails more than a simple generalization. It is possible to satisfy a generalization without satisfying any of the disjuncts, but in order to satisfy a disjunction, an object must satisfy one of the disjuncts.

Instead of unifying a partial description of a constituent with a grammatical description, we classify the description of the constituent with respect to an object-oriented representation of the grammar, in which each object stores information and constraints associated with a particular type of grammatical constituent. The classifier determines which grammatical classes the constituent instantiates, and the constraints associated with these classes can be used to give a more complete (grammatical, semantic, pragmatic) description of the constituent.

Classification provides a way of decomposing a large description into types, and organizing these types into a lattice so that they can be efficiently searched. Thus, it can provide a more efficient mechanism for unifying large descriptions. Two immediate benefits of the lattice representation are:

1. the descriptions of the grammar do not need to be copied each time that they are unified with a constituent, and

2. each constituent does not need to have an explicit representation of a complete set of its grammatical features, because many of these are entailed by its type.

Classification has the effect of abstracting frequently used combinations of features into a type hierarchy. By factoring descriptions of types of objects out of the feature structures that represent constituents, it is often possible to reduce the numbers of features that need to be unified (recursively) when the constituent is used as a role-filler (in multiple parses), because the type encapsulates restrictions on those features.

A Simple Example

In an example, consider how classification with respect to a simple grammar may be used in parsing the sentence: *David likes computers.* Assume that a lexical/morphological analyzer gives the following type membership information for each word:

```
David: Noun.
computers: Noun.
likes: Verb Transitive Present.
```

Also assume that a rather simple context-free grammar can be used to recognize possible constituents, and that it can be annotated to assign grammatical functions[1]. In the example

[1]Using the classification-based approach outlined here, it is theoretically possible to perform the parsing completely using only classification. However, such a parser would have to examine all substrings of the input in order

sentence, this grammar proposes a constituent c with the type Clause and the following grammatical functions:

```
subject : david
process : likes
dobject : computers
```

This initial description of the constituent, c, is then given to the classifier, which deduces the most specific types that it belongs to. The classifier begins by considering types that are directly below the initial type, i.e., Intrans-Clause and Trans-Clause. The definition of Intrans-Clause states that it is a Clause with a process of type Intransitive. This definition is not satisfied by c, because it does not have a process of type Intransitive. Next, the classifier considers Trans-Clause, which has a definition stating that it is a Clause with a process of type Transitive. This definition is satisfied by c. In addition, Trans-Clause has a constraint: it implies the type Active OR Passive, which means that any object which is a member of Trans-Clause must also be a member of Active OR Passive (that is, Active and Passive form a disjoint covering of Trans-Clause). Therefore, Active OR Passive is added to the list of types that c belongs to.

Because Active OR Passive is a disjunction, it is possible to infer membership in one of the disjuncts by proving incompatibility with all other disjuncts. c is compatible with all of the constraints of Active, but it is not compatible with the constraints of Passive: it has a process of type Present, Passive requires a process of type PastPart, and the types Present and PastPart specialize the disjoint types, Finite and Nonfinite. By eliminating the Passive disjunct from consideration, membership in the Active disjunct can be inferred. Active is the most specific type that can be inferred for c, because it specializes all other types that c belongs to (and there are no more specific types defined in this simple example).

As a consequence of acquiring membership in the type Active, c inherits all constraints that are associated with Active. These constraints require that the actor and subject roles are identical (i.e., that the values of these two roles should be unified), and that the goal and dobject roles are identical. Satisfying these constraints yields the following information about the roles of c:

```
actor : david
goal : computers
```

Thus, given the initial assumption that c is a clause with particular constituents filling the grammatical functions process, subject and dobject, classification deduces:

1. a more specific type: that c is an active clause;
2. values for previously unspecified roles: actor and goal.

The classifier uses the lattice representation of defined types to guide its search for types that are satisfied by a given object. It does not need to consider any types that fall below a type that the object is known not to specialize, such as all types below Intrans-Clause and Passive for the object c.

The power of using this kind of classification scheme may be further exploited by associating semantic and pragmatic constraints with each grammatical type, in addition to the grammatical constraints which have been illustrated.

Integrating Semantic Information into the Parsing Process

One of the greatest advantages of this method of parsing is the possibility of performing integrated semantic and syntactic processing. KL-ONE systems such as Loom were traditionally developed to represent semantic information, and with the inclusion of syntactic information as required for the work described here, both types of knowledge reside in the same system and are accessible to a single classification process.

The advantages for processing are clear. By being able opportunistically to access both semantic and syntactic knowledge at any point during the process, the parser can resolve ambiguities sooner than in the traditional pipeline model, in which syntactic parsing is completed before semantic parsing commences. Many of the structural ambiguities that arise during parsing are only resolvable by semantic knowledge, and pipeline parsers have to maintain all the syntactic possibilities until the semantic parsing phase. Non-pipeline parsers have to perform a complex interweaving of semantic and syntactic processing, requiring increased bookkeeping and more complex system architecture. In the method outlined in this paper, the parser's single call to the classifier will result in the most appropriate information — both semantic and syntactic — being found *and being reconciled, if possible* by the normal action of the classifier.

Another benefit is the increased portability provided by a knowledge representation paradigm used in the Penman system. In order to achieve greater portability, Penman contains a general taxonomic ontology of concepts called the Upper Model [1], under which the concepts from various application domains are subordinated. By inheriting information from the Upper Model, domain concepts can be handled appropriately by the Penman language generator without the generator ever having to be explicitly informed of their individual nature. Similarly, the parser can exploit inherited Upper Model information when trying to place words appropriately into structures. More information can be found in [7].

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 with inheritance. In particular, it may provide substantial improvements for some of the abovementioned sources of

to find all possible constituents, unless sufficient constraints on constituent ordering can be applied early enough in the parsing process. By performing a shallow structural parse before starting the deep classification-based parse, one gains a large improvement in efficiency, because even a skeletal context-free grammar can provide the basic segmentation of the input sentence into its major constituents. Thus, a simple context-free parsing component was used for this purpose with success in the prototype system.

inefficiency that have been observed with unification-based parsers:

Structure Sharing: 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. Hence the organization of descriptions into a lattice automatically provides a great amount of structure sharing.

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. This contrasts with most unification-based systems, in which feature structures are represented by directed graphs (or by first order terms, as in Prolog).

Avoiding Redundant Computations: With un-typed feature structures, each unification is performed on a pair of structures without reference to any stored knowledge, i.e., there is no way for simple unification to use the results of previous unification and subsumption computations, even if many objects with identical features have already been unified. By explicitly representing the types of objects in a lattice, information can be stored for classes of objects, making it possible to avoid repeated computations for multiple objects having the same type (or any more specific type). Thus the first time a component of a description is classified, it is placed into the lattice containing all other descriptions in the knowledge base. Since the lattice explicitly represents the types of objects, it makes full-depth consistency checks unnecessary between objects that are known to be in a subsumption relationship, and subsumption (success) and consistency (failure) tests only need be computed once for all objects that belong to the same types.

Using Classification as a Grammar Compiler: In summary, classification can be seen as providing a capability similar to that provided by compilers in programming systems. Although a simpler unification-based system may provide acceptable results with somewhat less overhead than a classification-based approach on a limited scale, a classification-based system is almost certainly to be preferable for applications that are necessarily knowledge-intensive.

Concluding Remarks

This work is part of an effort to provide the Penman system at ISI/USC with full natural language input and output capabilities. An experimental prototype of this parser using unification and a feature structure representation of part of Penman's grammar has been completed successfully. Most of the work in constructing a parser using the classification-based architecture of Loom and to reproduce the functionality of the unification-based system, now operating on the whole of the grammar, has been completed.

If successful, this experiment should enable a comparison of classification and unification as mechanisms for parsing. The 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.

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.

Acknowledgment

This research was sponsored in part by the United States Defense Advanced Research Projects Agency under contract MDA903-87-C-641 and in part by the United States Air Force Office of Scientific Research under contract F49620-87-C-0005. The opinions expressed here are solely those of the authors.

Thanks to Bob MacGregor for many cooperative discussions and help with a partial implementation of these ideas using Loom.

References

[1] Bateman, J.A., Kasper, R.T., Moore, J.D., Whitney, R.A. A General Organization of Knowledge for Natural Language Processing: The Penman Upper Model. USC/ISI Technical Report, Marina del Rey, 1990.

[2] Bobrow, Robert and Webber, Bonnie. Knowledge Representation for Syntactic/Semantic Processing. In *Proceedings of AAAI-80, The First National Conference on Artificial Intelligence*, Stanford, CA, August 1980.

[3] Brachman, Ronald and Schmolze, James. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*, Vol. 9:2, 1985.

[4] Eisele, Andreas and Doerre, Jochen. Unification of Disjunctive Feature Descriptions. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, NY, June 1988.

[5] Halliday, Michael. *System and Function in Language.* Kress G., (ed.), Oxford University Press, 1976.

[6] Karttunen, Lauri. Features and Values. In *Proceedings of the 10th International Conference on Computational Linguistics: COLING 84*, Stanford, CA, July 1984.

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

[8] Kasper, Robert. Conditional Descriptions in Functional Unification Grammar. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, NY, June 1988.

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

[10] Kasper, Robert. A Unification Method for Disjunctive Feature Descriptions. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, Stanford, CA, July 1987. Also available as USC/Information Sciences Institute Reprint RS-87-187.

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

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

[13] The Penman Project. The Penman Primer, User Guide, Reference Manual, and Nigel Manual. System documentation, USC/ISI Technical Report, Marina del Rey, 1989.

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

[15] Rounds, William and Kasper, Robert. A Complete Logical Calculus for Record Structures Representing Linguistic Information. In *Proceedings of the IEEE Symposium on Logic in Computer Science*, Cambridge, MA, June 1986.

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

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

[18] Sondheimer, Norman K., Weischedel, Ralph M. and Bobrow, Robert J. Semantic Interpretation Using KL-ONE. In *Proceedings of the Tenth International Conference on Computational Linguistics: COLING 84*, Stanford, CA, July 1984.