in: David Farwell, Laurie Gerber, and Eduard Hovy (Eds.). *Machine Translation and the Information Soup. Third Conference of the Association for Machine Translation in the Americas, AMTA '98, Langhorne, PA, USA, October 28-31, 1998, Proceedings.* Lecture Notes in Artificial Intelligence Vol. 1529. Springer: 1998, pp.286-299.

Making Semantic Interpretation Parser-Independent

Ulrich Germann

USC Information Sciences Institute,
Marina del Rey, CA
germann@isi.edu

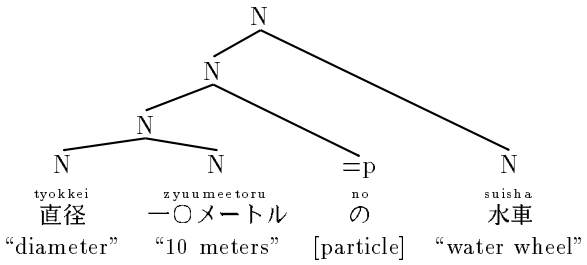
Abstract. We present an approach to semantic interpretation of syntactically parsed Japanese sentences that works largely parser-independent. The approach relies on a standardized parse tree format that restricts the number of syntactic configurations that the semantic interpretation rules have to anticipate. All parse trees are converted to this format prior to semantic interpretation. This setup allows us not only to apply the same set of semantic interpretation rules to output from different parsers, but also to independently develop parsers and semantic interpretation rules.

1 Introduction

Most machine translation systems currently available employ more or less sophisticated glossing techniques: The words and phrases of a source text are replaced by their translations into the target language and rearranged according to syntactic correspondences between the source and the target language. In many cases, this approach leads to acceptable results. For a number of European languages, there are now systems which provide sufficient coverage and quality of translation for applications such as indicative translations and gisting. However, when dealing with languages that show significant differences in their structure, such as Japanese and English, a deeper analysis of the source texts is necessary.

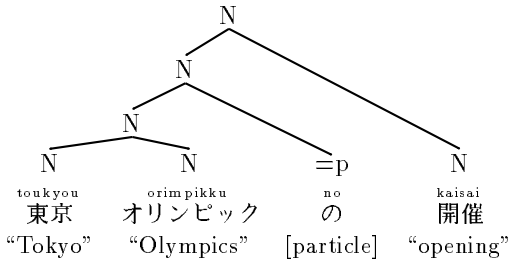
Consider, for example, the following two noun phrases:

(1)



"a water wheel with a diameter of 10 meters"

(2)



"the opening of the Olympics in Tokyo"

While the syntactic structures of the two noun phrases in Japanese are almost identical, their English translations are quite different in structure. Clearly, the information provided by a syntactic and morphological analysis of the Japanese phrases is not sufficient for accurate translations. However, if we consider the *meaning* of the constituents, we can, with the help of an ontology such as the SENSUS ontology (Hovy and Knight 1993, Knight and Luk 1994), achieve accurate translations into English. For example, SENSUS tells us that *Tokyo* is a location, and *the Olympics* are an event. With this knowledge, we can design a special rule that handles all combinations of locations and events. Similarly, knowing that 'diameter' is an attribute, and '10 meters' is a measurement, we can map a structure of the type `[attribute A] + [measurement M]` の `X` into a structure that expresses that X has the attribute A and A measures M (e.g., that the *water wheel* has a *diameter*, and that the *diameter* measures *10 meters*).

At the USC Information Sciences Institute, we are currently developing a system that tries to exploit such knowledge. The GAZELLE machine translation system aims at providing English translations with reasonable quality for unrestricted Japanese newspaper texts via semantic interpretation and subsequent text generation (Knight *et al.* 1995). At this stage, the system translates texts on a sentence-by-sentence basis. Figure 1 sketches the system. After some preprocessing, which ensures the correct character encoding of the input, etc., the input sentence is first segmented and tagged with morphological information, using the JUMAN segmenter/tagger developed at Kyoto University (Kyoto University 1997a). The JUMAN output is then parsed[1] and subsequently interpreted semantically (Knight and Hatzivassiloglou 1995b). The resulting intermediate representation is fed into a text generator, which produces a vast number of potential English renderings of the semantic content of the input sentence (Langkilde and Knight 1998a, 1998b). A statistical extractor picks the most likely rendering based on bigram and trigram models of English (Knight and Hatzivassiloglou 1995a).

[1] For parsing, we have used both a bottom-up chart parser with hand-crafted rules and a trainable shift-reduce parser (Hermjakob and Mooney 1997). Originally developed for English, the trainable parser was adapted to Japanese by Ulf Hermjakob, and trained on the Kyoto University Corpus (Kyoto University 1997c), a corpus of 10,000 parsed and annotated sentences from the *Mainichi* newspaper.

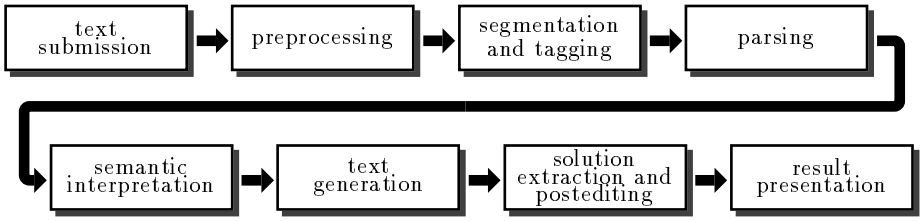


Fig. 1. The machine translation process.

A modular system architecture such as the one above allows us to use and evaluate different engines and strategies for solving particular aspects of the translation task while preserving and reusing as many of the remaining resources of the system as possible. Particularly in the areas of parsing and semantic interpretation, a modular approach is well justified: With more and more raw and annotated data such as corpora, annotated corpora, and treebanks becoming available (Marcus *et al.* 1993, Kyoto University 1997c), trainable parsers (e.g., Collins 1997, Hermjakob and Mooney 1997) have become a feasible and powerful alternative to hand-crafted systems. In contrast, semantically annotated data is still very rare and by no means sufficient as a basis for statistical approaches to semantic interpretation of texts, so that there is currently no serious alternative to a hand-crafted system for semantic interpretation. Even though one could argue — and we do support this argument — that accurate parses often cannot be achieved without the consideration of semantic criteria[2], we nevertheless claim that it is reasonable to perform parsing and semantic interpretation in separate steps. First of all, the amount of knowledge, rules, and processing needed for full-fledged semantic interpretation goes far beyond the requirements for accurate parsing. Therefore, adding semantic interpretation capabilities to an existing parsing system is labor-intensive and expensive, especially when statistical techniques used for parsing have to be coordinated with hand-crafted rules. And secondly, given the number of parsers that are becoming available for Japanese[3], it is highly desirable to be able to interpret output of different parsers with the same set of semantic interpretation rules.

The purpose of this paper is to describe how a system can be designed so that the semantic interpretation of parse trees does not depend on parser-specific

[2] A classical example is the sentence pair [*He ate spaghetti with Alfredo sauce*] and [*He ate spaghetti with a fork*], where the decision that the PP [*with Alfredo sauce*] is attached to N' [*spaghetti*], and the PP [*with a fork*] is attached to V' [*ate spaghetti*] can only be made on the grounds that *spaghetti* and *Alfredo sauce* are both some kind of food, whereas a *fork* is a tool for eating.

[3] We are currently aware of the KNP parser developed at Kyoto University (Kyoto University 1997b), and an HPSG parser developed at Tokyo University (Makino *et al.* 1998, Mitsuishi *et al.* 1998), in addition to the parsers currently used in our system (cf. Fn. 1).

characteristics such as the set of category symbols used, or the particular order of attachments within the parse tree. In the remainder of this paper, we first describe how semantic interpretation works and how it is implemented in our system. We then discuss the problems that arise when trying to interpret parse trees robustly, and how these problems can be overcome.

2 Semantic Interpretation

The idea of semantic interpretation is based on the notion of compositional semantics, i.e., the assumption that the meaning of an utterance can be computed by applying combinatorial rules to the meanings of the immediate constituents of the utterance. While semantic interpretation is usually implemented as a bottom-up algorithm, we will discuss the idea of compositional semantics in a top-down fashion in this section.

In a highly simplified framework, we assume that a sentence consists of one or more *basic propositions*. A basic proposition is the claim that a certain predicate or relation holds of, or holds between certain objects in the world. These objects are either sufficiently characterized by linguistic means (such as descriptive phrases that distinguish these objects from others in the respective domain of discourse), or obvious from the context.

For example, the sentence

(3) *The video lasts about eleven minutes, and [it] describes the company's activities.*

consists of two basic propositions:

(3a) *The video lasts about eleven minutes.*
(3b) *The video describes the company's activities.*

(3a) is the claim that the relation *last* holds between the video being talked about and a time span of eleven minutes: *last*(*video*, *eleven minutes*). (3b) claims that the relation *describe* holds between the video and the company's activities: *describe*(*video*, *the company's activities*).

The arguments of the predicates can be further analyzed as restricted variables: *The video* refers to that object X (in the respective domain of discourse) that has the property of being a video: $X \mid video(X)$[4]. Similarly, *the company's activities* are those Y that have the properties of (a) being activities, and (b) being somehow related[5] to the company: $Y \mid (activities(Y) \wedge be\_related\_to(Y,$ *the company*$)$. Finally, *the company* is that Z that has the property of being a company: $Z \mid company(Z)$. If we replace the arguments in (3b) by these restricted variables, we get the expression

[4] We do not consider the issue of quantifiers here.

[5] The specific character of this relation cannot be determined on syntactic grounds. We therefore leave the relation between the company and its activities unanalyzed here.

(3b') $describe(\mathrm{X} \mid video(\mathrm{X}),$
$\mathrm{Y} \mid (activities(\mathrm{Y}) \wedge be\_related\_to(\mathrm{Y}, \mathrm{Z} \mid company(\mathrm{Z}))))$

During semantic interpretation, a representation of the content of a sentence is built up by applying combinatorial rules first to the configurations of the individual words and then to the configurations of the higher-level constituents, whose meanings have been determined by previous interpretation steps. The meanings of the individual words are retrieved from a lexicon. As a rule of thumb, noun phrases (*referential constituents*) are interpreted as restrictions of variables, and verbs (*predicative constituents*) as predicates. It is assumed that the predicate–argument relations between the referential constituents and the predicative constituents of a sentence are reflected in its morphological and syntactic structure, so that the range of interpretations that a sentence may have is restricted by its morphological and syntactic properties. As we have seen above, these properties are not always sufficient for an accurate interpretation. In this case, knowledge about the world is utilized.

In the GAZELLE system, semantic interpretation is implemented in the following manner. The parser returns an annotated tree structure. Each node of the parse tree has a category label and a feature structure associated with it, which stores the various properties of the constituent represented by the node. Traversing the tree from the bottom to the top, and augmenting the feature structures associated with the nodes of the parse tree, the semantic interpretation engine gradually builds up an intermediate representation of the semantic content of the sentence, using information from the annotated tree structure, knowledge provided by a lexico-ontological database, and knowledge encoded in the semantic interpretation rules.

A feature structure is a bundle of attribute-value pairs (*features*), e.g. ⟨NUMBER, *sg*⟩, where NUMBER is the attribute and *sg* is its value. A value can be either an atomic feature structure, i.e. an empty bundle of features (which corresponds to an entity that cannot be described in terms of having particular properties), or it can be a complex feature structure itself. Mathematically speaking, features structures are rooted, directed graphs with labeled arcs. Figure 2 shows a feature structure that partially models the English word *sees* in a graph representation and as a so-called *attribute-value matrix* (AVM). The nodes of the graph represent values, the arcs attributes. Each attribute can be identified by its *path*, the sequence of arcs from the root node to the node representing its value. For example, in Fig. 2 the path SYN leads to a complex value representing the syntactic properties of the word *sees*, whereas the path SYN·PERSON points to the 'person' value of the word. By organizing features as a feature structure, features can be grouped and be passed along to higher nodes in the parse tree by declaring *path identity* between an attribute of the mother node and an attribute of a daughter node. Two paths are considered identical if they lead to the same node in the feature structure, i.e., if their values are token-identical. As a matter of convenience, we refer to a feature consisting of an attribute with the path X–Y–Z and its value as the X·Y·Z feature.

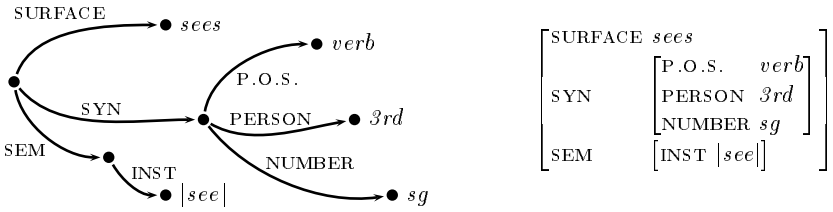


Fig. 2. A feature structure describing the word *sees* in a graph representation (left) and as an attribute-value matrix (AVM; right). Adapted from Germann (1998).

Each semantic interpretation rule consists of two elements: a context-free phrase structure rule which specifies the class of syntactic configurations that the rule should be applied to, and a set of instructions for the manipulation of the feature structures associated with the nodes. Syntactic configurations are identified by the labels of the mother and the daughter nodes in the parse tree. The basic mechanism for the manipulation of feature structures is *unification* (Shieber 1986, Moore 1989). Unification is an operation over two feature structures that returns a feature structure that contains (a) all features that occur in *either one* of the two feature structures, and (b) the unifications of all the features that occur in *both* of them. If the respective values of a feature that occurs in both structures are not unifiable, unification fails. Atomic feature structures are unifiable if they are either type-identical or in a subtype–supertype relationship with respect to a type hierarchy. In the latter case, the result of the unification is the more specific type. For example, the result of the unification of

$$\begin{bmatrix}\text{ATTR1} & \text{A}\\ \text{ATTR2} & \text{B}\end{bmatrix} \text{ and } \begin{bmatrix}\text{ATTR1} & \text{a}\\ \text{ATTR3} & \text{C}\end{bmatrix} \text{ is } \begin{bmatrix}\text{ATTR1} & \text{a}\\ \text{ATTR2} & \text{B}\\ \text{ATTR3} & \text{C}\end{bmatrix}$$

if *a* is a subtype of *A*. If *a* and *A* are not identical or in a subtype–supertype or supertype–subtype relationship, the unification of the two feature structures fails.

The semantic interpretation engine recognizes two kinds of unification instructions. *Conditioned* unification instructions (`<path1> =c <path2|value>`) succeed only if `<path1>` already exists. *Unconditioned* unification instructions (`<path1> = <path2|value>`) will create any path specified if it does not already exist. In order to distinguish the feature structures associated with the different parse nodes, path names are prefixed by the identifiers *x0, x1 ... xn*, where *x0* refers to the root node of the feature structure associated with the mother node in the configuration, and *x1 ... xn* refer to the root nodes of the feature structures associated with the daughter nodes from left to right.

The keyword **xor** in a rule introduces a hierarchical list of sets of instructions for the interpretation engine. The engine applies these sets of rules sequentially until unification succeeds. As soon as a set of instructions succeeds, the engine proceeds to the next configuration in the parse tree.

For example, the rule

```
(4) ((V -> N V) (*xor* (((x1 syn focus) =c wa)
                        ...
                       )
                       (((x1 syn case) =c ga)
                        ...
```

checks first whether the path X1·SYN·FOCUS already exists **and** whether its value is unifiable with *wa*. If these conditions are met, and unification also succeeds for the other instructions contained within the first set of instructions, the engine proceeds to the next parse node, otherwise it tries to execute the next set of instructions from the **xor** list.

Figures 3 and 4 illustrate the interpretation process for the sentence ビデオは約十一分。 (*bideo wa yaku zyuuippun. — "The video lasts about eleven minutes."*). The semantic content of each constituent is represented in the SEM feature of the feature structure associated with the respective parse node. For predicative constituents, the SEM·INST(ance) feature contains the predicate, and the values of the attributes ARG1 ... ARG*n* contain restrictions that characterize the arguments of this predicate. For referential constituents, the SEM·INST feature contains the 'head' restriction[6] of the respective argument. Various other features are used to store semantic modifications of constituents and other information.

The values of the INST features are usually concepts from the SENSUS ontology. In generation, INST values surrounded by vertical bars (|) in the intermediate representation are treated as concepts, INST values surrounded by double quotes are treated as literal strings. Each concept is associated with one or more English words that can express it.[7] Using concepts from an ontology does not only provide a wider range of expressiveness in the generator, but also allows us to use hierarchical information from the ontology during interpretation. For the sake of simplicity, all semantic values are represented as SENSUS concepts in the figures, even though they may look and be treated differently in the actual system.

3 Making Semantic Interpretation Robust

As mentioned above, the semantic interpretation rules identify syntactic configurations primarily by 'bare' phrase structure rules. In order to achieve robustness (coverage) in the system, we thus need at least one default rule for every conceivable combination of category symbols within a parse tree. Accuracy is then a matter of refining these rules. In order to achieve accuracy, we must have access to the properties of the constituents, i.e., we must know what features may occur in the feature structures associated with the parse nodes.

[6] A *blue car* is both something blue and a car, but in accordance with the syntactic structure of the phrase we consider *car* to be the 'head' restriction and *blue* a supplementary restriction of what the phrase can refer to.

[7] The lower areas of the SENSUS hierarchy are based on WordNet (Miller *et al.* 1990, Fellbaum 1998), version 1.5.

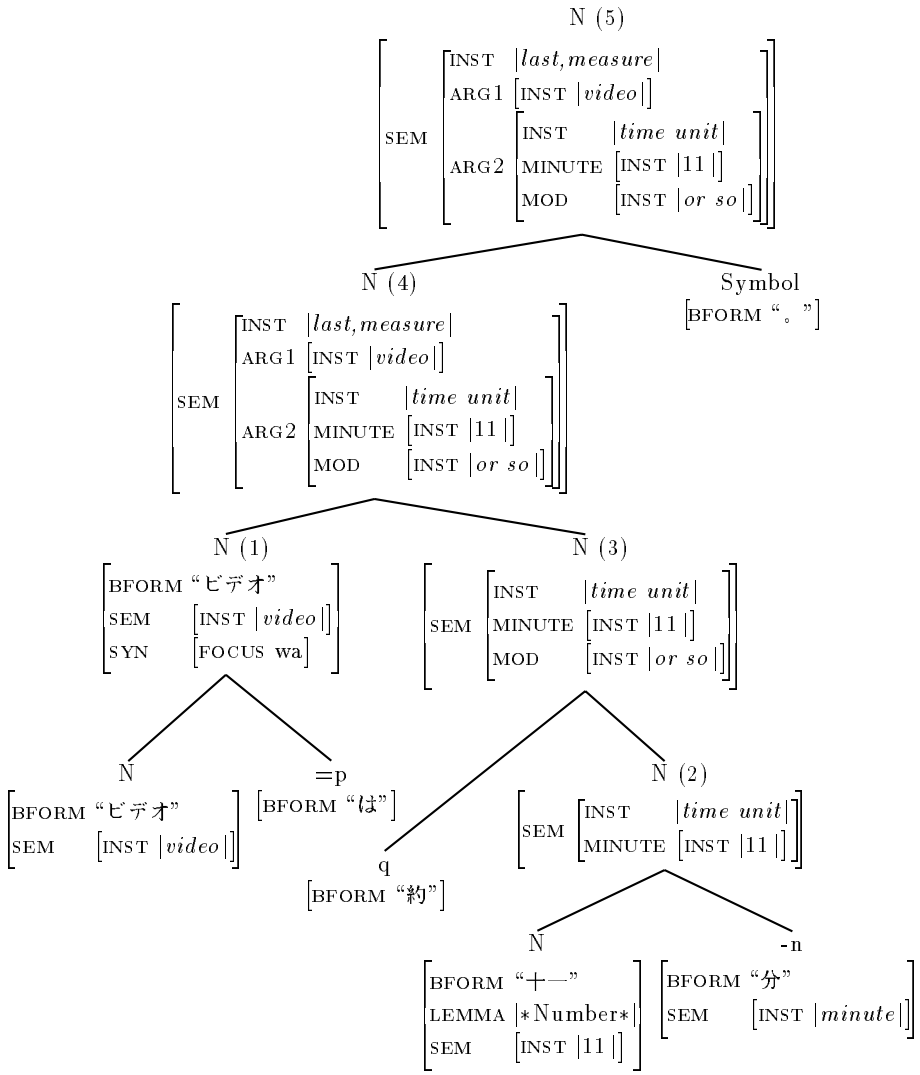


Fig. 3. The semantic interpretation process for the sentence ビデオは約十一分 (*bideo wa yaku zyuuippun — "The video lasts about eleven minutes."*). Information associated with the leaf nodes is provided by the segmenter/tagger and the lexicon. The feature structures associated with the higher nodes are computed in accordance with semantic interpretation rules (cf. Fig. 4). The numbers in parentheses after the node labels correspond to the numbers of the semantic interpretation rules in Fig. 4.

```
(1)   ((N -> N =p) (*xor* (((x2 bform) =c は)
                           ((x0 bform) =  (x1 bform))
                           ((x0 sem)   = (x1 sem))
                           ((x0 syn focus) = wa)
                          )
                          ...
                   ))

(2)   ((N -> N -n) (*xor* (((x1 lemma)      =c |*Number*|)
                           ((x2 bform)      =c "分")
                           ((x0 sem inst)   =  |time unit|)
                           ((x0 sem minute) =  (x1 sem))
                          )
                          ...
                    ))

(3)   ((N -> q N) (*xor* (((x1 bform)        =c "約")
                          ((x0 sem)          = (x2 sem))
                          ((x0 sem mod inst) = |or so|)
                         )
                         ...
                  ))

(4)   ((N -> N N) (*xor* (((x2 sem inst)  =c |time unit|)
                          ((x1 syn focus) =c wa)
                          ((x0 sem inst)  =  |last,measure|)
                          ((x0 sem arg1)  =  (x1 sem))
                          ((x0 sem arg2)  =  (x2 sem))
                         )
                         ...
                  ))

(5)   ((N -> N Symbol) (*xor* (((x2 bform) =c "。")
                               ((x0 sem) = (x1 sem))
                              )
                              ...
                       ))
```

Fig. 4. Excerpts from the semantic interpretation rules applied in the semantic interpretation process sketched in Fig. 3.

With regard to the implementation of a set of interpretation rules, this raises the following questions:

- Which category symbols does the parser use?
- In which configurations do these symbols occur?
- What features does the parser introduce, if any?

One way to answer these questions would be to analyze a sufficient amount of parser output in order to determine which symbols are used, in which configurations they occur, and so on. However, this approach has several disadvantages. First of all, it makes semantic interpretation vulnerable to changes in the parser. Every change in the set of category symbols used, or in the order of syntactic attachments will require adaptations of the semantic interpretation rules. Moreover, once the rules are tailored to one particular parser, switching to a different parser becomes expensive, since the set of rules will have to be ported to the new parser. Finally, depending on the parser, the set of additional features provided may vary considerably. While one parser may use a small set of category symbols and a rich vocabulary of additional features, others may make extensive use of different category symbols and not provide any additional features at all. Overall, this approach turns out not to be an attractive option.

Instead, we established a standardized parse tree format to which the parse trees are converted prior to semantic interpretation. Semantic interpretation then operates over trees in the standardized parse tree format. All features that serve as criteria for semantic interpretation are either introduced by the tree converter (the module that (a) replaces the category labels of the original parse tree with labels that conform to the standardized parse tree format, and (b) integrates additional information from the lexical database), or by the semantic interpretation rules themselves.

The standardized parse tree format is characterized by the following restrictions:

- We restrict the number of immediate constituents for each parse node to a maximum of three, that is, we allow unary ($A \rightarrow B$), binary ($A \rightarrow B\ C$), and ternary ($A \rightarrow B\ C\ D$) phrase structure rules. Ternary rules are restricted to cases of bracketing with symbols that have corresponding left-and-right counterparts such as quotes, parentheses, brackets, etc. All other rules must be unary or binary.
- We restrict the set of nonterminal symbols in the parse tree to the set of basic part-of-speech labels. There are no bar-level distinctions reflected in the labels. The unary sequence N – N' – NP in a parse tree will thus be converted to N – N – N. In our actual system, we distinguish 20 parts of speech,[8] as listed in Fig. 5.
- The label of the mother node in each configuration is unambiguously determined by the following principles. In unary configurations, the label of the mother node is the same as that of the daughter node. In ternary ones, the

[8] This part of speech inventory is based on Rickmeyer's (1995) analysis of Japanese.

part of speech	derivational			non-derivational
	lexeme	suffix	particle	
Verb	V	-v	=v	
Noun	N	-n	=n	
Adjective	A	-a	=a	
Nominal Adj.	K	-k	=k	
Adverb	M	-m		
Adnominal	D	-d		
Interjection	I			
Particle				=p
Prefix				q
Symbol				Symbol

Fig. 5. Category symbols used in our semantic interpretation rules.

category label of the middle node prevails. In binary configurations, the label of the mother node is determined by two factors. First, if the configuration contains a lexeme, the label of the mother node is a lexeme symbol. Secondly, the part of speech of the resulting label is determined by the part of speech of the last derivational symbol (cf. Fig. 5). For example, the configuration N *=v* will be reduced to V by the rule N *=v* $\leftarrow V$, and V N will be reduced to N, whereas the configuration *-v -a* will be reduced to *-a*, because it does not contain any lexeme. The label for the remaining nine possible binary configurations of non-derivational symbols (*=p*, *q*, *Symbol*) is determined by the label of the left-hand daughter node, except in the case $q \rightarrow$ *Symbol q*.

- The part-of-speech classification of the lowest nodes is based on the information from the segmenter/tagger, which we expect to be preserved in the feature structures associated with these nodes. Thus, dependence on information from the parse tree is restricted to knowing in which features of the feature structures this tagging information is stored.

With these restrictions, we have limited the number of possible configurations in a parse tree (including configurations that are completely bogus from a linguistic point of view such as a prefix suffixed to the preceding word) to $k^2 + 2k$, where k is the number of part-of-speech symbols. In our case, with 20 part-of-speech symbols, the number of possible configurations is hence limited to 440: 20 unary rules $X \rightarrow X$, 400 binary rules for every combination of two of the 20 category symbols, and 20 ternary rules $X \rightarrow$ *Symbol* X *Symbol* for bracketing phenomena.[9] Figure 6 shows the complete set of phrase structure rules for the subset { V, N, -v, Symbol } of our set of category symbols.

Since we rely on the parser only for the bare tree structure, while lexical information comes from the segmenter/tagger and the lexicon, and since the

[9] For comparison, our current parser employs over 85 category labels, which would increase the number of possible configurations to well over 7000. Of course, most of them will never show up in 'real life', but it is difficult to determine in advance which configurations may or may not occur.

unary rules

V → V	N → N	-v → -v	Symbol → Symbol

binary rules

V → V V	V → V Symbol	N → V N	N → Symbol N
V → N V	V → Symbol V	N → N N	-v → -v -v
V → V -v	-v → -v Symbol	N → -v N	-v → -v Symbol
V → -v V	-v → Symbol -v	N → N Symbol	-v → Symbol -v

ternary rules

V → Symbol V Symbol	-v → Symbol -v Symbol
N → Symbol N Symbol	Symbol → Symbol Symbol Symbol

Fig. 6. The complete set of phrase structure rules for the categorial vocabulary { *V, N, -v, Symbol* }. Our actual set of category symbols consists of 20 symbols.

new parse node labels are assigned to the nodes deterministically based on the principles stated above, we gain a high degree of independence from parser-specific information.

4 Coordinating Parsing and Semantic Interpretation

In order to coordinate parsing with semantic interpretation, the following steps have to be taken:

- If the parser's output does not conform to the structural requirements of our system, the parse trees have to be 'binarized'. This could also be considered a separate 'microparsing' step after some 'macroparsing' accomplished by the first parser. It is currently implemented as part of the parsing process but could also be integrated in the tree conversion step and take place after the category symbols of the lowest nodes in the parse tree have been replaced by our category symbols.
- The category symbols in the parse tree have to be replaced by the category symbols used in semantic interpretation, based on information from the segmenter/tagger, which is preserved in the parse tree, and on the label assignment principles for higher nodes.

Figure 7 illustrates the flow of processing: After parsing, the parse tree is first binarized. The tree converter subsequently replaces the original category labels by labels that conform to the standardized parse tree format and adds information from the lexicon. The standardized, enriched parse tree is then fed into the semantic interpretation engine.

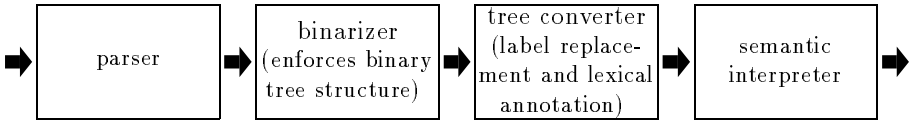


Fig. 7. From parsing to semantic interpretation: Parse trees are first converted to binary format, then category labels are replaced and lexical information is added to the parse nodes. This enriched structure is subsequently fed into the semantic interpretation engine.

5 Summary

We have shown how a semantic interpretation system can be set up to handle output from various parsers by replacing the category symbols provided by the parsers with its own symbols, and by relying only on information provided by the segmenter/tagger and a lexical knowledge base. The advantage of this approach is that the set of semantic interpretation rules can be developed and maintained independently from any specific parser. The costs of the adaptation of a new parser and of the coordination between the parser and the semantic interpretation module are reduced to the creation of mapping tables for symbols on the part-of-speech level and the binarization of the parser's output, if necessary. The modular setup of the system allows us to integrate additional resources as they become available without having to change the actual knowledge base for semantic interpretation.

6 Acknowledgments

The GAZELLE machine translation project is funded by the US Government under contract MDA904-96-C-1077. I am very grateful to Kevin Knight and Daniel Marcu for various comments on earlier versions of this paper.

References

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Christiane Fellbaum (Ed.). 1998. *WordNet*. M.I.T. Press, Cambridge, MA.

Ulrich Germann. 1998. Visualization of protocols of the parsing and semantic interpretation steps in a machine translation system. In *COLING-ACL '98 Workshop on Content Visualization and Intermedia Representations (CVIR '98)*.

Ulf Hermjakob and Raymond J. Mooney. 1997. Learning parse and translation decisions from examples with rich context. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Eduard Hovy and Kevin Knight. 1993. Motivating shared knowledge resources: An example from the Pangloss collaboration. In *Proceedings of the Workshop on Knowledge Sharing and Information Interchange (IJCAI)*.

Kevin Knight, Ishwar Chander, Matthew Haines, Vasileios Hatzivassiloglou, Eduard Hovy, Masayo Iida, Steve K. Luk, Richard Whitney, and Kenji Yamada. 1995. Filling knowledge gaps in a broad-coverage machine translation system. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Kevin Knight and Vasileios Hatzivassiloglou. 1995a. Two-level, many-paths generation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kevin Knight and Vasileios Hatzivassiloglou. 1995b. Unification-based glossing. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Kevin Knight and Steve K. Luk. 1994. Building a large-scale knowledge base for machine translation. In *Proceedings of the National Conference on Artificial Intelligence*.

Kyoto University. 1997a. Juman. `http://www-lab25.kuee.kyoto-u.ac.jp/nl-resource/juman.html`. As of 05/22/1997; URL valid on 06/11/98.

Kyoto University. 1997b. KNP. `http://www-lab25.kuee.kyoto-u.ac.jp/nl-resource/knp-e.html`. As of 05/28/1997; URL valid on 08/24/98.

Kyoto University. 1997c. 京都大学テキストコーパス Version 1.0 (Kyoto University text corpus, version 1.0). `http://www-lab25.kuee.kyoto-u.ac.jp/nl-resource/corpus.html`. As of 09/23/1997; URL valid on 06/11/98.

Irene Langkilde and Kevin Knight. 1998a. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL)*.

Irene Langkilde and Kevin Knight. 1998b. The practical value of n-grams in generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*.

Takaki Makino, Minoru Yoshida, Kentaro Torisawa, and Jun'ichi Tsujii. 1998. LiLFeS – towards a practical HPSG parser. In *Proceedings of the 36th Annual Meeting of the Association for Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL)*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinskiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography*, 3(4). `ftp://ftp.cogsci.princeton.edu/pub/wordnet/5papers.ps`.

Yutaka Mitsuishi, Kentaro Torisawa, and Jun'ichi Tsujii. 1998. Underspecified Japanese grammar with wide coverage. In *Proceedings of the 36th Annual Meeting of the Association for Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL)*.

Robert C. Moore. 1989. Unification-based semantic interpretation. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jens Rickmeyer. 1995. *Japanische Morphosyntax*. Groos, Heidelberg.

Stuart M. Shieber. 1986. *An Introduction to Unification-Based Approaches to Grammar*. CSLI, Stanford, CA.