Knowledge Representation Support

Sergei NIRENBURG and Lori LEVIN

ABSTRACT: The representation of world knowledge in a machine translation system is discussed, especially in terms of the interlingua text that is adopted in KBMT-89. While knowledge about the world is at the core of a knowledge-based machine translation program, it cannot be extracted from a text without knowledge about how languages encode it syntactically. The paper therefore also presents the grammar formalism with which language-specific syntactic knowledge is presented. Details are presented of our version of unification-based grammar, its basis in linguistic theory and its role in a knowledge-based machine translation system.

KEYWORDS: constituent structure (c-structure), functional structure (f-structure), interlingua text (ILT), Lexical Functional Grammar, microtheories, unification

1. THE INTERLINGUA TEXT

The use of world knowledge is widely and correctly perceived as the salient property of a knowledge-based machine translation system. Many researchers, however, do not proceed to a detailed discussion of the peculiarities of the use and representation of world knowledge in such systems. This leads to a number of misconceptions and inaccurate assessments regarding the amount and nature of research and development necessary for producing a large natural language processing system.

First, the relationship between the interlingua text (ILT — the results of a comprehensive analysis of an input text) and the concept lexicon (the domain model) is typically not well understood, and the important differences between the two not well appreciated — either with respect to the nature of their elements or to the differences in the knowledge representation languages used for their specification.

Second, uncertainty prevails over the status and utility of speech situation or pragmatic knowledge encoded or inferrable from the source language text. We believe that the pragmatic aspects of a source language text are an indispensable component of its overall meaning, and have undertaken to represent this kind of information in the KBMT-89 ILTs and to realize it in our target languages. In the following subsections we discuss the types of knowledge we cover in our system.
1.1. Varieties of World Knowledge in a Knowledge-Based MT System

The concept lexicon contains an ontological model, which provides uniform definitions of basic categories (such as objects, event-types, relations, properties, episodes and many more) used as building blocks for descriptions of particular domains. This world model is relatively static and organized as a multiply interconnected network of ontological concepts.

An interlingua text (ILT) is a representation of actual events that occurred in the world, as reported in a source language text. ILTs can be represented as networks of event- and state-tokens, complete with their participants and connected on a number of causal, temporal, spatial and other propositional links. These event-tokens, or episodes, are indexed through their corresponding ontological types. We do not produce the representations of these actual events, states, and so forth on the fly. Our main device in this process is the instantiation of tokens of appropriate concept types from the concept lexicon. Thus, the representation of the meaning of a text is understood as creating an episode. The participant roles in the episodes can be either instantiations of object and event types in the ontology or references to already existing named instances (such as, for instance, IBM or The Washington Post), permanently stored in a non-ontological part of the domain model. In principle, however, the episodes are relatively much more dynamic than the ontology — they are "forgotten" as well as merged with other episodes in memory.

Figure 1 illustrates the distinctions among the various types of world knowledge that have to be used in a comprehensive natural language processing system.

1.2. Integration of Discourse and Propositional Knowledge

It is a common and perhaps epistemologically required phenomenon in science that research concentrates on a particular facet of a complex problem, either theory- or application-oriented, often to the exclusion of other, equally important problems. NLP is, of course, no exception. Thus, specific projects have been devoted to such components of an NLP system as the design of knowledge representation languages in which to describe the ontology of an application (sub)world (e.g., Wilensky, 1984; Brachman and Schmolze, 1985); languages to describe events that take place during the use of a knowledge-based system (e.g., Kolodner, 1984); and languages that help capture such textual peculiarities as cohesion forces, characteristics of participants in the speech situation, thematic information, etc. (e.g., Tucker et al., 1986).

Other types of projects have dealt with using one or more of the above languages to acquire actual compendia of knowledge, viz., describing the ontologies of application (sub)worlds (e.g., Lenat et al., 1985; Hobbs, 1986); collecting caches of events in an application (sub)world; describing the discourse/pragmatic structure of texts or dialogues (e.g., Grosz and Sidner, 1985);
and writing lexicons that connect lexical units of a natural language with elements of the above ontologies, histories and structures (e.g., Nirenburg and Raskin, 1987).

Yet another family of projects has involved designing the control structures of computer programs that, making use of the above knowledge, translated NL inputs into representations in the above artificial languages or vice versa (e.g., Fass, 1986; and Jacobs, 1985).

An important point is that, while even these tasks are universally recognized as extremely complex, the extent of meaning extraction and manipulation in application-oriented NLP has been typically limited. Some systems limit themselves to syntactic analysis only. Others add the analysis of “logical form.” Those dealing with pragmatic/discourse-related issues are typically miniature systems designed specifically to test certain discourse/pragmatic hypotheses. The relatively shallow and/or fragmented approach to natural language understanding was justified not only in terms of the overall task’s complexity, but also by the supposed requirements of certain applications. One example would be the transfer-based machine translation systems of the 1970’s. But note also the relatively shallow requirements for the semantic description of input into many NL generation systems (see, e.g., Marcus, 1987, for a critique of lexical selection in current NL systems). At the same time, some NLG systems (cf. McKeown, 1985) attend to the discourse structure of the input. Some are even coupled with a separate discourse manager in a particular application (cf. the Counselor project at the University of Massachusetts (e.g., Pustejovsky, 1987)). This attitude of workers in generation may be partially attributable to the nature of the typical environments for their systems: usually, natural language front-ends to various databases or other dialog systems. With other applications, generation will have to attend more closely to all types of meaning. There are, however, a number of applications in which progress depends on a simultaneous use of more kinds of knowledge than before; specifically, progress depends on integrating the propositional and nonpropositional knowledge about concept types, concept tokens and text units.

1.3. Representative Classes of Discourse Knowledge

Knowledge requirements for knowledge-based machine translation are usually perceived as involving representations of propositional knowledge, similar to the ones described above. However, to ensure adequate levels of understanding of input texts, it is necessary to extract and overtly represent nonpropositional pragmatic and discourse meaning about concept instances. Pragmatic meaning is usually understood as pertaining to the attitudes of the speaker/hearer to the set of uttered propositions. Discourse meaning reflects the (language-dependent) rules of combining separate utterances into coherent texts.

Problems whose solutions depend, in whole or in part, on nonpropositional knowledge are not esoteric, but quite pervasive in meaning analysis. These problems include referential ambiguity, thematic structure of text, understanding of indirect speech acts and interpretation of discourse cohesion markers, such as moreover or in any case, etc.

1.4. Interlingua Text — Combining Concept Tokens into Networks

This subsection is devoted to a discussion of the syntax of interlingua text. Unlike a natural language text, an interlingua text is not linear. It is a potentially very complex network of interlingua units of sentence size, linked by interlingua cohesion markers. An interlingua text is represented as a frame that serves as an index for the interlingua clauses that compose the text. ILLT clauses are represented as frames. The ILLT clause is the place where event instances (tokens) are put into their modal, discourse and speech-situation context. Event and object tokens that appear in ILLTs are produced by obtaining tokens of the appropriate concept types in the domain model and augmenting them with various property values identified during source-language text analysis in translation. It follows that the slots whose values express a component of contextual propositional meaning (e.g., negation) or any type of nonpropositional meaning (including discourse meaning) appear only in ILLT frames for event and object tokens, and not in the domain model.

Every token of an interlingua concept stands in the is-token-of relationship to its corresponding type. The frame for a type and the frame for a corresponding token are, however, not identical in either structure or semantics, even though they share some slot names. There are, of course, regular correspondences between units of the concept lexicon (the domain model) and ILLT. Property values in concept tokens are typically elements or subsets of data types listed as ontological constraints in the corresponding slots of the concept lexicon. Thus, for instance, the color property slot in the concept lexicon frame for flower can be occupied by a list white yellow blue red purple .... But an ILLT frame rose11, which is a token of a subclass of the class flower, will have the value red as the contents of its color slot.

The nonpropositional knowledge derived from the natural language input is overtly represented in interlingua text. Interlingua sentences, clauses and events carry this information. The overt representation of interlingua text units, not just as a sequence of events, is one of the technical innovations of our project. Note that for representing nonpropositional relations (such as discourse or focus) we use the same knowledge structures as those traditionally used for representing the propositional content of natural-language input.

The formalism used for representing ILLTs is a kind of semantic network. The types of nodes in this network include:

- ILLT clauses, which are nodes that represent the combined propositional and nonpropositional meanings of a “unit” utterance; the representations of a proposition, a speech act and a focus value are combined at this
level. Natural language sentences are syntactically defined entities and can contain more than one clause.

- **ILT-propositions**, which typically represent the meaning of an action or a state; modality and aspect are specified in this node.
- **ILT-roles**, which typically represent a conceptual object connected to a proposition through a well-defined case link.

The links in this semantic network are of the following kinds:

- Role relations (agent, theme, etc.) that connect roles to propositions;
- Temporal relations that introduce a partial ordering of the temporal values of propositions and speech acts;
- Causal relations (causality and enablement) that are defined on propositions. These mark certain preconditions and effects that have been mentioned elliptically in the source language text (in a machine translation generator) or have been explicitly mentioned in the text plan (in interface-supporting generators);
- Discourse cohesion relations that are defined on clauses and represent the rhetorical structure of the text to be generated; and
- Focus-related links (given and new) that connect a clause node with some of its component nodes.

Computer programs require formal symbolic representations of the above networks. However, for human inspection it is desirable to be able to represent these networks graphically. We have designed a special graphics editor for this purpose.

### 1.5. Interlingua Text and The Concept of Microtheories

Significant progress has been made recently in natural language processing with respect to the theories of syntax. Semantic and pragmatic phenomena have traditionally been less amenable to computational analysis. It does not seem likely then that an integrated semantic/pragmatic theory that covers all lexical and compositional phenomena, as well as the various pragmatic considerations, will be formulated in the near future. This assessment becomes even more plausible if one recognizes the need to provide heuristics for automatic recognition of the multiple meaning facets of natural language texts as a part of the theory. At the same time, linguistics has accumulated a significant body of knowledge about the various semantically laden phenomena in natural languages.

The foregoing suggests that one of the more productive ways of building a comprehensive computational model of human language understanding and generation behavior would be to develop a large number of microtheories; each would treat a particular linguistic phenomenon in a particular language or group of languages and then provide a computational architecture that in turn would integrate the operation of all the modules. Thus, one can envisage microtheories of time, modality, speech act, causality, etc.

In what follows we will illustrate how an integrated representation scheme such as our ILT can be used to capture the phenomena belonging to the various microtheories. The structure of the interlingua text as described is a superset of the actual interlingua text used in KBMT-89. In this project, certain ILT frames, slots and fillers will not be used because the constraints on the application render such microtheories spurious. For instance, in KBMT-89 there has been no need for the text frame, since we are working with single sentences; and there seem to be no sentences with modality ‘desirable’; and the information about focus is not used to its maximum capacity. In addition, some of the linguistic microtheories have not yet been developed sufficiently to allow computational treatment. Therefore, some simplifications occur in the KBMT-89 representations. For example, only a subset of the quantifiers is actually used, and symbolic rather than numerical values have been assumed for most of the attributes and intensifiers, etc. The specification below can be used as the semantic and pragmatic representation component in machine translation and general NLP systems that process multilingual text and require a finer grain size of description than the one found sufficient for our research.

### 1.6. Meaning and Representation

In this subsection we present some of the decisions made with regard to the representation of natural-language meaning components.

#### 1.6.1. Clause: Where the Propositional Meets the Nonpropositional

\[
\text{clause ::= } \{ (\text{id <symbol>} ) \\
(\text{proposition <event-token>}) \\
(\text{modality modality-set}) \\
(\text{speech-act speech-act}) \\
(\text{focus}) \\
(\text{scope text | clause+ | proposition+ | role+}) \\
(\text{given text | clause+ | proposition+ | role+}) \\
(\text{new text | clause+ | proposition+ | role+}) \\
(\text{discourse-relation discourse-expr}) \}
\]

The **focus** slot refers to the thematic structure of a component of text. Its **scope** subslot is needed to determine the “background” against which given and new information are determined. In some cases there could be more than one focus nucleus inside one (usually compound) sentence, as in (1):

(1) When it comes to work, nobody beats him, but when it comes to relaxation, Jack could learn something from Fred.
One plausible analysis of the thematic structure of this sentence would be that there are two focus nuclei, corresponding to the two coordinate clauses, with both the subordinate clauses serving as values of given and both main clauses as values of new. Note that other analyses are possible. (A more complete microtheory of focus will involve nested focus scopes and a taxonomy of given and new values.) Thus another treatment of the thematic structure of this text would involve four focus nuclei, corresponding to each clause, with the following distribution of the values of given and new:

<table>
<thead>
<tr>
<th>Scope</th>
<th>Given</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clause1</td>
<td>undefined</td>
<td>work</td>
</tr>
<tr>
<td>Clause2</td>
<td>beats kim</td>
<td>nobody</td>
</tr>
<tr>
<td>Clause3</td>
<td>undefined</td>
<td>relaxation</td>
</tr>
<tr>
<td>Clause4</td>
<td>learn</td>
<td>from Fred</td>
</tr>
</tbody>
</table>

1.6.2. Basic Propositional Components: Propositions and Roles

```
proposition ::= (((id <symbol>)
  (is-token-of event-type | property-type)
  (property value)*)
  (proposition-quantifier p-quantifier-set)
  (temporal-relation temporal-exp)
  (spatial-relation spatial-exp)
  (subworld computer-world | medicine-world | ...)
  (aspect
    (phase begin | begin-end | end)
    (duration <positive-real> | indefinite)
    (iteration (times integer) | (lap <integer>))
    (role-list (role-type role)*)
  ))

role ::= (((id <symbol>)
  (referent <symbol>)
  (description
    (is-token-of object | property)
    (property value)*)
  (role-quantifier r-quantifier-set)
  (subworld computer-world | medicine-world | ...)
  (set-count integer))
  (role-type agent | theme | patient | source | instrument | destination | ...)
```

All propositions are tokens of events or properties (subsuming states). All roles are tokens of objects, events or properties. Property values listed in their corresponding type frames in the concept lexicon serve as default value ranges for the proposition and role slots, unless concrete values are listed in the proposition. Thus, if the concept lexicon frame for *rose has the slot (color (value pink red white)), this range will be assumed as default for the instance rose25, unless the frame for it will explicitly include a slot like (color (value white)). The ILT formalism as such does not preclude the substitution of values from outside the ranges specified in the concept lexicon, that is, having, for instance, (color (value blue)) in the example above. The decision to allow this should be made depending on whether the analysis module in the system is expected to process metonymy and metaphor.

The value of subworld effectively works as a "view" mechanism that allows a knowledge-based machine translation system to use only that part of the concept lexicon which actually describes the current domain, while having additional domain descriptions in the same lexicon. Note that markers such as this constitute the first (and often only) semantic constraints used in earlier MT systems for word sense disambiguation.

The aspectual properties of events are factored out of the basic set of concept lexicon events. For each event type used as the head of a proposition we determine the three aspectual properties phase, duration and iteration. This approach helps keep in check the number of primitive concepts in the concept lexicon. The aspect microtheory we are using in the ILT was developed by Pustejovsky (1988) and restructured in computational terms, as described in Nirenburg and Pustejovsky (1988).

1.6.3. Speech Acts

```
speech-act ::= (speech-act
  (type statement | definition | request-info | request-action)
  (direct? yes | no)
  (speaker object)
  (hearer object)
  (time temporal-exp)
  (space spatial-exp))
```

The taxonomy of speech acts is underdeveloped in this microtheory. The reason is not the difficulty of defining additional types of speech acts but rather the difficulty in formulating heuristic rules that would allow the analyzer to distinguish between, say, a promise and a threat. The value of the time of the speech act contributes to the representation of the meaning of tense in the source language sentence.

1.6.4. Quantification

```
r-quantifier-set ::= n, 0 < n < 1>
```
Thus, the value 0.8 means 'most'; 1 means 'all' or 'each'; 0 means 'no'; 0.2 means 'few.' This microtheory is definitely underdeveloped, and will be extended. One direction of improvement will be taking into account the concept of argumentative scales which would allow us to distinguish between (2a) and (2b).

(2) (a) I have little money (− only $10)
(b) I have a little money (− almost $10)

Even though the numerical values for the quantifiers little and a little will be identical (e.g., 0,2), the meaning of the two phrases is quite different. Speaking from an application-oriented standpoint, however, having an involved theory of quantification is relatively much less important than one would be led to believe in a reading of recent semantic literature. So,

p-quantifier-set ::= negation | universal | existential | ...

may be representable in the same manner as r-quantifiers, e.g., negation is 0, universal is 1, existential is any number in between, corresponding to a range from 'almost never' to 'very seldom' to 'seldom' to 'sometimes' to 'often' to 'usually' to 'almost always'.

1.6.5. Space

spatial-expr ::= absolute-space | relative-space

absolute-space ::= (space
(coordinate1 real)
(coordinate2 real)
(coordinate3 real))

relative-space ::= (spatial-operator object+)

spatial-operator ::= left-of | equal | between | in | near | above | ...

1.6.6. Time

temporal-expr ::= absolute-time | (time-operator time time)

time-operator ::= before | after | at | during

time ::= absolute-time | relative-time

absolute-time ::= number | range

relative-time ::= proposition.time

The set of temporal operators is simpler in this microtheory than in the set used by Allen (1984) because it is sufficient for our subworld.

1.6.7. Modality

modality-set ::= real | desirable | conditional | probable | possible | necessary | ...

Our microtheory of modality reflects the paucity of modal meanings in the sublanguage of computer hardware. If the system is to be used in a subworld rich in modal expressions, the microtheory would require a significant amount of improvement, especially with respect to heuristic rules.

1.6.8. Discourse Cohesion

discourse-expr ::= (cohesion-marker clause clause)*

cohesion-marker ::= condi | expansion | similar | generalization | contrastive | digression | ...

Discourse cohesion markers are used to represent nonpropositional relations among clauses and sentences. The microtheory above is based on Tucker et al. 1986. A number of other theories of discourse cohesion have been suggested (e.g., Groz and Sidner, 1985) and in future work we plan to incorporate some of these findings.

1.7. An Example of an ILT

The particular tasks of KBMT-89 as well as the current unavailability of certain microtheories have constrained the use of the various constructs in the ILT. This is expected, as should be clear to anybody who has attempted to implement a theoretical construct as a component of a large AI-oriented software system.

Consider an example of an actual KBMT-89 ILT: The Japanese sentence at input is given in (3a), the English in (3b).

(3) (a) Kakou souni no setzuko ga syuuuryou si ta ra sisutemun yunitto to purin-taa no den-gen-suitti ga 'kiru' gawa ni nattte iru koto okakunin-si te kudasai.

(b) Confirm that the power unit switches of the system unit and the printer are in the 'off' position when the connection of each device is complete.
In the sample ILT that follows, a number of frame slots are included as indices (e.g., all the –id slots). The asterisk marks a reference to a concept from the domain model, while the ampersand marks a reference to a value from a value set in the domain model:

(make-frame clause1
  (ilt-type (value clause))
  (clauseid (value clause1))
  (propositionid (value proposition1))
  (discourse-cohesion-marker (value (conditional clause2)))
  (speechactid (value speech-act1)))

(make-frame proposition1
  (ilt-type (value proposition))
  (propositionid (value proposition))
  (clauseid (value clause1))
  (aspect (value aspect1))
  (complete (value yes))
  (is-token-of (value *connect*))
  (agent (value unknown))
  (theme (value role2))
  (time (value time1))))

(make-frame role2
  (ilt-type (value role))
  (clauseid (value clause1))
  (is-token-of (value *device))
  (r-quantifier (value universal))
  (reference (value definite)))

(make-frame aspect1
  (ilt-type (value aspect))
  (clauseid (value clause1))
  (phase (value end)))

(make-frame speech-act1
  (ilt-type (value speech-act))
  (speech-act (value statement))
  (direct? (value yes))
  (speaker (value author))
  (hearer (value reader))
  (time (value (before time1))))

(make-frame clause2
  (ilt-type (value clause))
  (clauseid (value clause2))
  (propositionid (value proposition2))
  (speechactid (value speech-act2)))

(make-frame proposition2
  (ilt-type (value proposition))
  (propositionid (value proposition2))
  (clauseid (value clause2))
  (aspect (value aspect2))
  (is-token-of (value *confirm)))

(agent (value role3))
(theme (value clause3))
(time (value (after time1))))

(make-frame role3
  (ilt-type (value role))
  (clauseid (value clause2))
  (is-token-of (value *reader))
  (reference (value definite)))

(make-frame aspect2
  (ilt-type (value aspect))
  (clauseid (value clause2))
  (phase (value end)))

(make-frame speech-act2
  (ilt-type (value speech-act))
  (speech-act (value command))
  (direct? (value yes))
  (speaker (value author))
  (hearer (value reader))
  (time (value (before time1))))

(make-frame clause3
  (ilt-type (value clause))
  (clauseid (value clause3))
  (propositionid (value proposition3))
  (speechactid (value speech-act3)))

(make-frame proposition3
  (ilt-type (value proposition))
  (propositionid (value proposition3))
  (clauseid (value clause3))
  (aspect (value aspect3))
  (is-token-of (value *discrete-position))
  (range (value off-position))
  (domain (value role4))
  (time (value (after time1))))

(make-frame role4
  (ilt-type (value role))
  (clauseid (value clause3))
  (is-token-of (value *set))
  (member (value *power-switch))
  (belongs-to (value role5)))

(make-frame role5
  (ilt-type (value role))
  (clauseid (value clause3))
  (is-token-of (value *set))
  (member (value *system-unit *printer))
  (type (value conjunction)))

(make-frame aspect3
  (ilt-type (value aspect))
  (clauseid (value clause3))
  (phase (value end)))
1.8. Historical Note

The approach to knowledge representation we described is an immediate descendant of that in Nirenburg et al. (1987). The other direct influence is Carbonell et al. (1981). Some ideas about representation (in particular, the nature of a frame slot filler, dictated by the need to prepare the data for eventual preference processing) are related to work by Wilks (e.g., 1975). In some other respects (for instance, in treating properties as full-fledged concepts), we agree with Wilensky (1984). Ultimately, our approach belongs to the family of frame-based non-truth-value-oriented empirical semantic and pragmatic knowledge representation systems.

2. GRAMMAR FORMALISMS

While knowledge about the world is at the core of a knowledge-based machine translation program, it cannot be extracted from a text without knowledge about how languages encode it syntactically. This language-specific syntactic knowledge includes, for example, how a verb’s arguments are distinguished from each other by case markings and word order and how tense and aspect are represented by auxiliary verbs and inflectional endings. In this section we describe the grammar formalism with which we represent language-specific syntactic knowledge.1

We have chosen to use a unification-based grammar formalism for syntactic analysis. This formalism has its roots in the work of Kay (1985) and the theory of Lexical Functional Grammar (LFG) described by Kaplan and Bresnan (1982). It is also related to other unification-based grammars, such as those found in the PATR-II system (Shieber, 1986) and the theory of Head Driven Phrase Structure Grammar (Pollard and Sag, 1987); nevertheless, our rules and output structures adhere most closely to those of LFG.

Syntactic parsing is part of analysis of the source language text.2 If the

![Figure 2. An example of English constituent structure.](image)

parser used only the syntactic grammar and syntactic features from the lexicon, it would produce a feature structure containing syntactic information such as tense, number, case, and person, and grammatical functions such as subject and object. However, parsing is actually concurrent with mapping rule interpretation. As the parser builds f-structures, the mapping rule interpreter takes lexical items from the f-structures and maps them onto concepts in the ontology (concept lexicon). The mapping rule interpreter also maps grammatical functions (e.g., subject, object and modifier) from the f-structure onto slots (e.g., agent, domain and beneficiary) in the concept lexicon. The augmentor takes parser output as its input and produces ILTs as described in the previous section.

2.1. Two Types of Syntactic Structure

A grammar describes two structures for each sentence, a constituent structure (c-structure) and a functional structure or feature structure (f-structure) (Kaplan and Bresnan, 1982). A c-structure is a tree generated by a context-free grammar. The nodes in the tree are parts of speech like N (noun) and V (verb) or labels for syntactic constituents like NP (noun phrase) and VP (verb phrase). The leaf nodes of the tree are words or characters of a sentence. In our version of unification-based grammar, there are no empty leaf nodes. Figures 2 and 3 show c-structures in English and Japanese.

![Figure 2. An example of English constituent structure.](image)

In unification-based grammars, c-structures represent only surface word order and constituency. Therefore, there are no deep structures or rules which transform one c-structure into another and all possible surface structures, including passive sentences and wh-questions, are generated by the phrase structure rules.

1 For the sake of expository convenience, only the analysis grammar will be discussed here. Some minor differences between the analysis and generation formalisms are mentioned in the paper by Gates et al. that immediately follows this article.

2 The KMBT-89 parser and mapping rule interpreter are described in the paper "Analysis" by Morrison, Kee and Goodman in Part II of this issue.
Figure 3. An example of Japanese constituent structure.

F-structures are sets of pairs of features and values. The features are usually grammatical functions like SUBJ (subject) and OBJ (object) or names of grammatical features such as TENSE, NUMBER and PERSON. ROOT is a special feature which enables a lexical item to be mapped onto a concept. Usually, the value of ROOT is the morphological root of a word. The values of the features can be atomic symbols or feature structures. For example, in the (simplified) Japanese f-structure below the value of the MOOD feature is the atom IMPERATIVE, but the value of OBJ is an f-structure which consists of features and values. Inside the f-structure, which is the value of the OBJ function, the feature WH has the value ‘-‘. A value of a feature can also be a disjunction of values or a set of values.

{
(PADJUNCT ((PART KARA) (CAT N) (SEM NIL)
(ROOT DISUKETTODORAIBU)))
(OBJ ((WH -) (CASE O) (CAT N) (SEM NIL) (ROOT DISUKETTO)))
(CAUSATIVE -) (OBJ-CASE O) (SUBJ-CASE GA) (PASSIVE -)
(SUBCAT TRANS) (FORMAL +) (TIME PRESENT) (MOOD ((ROOT IMP))))
(VTYPE V-5DAN-KI) (CAT V) (ROOT TORINOKO))

For the most part, our grammars draw from the following set of grammatical functions (based on Bresnan, 1982); each feature and name is followed by one or two sentences in which an italicized string exemplifies the feature. However, we often deviate from this list because of vagaries of implementation.

SUBJ — subject: An expansion unit is attached. Do you have an expansion unit?

OBJ — object: Disconnect the expansion unit cable from the system unit.

OBJ2 — second object: Chapter 3 gave you a better understanding of how to handle diskettes.

See Tomita and Knight, 1988, for a description of how pseudo-unification f-structures differ from full-unification f-structures.

SCOMP — a complete sentential complement: Verify that it is the diagnostics diskette.

XCOMP — sentential complement not containing a SUBJ: The printer is prepared to receive information. The number should continue to increase.

OBL-agent — the agent of a passive sentence in oblique case: Your printer is also controlled by program commands from the system unit.

PPADJUNCT — any prepositional phrase: Get the diagnostics diskette from the back of this manual. Get the diagnostics diskette from the back of this manual.

ADVADJUNCT — adverb phrases that modify verbs: The printer should also work well. Operate your TV normally.

SADJUNCT — clauses that are not arguments of the main verb: Operate your TV normally to verify color quality. If an expansion unit is attached, set the Power switch to On.

MODIFIER — a modifier of a noun: This is the typematic test. For proper operation, follow the procedure below.

Grammatical functions such as SUBJ and OBJ as used here are quite different from subject and object as defined in traditional grammar books. In traditional grammar, a subject might be the main actor or agent in a sentence and an object might be the thing that is affected by the action. However, we follow LFG and Relational Grammar (Perlmutter, 1983) in treating 'subject' and 'object' as purely syntactic categories, membership in which is determined exclusively by syntactic behavior.

Grammatical features such as number and tense can vary a great deal from language to language. For example, some languages have singular, plural, and dual nouns (Warlpiri), some have only singular and plural nouns (English) and some do not mark number on nouns at all (Japanese). There is also considerable variation in grammatical systems of case, tense and aspect.

Grammatical functions like SUBJ and OBJ, on the other hand, do not vary as much from language to language. Since all languages draw from the same basic set of grammatical functions and since f-structure does not encode surface word order (the elements in an f-structure are unordered), languages which have very different c-structures can have very similar f-structures. We have already seen an example of a Japanese f-structure. Here is the English f-structure (like simplified from parser output) for the sentence examined above (Remove the diskette from the diskette drive):
Notice that except for the language-specific grammatical features, the English and Japanese f-structures are the same. The OBJ is diskette/disuketto, the PPADJUNCT is diskette drive/disuketoodoraibu, and the main verb of the sentence is remove/toirinouzoku.

An English sentence and its Japanese translation will not always have the same f-structure. (They will only be the same if the subject, object, verb, prepositional phrases, etc. are the same.) But they will often be the same and they will always draw from the same universal inventory of grammatical functions.

Similarly, within any one language, sentences only have the same f-structure or very similar f-structures if they have the same grammatical functions. For instance, the sentences Is the repairman fixing the computer? The repairman is fixing the computer and What is the repairman fixing? have similar feature structures because they all have the same subject, verb and object (except that the object has been replaced by what in the third sentence). In contrast, an active sentence and its corresponding passive sentence have different f-structures because they have different SUBJ's and OBJ's. Compare We gave books to the students and We gave students books. Observe that the former has a PPADJUNCT, to the students, whereas the other has a second object OBJ2, books. We represent the similarity in meaning of these two sentences by mapping them onto the same ILT.

2.2. F-Structures in Machine Translation

F-structure is important in linguistic theory because grammatical functions such as SUBJ and OBJ abstract away from surface differences among languages. Thus, many cross-linguistic generalizations can be described more easily in terms of grammatical functions (Bresnan, 1982; Perlmutter and Postal, 1983). Also, even within one language, some rules of grammar are stated more easily in terms of grammatical functions than in terms of word order and tree structures. This is especially true in languages with free word order where sentences with very different c-structures can have the same f-structure.

An important use of f-structures in parsing is to store features of words (e.g., number, person and tense) to be matched against features of other words in order to handle phenomena like agreement, case marking and verbal morphology.

In KBMT-89, f-structure is the input to mapping rules which produce ILTs. It is more convenient to map to the ILTs from f-structure than from c-structure because it is sometimes the case that more than one c-structure maps onto essentially the same f-structure. If we mapped from c-structure, we would have to write more mapping rules. Also, if we map from f-structure, our mapping rules for different languages will be more similar. This will allow us to take advantage of principles of linguistic theory and lexical semantics in the organization of mapping rules (see Mitamura, 1988).

The following presents an informal introduction to unification-based grammars and to some details of our grammar formalism. 4

In unification-based grammars, context-free phrase structure rules are augmented with equations which are actually calls to the process of unification and result in the building of f-structures.

The basis for the notation used in equations is that each node in the phrase-structure tree puts information into an f-structure. In the sentence you have a computer, the first NP node puts information such as number, person and the root of the head noun into the f-structure. Here the f-structure is (4a),5 the f-structure (4b):

(4) (a) [NP[N you]]

(b) (ROOT you) (PERSON 2))

An S node can also put information into an f-structure. For example, the f-structure for an S might indicate that the NP's f-structure is the value of the SUBJ feature, resulting in the f-structure in (5b):

(5) (a) [S[NP[N you]] [VP]]

(b) (SUBJ ((ROOT you) (PERSON 2))

An f-structure corresponding to a VP might contain information about the root and tense of the main verb, and it might incorporate the f-structure of an NP as the value of the OBJ function. A c-structure and f-structure for a VP are given in (6):
(a) [VP [V have] [NP [DET a] [N computer]]]

(b) ((ROOT HAVE)
  (TENSE PRES)
  (OBJ ((DETERMINER ((REFERENCE INDEFINITE))) (ROOT COMPUTER) (NUMBER SG) (PERSON 3)))))

It is possible for more than one phrase-structure node to put information into the same f-structure. The f-structure in (7a) contains the information from the S and VP and represents a complete sentence.

(7) (a) [S [NP [N you]] [VP [V have] [NP [DET a] [N computer]]]]

(b) ((SUBJ ((ROOT YOU) (PERSON 2)))
  (ROOT HAVE) (TENSE PRES)
  (OBJ ((DETERMINER ((REFERENCE INDEFINITE)))
        (ROOT COMPUTER) (NUMBER SG) (PERSON 3)))))

2.2.1. Names of F-Structures

In our parser, the variables x₀, x₁, x₂, x₃, etc. take f-structures as their values and are used in naming sub-f-structures. As an example, assume that x₀ is bound to this f-structure:

x₀ ((SUBJ ((ROOT YOU) (PERSON 2)))
  (ROOT HAVE) (TENSE PRES)
  (OBJ ((DETERMINER ((REFERENCE INDEFINITE)))
        (ROOT COMPUTER) (NUMBER SG) (PERSON 3)))))

The entire f-structure is bound to the variable x₀, so we can call it x₀. The sub-f-structure, ((ROOT YOU) (PERSON 2)), is not bound to a variable. However, since it is the value of the SUBJ function in x₀, we can call it (x₀ subj). In general, (x₀ f), where x₀ is a variable bound to an f-structure and f is a feature in x₀, refers to the value of the feature f in x₀.

In order to refer to the atom 'you', start by finding the SUBJ of x₀ — ((ROOT YOU) (PERSON 2)) — and then find the root of that. We would write this as (x₀ subj ROOT). In general, (x₀ f₁ f₂) means 'find the value of f₁ in x₀ and then find the value of f₂ inside of that'. Expressions of the form (x₀ f₁ f₂) are called 'path names' and can be arbitrarily long (e.g., (x₀ f₁ f₂ f₃ f₄ f₅ f₆)); in practice we rarely have need for a path name of length greater than 3.

2.2.2. Unification

Unification is an operation that combines information from two f-structures provided that they do not contain conflicting information. Here follows a very informal description of unification to provide enough background to understand how the grammar formalism works.

To unify two feature structures, f₁ and f₂ (f₁ U f₂):

- If x₁ and x₂ are both atomic and are the same entity, then x₁ U x₂ is that entity: sg U sg = sg.
- If x₁ and x₂ are atomic and are not the same, then unification fails: sg U pl = fail.
- If x₁ is atomic and x₂ is not atomic or vice versa, unification fails.
- If x₁ and x₂ are non-atomic f-structures, then the unification of x₁ and x₂ is an f-structure whose features are the union of the features in x₁ and x₂ and the value of a feature A is (x₁ A) U (x₂ A). That is, the value of a feature A is the value of A in x₁ unified with the value of A in x₂. If A has no value in an f-structure x₀, then (x₀ A) = [] or an empty f-structure, which is an identity element for unification.) Thus:

1. ((NUM SG) (PERS 3)) U ((CASE NOM) (GENG MASC)) =
   ((NUM SG) (PERS 3) (CASE NOM) (GENG MASC))

2. ((NUM SG) (PERS 3)) U ((CASE NOM) (NUM SG)) =
   ((NUM SG) (PERS 3) (CASE NOM))

3. ((NUM SG) (PERS 3)) U ((CASE NOM) (NUM PL)) = fail

4. ((SUBJ ((NUM SG) (PERS 3))) (ROOT SEE) (OBJ (ROOT MOUSE)))
   U ((SUBJ ((ROOT CAT) (DEF +1)) (OBJ (NUM SG) (PERS 3)))
   (SUBJ ((ROOT CAT) (DEF +) (NUM SG) (PERS 3))) (ROOT SEE) (OBJ
   (NUM SG) (PERS 3) (ROOTouse))

- And when one of the entities being unified is empty:
  sg U [] = sg

((NUM SG) (PERS 3) U [] = ((NUM SG) (PERS 3))

2.3. Equations and Unification

Each grammar rule consists of a context-free phrase structure rule followed by a list of equations. In the first rule below, x₀, x₁ and x₂ are variables bound to f-structures. x₀ is the f-structure containing information from the S node, x₁ is the f-structure containing information from the NP node, and x₂ is the f-structure containing information from the VP node.

In general, x₀ is bound to the f-structure corresponding to the node on the left-hand side of the arrow, x₁ is bound to the f-structure corresponding to
the first element on the right side of the arrow, \( x_2 \) is bound to the f-structure corresponding to the second element on the right, and so on.

The equal sign in an equation means

1. Find the f-structures referred to by each side of the equation;
2. (Pseudo) unify them; and
3. Replace both f-structures with the result of (pseudo) unification.

Here then are some simplified rules that could be used to parse the sentence *A message appears on the screen.*

\[
(\text{NP} \rightarrow (\text{NP} \text{VP})
\]
\[
(\text{VP} \rightarrow (V \text{PP})
\]
\[
(\text{PP} \rightarrow (P \text{NP})
\]
\[
(\text{NP} \rightarrow (\text{DET} \text{NP})
\]
\[
(\text{NP} \rightarrow (\text{ROOT MESSAGE} \text{NP}))
\]

In addition to the f-structures described by these rules, each word in the analysis lexicon carries a feature structure. When the word is attached to a node in the c-structure tree, the word's feature structure becomes the feature structure of the node.

Since our parser builds the c-structure from left to right, it will start with the word *a,* which has this f-structure:

(\text{NUMBER SINGULAR} \text{REFERENCE INDEFINITE})

A will be attached to a DET (determiner) node and its f-structure will be the determiner's f-structure.

The next word is *message,* which has this f-structure:

(\text{ROOT MESSAGE} \text{NUMBER SINGULAR} \text{PERSON 3})

This will become the f-structure for an N (noun) node when *message* is attached.

Now that there is a determiner and a noun, they can be combined into a noun phrase (NP) using this rule:

\[
(\text{NP} \rightarrow (\text{DET} \text{NP}) (x_0 \text{DET} \text{NP}) (x_0 \text{DET} \text{NP}))
\]

In this rule, \( x_0 \) is the f-structure for the NP, which is currently empty; \( x_1 \) is the f-structure for the DET node, which is currently the f-structure from the lexical entry of *a*; and \( x_2 \) is the f-structure for the N node, which is the f-structure from the lexical entry of *message.* Examples (8a and b) show the c-structure a *message* and f-structures \( x_0, x_1, \) and \( x_2 \) before unification.

(b) \( x_0 \) \[
(\text{NP})
\]
\[
(\text{VP})
\]
\[
(\text{PP})
\]
\[
(\text{NP})
\]
\[
(\text{DET}) \text{NP}
\]
\[
(\text{ROOT MESSAGE}) \text{NP}
\]

The first equation, \( x_0 = x_2, \) causes \( x_0 \) to unify with \( x_2. \) After unification, \( x_0 \) and \( x_2 \) are replaced with the result of unification, resulting in these f-structures:

\[
(\text{NP})
\]
\[
(\text{VP})
\]
\[
(\text{PP})
\]
\[
(\text{NP})
\]
\[
(\text{DET}) \text{NP}
\]
\[
(\text{ROOT MESSAGE}) \text{NP}
\]

The next equation, \( x_0 \text{DETERMINER} = x_1, \) unifies the value of \( x_0\)'s DETERMINER feature with \( x_1. \) Since \( x_0 \) does not yet have a DETERMINER feature, the expression \( x_0 \text{DET} \) refers to an empty f-structure. So, \( x_1 \) will be unified with an empty f-structure. The result is \( (\text{NUMBER SINGULAR}) \text{REFERENCE INDEFINITE}). \) After unification, \( x_1 \) and \( x_0 \) DETERMINER are replaced by the result of unification. Since \( x_1 \) was unified with an empty f-structure, it is not changed. \( x_0 \) DETERMINER now has a value, which is the result of the unification with \( x_1. \) Here is what \( x_0, x_1, \) and \( x_2 \) look like now:

\[
(\text{NP})
\]
\[
(\text{VP})
\]
\[
(\text{PP})
\]
\[
(\text{NP})
\]
\[
(\text{DET}) \text{NP}
\]
\[
(\text{ROOT MESSAGE}) \text{NP}
\]

The next word is *appears,* which has the following f-structure. This becomes the f-structure of the V node to which *appears* is attached.

(\text{ROOT} \text{APPEAR} \text{TENSE PRESENT})

The next words are on the screen. Their f-structures become the f-structures for the P, DET and N nodes to which they attach.
The phrase-structure rule and c-structure for the screen, and initial f-structures for \( x_0 \), \( x_1 \) and \( x_2 \) before unification, are given in (9). The and screen can be combined into an NP using the NP rule. The f-structure for the is \( x_1 \), the f-structure for screen is \( x_2 \) and the f-structure for the whole NP is \( x_0 \). Note that \( x_0 \) is currently empty.

(9) (a) \(<\text{NP}> \rightarrow \langle\text{DET} \ <\text{N}\rangle\)
\( (x_0 = x_2) \)
\( ((x_0 \ \text{determiner}) = x_1)) \)

(b) [NP [DET the] [N screen]]

(c) \( x_0 \) [ ]
\( x_1 \) ((REFERENCE DEFINITE))
\( x_2 \) ((ROOT SCREEN) (NUMBER SINGULAR) (PERSON 3))

The equations associated with the NP rule trigger the unification of \( x_1 \) with (\( x_0 \ \text{determiner} \)) and \( x_2 \) with \( x_0 \). Then \( x_0 \), \( x_1 \), and \( x_2 \) are replaced with the result of unification.

\( x_0 \) ((DETERMINER ((REFERENCE DEFINITE)))
\( (\text{ROOT SCREEN}) (\text{NUMBER SINGULAR}) (\text{PERSON 3})) \)
\( x_1 \) ((REFERENCE DEFINITE))
\( x_2 \) ((ROOT SCREEN) (NUMBER SINGULAR) (PERSON 3))

The preposition and NP can be combined into a PP using the PP rule. The f-structure for on is \( x_1 \) and the f-structure for the NP that was just built is \( x_2 \). (This f-structure was bound to the variable \( x_0 \) in the previous rule, but the variable bindings change with every rule.) So, \( x_0 \), currently empty, is the f-structure corresponding to the PP node.

Example (10) shows the c-structure for on the screen and the f-structures \( x_0 \), \( x_1 \) and \( x_2 \) before unification.

(10) (a) [PP [P on] [NP [DET the] [N screen]]]

(b) \(<\text{PP}> \rightarrow \langle\text{P} \ <\text{NP}\rangle\)
\( ((x_0 = x_2) \)
\( ((x_0 \ \text{preposition}) = (x_1 \ \text{root}))) \)

(c) \( x_2 \) ((DETERMINER ((REFERENCE DEFINITE))))
\( (\text{ROOT SCREEN}) (\text{NUMBER SINGULAR}) (\text{PERSON 3})) \)
\( x_1 \) ((ROOT ON))
\( x_0 \) [ ]

The first equation unifies \( x_0 \) with \( x_2 \), and \( x_0 \) and \( x_2 \) are replaced with the result of unification:

\( x_2 \) ((DETERMINER ((REFERENCE DEFINITE))))
\( (\text{ROOT SCREEN}) (\text{NUMBER SINGULAR}) (\text{PERSON 3})) \)
\( x_0 \) ((DETERMINER ((REFERENCE DEFINITE))))
\( (\text{ROOT SCREEN}) (\text{NUMBER SINGULAR}) (\text{PERSON 3})) \)

The second equation unifies \( x_0 \ \text{preposition} \) with \( (x_1 \ \text{root}) \). Since \( x_0 \ \text{preposition} \) is currently undefined, its value is the empty f-structure [ ] . The value of \( (x_1 \ \text{root}) \) is 'on'. So, 'on' is unified with [ ], and \( x_0 \ \text{preposition} \) and \( (x_1 \ \text{root}) \) are replaced with the results. The effect of this unification is that a preposition feature is added to \( x_0 \) with the same value as \( (x_1 \ \text{root}) \):

\( x_0 \) ((PREPOSITION ON) (DETERMINER ((REFERENCE DEFINITE))))
\( (\text{ROOT SCREEN}) (\text{NUMBER SINGULAR}) (\text{PERSON 3})) \)
\( x_1 \) ((ROOT ON))

Now the PP and V can be combined into a VP. Here, \( x_0 \) is the f-structure for the VP, \( x_1 \) is the f-structure for V and \( x_2 \) is the f-structure for PP. Example (11) atop the following page shows the c-structure for appears on the screen and f-structures \( x_0 \), \( x_1 \) and \( x_2 \) before unification.

The first equation unifies \( x_0 \) with \( x_1 \):

\( x_0 \) ((ROOT APPEAR) (TENSE PRESENT))
\( x_1 \) ((ROOT APPEAR) (TENSE PRESENT))

The next equation unifies \( x_2 \) with \( (x_0 \ \text{ppadjunct}) \). Both are replaced with the result of unification.

\( x_0 \) ((PPADJUNCT ((PREPOSITION ON))
\( (\text{DETERMINER ((REFERENCE DEFINITE))}) \)
\( (\text{ROOT SCREEN}) (\text{NUMBER SINGULAR}) (\text{PERSON 3})) \)
\( (\text{ROOT APPEAR}) (\text{TENSE PRESENT}) \)
\( x_2 \) ((PREPOSITION ON) (DETERMINER ((REFERENCE DEFINITE))))
\( (\text{ROOT SCREEN}) (\text{NUMBER SINGULAR}) (\text{PERSON 3})) \)
The last rule puts NP and VP together to form an S. Here, x0 is the f-structure for S, x1 is the f-structure for NP and x2 is the f-structure for VP. The c-structure for "a message appears on the screen" and the f-structures for x0, x1, and x2 before unification are given in example (12).

The first equation unifies x0 and x2:

x0 (PPADJUNCT (PREPOSITION ON) (DETERMINER (REFERENCE DEFINITE)) (ROOT SCREEN) (NUMBER SINGULAR) (PERSON 3))

The second equation unifies x1 with (x0 sub j), resulting in these f-structures:

Finally, the value of x0 is returned for the parse.

REFERENCES


Lenat, D., M. Prakash, and M. Shepherd. 1985. Cyclic knowledge to overcome brittleness and knowledge acquisition bottlenecks. AI Magazine, VI.


