Lexicons

Donna GATES, Dawn HABERLACH, Todd KAUFMANN, Marion KEE, Rita McCARDELL, Teruko MITAMURA, Ira MONARCH, Stephen MORRISSON, Sergei NIRENBURG, Eric NYBERG 3rd, Koichi TAKEDA and Margalit ZABLUDOWSKI

ABSTRACT: The three lexicons used by KBMT-89 are described: A concept lexicon constitutes the sublanguage domain model for specifying semantic information; it is maintained by ONTOS, a knowledge-acquisition and maintenance system. An analysis lexicon is a dictionary containing syntactic information and mapping rules required for semantic parsing. And a generation lexicon, similar to the analysis lexicon, is employed in the generation phase.

KEYWORDS: analysis lexicon, concept lexicon, domain model, episodic memory, events, generation lexicon, knowledge acquisition, mapping rules, ONTOS, semantic memory

1. THE CONCEPT LEXICON

The KBMT-89 ontology or concept lexicon\(^1\) is a language-independent\(^2\) conceptual representation of the interactions between personal computers and their users. It provides the semantic information necessary in our sublanguage domain for parsing source text into ILTs and generating target texts from ILTs.

We are extending and modifying the ontological model to the level of detail required for specifying the semantic restrictions for appropriately constraining the output of the parser and generator. In order to provide sufficient semantic restrictions, the ontological model must provide uniform definitions of fundamental categories that are employed in crafting descriptions of particular domains.

\(^1\) 'Concept lexicon' and 'ontology' are nearly interchangeable, although we especially use the former when we want to refer to the ontology as containing lexical mapping rules (described in Section 2, below).

\(^2\) Although node-names and slot-names of the encoding are in English, they represent language-independent concepts in the sense that both English speakers and Japanese speakers can use them to make meaning assignments to lexical items in their respective languages.

1.1. **ONTOS: A Knowledge Acquisition and Maintenance System**

Ontological and domain models are acquired with the knowledge acquisition and maintenance system ONTOS. The system provides an interactive environment that includes facilities for interacting in multiple windows through menus and graphics. ONTOS contains ontological postulates that are the basis for:

- Memory and decision aids to help users find the appropriate way to describe a newly-entered concept; and
- A filtering capability that helps identify potential inconsistencies and problems in the knowledge base.

These ontological postulates are computationally inscribed in a frame network representation of objects, events and situations, characterized by attributes and relations and organized in subcategorical and partonomical hierarchies. Integrated acquisition of domain concepts proceeds through elaboration and specification of concepts framing ontological postulates. The result is what we call an *ontology*. We maintain that such an environment should support both:

- Knowledge update by humans through intelligent interactive aids; and
- Semi-automatic and automatic knowledge update in which a knowledge acquisition maintenance system learns from texts by suggesting new, partially characterized concepts to be elaborated on, refined and sometimes deleted by humans or by the system itself as new information is encountered.

Most importantly, we believe that the second of these two requirements is essentially dependent on the first in the sense that automating knowledge and lexicon building depends on hand-crafting a substantial base of real-world knowledge (Lenat et al., 1985) for limited subworlds (Nirenburg and Raskin, 1987) and sublanguages (Kitredge and Grishman, 1986).

1.2. **Ontological Postulates**

World knowledge is organized as a multiply interconnected, hierarchical network of frames constructed with ONTOS. Ontological postulates which are resident in ONTOS are encoded in the higher, more abstract nodes of this network and serve as a map to help knowledge enterers determine where domain concepts fit into the knowledge hierarchy. They are also a source of properties and constraints that can be further specified into domain concepts. For KBMT-89, we have built a domain ontology representing the interactions among personal computers, their users and the manuals that guide this interaction.

The knowledge representation language FRAMEKIT (Nyberg, 1988) represents ONTOS concepts and serves as a grammar of the *lingua mentalis* or language-independent description of the subworld of human-computer interaction. Our attention is focused on the task of defining and creating the concept lexicon for this *lingua mentalis*, that is, the set of interrelated concepts.

In this section we present five postulates specified in natural language that regulate the relations between and among ONTOS concepts and their properties. We first discuss some of the extensions to any general-purpose knowledge representation system that enable it to facilitate ontological modeling. Next, we discuss the types of knowledge in a domain model, and how that domain model is used in representations of text meaning.

1.2.1. **Knowledge Representation for Domain Modeling**

To construct ontological domain models, we need to extend the semantic definitions and constraints already available in FRAMEKIT. These extensions provide an ontological interpretation or semantics for the FRAMEKIT representation language.

Thus, the first four postulates:

1. **Each frame represents an ONTOS concept.**
2. **Concepts are subdivided into the types of things that can be referred to, such as objects, events and their properties.**
3. **Properties of concepts are subdivided into relations and attributes, with each slot corresponding to a property.**
4. **Relations map concepts into concepts, while attributes map concepts into value sets. The elements of value sets are literals, not concepts.**

So, for instance, *part-of* is a relation. (Consider `((part-of) wheel car)`.) On the other hand, *age* is an attribute whose value set is defined in ONTOS as a semi-open integer range (> 0) whose dimensionality is seconds. Further, *color* is an attribute that maps physical objects into a set of symbols (a value set) without a strict ordering.

The distinction between relations and attributes captures the intuitive distinction between them, namely that relations involve two or more things in the world, whereas attributes only involve one. Making reference to an element in a value set is not referring to one thing in terms of another, but referring to a

---

3Our goal was to define all scalar attributes using standard units of measurement, e.g., seconds in the case of temporal ranges. However, for KBMT-89, qualitative restrictions on ranges were assumed. We used such literal value sets elements as 'small,' 'medium' and 'large' instead of numerical ranges.

4We have considered using a quantitative standard for color, such as the Munsell color notation, which specifies colors on numerical scales of hue, value and chroma, and can be expressed as accurately as desired. The boundaries of each color name can be specified and defined in terms of this or some like numerical notation. As with scalar attributes in KBMT-89, we have adopted the qualitative or "literal" representation of color.
given thing in a certain way. This method of encoding an ontological distinction has computational benefits as well. It establishes a syntactic criterion that facilitates consistency and type checking in creating and extending a knowledge base. Moreover, the distinction provides the device for restricting the conceptual granularity of an ontology. For example, while there should be a concept of number or numeral, it is not necessary for there to be a concept for every number (1, 2, 3, ...). Likewise for color: Although there needs to be a concept of color, there does not need to be a concept for every color (red, blue, black, ...). However, with regard to representing particular numbers, we discovered in the process of integrating the ontology into the analysis lexicon that virtual frames\(^3\) made it easier to build parts of ILTs to represent those numbers. Further, special lexical entries were needed to facilitate building ILTs involving value-set elements.

A number of legal facet fillers are allowed in our knowledge representation language.\(^4\) There are these kinds of directly listed values: single concept names, multiple concept names, single value set elements, multiple symbolic value set elements and multiple numerical value set elements.

### 1.2.2. Semantic Memory

The requirements of knowledge representation extend beyond providing an adequate formalism in which to record knowledge and beyond adequate storage and retrieval. One has to specify the contents or semantics of the knowledge units. This task is less formalizable than the syntactic aspects of knowledge representation; this is one reason why relatively little progress has been made in AI with respect to ontological world modeling. Indeed, this area of scientific research remains, as it has been for over 2,500 years, within the purview of philosophy. Understood philosophically, the task of selecting conceptual primitives or categories has been a challenge and vexation since Aristotle and, in the modern era, since Kant. We make no claims here about the metaphysical or philosophical status of our categories; to do so would take us too far from the empirical enterprise of knowledge-based machine translation. While a number of important theories have been propounded in philosophical ontology, we believe that it is necessary to reformulate the goals and methodology of this inquiry. Such reformulation is similar to the way in which the finite time/space constraints of practical computation changed the style and attitudes of certain areas of discrete mathematics, which gave birth to the theory of computation.

An ontological model must define a large set of generally applicable categories for world description. Several requirements emerge:

- Perceptual and common-sense categories are needed for an intelligent agent (such as, for instance, a planning or learning program) to interact with and manipulate the states of the outside world (see for instance articles in Hobbs and Moore, 1985).
- Categories are required for encoding interagent knowledge that involves firstly one's own as well as other agents' intentions, plans, actions and beliefs; and secondly communication-related knowledge, including syntax and semantics of at least one natural language as well as discourse and pragmatics rules of linguistic communication (Nirenburg, 1985).
- Categories must be found to help describe metaknowledge, that is, knowledge about knowledge and its manipulation, including rules of behavior, and heuristics for constraining search spaces in various domains.
- Means are needed to encode categories generated through the application of the above inferential knowledge to the contents of an agent's world model (see articles in Brachman and Levesque, 1985).

The choice of categories is not a straightforward task, as anyone who has tried realistic-scale world description knows all too well. As an illustration of the difficulties encountered in such an undertaking, consider the questions:

Which of the set of attributes pertinent to a certain concept should be singled out as concept-forming and thus have named nodes in the conceptual network corresponding to them, and which other ones should be accessible only through the concept of which they are properties? Consider, that is, whether the class vehicle would be further subdivided into water-vehicle, land-vehicle or air-vehicle or, rather, into engine-vehicle, animal-propelled-vehicle or gravity-propelled-vehicle; or perhaps into cargo-vehicle, passenger-vehicle, toy-vehicle or mixed-cargo-and-passenger-vehicle? Or might it be preferable to have a large number of small classes, such as water-passenger-animal-propelled-vehicle, of which, for instance, will a rowboat be a member?

Also: Which entities should be considered objects and which ones relations? Should we interpret a cable connecting a computer and a terminal as a relation? Or should we rather define it as a physical object and then specify its typical role in the static episode or "scene" involving the above three objects?

Although we have distinguished between relations and attributes, we might have defined attributes as one-place relations. Is it a good idea to introduce the ontological category of attribute value set with its members being primitive and unstructured (such as the various scalars and other, unordered, sets of properties)? Or is it better to define them as full-fledged ontological concepts? If so, how do we deal with attributes with infinite or even very large ranges?

Further, should we represent colors symbolically, as, say red, blue, etc. or should we rather define them through their spectrum wavelengths, position on the white/black scale and brightness? How should we treat sets of values? Should we represent The Three Musketeers as one concept or a set of three?
What about The Pittsburgh Pirates? What's an acceptable way of representing complex causal chains? How does one represent a concept like fake gun? Is it a gun? Or a toy? Or neither? Or is it maybe the influence of natural language and a peculiar choice of meaning realization on the part of the speaker that poses this problem — and perhaps we do not need to represent this concept at all . . .?

Whereas we advocate scaled representations discussed above, we must, for any level of detail, provide concrete answers to these questions. A number of such decisions have already been made in our work on the prototype ontological model which we acquired using ONTOS. Figure 1 shows several subnetworks from this ontology produced by ONTOS from its resident ontology with the help of its browsing facility. This output displays some of the answers we suggested for questions of the type sampled above.

1.2.3. Episodic Memory

A general-purpose world model must cover the union of needs for multiple application domains and for any type of rational process. The knowledge required to solve problems in various domains and types of activity includes not only an ontological world model, as described above, but also records of past experiences (including learned or reported ones). To resume the linguistic analogy, in addition to grammar and lexicon our lingua mentalis makes itself manifest in "texts" encoded in it. In KBMT-89, ILTs are literally such texts. Hence our final postulate:

5. The lingua mentalis equivalent of a text is an episode — a unit of knowledge that encapsulates a particular experience of an intelligent agent and that is typically represented as a temporally and causally ordered subnetwork of frames.

The ontology and the episodes are sometimes discussed in terms of the contents of two different types of memory: semantic and episodic (Tulving, 1985). This distinction seems useful in computational modelling as well. In our work we are representing both ontological concepts and episodes with varying degree of specificity. The same knowledge representation apparatus is used for storing the contents of both the ontology and the episode cluster.

It becomes important, from the theoretical, descriptive and implementational points of view to define an adequate interface between the two memories. We suggest the following approach. The episodes (remembered experiences) are temporally and hierarchically organized sets of tokens of event-types. They are indexed through the type they correspond to and can be interrelated on temporal, causal and other such links. The participant roles in the episodes can be either instantiations of object and event types in the semantic memory or references to existing named instances, stored outside semantic memory, but having links to their corresponding types (again see Figure 1).
The presence of a systematic representation and indexing method for episodic knowledge is an enablement condition for case-based reasoning (Kolodner, 1984; and Schank, 1982) and analogical inference (Carbonell, 1983; and Carbonell, 1986).

1.3. The Domain Ontology

The concept lexicon comprises the domain ontology and the lexical mapping rules entered as the fillers of lexical slots in frames encoding concepts. The addition of the lexical mapping rules is not an addition to the content of the ontology. Rather, the ontology provides the basis for assigning meanings to lexical items by way of the lexical mapping rules. However, associating lexical items with items in the ontology has required modification of the ontology itself.

The concept lexicon plays an important role throughout the KBMT-89 system. It is used to assign semantics via lexical mapping rules to all the lexical items of the 150 English and 150 Japanese sentences selected for translation. The analysis lexicon is built by adding information from the lexical mapping rules to syntactic and morphological information associated with the corresponding lexical items. At parse time, f-structures are built from source language text by the syntactic parser. A mapping rule interpreter analyzes these f-structures using lexical mapping rules, structural mapping rules, the analysis lexicon and the concept lexicon (here to determine semantic restrictions) to produce parser output. The augmentor builds an ILT from the parser output. When there is an ambiguous parse, the augmentor uses the parser output and information from the concept lexicon to formulate alternative meaning options and builds an ILT in accord with user response. The generation lexicon is also based on the concept lexicon and is used by the generator to produce target language text. Here follow descriptions of some of the decisions that were made during creation of the KBMT-89 concept lexicon.

Types and Tokens: We distinguish between the meaning of concepts on the one hand and the meaning of propositions (ILT structures) on the other. The meanings of concepts are types. The meanings of propositions are composed of instances of these types. The concept lexicon specifies, for example, the concept of a computer, the concept of having parts and the concepts of a computer's components. The ILT represents actual states of affairs in the world, e.g., that a particular computer has a CPU, a keyboard, a disk drive and a monitor as components. One important implication of this is that the concept lexicon cannot reference particular propositions. It is only permitted to reference heads of propositions, such as events or properties which, in a sense, classify types of propositions — that is, propositions describing changes, relationships and

---

7 Actually, other information is also included, but it is not relevant to the discussion here; for an account, see the discussion of ILTs in the paper "Knowledge Representation Support" by Nirenburg and Levin in this issue.

Attributes and States: From a commonsense point of view, there seems to be a fairly clear distinction between properties and states. Properties are the attributes objects can have and states are conditions that objects are in. However, on closer scrutiny, the initial clarity of this distinction begins to fade. Does the distinction between states and properties warrant the introduction of states into the ontology? What does the distinction really amount to? It comes, we suggest, to this: a state is more temporary and more likely to change than a property. However, this difference only represents a difference of degree. Such distinctions allow different users to make different decisions in different situations about whether something should be a state or a property. But this is precisely what ontological analysis is supposed to avoid. Consequently, we decided to treat what are deemed 'states' from a commonsense point of view as properties in our ontology. Thus we ruled out states like *on-state in (a), below. Rather, we handle the meaning of on and off as the value-set elements %on or %off of the attribute *electric-current, as in (b).

a. b.

(((on-state
(is-a *state)
(theme *light-switch))
(electrical-current %on %off))

**(light-switch
(is-a *switch
(theme *light-switch))
(electrical-current %on %off))

States and Events: We also initially assumed a traditional distinction between states and events. For example, in analyzing the meaning of the sentence John is hitting the ball one can say that in the time between the beginning of the hitting event and the end of the hitting event, John is in a state of hitting, thus again seeming to require that states be postulated in the ontology. However, postulating states would require the creation of nearly identical frames:

a. b.

(((hit
(is-a *physical-event)
(agent *intentional-agent)
(theme *physical-object)
(instrument *stick))

((hitting
(is-a *physical-state)
(agent *intentional-agent)
(theme *physical-object)
(instrument *stick))

Rather than create more frames than needed, we handle the meaning of
present progressives by instantiating an appropriate event frame and recording aspeclual information. Specifically, an aspect frame containing a (phase) slot filled with \((\text{and begin (not end)})\) is instantiated and embedded in the event frame. Only the event frame is part of the concept lexicon.

The concept *event as specified in the lexicon is used in the analysis of tense and aspect. The meaning of most nominalizations can be mapped to events so that a sentence such as The destruction of the city was a mistake can be paraphrased as 'It was a mistake to destroy the city.' By taking The destruction as an event, we gain the generalization that the frame should contain an agent and a theme.

Complex Events: From the point of view of the concept lexicon an event is complex just in case it *has-as-part other events in the concept lexicon. Thus, even if an event might seem outwardly complex (e.g., a computer test), we do not make it a complex event unless there is a need to refer to an overriding event and talk about its component events.  

A primary example of this is the complex event *to-press-button, which be divided into three subevents: *press-button, *hold-down-button and *release-button. These subevents are temporally related; this relation is captured in the concept lexicon through slots internal to representations of the subevents. For instance, the *press-button concept contains a *before relational slot filled with *hold-down-button; and *hold-down-button in turn contains an *after slot filled with *press-button and a *before slot filled with *release-button.

So, in order to capture the meaning of the English sentences

(1) a. Hold down the key for 10 seconds.

b. Release the key.

it is necessary to posit subevents for the complex event *to-press-button. In Japanese however, the concept of holding down a key might be expressed as 'pressing,' with an aspeclual marker suggesting 'long duration;' so only one event is needed. (In Japanese, one expresses a subevent for 'release-button,' but not 'hold-down-button.')

In the English analysis lexicon, sentences that mean 'type a button' are mapped onto the *to-press-button frame, while those meaning either 'hold down' or 'release' are mapped onto the subevents. The *duration attribute tells the system that if a duration is expressed in the subevent *hold-down-button, that duration is also the duration of the complex event.

In the Japanese analysis lexicon, we would map the sentence meaning 'hold down the key for 10 seconds' onto the *to-press-button frame and insert '10 seconds' into the *duration slot of the complex event. The *duration slot tells the system that if a duration is expressed in the complex-event, then it also is the duration of the subevent, in this case *hold-down-button. In this way, the (English) generator will produce 'hold down the key for 10 seconds' as opposed to 'type the key for 10 seconds'; and this is what is wanted as the latter sentence has the preferred meaning 'type the key over and over again for 10 seconds.'

While the notation described above is adequate for this project, it is perhaps flawed in that two complex events might share some subevents but order them differently. Similarly, a single subevent of two different complex events might have a different cause or effect depending on the complex event. An ideal notation for complex events must therefore relate the subevents within the complex-event.

Determining Lexical Unit Boundaries Ontologically: Semantically, the determination of lexical unit boundaries is dictated by conceptual unit boundaries specified ontologically. We offer two examples, reflecting problems that arose as we began to associate lexical mapping rules with the ontology. The first also exhibits a problem arising from our assumption that ontology and meaning are language-independent. In the context of KBMT-89, however, such problems were easily handled and in any event not very numerous, thus vindicating, to some degree, our assumption.

In the first case we had to decide whether the English shipping card-board should be one lexical unit or two. Its lexical mapping is to the concept *protection-material. Now, in Japanese the corresponding lexical unit is hogosito, which has a more specific meaning than *protection-material and as such is lexically mapped to a more specific concept — *protection-sheet. Hence, in this case, the Japanese lexical unit has a more specific corresponding concept than the English. The solution was to make shipping cardboard one lexical unit associated with *protection-material. Further, since the corresponding Japanese lexical unit is associated with a child concept of *protection-material, namely *protection-sheet, locating it for translation purposes is computationally simple.

The second example concerns the sentence The basic IBM Personal Computer XT consists of a system unit and keyboard. For KBMT-89, system unit is one lexical unit while IBM Personal Computer XT is not. The rationale behind this decision can be explained in terms of compositionality. The head of the latter phrase is computer and, after analysis, would be (structurally) mapped into the concept *computer. At this point, the phrase could be composed in the following manner: In order to go from *computer to *ibm-computer, the
relation *manufactured-by (which is a concept in the ontology) becomes filled with IBM. From there, the attribute *number-of-users (also a concept) is filled with the slot value single (which is indicative of personal) and goes to *ibm-pc. This concept's relational slot *make (likewise a concept) then becomes filled with XT. For the phrase system unit, there does not exist a relation or attribute in the concept lexicon that would allow for a comparable traversal and, hence, composition. Thus, system unit is one lexical unit.

Objects: There are two sorts of "objects" which caused problems in building the concept lexicon: *2d-object and *representational-object.

Because our domain focuses on human-computer interaction and in particular the information on computer screens and in printed manuals, we had to create two classes of the concept *physical-object: the ordinary *3d-object and the rather unusual *2d-object (illustrated in Figure 3).


It was not obvious that objects such as those represented by *display object were physical because of the strong tendency to think of physical objects as three-dimensional. But it was clear that two-dimensional objects have many physical properties — they can be seen by more than one person, stand in spatial relations, be pointed to and so forth.

Once we made the decision to consider display objects as two-dimensional physical objects, it was easier to introduce other two-dimensional objects to handle a related but somewhat different problem. Consider the sentences:

(2) a. An indication of the memory test will appear in the upper-left corner of your screen.

b. Before arranging your system read the note below.

What sort of meaning should be ascribed to 'upper-left corner' and 'below'? There is a strong tendency to think of 'below' as a relation, e.g., 'A is below B.' But in sentence (2b), what is the note related to? Again, there is a strong tendency to think of 'left-of' as being a relation as well — 'A is to the left of B.' However, in both the case of 'below' and 'upper-left-corner', the lexical items seem to be referring to objects; now, these objects might be insubstantial, but they are nevertheless in some sense physical. We decided to make these *2d-objects as well. In this way, *2d-objects like *display-object and *corner can be *spatially-part-of other *2d-objects like *2d-areas. Figure 3 illustrates *2d-object frames.

Representational objects also required difficult ontological decisions. A *representational-object has three subdivisions: *information, *temporal-object and *language.

Figure 2. The 2d-object hierarchy.
We maintain that postulating objects like *information in the ontology is necessary to capture a unique relation like the 'convey/conveyed-by' relation, which is not a relation between physical objects, but a relation between a physical object and a non-physical object or representational object. For example, we maintain that a picture of a horse does not *convey a horse. Rather, we hold that the picture conveys information the referent of which is a horse. That is, the horse is the *referent of the *information that is *conveyed-by the picture.

Consider *temporal-object. These objects represent the temporal aspect of events; hence they are *representational objects. They were introduced for reasons similar to those for introducing *2d-objects containing *2d-objects such as corners. *temporal objects like *present, *past and *future temporally contain other temporal-objects called *moments and *events. They are used to analyze the meaning of words like later, now and for now. *moments are temporally part of any *temporal-object, but also have *moments as parts.10 Also, *past, *present and *future are represented as *temporal-objects analogous to the way *2d-areas are represented as *2d-objects. And *past, *present and *future have as parts *moments which can be related by *temporal-relations such as *before and *after.11 Given present distinctions, the word later can then be analyzed as a *moment that is temporally part of the *future, the word now as a *moment temporally part of the *present and the phrase for now as a duration starting in the *present and extending to a moment in the *future. This last is somewhat more complicated than what is needed for our corpus, so we interpret 'for now' as temporally part of the *present-future, a concept which is a child of both *present and *future.

2. THE ANALYSIS LEXICONS

2.0.1. Introduction and Overview

An analysis lexicon is a dictionary (indexed by word and by part of speech) that contains the syntactic information, mapping rules and semantic data required for semantic parsing. During the loading phase of parsing, an analysis lexicon is built up from declarative knowledge sources for each language. We usually refer to the final form of the lexicon as the dictionary for that language. When

---

10 This commit us to their being indefinitely divisible. In our present domain, this is not a problem. However, we could have put a stop to the indefinite divisibility of *temporal-objects by introducing *temporal-points as the parts of *moments. Then *temporal-points would be *temporal-objects that have no parts.

11 KBMT-89 cannot handle sets. Again, this is not a problem in the present domain.
a word is encountered in the input, the run-time parser consults the dictionary and retrieves the features and mappings associated with that word.

The English and Japanese analysis lexicon entries generally include the following parts: a word, its category, its inflection, its root form, and the syntactic features and mapping rules associated with that word. A top-level outline for an entry is shown below in Backus-Naur Form (BNF), followed by a informal explanation of the terms in this example and where they originate. Then a more detailed BNF specification for lexicon entries appears in Section 2.0.2.

```xml
<analysis-lexicon-entry> ::= 
  <word>
  <category>
  ?<inflection>
  ?<infl-form>
  ?<root>
  ?<features>
  ?<mappings>
```

The dictionary for each language is created from the following knowledge sources: a base lexicon for each word category in the language, the syntactic feature hierarchy for each category, the morphology rules and mapping rules for that language, and the concept lexicon entries. We refer to the process of building an analysis lexicon as expansion, because it begins with the base lexicon and adds to it information from the other knowledge sources. The format of the entries is the same throughout this process. During expansion, most existing entries will have more information added to them, and a new entry will be created for each morphological variation on the basic (root) form of the word.

The base lexicon is the original source of <category>, <word>, <inflection>, <features> and <mappings> as given in the example above. The <category> of the word indicates a part of speech, such as V for verb or N for noun. If the same word has more than one category, separate entries are created. The <inflection> consists of an inflection type, such as past or present, and its form. Information about any irregular forms in the word's inflection appears here, and will override the default, which is to assume that all the forms of a word have regular morphology. The syntactic features are used during syntactic parsing and include number, agreement and valency category.12 The parts of the entry represented by <features> and <mappings> contain, in the base lexicon, either a pointer to default information contained in a separate hierarchy (see below), or information specific to the individual entry, or both. In the case of <features>, specific information will override the default for any given feature; in the case of <mappings>, specific information will simply be added to the default information.

Before expansion, the entry <word> is defined to be the root of the word. (The root is a basic form of the word. For example, the conjugated English forms removed, removes and removing have the same root, remove.) In the completed dictionary, the entry <word> is an actual input string.

The function of the mapping rules is to unite the concept lexicon's frame definitions for a particular domain with the syntactic rules for a given language. There are two kinds of mapping rules for a particular word: lexical mapping rules and structural mapping rules. The lexical mapping rules map a word onto a head concept or a slot and a concept. The structural mapping rules map a syntactic slot of an f-structure, such as subject or object, onto a semantic slot in a concept (such as agent or theme).13

Both syntactic features and structural mapping rules can be defined in an analysis lexicon entry either locally using (local) or non-locally using (class). (local) is an information holder for 'word specific' syntactic features and mapping rules. (class) contains a pointer to a general structure in which common information can be stored. These general structures for the syntactic features and the structural mapping rules are called the syntactic feature hierarchy and the structural mapping hierarchy, respectively. The analysis lexicon will inherit information from these hierarchies, through the pointers contained in the individual entries.

2.0.2. Analysis Lexicon Notation, Structure and Entries

The notational conventions for an analysis lexicon entry are presented in this subsection. We use a BNF-like specification here to show how entries are structured, but it should be understood that this specification is a human-readable

12Valency refers to the argument structure of a verb or a de-verbal noun. In semantic terms, a verb usually corresponds to a proposition, which has certain roles associated with it. The roles usually correspond to noun phrases or prepositional phrases, which appear in the syntax as arguments to the verb, such as the verb's subject or object. However, sometimes a role is filled with an understood reference which is not syntactically realized. For example, the English verb move corresponds to a proposition which has the following roles: agent, for the actor doing the moving; theme, for the object being moved; source, for the location being moved from; and goal, for the location being moved to. All of these may be syntactically realized, as in the sentence John moved the book from the chair to the table. The arguments corresponding to the source and goal roles may be left out of the sentence (John moved the book) but fillers for those roles will still be understood to exist, even if they are not specified. Now consider the English verb put, which corresponds to the same propositional structure. Put will not allow the syntactic realization of the source role: *John put the book from the chair to the table, but John put the book on the table. Neither will it allow the deletion of the argument corresponding to the goal role: *John put the book. Despite their semantic similarities, put and move have different valencies. Generally, the valency of a verb signifies those semantic roles which may be filled by actual syntactic arguments to the verb, the permissible lexical categories and features of these arguments (e.g., whether they may be nouns or prepositional phrases), and the order in which the arguments may appear in the given language.

13An explanation of the mapping rules appears below. The function of the KBMT-89 mapping rule interpreter is described in the paper "Analysis" by Morrison, Kee and Goodman in Part II of this issue.
convenience. The actual entries in the system are built up using the notation that results when the BNF specifications used here are fully expanded.

- **Miscellaneous characters**
  - ? = [optionality] 0 or 1
  - { + } = 1 or more
  - { * } = 0 or more
  - * = name of concept
  - ; = any comments
  - & = atomic value
- **<word>** expands to the string of an entry word. **<word>** corresponds to a terminal input string in a syntactic grammar. If a phrasal expression is written as a single string in a grammar (e.g., turn-on as one word), then it is treated as a single **<word>**.
- **<category>** expands to (CAT **<cat>**).
- **<cat>** expands to the category of an entry word. The name of the category of an entry word is the same as the category name used in the syntactic grammar and the concept lexicon. The category names for English include N, V, Adj, Adv, P, Det, Conjunct and so forth.
- **<inflection>** expands to (INFL [ + {<inf-
type> <form>}]). This structure is used for special-case (irregular) morphology only. During the expansion phase, the morphology package checks for **<inflection>** and uses any information it contains instead of the default morphological expansion. **<inflection>** can apply to verbs, adverbs, nouns, adjectives, etc., depending on the language.
- **<form>** expands to the string of some inflected form of the root word. The value contained in **<form>** is an actual, inflected input string. It is used to hold irregular forms, which will be used during expansion (instead of regular morphology) as the basis of new lexical entries. For example, **<form>** for the **<word>** see will contain saw (paired with the <inf-type> 'past'), and seen (paired with the <inf-type> 'past participle').
- **<inf-
type>** expands to the name of an inflection type. It indicates a type of inflected form such as 'past' or 'past participle' for a verb and 'singular' or 'plural' for a noun in English.
- **<inf-
form>** expands to (INFL-FORM <inf-type>). This is created when the expansion of the analysis lexicon is performed, and indicates an entry's inflection type. All inflected entries contain this information. If **<inf-form>** appears in a base lexicon entry, the entry passes into the dictionary completely unchanged from its base lexicon form.
- **<root>** expands to (ROOT &value). This will be created when the expansion of the analysis lexicon is performed if the expanded form is not the root form of the word. For example, with see as the root form, <word> = see \Rightarrow <word> = saw; and <root> = see for the past tense. But <word> = see \Rightarrow <word> = saw; and no <root> for the present tense (excluding third-person singular).
- **<features>** expands to (FEATURES ?<class-feature> { * <local-feat> } ?<all-features>). Here, **<features>** includes syntactic features and equations. These features can be defined either locally or non-locally. **<features>** will not appear in either the base or expanded lexicon for entries which have no syntactic features, e.g., adverbs and most prepositions in English.
- **<local-feat>** This designates locally specified features, and expands to (LOCAL { + (<feature> <value> ) } ) . Following is a fully-expanded example; valency and abbr are feature names, and ditrans and + are values:

```
(LOCAL (valency ditrans)
  (abbr +)
)
```
- **<feature>** The name of a feature.
- **<value>** The value of a named feature.
- **<class-feature>** expands to (CLASS { + <class-name-FEAT> }), which points to non-locally specified feature-value pairs.
- **<class-name-FEAT>** expands to the name of a syntactic feature class. Such a class specifies default feature-value pairs. If a feature is not specified locally in a lexical entry, but appears in the CLASS specified for that entry, then the expanded version of the entry will inherit the feature and its associated value from its CLASS.
- **<all-features>** expands to (ALL-FEATURES <f-structure>). This information is included in an entry when the morphological expansion and syntactic feature class inheritance are performed by the system. An entry which has not undergone morphological expansion will not contain <all-features> (e.g., adverbs and prepositions in English.)
- **<f-structure>** expands to the parser's internal representation of the f-structure representing the syntactic features of an entry. It is created during expansion, and includes all features which were added by the morphological rules and inherited from the syntactic feature classes. The **<f-structure>** does not include information from any of the mapping rules associated with the entry. The primary reason for including this f-structure is to enable the grammar writer to check the post-expansion features of an entry at a glance, which can be very helpful in debugging grammars.
• \texttt{<mapping>} expands to 
  (\texttt{MAPPING ?<class-map> {<local-map>}}). This is where all 
  mapping rule information for a lexical entry is stored. Mappings can be 
  defined either locally or non-locally. \texttt{<mapping>} will not appear in 
  either the base or expanded lexicon for entries which have no mapping 
  rules, e.g., prepositions in English.

• \texttt{<class-map>} expands to \texttt{(CLASS {<class-name>}}). 
  This simply specifies the notation used to reference one or more 
  classes of structural mapping rules.

• \texttt{<class-name>}. This expands to the name of a structural mapping 
  rule class. Mapping rule classes are a method of storing structural mapping 
  rules in an inheritance hierarchy (similar to the syntactic feature hierarchy 
  described above). Structural mapping rules are explained below under 
  \texttt{<structural-mapping>}. 

• \texttt{<local-map>} expands to \texttt{(LOCAL {<map>} ?<map-test>)}. Local 
  mapping rules are used for lexical mapping as well as structural mapping. 
  Local mappings are specific to the entry in which they appear, and 
  add information to the inherited mappings, but cannot override inherited 
  information.

• \texttt{<map>} expands to \texttt{<lexical-map> | <structural-map>}. This 
  distinction corresponds to our organization of mapping rules into two 
  broad categories, lexical and structural.

• \texttt{<lexical-map>} expands to 
  \texttt{<head-lexical-map> | <unhead-lexical-map>}. These 
  are the two basic types of lexical mapping rules.

• \texttt{<head-lexical-map>} expands to 
  \texttt{(HEAD <sem-head> {<slot-value-pair>})). \texttt{HEAD} is used 
  for lexical mapping, in which a \texttt{<word>} maps onto the head of a concept. 
  The \texttt{<slot-value-pair>} is used for adding finer-grained distinctions 
  than concept names alone will allow.

• \texttt{<unhead-lexical-map>} expands to 
  \texttt{(UNHEAD <slot-value-pair>}). \texttt{UNHEAD} mapping\footnote{Or \textit{embedded} mapping; these terms are synonymous. They are explained in the paper "Analysis," op. cit., Part II of this issue.} is used for 
  lexical mappings in which a \texttt{<word>} maps onto a slot and a concept.

• \texttt{<slot-value-pair>} expands to \texttt{(<sem-slot> <sem-value>)}.
  This is used to assign a value to a slot.

• \texttt{<sem-slot>} expands to the name of a slot in a concept lexicon entry.

• \texttt{<sem-value>} expands to an ontological value.\footnote{See the previous section, on the concept lexicon. An ontological value is a concept or a value-set element in the concept lexicon.}

• \texttt{<structural-map>} expands to 
  \texttt{(SLOT {<structural-mapping>})). \texttt{SLOT} is used to group 
  structural mappings, if the entry has more than one structural mapping.

• \texttt{<structural-mapping>} expands to a rule of the form: 
  \texttt{(<sem-slot> = (<syn-slot> {<syn-test>}))}. Structural 
  mappings associate syntactic functions with semantic relations. Some 
  examples of structural mappings include: (agent = subj), (theme 
  = obj) and (goal = (ppadjunct (prep = to))).

• \texttt{<syn-slot>} expands to the name of a slot in the syntactic grammar.

• \texttt{<map-test>} expands to \texttt{(<sem-map-test> | <syn-map-test>)}.
  These correspond to a semantic test and a syntactic test, respectively.
  These tests determine whether or not the system will apply the mapping 
  rules within the \texttt{LOCAL} where the test appears.

• \texttt{<sem-map-test>} expands to \texttt{(SEM-TEST {<sem-test>})}. 
  This tests semantic constraints on rule application.

• \texttt{<sem-test>} expands to \texttt{(SEM-SLOT <sem-value>)}. Here the 
  slots will be relation names, and the values will be ontological values. It 
  is unified with the semantic structure being built; if the unification fails, 
  none of the mapping rules inside the \texttt{LOCAL} where the test is located will 
  be applied.

• \texttt{<syn-map-test>} expands to \texttt{(TEST {<syn-test>})}. This tests 
  syntactic constraints on rule application.

• \texttt{<syn-test>} expands to \texttt{(SYN-SLOT <syn-value>)}. This test 
  is unified with the syntactic f-structure being built; if the unification fails, 
  none of the mapping rules inside the \texttt{LOCAL} where the test is located 
  will be applied. An example is the test for a passive feature structure: 
  \texttt{(passive = +)}.

• \texttt{<syn-value>} expands to a syntactic feature value.

\subsection*{2.0.3. Structure}

The general structure of an expanded analysis lexicon entry is shown below. 
In the course of analysis lexicon expansion, \texttt{<inf1-form> and <root>} will 
be added to those entries which are created by the morphology package 
during expansion. Currently, the Japanese analysis lexicon does not need to be 
expanded because the morphological derivation is done in the grammar. The 
English morphological functions are described below. Also, because features are 
incorporated into the grammar, they are not contained in the Japanese analysis 
lexicon.

\begin{verbatim}
("word"
  (CAT <cat>))
\end{verbatim}
nouns. Morphological rules expand this base lexicon by generating all of the
inflected forms of nouns and verbs.

The morphological rules for English do two things: They add and delete
letters to produce new lexical entries, and they add syntactic features to the new
entries so that the entries can be used with the English analysis grammar.

The input to the morphological rules is a file containing lexical entries for
uninflected nouns and verbs. Each lexical entry includes syntactic features which
are common to all inflected forms of the given word. Following is an example
of a lexical entry for an uninflected noun (before expansion):

```plaintext
("arrow" {CAT N}
    {FEATURES (COUNT YES) (PERSON 3) (MEAS-UNIT NO)
     (PROPER NO) (ROOT ARROW))
    {MAPPING (CLASS OBJECT-MAP)})
```

The rules that strip letters from and add suffixes to words are written in
a notation similar to that of phonological rules in linguistics. The rules are
mutually exclusive so that only one rule should be able to fire on any given
lexeme. The following is an example of a morphological rule that forms plural
nouns. Five of the lines are annotated with numbers, which correspond to
enumerated comments in the text that follows the rule.16

```plaintext
(N plural ; Part of speech and inflectional form
  (**" <> "s" / NOT-SIBILANTorOry _ #) ; (1)
  (**" <> "es" / SIBILANTorO _ #) ; (2)
  ("y" <> "ys" / V _ #) ; (3)
  ("y" <> "ies" / C _ #) ; (4)
  irregular ; (5)
  ((x0 number) = plural)
  ((x0 person) = 3)
  ((x0 count) = yes))
```

The rule says, in essence, that nouns become plural in the following ways:

1. If the last letter in the word is not a sibilant s, x, sh, ch, z, not o and not y, then add s to the lexeme, e.g., "computers" ->
   "computers" and "diskette" -> "diskettes".

2. If the last letter is a sibilant or o, then add es, e.g., "switch" ->
   "switches", "loss" -> "losses" and "potato" -> "potatoes".

---

16 Note in the rule that V is a variable that represents the set of English vowels; C represents the
   English consonants; SIBILANTorO represents the set of sibilants; and NOT-SIBILANTorOry
   represents the set of alphabetical characters excluding the sibilants o and y.
3. If the lexeme ends in y and is preceded by a vowel, then add s. (Actually, the program strips the y off and adds ys.) Thus, "key" -> "keys".

4. If the lexeme ends in y preceded by a consonant, then change the y to ies: "memory" -> "memories".

5. If the lexical unit is marked as irregular, do not strip or add anything. Instead, get the plural form from the INFL or CONJ slot of the uninflected word. For example, the analysis lexicon entry for mouse shows that the plural form is mice: ("MOUSE" (CAT N) (CONJ (PLURAL "mice"))...). Hence, "mouse" -> "mice".

These suffixation rules are accompanied by a list of equations. The final lexical entry for plural words will include the results of applying these equations, in addition to the features contained in the original uninflected lexical entry.

The equations serve two purposes. First, they add syntactic information about the inflected form of the word. This information is used in the analysis grammar. For example, the morphological rule that produces plural nouns also adds information about the noun's number; this information is used for checking subject-verb agreement in the grammar. Second, the equations can prevent incorrect words from being generated. Such errors can occur when the equations introduced by the morphology rule contain information that conflicts with information in the original, uninflected lexical entry.

These equations are the same as those used in the analysis grammar and any operator used in the analysis grammar can also be used in these morphology rules. (One can think of X0 as the FEATURE-STRUCTURE found in the lexical entry of a word.)

The three equations in the above example do the following:

1. Unify (number plural) with the feature-structure of the lexeme. Given that the lexical entry will not yet contain number information, this equation will add the feature-value pair to the feature structure.

2. Unify (person 3) with the feature structure of the lexeme.

3. Unify (count yes) with the feature structure of the lexeme. This feature is already in the unaugmented lexical entry of each noun. This unification prevents mass nouns (designated in contradistinction to count nouns: (count no)) from becoming pluralized as, for instance, in "knowledge" -> "knowledges".

The following, then, gives the result of a morphological expansion for a noun:

---

("arrow" (CAT N)
  (FEATURES ((COUNT YES) (PERSON 3) (MEAS-UNIT NO)
    (PROPER NO) (ROOT ARROW))
  (MAPPING (CLASS OBJECT-MAP)))
("arrows" (CAT N) (ROOT ARROW)
  (FEATURES ((COUNT YES) (PERSON 3) (MEAS-UNIT NO)
    (PROPER NO) (ROOT ARROW) (NUMBER PLURAL))
  (MAPPING (CLASS OBJECT-MAP)))

Passivization is a product of the morphology rules. Verbs are expanded by morphological rules, and all of their forms (infinite, 3rd person singular present tense, present participle, past participle and past tense) are put into the analysis lexicon. The morphological rules produce two past-participial lexical entries for each transitive verb: One contains the features of the original uninflected entry and the other has the feature (passive +) and (valency intrans) instead of the original (valency trans).

2.2. Lexical Mapping Rules

There are two types of rules that map between an input string and a concept: open-class lexical mapping rules and closed-class lexical mapping rules.

Open-class lexical items are defined to be nouns, verbs, adjectives and adverbs. All other parts of speech, for example, prepositions, determiners, conjunctions, etc., constitute the set of closed-class lexical items. However, some special items, such as phrasal verbs (i.e., verbs of phrase such as begin, continue, end, etc.); semi-functional verbs (do, make, proceed, etc.); and aspectual adverbs (already, always, still, etc.) are defined to be closed-class as well. Hence, the delineation between these two types of lexical items was determined by part of speech as well as by function.

2.2.1. Open-Class Lexical Mapping

The function of the lexical mapping rules is to unite an input word with a concept. Open-class rules are divided into those in which a word may be mapped onto a head of a concept, and those in which a modifier may be mapped onto a semantic slot and a head of a concept. In one such mapping, a word such as the English remove or the Japanese torinosoku represents a concept, such as *remove. This mapping is called head mapping and is illustrated in the following rule:

("torinosoku" (cat V)
  (mapping (local (head (*remove)))))

The second type of mapping occurs when a modifier maps onto a slot and a
concept. This mapping is called embedded mapping.\textsuperscript{18} For example, the noun phrase *sissemu yunito no deisuketto doraibu no rebaa ('the lever of the diskette drive of the system unit') contains modifiers, *sissemu yunito ('system unit') and deisuketto doraibu ('diskette drive'), for a head noun rebaa ('lever'). The head noun concept is *lever, and *lever is a part of *diskette-drive; in turn, *diskette-drive is a part of *system-unit. Therefore, the following output is expected:

\begin{verbatim}
[*LEVER
  (part-of [*DISKETTE-DRIVE
   (part-of [*SYSTEM-UNIT])))]
\end{verbatim}

In order to map *system-unit onto the slot part-of in *diskette-drive, and *diskette-drive onto the slot part-of in the head noun concept *lever, the slot must be defined in the mapping rule of the modifier. We used two embedded mapping rules to define the mappings of the modifiers. This reduces the number of ambiguities because the system does not try to map all the possible semantic slots in the head noun concept.

The embedded mapping rules for *diskette-drive and *system-unit are:

\begin{verbatim}
(*deisuketto doraibu
  (cat N)
  (mapping (local (unhead (part-of [*DISKETTE-DRIVE])))))))

(*sissemu yunitito
  (cat N)
  (mapping (local (unhead (part-of [*SYSTEM-UNIT])))))))
\end{verbatim}

In both headed and embedded mapping, some words map onto general concepts, which must be further specified to capture the meaning of a word as completely as possible. In these cases, a value can be assigned to a semantic slot to make the concept more specific. For example, the Japanese \textit{akai} ('red') maps onto the concept *color. In that *color is a general concept for all colors, distinctions need to be made among all color terms, such as, in English, \textit{red}, \textit{yellow} or \textit{blue}. So the mapping of \textit{akai} looks like this:

\begin{verbatim}
(*akai
  (cat ADJ)
  (mapping
   (local (head (*COLOR (@attribute-range "red")))))
\end{verbatim}

For the convenience of the rule writer, the lexical mapping rules can be written in the concept lexicon as well. The notation is slightly different than the one in the analysis lexicon, but the lexical mapping rules in the concept lexicon are converted automatically into the analysis lexicon format during loading.

The format of the lexical mapping rules in the concept lexicon is shown below. Keeping in mind the previous footnote, observe that \textit{head} indicates Japanese head mapping and \textit{unhead} is Japanese embedded mapping. For English, these would be \textit{ehead} and \textit{eunhead}.

\begin{verbatim}
(HEAD-OF-CONCEPT
  (jhead ((word (cat X)))
   (jhead ((word (cat X)) (@slot-name @value)))
   (junhead ((word (cat X)) (@slot-name @value)))
   (junhead ((word (cat X)) (@slot-name @value)))
)
\end{verbatim}

The example below illustrates a lexical mapping rule in the concept and analysis lexicons.

\begin{verbatim}
; In the concept lexicon:

(REMOVE
  (jhead ((torinozoku (cat V)))))

; In the analysis lexicon:

("torinozoku"
  (cat V)
   (mapping (local (head (*remove))))))
\end{verbatim}

\subsection{Closed-Class Lexical Mapping}

Closed-class lexical items are non-productive sets of words with a small finite number of members. These words are sometimes called function words because they are generally non-referential. As a rule, the number of closed-class items in a language does not increase. In other words, while one can coin new nouns and verbs, new prepositions or numerals are very seldom, if ever, coined.

Analysis-lexicon entries for closed-class items usually carry information about some properties of the concept tokens instantiated for the open-class words in the sentence. It follows that closed-class items do not usually map into domain concepts. In KBMT-89, however, we have created abstract concepts in the concept lexicon for some of these words, thus treating them as regular open-class items. The reasons for this decision turn largely on the relative ease and difficulty of implementing the special treatment for closed-class items in the analysis environment. Some of the closed-class items are treated at syntactic parsing time and their contributions to the overall meaning representation of the sentence take the form of certain property-value pairs that are included in the representations of syntactic constituents. We will illustrate this mechanism below. Some other closed-class items add property-value pairs to the interlingua representations of the meanings of open-class items. The emphasis in this
description is on the closed-class items from the KBMT-89 corpora (personal-computer instruction manuals). Treatment of additional closed-class items is illustrated whenever a solution for them could be found within the bounds of our computational architecture and knowledge-interaction constraints. The items and item classes not in the KBMT-89 corpora are marked with a #.

It should be stated quite prominently that the decisions made with respect to the closed-class items are not necessarily informed by the wealth of the findings about some of them (e.g., quantifiers and determiners) in the linguistic literature. Our determination was very often determined by the size and nature of our corpora, by the necessary granularity of description and even by the constraints imposed by the formalisms and algorithms used in our computational implementation. We intend to improve our treatment of closed-class items by creating or incorporating more advanced (micro)theories of these phenomena.19

2.3.1. Demonstratives, Determination and Quantifiers

These items are treated at parsing stage. The features that they add to the parser results are given in the table on the following page.

2.3.2. Numerals

We have divided numerals into five groups:

- number bullets (L. Turn on the power)
- cardinals (3 books)
- measurement cardinals (A six-foot cable)
- virtual ordinals (disk drive number 1)
- #ordinals (second time.)

The parser places number bullets in the field number-bullet which is inserted unchanged into the ILT at the clausal level. Cardinals fill a quantity slot in the f-structure. This slot is mapped into the cardinality slot in the ILT. The measurement expressions in the corpus are hyphenated forms such as 128K-byte, six-foot and so forth. The parser converts each such into two property-value pairs, e.g.,

\[(\text{root foot})
\quad (\text{quantity 6})\]

In future implementations the augmentor will convert non-standard units into our interlingua standard. Each type of measurement has a standard in the concept lexicon. For example, length is always converted into meters, volume

<table>
<thead>
<tr>
<th>Item</th>
<th>Feature-Value Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>a/an</td>
<td>(reference indefinite) (number singular)</td>
</tr>
<tr>
<td>some</td>
<td>(reference indefinite) (number plural)</td>
</tr>
<tr>
<td>the</td>
<td>(reference definite) (specific +)</td>
</tr>
<tr>
<td>this</td>
<td>(reference definite) (specific +) (number singular) (near +)</td>
</tr>
<tr>
<td>that</td>
<td>(reference definite) (specific +) (number singular) (near -)</td>
</tr>
<tr>
<td>these</td>
<td>(reference definite) (specific +) (number plural) (near +)</td>
</tr>
<tr>
<td>those</td>
<td>(reference definite) (specific +) (number plural) (near -)</td>
</tr>
<tr>
<td>any/any of</td>
<td>(specific -) (r-quantifier universal)</td>
</tr>
<tr>
<td>specific</td>
<td>(specific +)</td>
</tr>
<tr>
<td>either</td>
<td>(specific +) (dual +) (r-quantifier universal)</td>
</tr>
<tr>
<td>additional</td>
<td>(new +)</td>
</tr>
<tr>
<td>other</td>
<td>(new +)</td>
</tr>
<tr>
<td>further</td>
<td>(new +)</td>
</tr>
<tr>
<td>#another</td>
<td>(number singular) (new +)</td>
</tr>
<tr>
<td>same</td>
<td>(new -)</td>
</tr>
<tr>
<td>no</td>
<td>(r-quantifier negation)</td>
</tr>
<tr>
<td>#none</td>
<td>(r-quantifier negation)</td>
</tr>
<tr>
<td>#some of</td>
<td>(r-quantifier existential)</td>
</tr>
<tr>
<td>#a number of</td>
<td>(r-quantifier existential)</td>
</tr>
<tr>
<td>multiple</td>
<td>(r-quantifier existential)</td>
</tr>
<tr>
<td>a variety of</td>
<td>(r-quantifier existential)</td>
</tr>
<tr>
<td>several</td>
<td>(r-quantifier existential)</td>
</tr>
<tr>
<td>many</td>
<td>(r-quantifier existential)</td>
</tr>
<tr>
<td>#much</td>
<td>(r-quantifier existential)</td>
</tr>
<tr>
<td>all</td>
<td>(r-quantifier universal)</td>
</tr>
<tr>
<td>whole</td>
<td>(r-quantifier universal)</td>
</tr>
<tr>
<td>each</td>
<td>(r-quantifier universal)</td>
</tr>
<tr>
<td>every</td>
<td>(r-quantifier universal)</td>
</tr>
<tr>
<td>#few</td>
<td>(r-quantifier minority)</td>
</tr>
<tr>
<td>a little</td>
<td>(r-quantifier minority)</td>
</tr>
<tr>
<td>most</td>
<td>(r-quantifier majority)</td>
</tr>
<tr>
<td>approximately</td>
<td>(precision -)</td>
</tr>
<tr>
<td>about</td>
<td>(precision -)</td>
</tr>
<tr>
<td>#precisely</td>
<td>(precision +)</td>
</tr>
<tr>
<td>#exactly</td>
<td>(precision +)</td>
</tr>
</tbody>
</table>

19For a discussion of microtheories, see the paper "Knowledge Representation Support" by Nirenburg and Levin in this issue.
into liters, etc. The information is then placed in the *conversion frame appropriate to the units used in the input. In this way, we retain the original numerals and units and still have a standard in which to express slot restrictions. Depending on the target language, the generator can use either the original units or the standard units. The result of the conversion can appear in the ILT as in the following:

(conversion-foot-meter
  (source-number 6)
  (source-units foot)
  (goal-number 1.83)
  (goal-units meter))

Virtual ordinals are placed in the syntactic f-structure field post-nom-mod. If the word number or any variation thereof occurs, it is discarded. The root of the post-nom-mod field is mapped into a value of the concept lexicon property object-label.\textsuperscript{20}

2.3.3. Prepositions

Prepositions are included in a field within the f-structure for prepositional phrases. No other semantic information is given, except in the case of the English by where one parse of the f-structure in which it is contained is labelled obl-agent.

Prepositions are categorized by the semantic roles which they indicate; they direct the system to relate the superordinate f-structure to their arguments by a particular relation. The right-hand side of a lexical rule for a preposition contains a cat field and a relation field, so that the rules for after would appear as follows:

after-1 (cat P) (relation after)

after-2 (cat P) (relation under)

That these words are prepositions signals the structural mapping rules to use this relation to relate the parent f-structure to the argument f-structure.

2.3.4. Aspectual Adverbs

Syntactically, these are all regular adverbs. They appear in the f-structure as advadjunct. For example, yet is represented as

(advadjunct (root yet)))

In the parser output, aspectual adverbs fill the aspect slot. For example, yet appears as:

\textsuperscript{20}This strategy is also used to handle the identifying letter in phrases like disk drive A.

\begin{tabular}{|l|l|}
\hline
\textbf{Item} & \textbf{Feature-Value Pairs} \\
\hline
always/every time & (*iteration (%attribute-range always)) \\
(once) again & (*iteration (%attribute-range again)) \\
usually/often/frequently & (*iteration (%attribute-range &frequent)) \\
first & (*phase (%attribute-range &begin)) \\
rarely & (*iteration (%attribute-range &frequent)) \\
one & (*iteration (%attribute-range &frequent)) \\
immediately & (*phase (%attribute-range &end)) \\
yet/already & (*phase (%attribute-range &end)) \\
#ever & (*iteration (%attribute-range &frequent)) \\
\hline
\end{tabular}

2.3.5. Attitudinal Adverbs

These adverbs convey the speaker's attitude to a proposition. They appear in the f-structure as regadverbs and are divided into two classes: those that are semantically the same as using the modal might, and those that are not.

The former are processed by setting the modal feature possibility to + in the ILT, and preventing concept lexicon mapping. The members of this class are #perhaps, #maybe and #possibly.

All other attitudinal adverbs trigger the creation of an attitude feature at the clausal level of an ILT. Possible values are negative, positive, probable, unexpected, only, and finally. These are presented below:

\begin{tabular}{|l|l|}
\hline
\textbf{Item} & \textbf{Feature-Value Pairs} \\
\hline
#unfortunately & (attitude negative) \\
#lucky & (attitude positive) \\
#probably & (attitude probable) \\
#particularly & (attitude unexpected) \\
#even & (attitude unexpected) \\
#only & (attitude only) \\
just & (attitude only) \\
at last & (attitude finally) \\
finally & (attitude finally) \\
\hline
\end{tabular}

2.3.6. Discourse Cohesion Markers

Discourse cohesion markers in our corpora can be divided syntactically into coordinate conjunctions, subordinate conjunctions, regular adverbs, a prepo-
2.3.7. Modals

Most modals cause the parser to create a modal feature filled with either ability, necessity, conditional or possibility. The only exception to this is the English will which sets the tense feature to future. These features are passed unchanged to the ILT:

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>can, be able</td>
<td>ability</td>
</tr>
<tr>
<td>could</td>
<td>ability</td>
</tr>
<tr>
<td>may</td>
<td>possibility</td>
</tr>
<tr>
<td>might</td>
<td>possibility</td>
</tr>
<tr>
<td>must</td>
<td>necessity</td>
</tr>
<tr>
<td>should</td>
<td>necessity</td>
</tr>
<tr>
<td>would</td>
<td>conditional</td>
</tr>
<tr>
<td>will/shall</td>
<td>(tense future)</td>
</tr>
</tbody>
</table>

2.3.8. Pronouns

The f-structures record all relevant constraint information on a pronoun and name the root pro. If the value for a certain feature is not determined, that feature is left out (e.g. number for you). Thus, it will translate to the following f-structure:

```
{root pro}
{person 3}
{number singular}
{gender neuter}
{human --}
```

Our system handles the following pronouns:

<table>
<thead>
<tr>
<th>Pronoun</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>nominative</td>
<td>I, you, he, she, it, we, they</td>
</tr>
<tr>
<td>objective</td>
<td>me, you, him, her, it, us, them</td>
</tr>
<tr>
<td>possessive</td>
<td>my, your, his, her, its, our, their</td>
</tr>
<tr>
<td>absolute possessive</td>
<td>mine, yours, his, hers, its, ours, theirs</td>
</tr>
<tr>
<td>demonstratives</td>
<td>this, that, these, those</td>
</tr>
</tbody>
</table>

Non-second person nominative, objective, possessive, and absolute possessive pronouns are mapped to `$pronom$` with the embedded features person, gender, and number.

Demonstrative pronouns are mapped to `$pronom$`, but they trigger the features number and near.

Quantified pronouns (e.g. everybody, anything, nothing) are treated as nouns and handled through our regular open-class mapping facilities. They instantiate either an object or intentional-agent frame and add additional feature-value pairs to those frames. For example, the pronoun anything would instantiate an *object* frame with embedded *quantifier* and specific fields, as follows:

```
{(*object
 {r-quantifier universal}
 {specific --})}
```

2.4. Structural Mapping Rules

The structural mapping rules²¹ map grammatical functions within the f-structures, such as subject and object, onto semantic roles in concept structures. For example, the f-structure of the Japanese sentence *Deisuketto doraibu kara deisuketto o nukito te kudasai* ("remove the diskette from the diskette-drive"), is as follows:

```
{TIME ((ROOT PRESENT))) (MOOD ((ROOT IMP)))
(PPADJUNCT
 {PART KARA} {CAT N} {ROOT DEISUKETTODORABI})
(OBJ {INH -->} {CASE O} {CAT N} {ROOT DEISUKETTO})
(CAUSATIVE -->) {OBJ-CAUSE O} {SUBJ-CAUSE GA} {PASSIVE -->}
(SUBCAT TRANS) {FORMAL -->}
(VTYPE V-5DAM-R) {CAT V} {ROOT NUKITORU})
```

²¹These and related rules are considered from the perspective of the KBMT-89 analyzer in the paper "Analysis," op. cit., Part II of this issue.
This f-structure contains only syntactic information and must be mapped onto the semantic structure. The verb, *torinokoru*, subcategorizes agent, theme, and source. The subject maps onto agent, the syntactic object maps onto theme and *ppadjunct*, with the particle *kara*, maps onto the source. Therefore, the structural mapping for this sentence would look like this:

```
(mapping (local (slot
  (agent = subject)
  (theme = object))
  (source = *ppadjunct (part = kara))))
```

And the following is the expected semantic parser output of the sentence:

```
[*REMOVE
  (MOOD IMP)
  (PASSIVE -)
  (THEME [*DISKETTE])
  (SOURCE [*DISKETTE-DRIVE])
  (TENSE PRESENT)]
```

Because the same structural mapping can be used for other verbs of the same class, such as *haasu* (‘detach’) and *nukitoru* (‘remove’), these rules can be written once and stored in the mapping rule hierarchy. Then the rules can be inherited by the analysis lexicon from the structural mapping rules.

### 2.4.1. Notation, Structure and Examples of Structural Mapping Rules

The notation of the structural mapping rules is as follows. Basically, the notation is the same as `<mapping>` in the analysis lexicon. The difference is that `<mapping>` in the structural mapping rules does not contain the lexical mapping; hence there are no heads and unheads.

- `<structural-mapping-rule>`

  ```
  (<class-name-MAP> <structural-mapping>).
  ```

- `<class-name-MAP>` = The name of a class for structural mapping rules, e.g., *perception-trans-MAP*.

- `<mapping>` = 

  ```
  (MAPPING {+ (?<class-map> ?<local-map>)}).
  ```

- `<class-map>` = 

  ```
  (CLASS {+ <class-name-MAP>}).
  ```

- `<local-map>` = 

  ```
  (LOCAL {+ (?<syn-test> ?<syn-map>)}).
  ```

- `<syn-test>` = 

  ```
  (TEST {+ <syntactic-feature>}). This is a syntactic test; e.g., (TEST (passive = +)).
  ```

- `<syn-map>` = 

  ```
  (SLOT (+( <structural-mapping>))). E.g.:
  ```

  ```
  (SLOT (+(<slot = agent)
    (obj = theme))
  ```

- `<structural-mapping>` = 

  ```
  (sem-slot) = (+(<syn-slot> (* <syn-test>)))).
  ```

  For instance:

  ```
  (agent = subj)
  (theme = obj)
  ```

- `<syn-test>` = 

  ```
  (+(<syn-slot> <syn-value>)). For example:
  ```

  ```
  (goal = (ppadjunct (prep to)))).
  ```

- `<sem-slot>` = Name of slot for semantic lexicon entry.

- `<syn-slot>` = Name of slot for syntactic structure.

- `<syn-value>` = Syntactic structure value.

Now we can display the general structure of the structural mapping rules:

```
(<class-name-MAP>
  (MAPPING
    (CLASS {+ <class-name-MAP>})
    (*
      (?<LOCAL
        (?<TEST {+ <test>})
        (SLOT {+ <structural-mapping>})))))
```

And we can use the following English example to illustrate how these rules work. The structural mapping rule for the English verb *change* is:

```
(*CHANGE* (cat V)
  (FEATURES (class CAUS-INCHO-VERB-FEAT))
  (MAPPING (local (slot
    (goal = (ppadjunct (prep to))))
    (class C-I-VERB-MAP)))
```

The verb belongs to the causative-inchoative verb class (*c-i-verb-map*). It can be intransitive, as in the English sentence *Therefore, even if you push the Kana key, the square on the screen will not change*. In this sentence the appropriate concept is *to-replace-display-object*. Because the subject NP (*the square*) is inanimate, it should be mapped to the semantic slot theme. The necessary mapping rule is inherited by change from the following:

```
(C-I-VERB-MAP
  (MAPPING
    (local
      (test (passive = -) (valency = intrans))
      (slot (theme = subj))
      (class CB-TH-VERB-MAP CB-TR-VERB-MAP CB-TP-VERB-MAP)))
```
The verb change can also be transitive, as in the English sentence Online button changes the online mode and offline mode. Here, the appropriate concept is again *to-replace-display-object. Since the subject NP (online button) is inanimate, it should be mapped to the semantic slot caused-by. (Only animate entities and the computer can be agents). The object NP (the online mode and offline mode) maps into the theme semantic slot. The necessary rule is inherited by the c-i-verb-class from the following:

\[(C\text{-TH-VERB-MAP})\]
\[
\begin{align*}
&\text{(mapping)} \\
&(\text{(local)}) \\
&\text{(test (passive = -)) (valency = trans)} \\
&\text{(slot (caused-by = subj))} \\
&\text{(theme = obj)}) \\
&(\text{(local)}) \\
&\text{(test (passive = +))} \\
&\text{(slot (caused-by = ob-l-agent))} \\
&\text{(theme = subj))} \\
&\text{(sem-test (focus theme))} \\
&(\text{(local)}) \\
&\text{(test (passive = +))} \\
&\text{(slot (theme = subj))} \\
&\text{(sem-test (focus theme))} \\
&\text{(caused-by unknown))} \\
&(\text{class AG-TH-VERB-MAP)})
\end{align*}
\]

This rule is also used for the sentence, 6. To change the top of page setting, repeat the procedure in steps 1 - 5. Here the object NP (top of page setting) is assigned to theme; there is no subject. The appropriate concept is *change-top-of-page-setting.

3. THE GENERATION LEXICONS

The KBMT-89 generation lexicons are used in the generation phase by the lexical selection module. The module accesses them to determine, for a target language, the correct open-class lexical items for a given interlingua text. These lexicons contain all open-class items (nouns, verbs, adjectives and adverbs) that are used in the project corpus.

Recall from the previous section that the analysis lexicons include open-class lexical items, closed-class items and structural mapping rules. The generation lexicons, in contrast, do not include closed-class items or structural mapping rules. There are several reasons for the difference. One is trivially a function of the different roles and analysis and generation. Another is that the two lexicons were developed independently and brought together in KBMT-89.

It is more difficult to design a generation lexicon and mapping rules if the generation grammar is based on a non-logical grammar. This is so because we have to guarantee that the lexicon and rules successfully recover f-structures — which are the same as those in the analysis grammars — from the ILT. If we design an optimized generation grammar, f-structures as the input to the grammar will be much simpler; that is, they will not contain unnecessary analysis information. Too, the design of the generation lexicon and mapping rules will therefore be simplified.

3.1. Generation Lexicon Entries

Each entry in the lexicon is defined as a FRAMEKIT frame for each content word by associating syntactic words with each semantic concept. Generation lexicon entries form a set of vocabularies which we use to compose sentences. Generation lexicon entries can be automatically obtained from lexical entry values of Ontos concepts. Additionally, valency of verbs, morphological information and subcategorization information are manually added to the entries.

Generation lexicon entries contain a frame name, a meaning pattern, a lexeme, syntactic features and other data, in the following format:

\[
\text{(make-frame} \\
\text{<frame-name>} \\
\text{<meaning-pattern>} \\
\text{<lexeme>} \\
\text{<syntactic-features>} \\
\text{<subcategorization-information>} \\
\text{<morphological-information>})}
\]

Following is an example of a Japanese generation lexicon entry, for arawareru (*to appear*):

\[
\text{(make-gl arawareru} \\
\text{<is-token-of (value "appear")>} \\
\text{<syntactic-info (cat v)>} \\
\text{<features (cat v) (root arawareru)>})
\]

generation, they are stored in a file that is accessed by the generator’s mapping-rule interpreter. Closed-class items are actually mapping rules that, using the same notation as structural mapping rules, create an f-structure for each closed-class item that is to be generated.

24. The paper “Analysis and Generation Grammars” in this issue discusses the differences between the English and Japanese grammars, especially as regards bi-directionality. Note particularly the mention of the tradeoffs entailed by each approach.

25. For example, subj-case and obj-case slots are attached to each verb in the analysis grammar to handle active, passive and causative sentences by matching these slot values with case markers of nouns. Once an f-structure is obtained, they are entirely irrelevant because we know what the case markers are, and correct case markers are available from case slots of these case fillers. Thus we can generate a sentence from an f-structure without the subj-case and obj-case slots.
Lexicons

3.2. Generation Mapping Rules

Generation mapping rules describe a mapping from semantic slot-value pairs to syntactic slot-value pairs. These rules consist of the following sets of sub-rules:

- Rules to determine tense, mood and modals.
- Rules which combine relative clauses with nouns; embedded clauses with main verbs; and subordinate clauses with main clauses and conjunctions.
- Rules which combine the f-structures of compound nouns.

To illustrate, the English generation mapping rule shown below can be used to create an f-structure for the closed-classaspectual adverbs yet or already depending on the input ILT.

(maprule e *PROPOSITION :any (if (equal $frame.completion 'yes)
  t
  nil)
  (lexpr (if (equal $frame.p-quantifier 'negation)
    'yet
    'already))
  =>
  (slot advadjunct)
  (list !false (cons '(position end !slots))))

The rule's contents are interpreted as follows:

1. This is an English mapping rule (maprule e).
2. It is applied to the *PROPOSITION frame of the ILT.
3. The rule can be used in conjunction with any (:any) mapping rule that applies to that frame.
4. If the value of the completion slot in this frame of the ILT is yes, then perform the mapping that follows; otherwise do not.
5. Create the slot advadjunct in the f-structure being built.
6. If the value of the p-quantifier slot of the frame is negation, give this slot the value ((root yet)); otherwise, assign the value ((root already)).

The annotated example at the end of this section provides an extended illustration of the role of the generation lexicon in generation mapping.

26 The annotated example at the end of this section provides an extended illustration of the role of the generation lexicon in generation mapping.

27 These rules are discussed in greater detail in the paper "Generation" by Nyberg et al. in Part II of this issue.
7. Lastly, add (position end) to the list that is the value of adverbial.

The resulting (truncated) f-structure is then either of the two below:

{...  
  (adverbial (position end) (root already))  
  ...}

{...  
  (adverbial (position end) (root yet))  
  ...}

The structural mapping rules map the ILT slots to appropriate syntactic slots to create an f-structure; this f-structure is then the input to the syntactic generation module, where a grammatical string is produced.

The remainder of this section presents the ILT for the Japanese sentence *Sin-dan tesuto sekusyon o mite kudasai* ('Please see the diagnostic testing section') and an annotated trace showing the f-structure output of the various mapping rules in the generation of the sentence. The ILT:

(dolist (frame '  
  '(clause1 proposition1 role1 role2 role3 role99  
    speech-act1 aspect1 timel))  
  (erase-frame frame))

(make-frame clause1  
  (clause1 (value clause1))  
  (clause-response (value role0))  
  (proposition-id (value proposition1))  
  (speech-act-id (value speech-act1)))

(make-frame proposition1  
  (ilt-type (value role))  
  (proposition-id (value proposition1))  
  (clause-id (value clause1))  
  (is-token-of (value *proceed-through-text))  
  (aspect (value aspect1))  
  (time (value (after timel)))  
  (agent (value role5))  
  (goal (value role2)))

(make-frame role1  
  (ilt-type (value role))  
  (role-id (value role1))  
  (clause-id (value clause1))  
  (is-token-of (value *reader))  
  (reference (value definite)))

(make-frame role2  
  (ilt-type (value role))  
  (role-id (value role2))  
  (clause-id (value clause1))  
  (is-token-of (value *section)))

 Lexicons

{(string-is (value Diagnostic_Testing))  
  (convey (value role3))  
  (reference (value definite))}

(make-frame role3  
  (ilt-type (value role))  
  (role-id (value role3))  
  (clause-id (value clause1))  
  (is-token-of (value *test))  
  (end (value no))  
  (purpose (value role99))  
  (reference (value indefinite)))

(make-frame role99  
  (role-id (value role99))  
  (ilt-type (value role))  
  (is-token-of (value *diagnose)))

(make-frame speech-act1  
  (speech-act (value request-action))  
  (direct? (value yes))  
  (speaker (value author))  
  (hearer (value role1))  
  (time (value timel)))

(make-frame aspect1  
  (phase (value begin))  
  (iteration (value 1)))

(make-frame timel  
  (is-token-of (value *time)))

Now the trace below displays a series of five successful rule applications.  
(Inapplicable rules are deleted for cogency.) Each instance of 'Firing Rule'  
designates an attempt to apply a rule.

Frame: ROLE3

Lex: TESTING

FS.: ((ROOT TESUTOSURU) (CAT N))

: Here, a role3 frame is found and realized as  
: a generation lexicon entry "testing" whose  
: features values are (ROOT TESUTOSURU) (CAT N))

: The following mapping rule determines if there is  
: a modifier of a role frame. If there is, the  
: syntactic structure represented by the role frame  
: is marked as a compound noun and the f-structure  
: of the modifier (filler of "attached-to", "convey",  
: "for-purpose", etc.) is mapped to an  
: "adjunct" slot in the f-structure of the role frame.

{DEFRULE J *ROLE
Predicate returns ROLE3.
Slot XADJUNCT added.
Feature (COMPOUND CN) added.

;; Here, the generator reports that it has found
;; role99 to be a modifier of the role3 frame. It
;; then returns the following new f-structure of role3:

(COMPOUND CN) (XADJUNCT ((ROOT SINDANSURU) (CAT N)))
(ROOT TESUTOSURU) (CAT N)

------------------------------
Firing Rule
------------------------------

Frame: ROLE2
Lex: SECTION
FS: ([VTYPE V-IDAN] (SUCAT TRANS) (ROOT MIRU) (SUBJ-CASE GA) (OBJ-CASE O) (CAT V))
;; The following rule maps semantic "source" and
;; "goal" case fillers into the syntactic slots
;; PFSJUNCT and OBJ respectively.
(DEFRULE J *PROPOSITION
  :ANY
  (OR (FRAME-ACCESSOR FRAME 'ATTACHED-TO)
       (FRAME-ACCESSOR FRAME 'CONVEY)
       (FRAME-ACCESSOR FRAME 'FOR-PURPOSE)
       (FRAME-ACCESSOR FRAME 'LABEL)
       (FRAME-ACCESSOR FRAME 'MATERIAL)
       (FRAME-ACCESSOR FRAME 'PROPERTY)
       (FRAME-ACCESSOR FRAME 'PURPOSE))
  ((SLOT ATTACHED-TO) => (SLOT XADJUNCT))
  ((SLOT CONVEY) => (SLOT XADJUNCT))
  ((SLOT FOR-PURPOSE) => (SLOT XADJUNCT))
  ((SLOT LABEL) => (SLOT XADJUNCT))
  ((SLOT MATERIAL) => (SLOT XADJUNCT))
  ((SLOT PROPERTY) => (SLOT XADJUNCT))
  ((SLOT PURPOSE) => (SLOT XADJUNCT))
  (FEATURE COMPOUND CN))

Predicate returns T.
Slot OBJ added.

;; The generator finds that role3
;; fills the "goal" slot of proposition1.
;; The f-structure of role3 is therefore mapped to
;; the "obj" slot in the f-structure of proposition1:

(OBJ
  (COMPOUND CN)
  (XADJUNCT
    ((COMPOUND CN) (XADJUNCT ((ROOT SINDANSURU) (CAT N)))
     (ROOT TESUTOSURU) (CAT N)))
  ([VTYPE V-IDAN] (SUCAT TRANS) (ROOT MIRU) (SUBJ-CASE GA) (OBJ-CASE O) (CAT V)))
Frame: PROPOSITION1
Lex: SEE
FS: ((OBJ
   ((COMPOUND CN)
      (KADJUNCT
         ((COMPOUND CN) (KADJUNCT ((ROOT SINDANSURU) (CAT N)))
          (ROOT TESUTOSURU)
          (CAT N)))
         (ROOT SEKUSYON) (CAT N)))
         (VTYPE V-1DAN) (SUBCAT TRANS) (ROOT MIRU) (SUBJ-CASE GA)
         (OBJ-CASE O) (CAT V))
   (CAT V))
   ;; The rule below can map a response ("yes" or "no")
   ;; or a number ("1.", "2.", ...) associated with
   ;; the sentence. It also attaches the
   ;; (formal +) feature to each proposition frame:
   (DEFRULE J *PROPOSITION
      :ARY
      T
      ((SLOT CLAUSE-RESPONSE) => (SLOT SENT-TAG))
      ((SLOT NUMBER-BULLET) => (SLOT SENT-NUM))
      (FEATURE FORMAL +))

Predicate returns T.
Feature (FORMAL +) added.

   ;; Since the proposition1 frame has no
   ;; CLAUSE-RESPONSE or NUMBER-BULLET, the
   ;; mapping rule attaches only
   ;; the (formal +) feature:
   ((FORMAL +)
      (OBJ
       ((COMPOUND CN)
          (KADJUNCT
             ((COMPOUND CN) (KADJUNCT ((ROOT SINDANSURU) (CAT N)))
              (ROOT TESUTOSURU)
              (CAT N)))
             (ROOT SEKUSYON) (CAT N)))
             (VTYPE V-1DAN) (SUBCAT TRANS) (ROOT MIRU) (SUBJ-CASE GA)
             (OBJ-CASE O) (CAT V))
       (CAT V))

LEXICONS

(((COMPOUND CN) (KADJUNCT ((ROOT SINDANSURU) (CAT N)))
  (ROOT TESUTOSURU)
  (CAT N)))
((ROOT SEKUSYON) (CAT N))
(VTYPE V-1DAN) (SUBCAT TRANS) (ROOT MIRU) (SUBJ-CASE GA)
(OBJ-CASE O) (CAT V))

;; This final rule determines the f-structure’s
;; "mood" slot value:
(DEFRULE J *PROPOSITION
   :ARY
   T
   ((EXPR
      (COND ((AND
       (EQUAL (FRAME-ACCESSOR FRAME 'CLAUSEID
         'SPEECHACTID 'SPEECH-ACT)
          'COMMAND)
       (EQUAL (FRAME-ACCESSOR FRAME 'CLAUSEID
         'SPEECHACTID 'DIRECT?)
          'YES))
       'IMP)
       ((EQUAL (FRAME-ACCESSOR FRAME 'CLAUSEID
         'SPEECHACTID 'SPEECH-ACT)
          'REQUEST-INFO)
       'QUES))
      (T 'DEC)))
   => (SLOT MOOD))
   ;; Proposition1 has a speech-act frame whose
   ;; semantic slots include [speech-act (value command)]
   ;; and [direct? (value yes)].
   ;; The mapping rule thus adds (mood ((root imp)))
   ;; to the f-structure:

Predicate returns T.
Slot (MOOD ((ROOT IMP))) added.

;; Hence the final f-structure:
((MOOD ((ROOT IMP))) (FORMAL +)
   (OBJ
    ((COMPOUND CN)
      (KADJUNCT
       ((COMPOUND CN) (KADJUNCT ((ROOT SINDANSURU) (CAT N)))
        (ROOT TESUTOSURU)
        (CAT N)))
       (ROOT SEKUSYON) (CAT N)))
       (VTYPE V-1DAN) (SUBCAT TRANS) (ROOT MIRU) (SUBJ-CASE GA)
       (OBJ-CASE O) (CAT V)))
REFERENCES


