The Subworld Concept Lexicon and the Lexicon Management System

Sergei Nirenburg

Center for Machine Translation
Carnegie Mellon University

Victor Raskin

Natural Language Processing Laboratory
Purdue University

Natural language processing systems require three different types of lexicons: the concept lexicon that describes the (sub)world ontology and the analysis and generation lexicons for natural languages. We argue that the acquisition of the concept lexicon must precede any lexical work on natural language and that a comprehensive lexicon management system (LMS) is necessary for lexicon acquisition in large-scale applications. We describe the interactive concept lexicon acquisition module of the LMS for TRANSLATOR, a knowledge-based, sublanguage-oriented machine translation project.

This project belongs to two of the fastest growing fields of computational linguistics and artificial intelligence in general: the lexicon and knowledge acquisition for AI systems.

The work in lexicon has centered on a) studies concerned with the utilization of conventional human-oriented dictionaries, newly available in machine-readable form, for computational tasks (e.g., Amsler 1984a,b; Chodorow et al. 1985; Ahlswede 1985; Markowitz et al. 1986) improving the ancillary capabilities for lexicon systems, such as, for instance, morphological processors and descriptions (e.g., Nirenburg and Ben Asher 1984; Byrd et al. 1986; Boguraev et al. 1987); c) hand-building of lexicons necessary for natural language systems, often with considerations about extensibility (e.g., Zernik and Dyer 1985; Bessemer and Jacobs 1987). An interesting perspective on the field is given in Miller (1985).

Knowledge acquisition is a central topic in AI and expert systems. A number of systems exist for assistance in acquiring specialized knowledge for natural language processing. Thus, TEAM (Grosz et al. 1985), TELI (Ballard and Stumberger, 1986), IRACQ (Ayuso et al. 1987) and some others are devoted to facilitating customizations in the framework of database query systems; INKA (Phillips et al. 1986) and CYC (Lenat et al. 1986) are examples of knowledge acquisition systems aimed at expert system design.

We would like to address some important methodological and strategic points in providing the lexicon support in the context of a text processing system, such as the machine translation system TRANSLATOR (Nirenburg et al. 1987).

1. Need Help Building Lexicon

A natural language lexicon is a necessary part of any natural language processing (NLP) system. Building such a lexicon is an important knowledge acquisition task. The format of the lexicon has been the subject of substantial number of research projects, both in linguistics and in artificial intelligence (AI). In linguistic semantics, the efforts have focused on the formal notation for representing semantic knowledge with a special emphasis on the involved categories and rules. The projects have ranged from simple feature notations (Katz, Fodor 1963; Bendix 1966) to script-based contextual analysis (Raskin 1986).

In AI, this type of work belongs to the area of knowledge representation. A number of methods of representing knowledge have been developed, ranging from first-order predicate calculus notation to quite sophisticated frame-based and advanced logical formalisms. (Brachman and Levesque 1985 summarizes the state of the art in knowledge representation.)

By comparison, the number of projects devoted to the design of procedures for compiling and maintaining

0362-613X/ 87 /030276-289$03.00

a large lexicon has been relatively small. The usual practice in AI has been to suggest a relatively well-developed knowledge representation language and then use it to represent a miniature world or a very restricted sublanguage. It is clear, however, that for an NLP system to be really useful, one needs not only to provide a language in which to record the meanings of linguistic units, but also, in fact, to record those meanings for a non-miniature world (and its corresponding sublanguage).

The crux of this problem is generating **linguistic heuristics** to support the growth of the representation of the subworld/sublanguage. Indeed, there can be no deterministic solution to the problem of knowledge acquisition; therefore, weaker, heuristic methods should be applied to (a) discovering the categories in terms of which to account for the morphological, syntactical, and semantical properties of words and phrases of natural language and (b) developing guidelines for compiling the lexicon for a natural language. It follows from the above that new, sophisticated ways of human-computer interaction must be devised in order to support the tasks of linguistic description in terms of AI-style knowledge representation.

The implication of many workers in knowledge representation that once the notation is developed it will be easy to use it for actually describing the world, is clearly anti-intuitive and anti-experiential, as anybody who has ever developed an NLP system knows only too well. Quite frequently, the lack of any reference to the procedures of lexical description spells out the difference between a system conceived and a system executed.

Thus, decisions to assign a word to a postulated category, for instance, are far from straightforward — many borderline cases have to be routinely considered and resolved. In any large application, lexicon building is typically done by a large number of relatively untrained people who are certain to make non-uniform decisions in many cases. The instructions these people obtain at the beginning of their work cannot be very precise, unambiguous, or designed to provide in advance for any contingency. As a result, even a good knowledge representation language cannot guarantee a quality lexicon.

One can in principle think of avoiding these difficulties through automation. Fully automated lexicon building procedures would definitely ensure the uniformity of description. There was a considerable enthusiasm about the possibility of a complete automation of lexicology some 15-20 years ago (see, for instance, Collin 1960; Grimes 1970; Kucera 1969; Venezky 1973). A more feasible scenario, however, is partial computational assistance for lexicology, probably along the lines suggested in this paper, as well as utilizing such new and promising resources as on-line dictionaries.

Recent work on machine-readable dictionaries offers new and interesting possibilities both for the computer-assisted lexicology (see Walker 1984) and for constructing lexical databases derived from the definitions in machine-readable dictionaries and utilized in NLP along with other fields (see Amsler 1982, 1984a; Walker, Amsler 1986, Calzolari 1984a,b). The premises and goals of these efforts are fully compatible with our belief, first, that no AI system is ready to make the kind of decisions that lexicon building requires and, second, that 'simply having an online version of an encyclopedia [or a dictionary] would be of little use, as there is practically nothing that current AI could draw from the raw text. Rather, we must carefully re-represent the encyclopedia's knowledge — by hand — into some more structured form' (Lenat et al. 1986:75). Such re-representation would be necessary for Amsler's (1984a:458) 'lexical knowledge base [which] is a repository of computational information about concepts' and which contains information derived from machine-readable dictionaries, the full text of reference books, the results of statistical analysis of text usages, and data manually obtained from human world knowledge.'

This paper deals primarily with the last item on Amsler's agenda. It is based on the following approach to the problem of lexicon building. The work is done by humans assisted by an interactive aid which enhances productivity and ensures uniformity. It is important to recognize that lexicon building in NLP involves the acquisition of not one entity but rather of **three interrelated but distinct lexicons,** namely

a) the world concept lexicon which structures our knowledge of the world

b) the analysis lexicon which is indexed by natural language words and phrases connected with concepts from the world concept lexicon, and

c) the generation lexicon, which is indexed by concepts in the world concept lexicon connected with natural language words and phrases.

In reality, AI systems deal not with the entire world but typically with a well-specified subworld of it. In other words, in any practical application, the world concept lexicon will, in fact, be a subworld concept lexicon. In the area of machine translation the analysis and generation lexicons involve two different natural languages, which creates the task of building such resources as, say, an English — subworld and subworld — Russian lexicons, as is the case, for example, in TRANSLATOR, the knowledge-based machine translation project for the computer subworld (see Nirenburg et al. 1985, 1986, 1987).

2. World First, Word Later

We assert that of the three lexicons described above, the first to be built must be the subworld concept lexicon. The availability of such a lexicon is a *sine qua non* for any subsequent lexical work in NLP.

The recognized necessity to describe the concept lexicon prior to dealing with natural languages is a feature which clearly distinguishes our approach from other work on lexical aids (e.g., Ahlswede 1985) and brings us closer to some non-NL work on knowledge acquisition (e.g., Lenat et al. 1986).

Ahlswede (1985) describes a typical approach to lexicon building in a constrained subworld. The subworld is that of stroke medicine. The interactive aid is designed for the analysis lexicon, in our terminology. The concept realm is not discussed at all, though the four major semantic features used in this lexicon are obviously underlain by subworld concepts. As a result, the structure and interrelations of concepts cannot be conveniently reasoned about. Thus, for instance, the distinction between 'general' and 'specific' features is not supported by the ontology and is not really used, since only one subworld is chosen.

We agree with those who believe that each subworld must be described fully and uniformly prior to the introduction of any semantic feature. This may not be feasible or practical for some very rich domains — it is definitely beyond reach as far as the whole world served by the whole language is concerned. However, most NLP systems are designed for constrained domains, and it will probably remain this way for some time.

The description of a subworld is a matter of distinguishing all the relevant concepts in it and arranging them in a logical, consistent, and meaningful way. We agree entirely with Lenat et al. (1986:75) that to create a knowledge acquisition system, 'we must encode all the world's knowledge down to some level of detail; there is no way to finesse this.' We differ from them by our overt, theoretically built-in emphasis on limited subworlds. More importantly, we want to assist the 'knowledge enterers' with an interactive aid it is not at all clear how the CYC system of knowledge acquisition which Lenat et al. describe can ensure the quality and uniformity of the descriptions without such a device.

Another reason why it is important as well as convenient to precede the lexical work with the description of the subworld is that such a description is natural language-independent. In other words, the same subworld may correspond to sublanguages of different natural languages. In MT, the representation of the subworld functions as — and is — the interlingua. Beginning directly with the lexicon of a natural language would make the implied subworld biased toward this language, and this is known to have an adverse affect on the description of other natural sublanguages corresponding to the same subworld.

The subworld concept lexicon determines the structure of the associated analysis and generation lexicons. Thus, the constrained nature of the subworld (see Raskin 1985 and references therein as well as Kittredge and Lehrberger 1982 and Grishman and Kittredge 1986) always severely limits polysemy/homonymy so rampant in the language as a whole. Thus, the concept lexicon for the computer subworld will make it clear that there is a need to accommodate only meanings (5) and (7) in the lexical entry for *operator* in the corresponding sublanguage of English, even though the English language possesses all of the meanings listed in Figure 1 (cf. SOED 1973:1453).

(1) one who does something, an agent
(2) one who is professionally engaged to perform a certain operation
(3) a surgeon
(4) one who carries on financial operations
(5) one who works a machine
(6) one who works a business
(7) a symbol

Figure 1. Meanings of *operator*

In fact, (5) will be narrowed down even further in the sublanguage, to 'one who works a computer' and this is what the English (or any other natural language) lexical entry should be limited to.

The subworld concept lexicon also determines, to a large extent, the inventory of semantic features used in the entries of the analysis and generation lexicons, the values of those features, and their status. Thus, a binary semantic feature is usually induced by a branching in the 'isa' hierarchy underlying the concept lexicon. Frame slots are suggested by a perceived link between some two hierarchically unrelated concepts in the concept lexicon. The relative importance and scope of features used in analysis and generation lexicons depends on the status of the corresponding concepts in the concept lexicon. We now rest our case for the priority of the concept lexicon over the analysis and generation lexicons and proceed to describe a system for facilitating lexicon acquisition and maintenance.

2.1 The Design of a Lexicon Management System.

In a large-scale NLP application, the process of acquiring and maintaining the various lexicons cannot be left unattended. Therefore, any such application requires a **lexicon management system** (LMS). Historically, relatively less attention has been devoted to the lexicons in this context than to either the grammars or the actual processing modules — parsers, inference engines, and generators. The importance of an LMS grows proportionally to the size of the lexicons necessary for an application and also to the depth of coverage. Thus, for instance, it is extremely important to use a principled LMS in a knowledge-based machine translation system where the lexicons must cover not only the usual morphological and syntactic knowledge but also the semantic and pragmatic knowledge about the translation area and where the typical number of concepts in such an area is large.

An LMS is a collection of programs that help create, augment, modify, and test the various lexicons in an NLP application. The particular LMS described in this paper is suggested for the TRANSLATOR project. The goal of TRANSLATOR is ultimately to develop a knowledge-based multilingual machine translation sys-

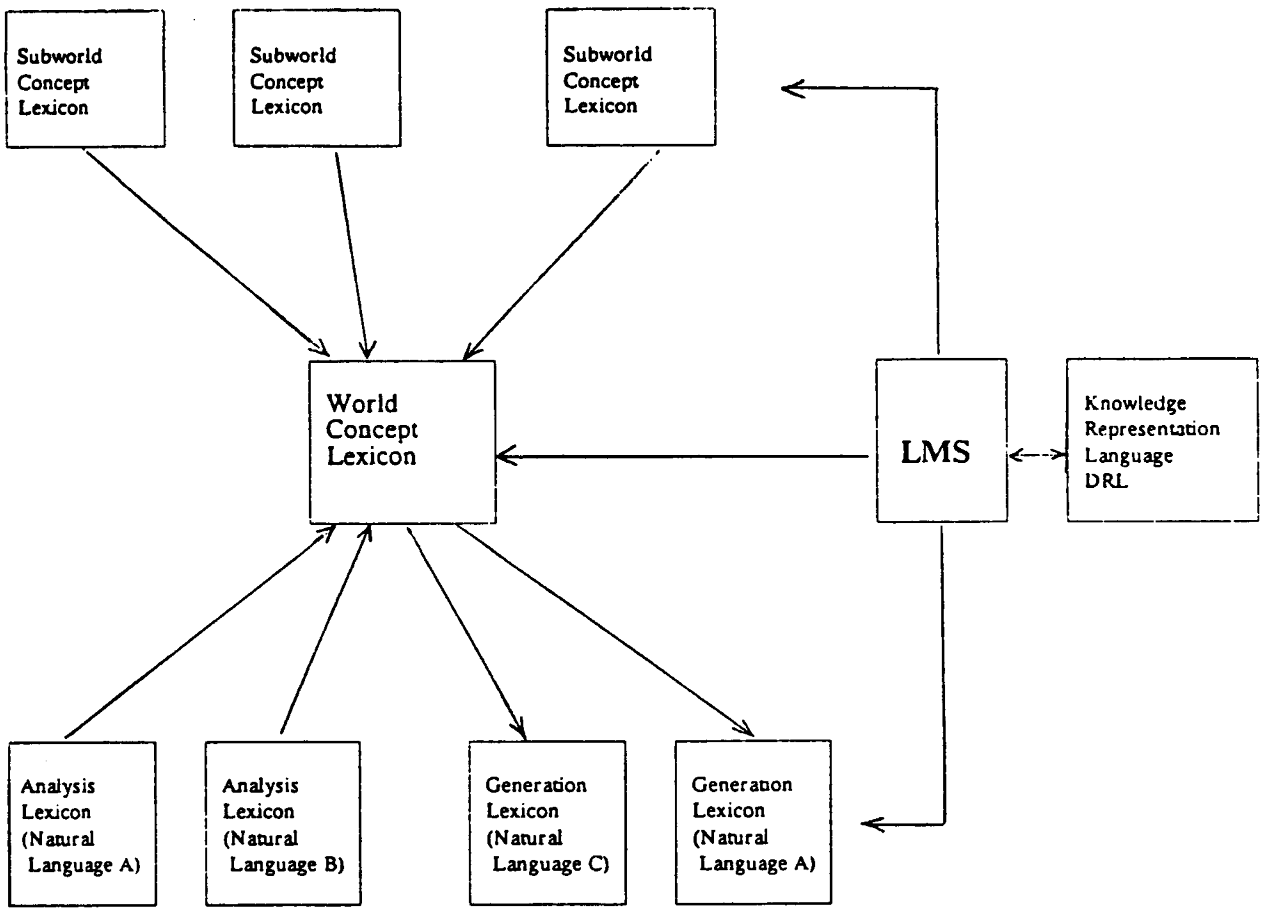

Subworld Concept Lexicon
Subworld Concept Lexicon
Subworld Concept Lexicon
World Concept Lexicon
LMS
Knowledge Representation Language DRL
Analysis Lexicon (Natural Language A)
Analysis Lexicon (Natural Language B)
Generation Lexicon (Natural Language C)
Generation Lexicon (Natural Language A)

Figure 2. The architecture of the TRANSLATOR Lexicon Module, including the Lexicon Management System (LMS). At present, the world concept lexicon is identical with the only subworld concept lexicon—that of computer science.

tem for multiple subject areas. Various modules in the system are designed so as to allow interactive human participation with no pre- or post-editing. The LMS is one such module.

The lexicon component of TRANSLATOR (Figure 2) contains four types of lexicons:

1. the subworld concept lexicons, which contain representations, in the DRL knowledge representation language (Nirenburg, Raskin, Tucker 1986), of concepts that belong to a particular subworld, such as, for instance, the subworld of weather reports or of research papers in computer science;
2. the world concept lexicon, in which the information from all the available subworld concept lexicons is merged;
3. the analysis lexicons, whose entries are indexed by lexical units in a natural language and which contain
   a) syntactic information about these units;
   b) a pointer to a corresponding concept in the concept lexicon (this link essentially assigns semantic meaning to the lexical unit; note that some lexical units do not correspond to concepts); and/or
   c) additional control and constraint information for the use of the analysis module of the underlying NLP system.
4. the generation lexicons, whose entries are indexed by concepts in the concept lexicon and which contain pointers to corresponding lexical units in the target language. (Note that the analysis and generation lexicons are not symmetrical; for a detailed analysis of the differences see Nirenburg and Raskin 1989.)

LMS maintains all four types of lexicons. In this paper we will concentrate on describing that part of the LMS which is devoted to the maintenance of one subworld concept lexicon (that of computer science research).

The primary purpose of an LMS at the first stage of the project is to support knowledge acquisition. At later stages in the life of the LMS, testing and modification will become the primary types of work it supports. An ordinary LMS user, i.e., an **enterer**, will obtain at this stage a list of concepts to enter in the concept lexicon and will code the information about them in DRL. The LMS assists the enterer by providing graphic and other aids for human decision making. In case of doubt, the enterer can try to resolve the difficulty or to refer it to the **lexicon manager**, whose responsibility it is to force solutions to problems in lexicon acquisition.

The task of the manager has much in common with that of a database administrator in the database system environment. In their respective capacities, they are both responsible for

a) maintaining the format and contents of the knowledge representation language ('data dictionary' in database terms);
b) defining and executing the security and integrity checks for the accumulated data;
c) developing and running statistic analysis routines to monitor access time, etc. (this becomes important in production-size lexicon systems);
d) interfacing with regular users (enterers for the lexicon manager).

In addition to the above, the lexicon manager also modifies the knowledge representation language in accordance with the evidence accumulated in the process of knowledge acquisition.

The LMS thus has two modes of operation: a mode for enterers and a mode that supports the activities of a lexicon manager. In what follows we will describe and illustrate how these modes are implemented in the TRANSLATOR LMS. The system has been implemented in Zetalisp on a Symbolics 3600 Lisp Machine.

2.2 A DRL Ontology for Computer Subworld

The intended result of the first stage of research reported in this paper is a computer subworld concept lexicon. The subsequent stages will add a number of analysis and generation lexicons. We have argued that for the purpose of lexicon building it is necessary to go beyond just proposing a knowledge representation framework. It is necessary to use it for the construction of actual lexicons and to keep using it until the lexicons are judged sufficiently complete. An LMS must contain a set of interactive aids to facilitate this type of activity.

The first step is, however, still to suggest a knowledge representation scheme for coding lexical meanings in a concept lexicon. Structurally, the concept lexicon can be viewed as a complex network, with concepts as nodes. The connections in this network place the nodes in various hierarchies and classify them on the basis of certain characteristics and constraints. We suggest a frame-based representation in which frames correspond to concepts, and slots convey constraints on the meaning of these concepts. The sets of values that can occupy certain slots, the **domains** of the latter, can be further classified. Thus, some slots take names of concepts in the world as values (such are hierarchy-related slots); some others, take values from specially defined **property sets.** The slots can be occupied by any number of members of the corresponding domain, and the logical operators *and*, *or*, and *not* can be used to augment the expressive power. Also, in every case, the semantics of the constraints in the lexicon is that of **default** knowledge: the contents of a slot are understood as **typically** constraining the meaning of the concept.

The semantic character of a slot in a concept lexicon frame underscores a careful distinction that should be made between concept types and concept tokens. Such a distinction is common in certain knowledge representation languages (see Brachman and Schmolze 1985; Nirenburg et al. 1986). Concept types belong in the lexicon; concept tokens are instantiations of concept types obtained as a result of analyzing natural language inputs. One kind of knowledge representation can be used for both types and tokens, a frame-based notation being one of them. (Of course, due to a number of possible reasons, a complete knowledge representation system can use one type of representation for the types and another for the tokens, cf. Hobbs 1985 for a discussion of such a position.) The frames for a type and its token will not, however, be identical in structure. The semantic content of slots in a lexicon (type) frame is different from that of the corresponding slots in the text (token) frame. Concept tokens have their slots occupied by actual values of properties; if information about a property is not forthcoming, then the default value (if any) is inherited from the corresponding type representations. For example, the frame for a verbal action type and a verbal action token can have a slot named 'agent.' However, in the former case, the slot can be occupied by a concept type, such as, for instance, 'human.' In the latter case, the slot must be occupied by a concept *token,* such as 'John23.' If the analysis program cannot make a decision as to what token(s) must occupy a slot, it produces a temporary filler token where constraints will be inherited from its corresponding type. In this case, it may be 'person1,' with all the constraints of the type 'human' inherited.

The concept lexicon forms a tangled ISA hierarchy with property inheritance. However, the examples below are simplified to make it look as a strict hierarchy. The current version of the top levels of this hierarchy is shown in Figure 3. The actual world ontology in a working system depends on the subworld for which it is developed. We will survey the concept frames in the order suggested by the ISA hierarchy.

```
all ::= (all
          (id string)
          (subworld subworld*))
```

This is the root of the *isa* hierarchy. The two slots mean that every node has an *id* and represents a concept that belongs to one or more *subworlds.*

```
process ::= (process
              (isa all)
              (patient object))
```

At this level we meet the 'isa' slot, the pointer to a node's parent in the hierarchy. Processes, as one can see from Figure 3, divide into actions and states. The only overtly mentioned property common to all processes is the conceptual case of 'patient' (this reflects our opinion that in the English sentence 'John is asleep' *John is not an agent,* but rather a patient). Note that 'patient' subsumes the semantics of 'beneficiary.'

```
action ::= (action
             (isa process)
             (consists-of process-sequence)
```

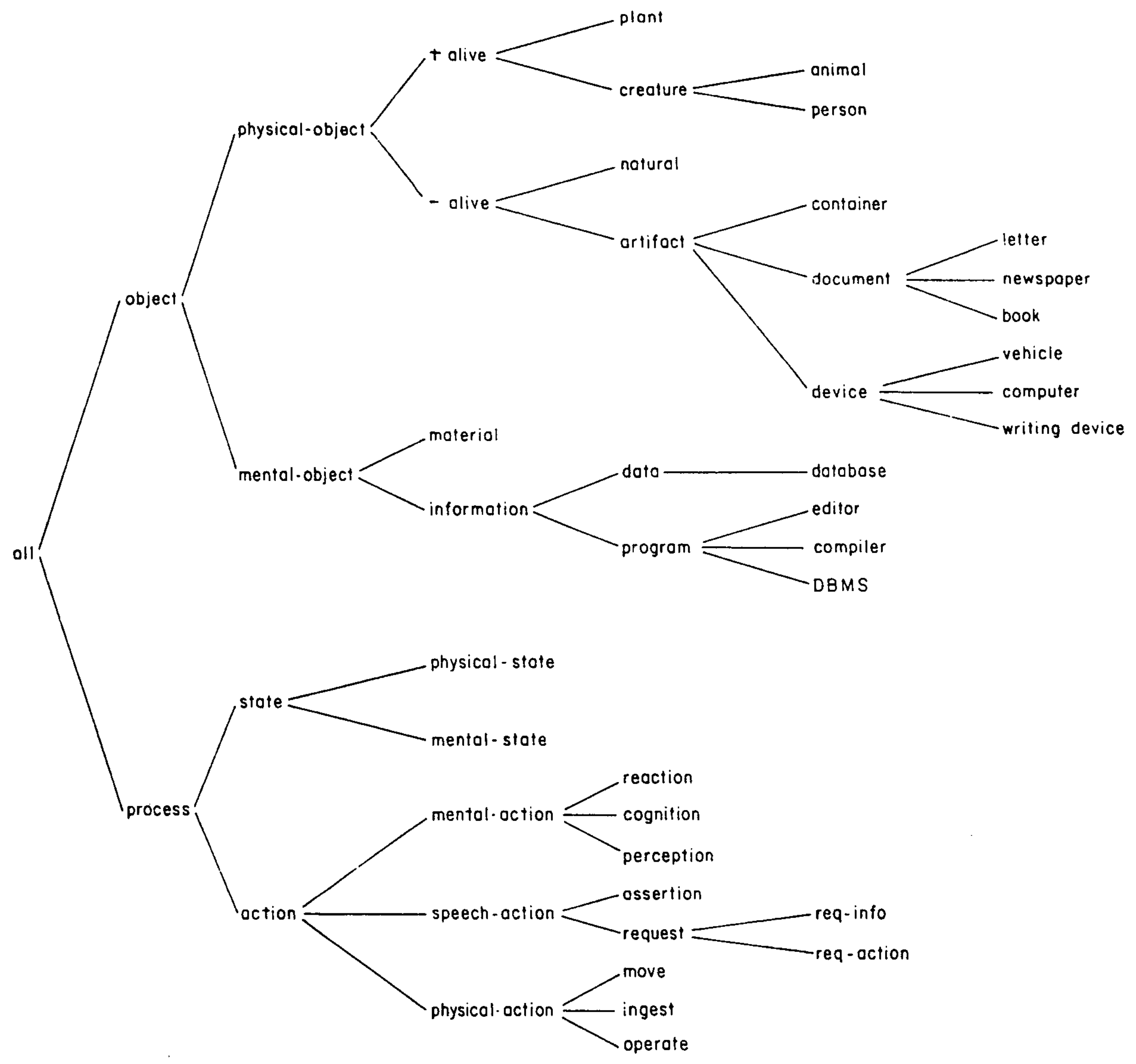


Figure 3. A fragment of the ISA hierarchy for the computer science world.

(*part-of* process*)
(*preconditions* state*)
(*effects* state*)
(*tempor* process*)
(*agent* creature)
(*object* all)
(*instrument* object)
(*source* object)
(*destination* object))

The action frame contains two groups of slots:

a) paradigmatic relation slots (*isa, consists-of, part-of, preconditions, effects, tempor*) that connect it with other processes;
b) syntagmatic relation slots (the conceptual case slots *agent, object, instrument, source and destination*; *patient* is inherited).

The *consists-of* slot contains either the constant 'primitive,' if the action is not further decomposable, or a description of the sequence of actions which comprise the given action. This sequence is a list of action names connected by the operators *sequential, choice* and *shuffle*. In other words, an action can be a sequence of subactions ('sequential'), a choice among several subactions ('choice'), a temporally unordered sequence of subactions ('shuffle') or any recursive combination of the above. This treatment of processes is inspired by Nirenburg, Nirenburg, Reynolds 1985. The *tempor* slot connects the action with other actions that typically occur proximally in time to it. Metaphorically, while *consists-of* describes children, *tempor* describes siblings.

physical-action ::= (*physical-action*
(*isa action*)
(*object* physical-object))

mental-action ::= (*mental-action*
(*isa* action))

Only creatures can be fillers for the *agent* slot.

Mental actions further classify into reaction (cf. the English *please* or *like*), cognition (*deduce*) and perception actions (*see*).

speech-action ::= (*speech-action*
(*isa* process)
(*consists-of* primitive)
(*agent* person)
(*patient* person* | organization*)
(*object* event)
(*source* agent)
(*destination* patient))

Speech processes are 'primitives'. The *agent* slot filler has the semantics of the speaker. The *patient* is the hearer.

state ::= (*state*
(*isa* process)
(*part-of* state*))

The actant in states, which is the patient rather than the agent, is inherited from the process frame.

object ::= (*object*
(*isa* all)
(*part-of* object*)
(*consists-of* object*)
(*belongs-to* creature | organization)
(*spatial* physical-object*)
(*object-of* mental-action speech-action)
(*patient-of* process)
(*instrument-of* process)
(*source-of* process)
(*destination-of* process))

Once again two groups of slots describe paradigmatic and syntagmatic relations. The *spatial* slot lists objects that typically appear proximally in space to the current object. The . . .*-of* slots have the semantics of 'typical . . .-of' and are complementary to the conceptual case slots in process descriptions.

The other type of slot fillers in the DRL frames is a set of property values. Properties are defined in the world, each with a corresponding set (domain) of property values. Only an illustration of property values is given here. Many more exist and are used in the lexicon description.

size-set ::= (infinitesimal . . . huge)
color-set ::= (black . . . white)
shape-set ::= (flat square spherical . . .)
subworld-set ::= (computer-world business-world everyday-world . . .)
boolean-set ::= (yes no)
texture-set ::= (smooth . . . rough)

A path of concepts from the root to a leaf node in the *isa* hierarchy is presented below and followed by the corresponding frames (the frames for *all* and *object* are given above).

all→object→physical-object→+alive→creature→→person→computer-user

physical-object ::= (*physical-object*
(*isa* object)
(*object-of* (+ (take put))
(*size* size-set)
(*shape* shape-set)
(*color* color-set)
(*mass* integer))

The '+' sign in slots means all inherited information plus the contents of the current slot.

+alive ::= (+*alive*
(*isa* physical-object)
(*edibility* boolean-set))

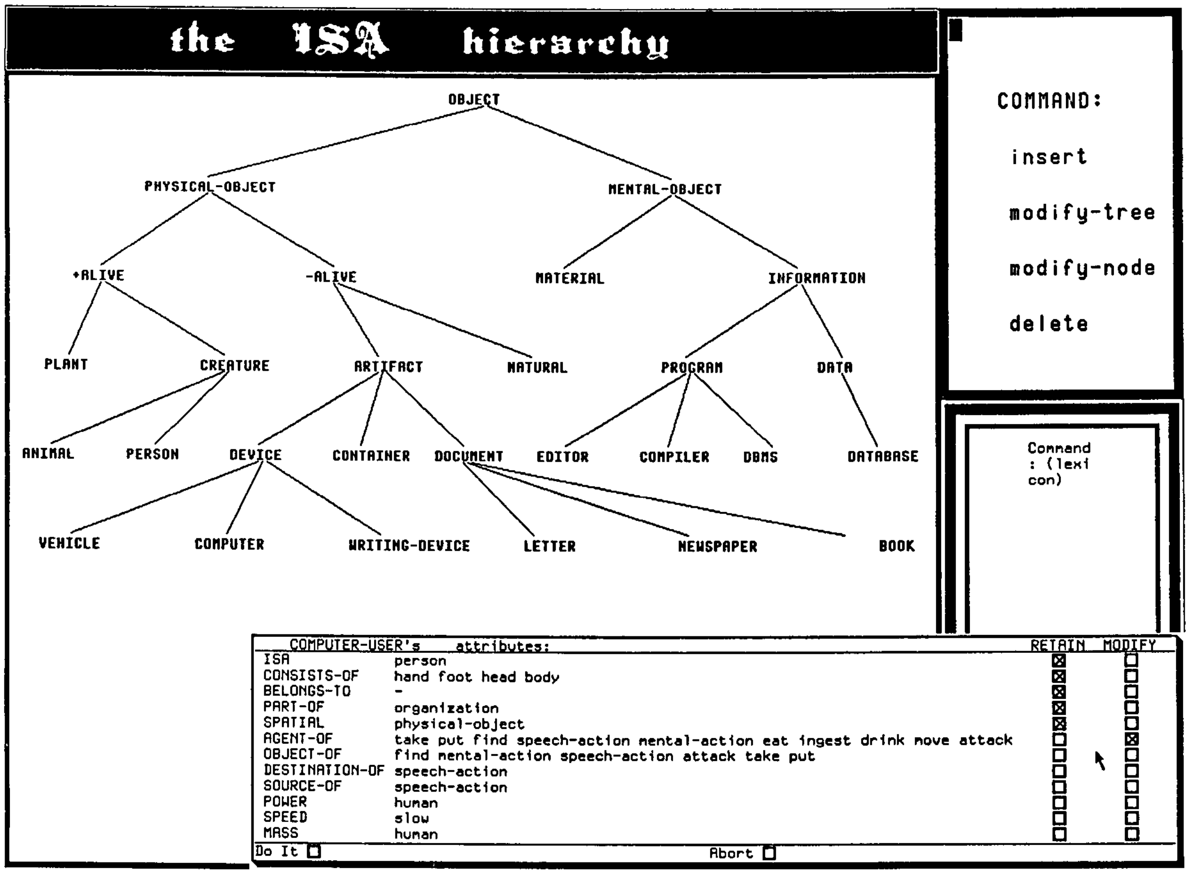
creature ::= (*creature*
(*isa* +alive)
(*agent-of* (eat ingest drink move attack))
(*consists-of* (head body))
(*object-of* (+ (attack))
(*power* power-set)
(*speed* speed-set))

person ::= (*person*
(isa creature)
(*agent-of* (+ (take put find speech-action mental-
action)))
(*source-of* speech-action)
(*destination-of* speech-action)
(*consists-of* (+ (hand foot . . .)))
(*power* human)
(*speed* slow)
(*mass* human))

computer-user ::= (*computer-user*
(*isa* person)
(*agent-of* (+ operate))
(*subworld* computer-world))

The complete frame of the leaf of this path, '*computer-user,*' including all inherited slots and default values, is listed below. In reality frames like this do not exist, because the tokens of this type do not contain all the possible slot fillers.

(computer-user
(*isa* person)
(*consists-of* (hand foot head body))
(*belongs-to* none)
(*part-of* organization*)
(*spatial* physical-object*)
(*agent-of* (operate take put find speech-action mental-action eat ingest drink move attack))
(*object-of* (find mental-action speech-action attack take put))
(*destination-of* speech-action)
(*source-of* speech-action)
(*power* human)
(*speed* slow)
(*mass* human)
(*subworld* computer-world))



COMPUTER-USER's	attributes:	RETAIN	MODIFY
ISA	person	☒	☐
CONSISTS-OF	hand foot head body	☒	☐
BELONGS-TO	-	☒	☐
PART-OF	organization	☒	☐
SPATIAL	physical-object	☒	☐
AGENT-OF	take put find speech-action mental-action eat ingest drink move attack	☐	☒
OBJECT-OF	find mental-action speech-action attack take put	☐	☐
DESTINATION-OF	speech-action	☐	☐
SOURCE-OF	speech-action	☐	☐
POWER	human	☐	☐
SPEED	slow	☐	☐
MASS	human	☐	☐

Figure 4. Entering *computer-user*, I.

3. Two Sample Sessions with the Subworld Module of the TRANSLATOR LMS

We will demonstrate the work of the subworld module of the TRANSLATOR LMS on two examples. They will illustrate the two modes discussed in 2.1, the enterer and manager modes. The first, simple example will involve the placing of a concept in an appropriate slot of the 'isa' tree and checking the slots of the inherited frame for appropriate values. All of these operations will be executed in the enterer mode. The second, more complicated example will present serious difficulties for the enterer and will require the manager's full authority for the resolution of the problem. In other words, the first example will leave the existing ontology intact while the second will require its modification.

3.1 The Enterer Mode: Adding the Concept COMPUTER-USER.

This concept represents meaning (5) in the dictionary entry for the English word *operator* (see Figure 1). The task of the enterer essentially involves the following operations:

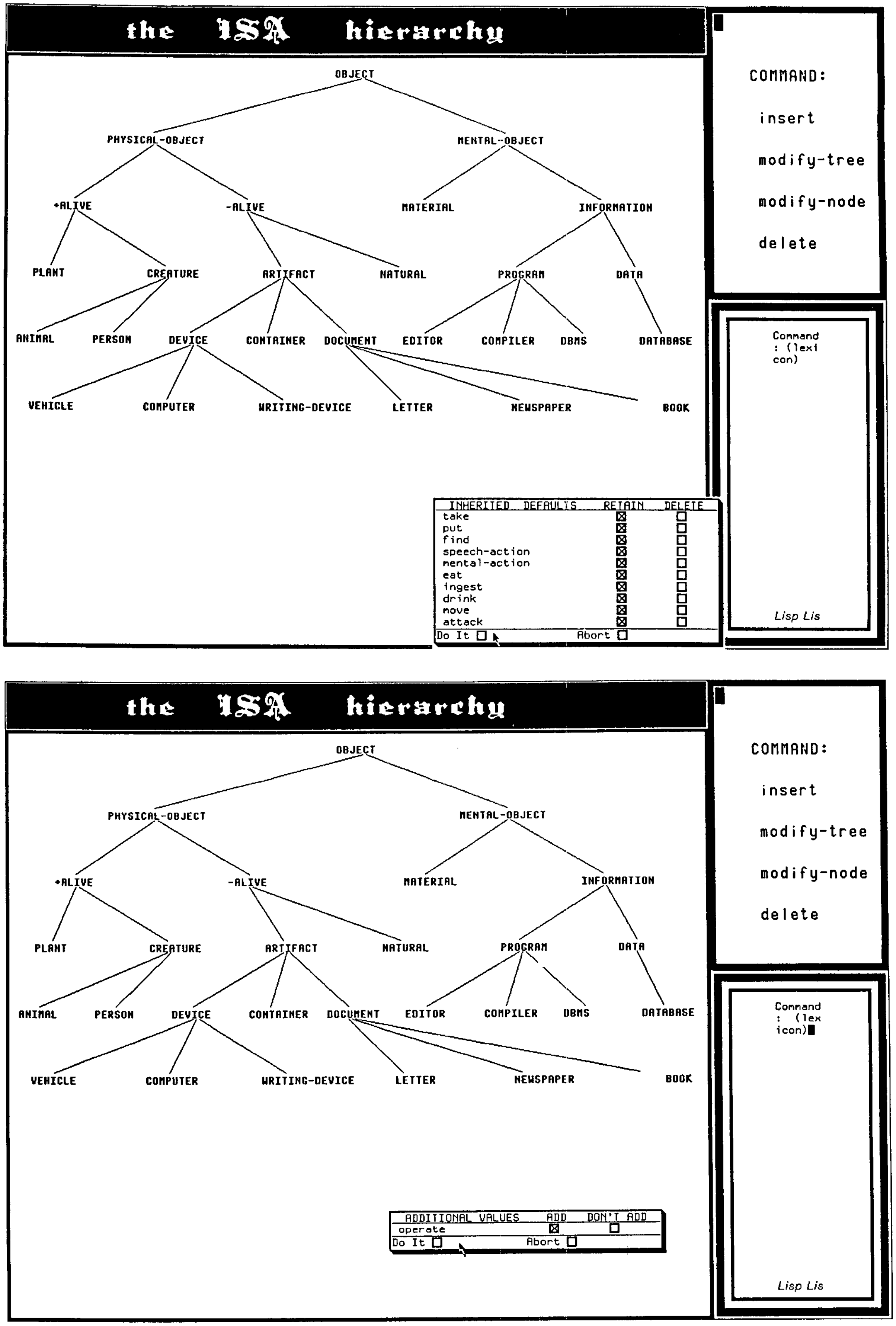
Insert: finding the appropriate slot in the *isa* hierarchy to which the new concept will be attached as a leaf child

Fill: describing all the properties of this concept by modifying, wherever necessary, the slot fillers of the inherited frame

Check: checking whether the property values chosen distinguish it from its parent and siblings.

The LMS helps the enterer with all three of these activities. To decide on the position of the concept in the hierarchy, the enterer can browse through its graphical representation and view the frames corresponding to any node (concept) in it. If the enterer has to consider more than one possibility, the frames for several candidate parents for the node to be inserted, can be displayed — in the current implementation up to three candidate frames can be displayed simultaneously. In the case of *computer-user* the enterer does not have a difficulty in selecting *person* as the only candidate parent node.

When the place for a concept is found, a menu containing all slots pertinent for the parent node frame which incorporates all the slots and slot fillers it inherited from its ancestors, appears on the screen. The enterer scans the menu and modifies, wherever necessary, the values in these slots. Figure 4[1] shows the state of the process for entering *computer-user* when the values for the *agent-of* slot are being considered. Note that the system provides additional help by listing candidate values for this slot. The set of candidates is produced automatically by determining the common



```
Command
: (lexi
con)
```

INHERITED DEFAULTS	RETAIN	DELETE
take	⊠	☐
put	⊠	☐
find	⊠	☐
speech-action	⊠	☐
mental-action	⊠	☐
eat	⊠	☐
ingest	⊠	☐
drink	⊠	☐
move	⊠	☐
attack	⊠	☐

Do It ☐ Abort ☐

```
Command
: (lex
icon)
```

ADDITIONAL VALUES	ADD	DON'T ADD
operate	⊠	☐

Do It ☐ Abort ☐

Figure 5. Entering *computer-user*, II.

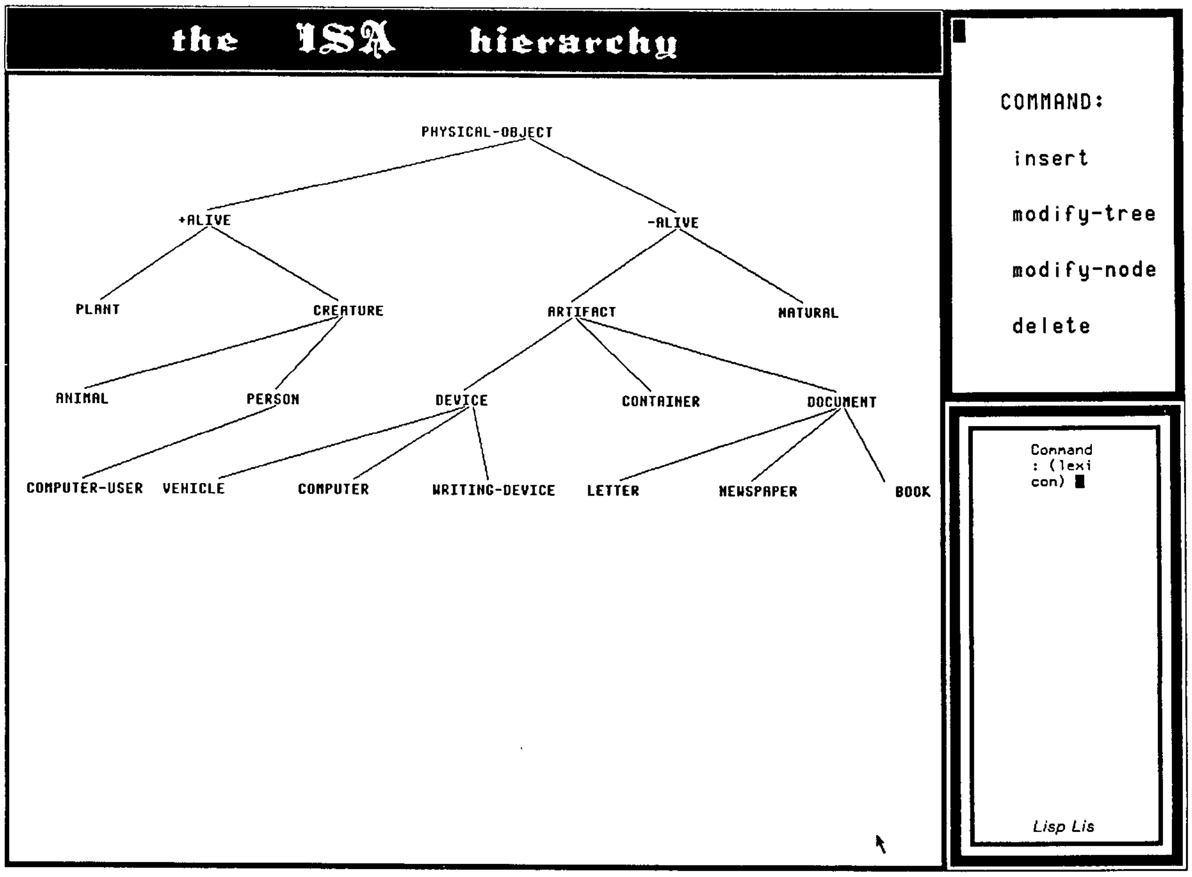


Figure 5A. Entering *computer-user*, III.

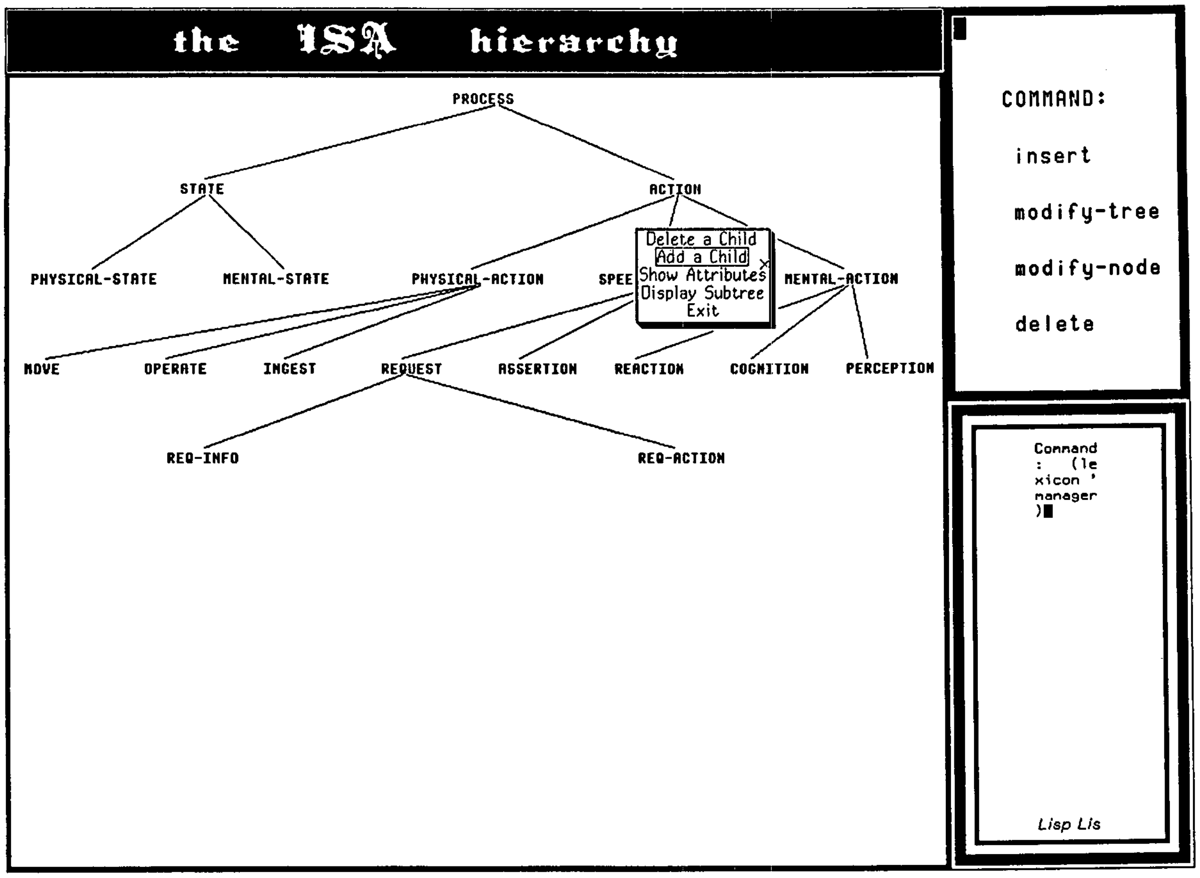
ancestor of the inherited slot values (in this case, 'action') and suggesting to the enterer all the descendants of this ancestor node that are not currently listed as the slot values. For slots whose value domains are not concepts but rather property values, the help mechanism simply lists the complete list of values for discrete domains and the appropriate ranges for the continuous ones. In our example, the enterer adds *operate* to the values of the *agent-of* slot. Another important ability (not illustrated in the example) is adding new slots to various frames and then supplying fillers for them.

When the slot filling stage is completed (this stage of the process is illustrated in Figure 5), it is necessary to check whether the new node is distinct from its parent and siblings. In our example, the new concept does not have siblings, therefore only the former distinction is tested. The parent distinction check is performed automatically — the system will alert the enterer if there has been no changes (except for the 'isa' slot filler) to the frame displayed for modification at the Fill stage. If this happens, the new concept is discarded as a duplicate of the parent node. In our example, an additional value has been added to the *agent-of* slot. The sibling distinction check is facilitated in the LMS by allowing the enterer to view the frames for the concept and its siblings alongside one another. In case no distinction can be found between the new concept and one of its siblings, the new concept is deleted as a duplicate. In general, before deleting a new concept because of duplication, the enterer makes an effort to save it by changing at least one slot value in either of the two duplicate frames. This may involve adding an existing concept or a property value from the current domain. However, in some cases, a new concept or a modification of a property domain may be required. Each such case is reported to the manager.

3.2 The Manager Mode: Adding the concept MANAGE(-DATABASE)

The enterer first decides to insert the new concept *manage(-database)* as a sibling of *operate* and, therefore, selects *physical-action* as the parent node. When the frame menu is displayed, however, he discovers a major discrepancy — *manage(-database)* requires the 'mental-object' filler for the *object* slot. The enterer then makes an attempt to attach the new concept as a child of the *mental-action* node but cannot accept *reaction, cognition,* and *perception* as the siblings of *manage(-database)* for reasons of significant conceptual dissimilarity. At this point, the enterer decides to refer the matter to the manager. This is accomplished by writing a report about the situation and sending it through the mail system.

At this point the responsibility for processing this



```
Command : (lexicon 'manager
)
```

Figure 6. Changing the descendents for *mental-action*, I.

concept passes to the manager. In addition to the enterer's message the manager also has access to the enterer's attempts to insert *manage(-database)* into the hierarchy. The manager is, naturally, capable of handling all the enterer's tasks; but the LMS manager mode has additional functionality. The enterer is allowed to perform only one operation on the 'isa' hierarchy — adding leaves to it. The manager can perform all the operations on this tree — insert new nodes anywhere in it; delete nodes (either with their subtrees or without, in which case the children of the deleted node become children of the latter's parent); and move subtrees to a different position. The manager is also entitled to create additional property sets (domains) and add and delete values in the existing ones. Finally, the manager's authority also covers adding, deleting and modifying the value domains of all slots in all concept frames. Each of these capabilities is supported by LMS routines.

In our example, the manager has to reorganize the subtree whose root is *mental-action*. First, he finds the necessary node and calls a menu of operations on a node (Figure 6). Clicking on 'delete-child', he obtains another menu, in which he indicates that he wants to detach the children of the node. The result of this operation is illustrated in Figure 7. At this point the insertion of the two intermediate concepts: *computer-mental-action* and *non-computer-mental-action* can be performed (which is supported in the manner illustrated in 3.1 above). During the insertion process the manager acts exactly in the same manner as an enterer. Temporarily disregarding the detached children, he inserts the intermediate concepts one by one.

At the next stage, the former children of *mental-action* are reconnected to the hierarchy as children of *non-computer-mental-action*. Once again, the manager acts here as an enterer, filling the frame slots and running the sibling and parent distinction checks. Next, *manage(-database)* is attached as a child of the *computer-mental-action* node. Again, all the insert-time checks are run. At this point the manager decides to develop the *computer-mental-action* node further deductively, that is, without waiting for empirical evidence for this to accumulate. He decides to enter the nodes *write-code* and *use-interactive-system* as siblings of *manage(-database)*. This operation is entirely at the manager's discretion and expresses the manager's understanding of the computer world and his expectations concerning the concepts that will have to be added to the ontology in the future. After these concepts have been added to the world, the subtree with the root *mental-action* will look as in Figure 8.

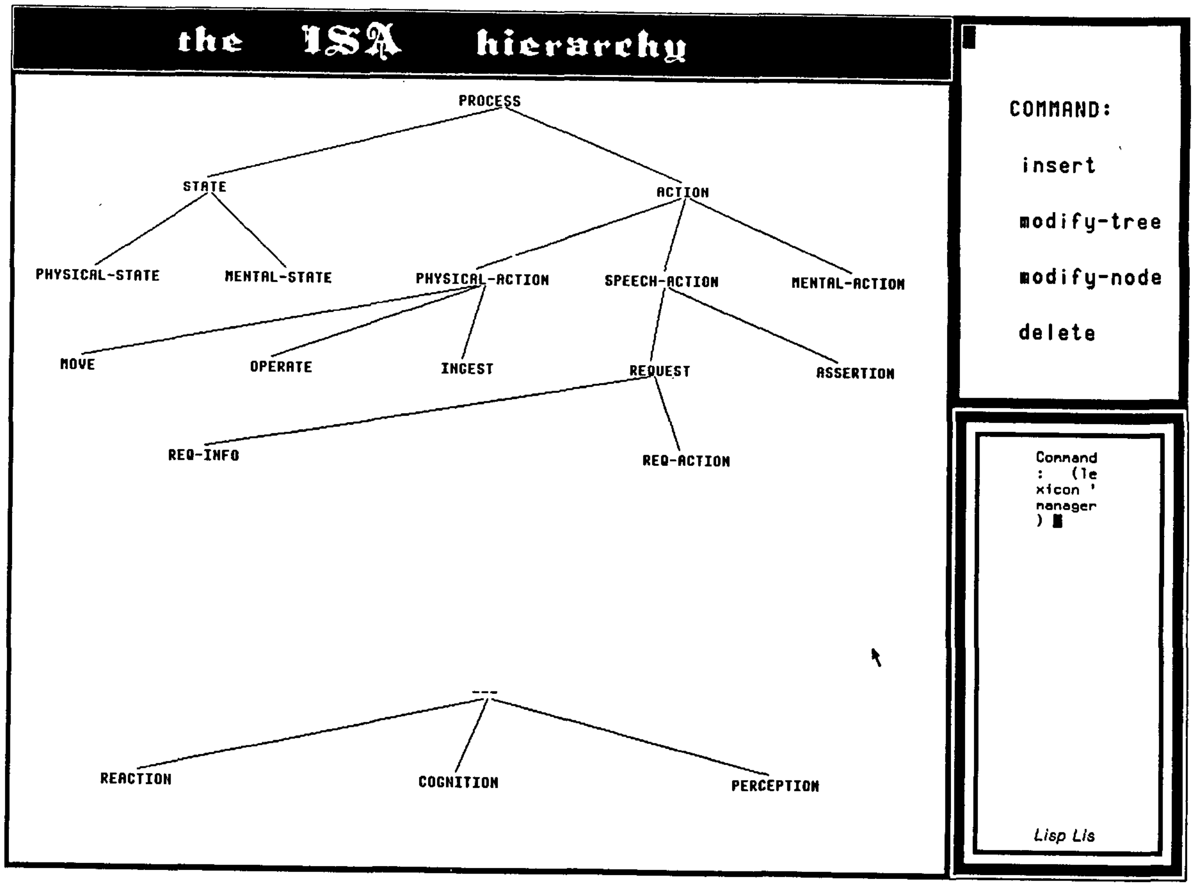


Figure 7. Changing the descendents for *mental-action,* II.

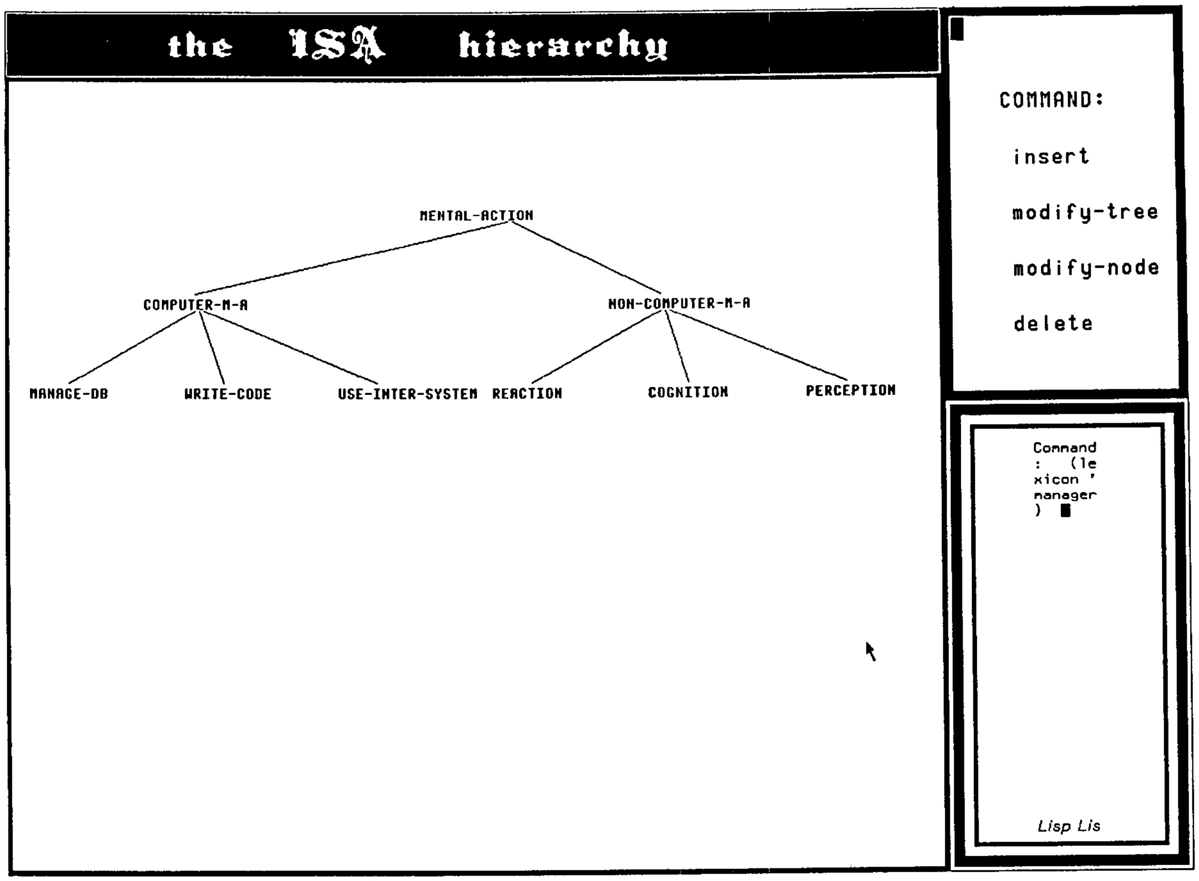
4. Discussion

The TRANSLATOR LMS has been implemented only in its initial stage, so that its functionality at present is more constrained than it will be when the LMS attains full capacity. We would like to discuss here the functionality upgrades under development at the present time.

First, the manager's ability to maintain integrity and consistency in the lexicons must be further supported. This capability includes modifications of the existing concept lexicon entries which are affected by the insertion of a new concept by an enterer or the manager or by the manager's revision of the domain of a frame slot. At present, the LMS can automatically find and modify the values in slots complementary to the ones modified. For example, in our first sample session, after the enterer adds 'operate' to the *agent-of* slot of *computer-user,* the system automatically adds *computer-user* to the *agent* slot of 'operate.' But more should be expected. Thus, in our second sample session, after the manager inserts the *computer-mental-action* and *non-computer-mental-action* as the children of *mental-action,'* the LMS should be able to generate a suggestion that he consider a symmetrical modification in the sibling branch (*physical-action*), which would make *operate* a child of *computer-physical-action.'* In general, we feel that the LMS should provide more heuristic support for the manager's thought processes concerning ontology modifications.

This latter modification, along with the one the manager actually executed in the sample session, would be a step in the transition from the world ontology in Figure 3 to the computer subworld ontology. Some of these changes accurately reflect the differences between the world and the subworld, i.e., in this case, the distinction between concepts of *computer-related* and *non-computer-related* is central for the subworld but not for the world. Other changes in the ontology include the correction of mistakes and the refining of the hierarchies suggested by the accumulation of data and experience. Thus, we will not be surprised if additional data will provide evidence for making *speech-action* a descendant of *mental-action* instead of *action.*

An additional source of insight into the improvement of the LMS will be provided by its use. Thus, it is difficult to know *a priori* how many enterers a single manager will be able to supervise; what forms of the interaction between the enterers and the manager are most effective; whether it is possible to support a concurrent mode of operation in which more than one enterers work on one hierarchy simultaneously; or should there be levels of priority assigned to the messages from enterers to the manager or mechanisms for clustering similar messages (if the messages are forced



```
Command
: (le
xicon '
manager
)
```

Figure 8. Changing the descendents for *mental-action*, III.

into a certain format). A separate concern is a detailed set of on-line instructions for quick reference both by the enterers and the manager. It should be added to the LMS in the tutorial, reference-manual and keyword-based help mode format.

So far in this section, we have been discussing the depth-wise improvements to the TRANSLATOR LMS. The breadth-wise development concerns the analysis and generation lexicons. The work on those can be completed only after the subworld concept lexicon is already reasonably complete.

In the analysis lexicons, indexed by the words or word equivalents, the entries will contain conceptual information from the subworld concept lexicon. Many words, however, will not have a concept-based lexicon entry but will contain information triggering certain parsing procedures. Some other entries will include both of these types of information. Every entry in an analysis lexicon contains syntactic information, and the current LMS already includes an interactive aid for acquiring it for English. It appears that generation lexicons are, in fact, indexed not only by subworld concepts but also by rhetorical and pragmatic markers in the representation of a text obtained after analysis because both typically have lexical — as well as grammatical — expression in natural language.

Acknowledgements. This research was supported, in part, by the National Science Foundation under Grant DCR-8407114 and, in part, by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, NY 13441–5700, and the Air Force Office of Scientific Research, Bolling Air Force Base, DC, 20332, under contract number F30602–85-C-0008, to the Northeast Artificial Intelligence Consortium. We would also like to thank Irene Nirenburg who implemented the LMS, and Sabine Bergler for a number of useful discussions of the subject matter.

References

Ahlswede, T. E. 1985. A Tool Kit for Lexicon Building. Proceedings of 23rd ACL, pp. 268-276.

Amsler, R. A. 1982. Computational Lexicology: A Research Program. AFIPS, Proceedings of the National Computer Conference, vol. V, pp. 657-663.

Amsler, R. A. 1984a. Lexical Knowledge Bases. Proceedings of COLING-84. Stanford University, Stanford, CA, July, pp. 458-459.

Amsler, R. A. 1984b. Machine-Readable Dictionaries. In: M. E. Williams (ed.), **Annual Review of Information Science and Technology**, vol. 19. White Plains NY: Knowledge Industry Publications, pp. 161-209.

Apresyan, Ju. D. 1974. **Leksicheskaya Semantika** (Lexical Semantics). Moscow: Nauka.

Ayuso, D., V. Shaked and R. Weischedel. 1987. An Environment for Acquiring Semantic Information. Proceedings of 25th ACL, pp. 32-40.

Ballard, B. and D. Stumberger, 1986. Semantic Acquisition in TELI: A Transportable, User-Customized Natural Language Processor. Proceedings of 24th ACL, pp. 20-29.

Bendix, E. H. 1966. **Componential Analysis of General Vocabulary.** Bloomington, IN: Indiana University Research Center in Anthropology, Folklore and Linguistics, Publication 41.

Bessemer, D. and P. Jacobs. 1987. FLUSH: A Flexible Lexicon Design. Proceedings of 25th ACL, pp. 186-192.

Boguraev, B. et al. 1987. The Derivation of a Grammatically Indexed Lexicon from the Longman Dictionary of Contemporary English. Proceedings of 25th ACL, pp. 193-200.

Brachman R. J. and H. J. Levesque (eds.) 1985. **Readings in Knowledge Representation.** Los Altos, CA: Morgan Kaufmann.

Brachman, R. and J. Schmolze 1985. An Overview of the KL-ONE Knowledge Representation System. Cognitive Science, 9, pp. 171-216.

Byrd, R., J. Clavans, M. Aronoff and F. Anshen. 1986. Computer Methods for Morphological Analysis. Proceedings of 24th ACL, pp. 120-127.

Calzolari, N. 1984a. Detecting Patterns in a Lexical Database. Proceedings of COLING-84. Stanford University, Stanford, CA, July, pp. 170-173.

Calzolari, N. 1984b. Machine-Readable Dictionaries, Lexical Database and the Lexical System. Proceedings of COLING-84. Stanford University, Stanford, CA, July, p. 460.

Chodorow, Martin S., Roy J. Byrd and George E. Heidorn. 1986. Extracting Semantic Hierarchies from a Large On-Line Dictionary. Proceedings of 23d ACL, pp. 299-304.

Collin, A. 1960. The Automatic Construction of a Glossary. *Information and Control,* IX, pp. 211-230.

Frawley, W. 1980-81. Lexicography and the Philosophy of Science. *Dictionaries,* 2-3, pp. 18-27.

Grimes J. E. 1970. Computing and Lexicography. *Linguistic Reporter,* V, pp. 1-4.

Grishman R. and R. Kittredge (eds.) 1986. **Analysing Language in Restricted Domains.** Hillsdale, NJ: Erlbaum.

Grosz, B., D. Appelt, P. Martin and F. Pereira. 1985. TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces. TR 356, SRI International, Menlo Park, CA, August.

Hobbs, J.R. 1985a. Introduction. In: J.R. Hobbs and R.C. Moore (eds.), **Formal Theories of the Commonsense World.** Norwood, NJ: Ablex, pp. xi-xxii.

Hobbs J. R. 1985b. Ontological Promiscuity. Proceedings of 23rd ACL, pp. 61-69.

Katz J. J. and J. A. Fodor 1963. The Structure of a Semantic Theory. Language , 39, pp. 170-210.

Kittredge R. and J. Lehrberger (eds.) 1982. **Sublanguage.** Berlin: de Gruyter.

Kucera, H. 1969. Computers in Language Analysis and Lexicography. In: **American Heritage Dictionary of the English Language,** NY: Houghton Mifflin, pp. XXXVII-XL.

Lakoff, G. 1972. Linguistics and Natural Logic. In: D. Davidson and G. Harman (eds.), **Semantics of Natural Language,** Dordrecht: Reidel, pp. 545-665.

Lakoff, G. and M. Johnson 1980. **Metaphors We Live By.** Chicago — London: University of Chicago Press.

Landau, S. 1984. **Dictionaries.** NY: Scribners.

Lenat, D., M. Prakash and M. Shepherd 1986. CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks. AI Magazine, VI, pp. 65-85.

Markowitz, J., T. Ahlswede and M. Evens. 1986. Semantically Significant Patterns in Dictionary Definitions. Proceedings of 24th ACL, pp. 112—119.

McCawley, J. D. 1972. A Program for Logic. In: D. Davidson and G. Harman (eds.), **Semantics of Natural Language,** Dordrecht: Reidel, pp. 157-212.

McCawley, James D. 1986. What Linguists might Contribute to Dictionary Making if They could Get Their Act Together. In: P. Bjarkman and V. Raskin (eds.), **The Real-World Linguist: Linguistic Applications in the 1980s,** Norwood, NJ: Ablex, pp. 3-18.

Miller, G. 1985. Dictionaries of the Mind. Proceedings of 23d ACL, pp. 305-314.

Nirenburg, S. and Y. Ben Asher. 1984. HUHU: The Hebrew University Hebrew Understander. *The Journal of Computer Languages,* 9, pp. 161-182.

Nirenburg, S. and V. Raskin. 1989. **Meaning in Human-Computer Interaction: Computational Lexical Semantics,** Amsterdam: Elsevier (in preparation).

Nirenburg, S., V. Raskin and A. B. Tucker 1985. Interlingua Design for TRANSLATOR. Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages , Colgate University, Hamilton, NY, August, pp. 224-244.

Nirenburg, S., V. Raskin and A.B. Tucker 1986. On Knowledge-Based Machine Translation. Proceedings of XI International Conference on Computational Linguistics, COLING-86, Bonn, Germany, August 1986, pp. 627-632

Nirenburg, S., V. Raskin and A.B. Tucker 1987. The Structure of Interlingua in TRANSLATOR. In: S. Nirenburg (editor), **Machine Translation: Theoretical and Methodological Issues,** ACL Series 'Studies in Natural Language Processing,' Cambridge University Press, pp. 90-113.

Nirenburg, S., I. Nirenburg and J.H. Reynolds 1985. Control Knowledge in POPLAR: a Personality-Oriented Planner. Proceedings of 3rd Annual Conference on Intelligent Systems and Machines. Oakland University, Rochester, Michigan, April, pp. 63-67.

Raskin, V. 1985. Linguistics and Natural Language Processing. Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages, Colgate University, Hamilton, NY, August, pp. 268-282. Also in: S. Nirenburg (ed.), **Machine Translation: Theoretical and Methodological Issues,** ACL Series 'Studies in Natural Language Processing,' Cambridge University Press, pp. 42-58.

Raskin, V. 1986. Script-Based Semantics. In: D.G. Ellis and W.E. Donohue (eds.), **Contemporary Issues in Language and Discourse Processes.** Hillsdale, NJ: Erlbaum, pp. 23-61.

SOED 1973. **Shorter Oxford English Dictionary.** Oxford: Clarendon Press, 3rd Ed.

Venezky, R.L. 1973. Computational Aids to Dictionary Compilation. In: R.Frank and A.F.Cameron (eds.), **A Plan for the Dictionary of Old English.** Toronto: University of Toronto Press, pp. 307-327.

Walker, D. E. 1984. Panel Session: Machine-Readable Dictionaries. Proceedings of COLING-84. Stanford University, Stanford, CA, July, p. 457.

Walker, D .E and R. A. Amsler 1986. The Use of Machine-Readable Dictionaries in Sublanguage Analysis. In: Grishman and Kittredge (eds.), pp. 69-83.

Zernik, U. and M. Dyer. 1985. Towards a Self-Extending Lexicon. Proceedings of 23d ACL, pp. 284-292.