Computers and the Humanities, Vol. 11, pp. 235-258. PERGAMON PRESS, 1978. Printed in the U.S.A.

0010-4817/78/0706-0235$2.00/0

Morphological Analysis and Lexicon Design for Natural-Language Processing

NICK CERCONE

1. Introduction

Language use presumes knowledge about words. Part of this knowledge is functional (how words are used) and part of it deals with the meanings of words. The lexical component of a computer memory model should be organized to deal adequately with both.

Before we begin designing a lexicon, we might observe how any standard dictionary is organized. This consideration quickly leads to chagrin. Words are not always assigned meanings in any consistent way; rather, various methods are used, be they extra-linguistic (e.g., diagrams) or explicit definitions. Also meanings are given in English, which is not a suitable formalism for lexical memory. This informal structure is not conducive to the construction of an "in toto" memory model.

The experimental program reported in Cercone (1975b) explores the nature and computational use of meaning representations for word concepts in an automated natural language understanding system. Word meanings are represented as extended semantic networks (see Schubert, 1974) based on propositions. These meaning representations are accessed via a lexicon.

Functional knowledge about words, on the other hand, can aid language interpretation when meaning analysis alone stalls or when making initial assumptions concerning anaphoric references or those associated with word order. For example, if a system were canonically to represent one sense of "give" as a three-place predicate *GIVE1(x,y,z) corresponding to the sentence "John gave Mary the book," where argument x is bound to John, y to Mary, and z to book, then the appearance of the preposition "to" in "John gave the book to Mary" signifies change in the order of arguments. As another example consider the definite determiner in the description "the red haired woman drinking wine" in contrast to the indefinite determiner in "a red haired woman drinking wine." Any two sentences in which these descriptions appear have quite distinct meanings.

The construction and maintenance of a lexicon, designed to operate with the meanings and functions of words, is described.

2. Morphological Analysis of English Words

The length of lexical items (the number of characters) and their codification can greatly affect subsequent processing times in automated language understanding systems. Lexical design must account for this. One obvious storage simplification is to perform morphological analysis on words. This facility would permit all entries with the same root form to be stored as a single lexical item along with indicators for permissible inflections. Morphological analysis is an integral component of the experimental program of Cercone (1975a) for many reasons, including the following advantages:

(i) Storage economy
It would be absurd to store all forms of lexical items directly since well-defined spelling rules exist which specify all normal word formations. A small, relatively simple analysis routine can save vast amounts of storage.

(ii) Interpretive assistance
A by-product of morphological analysis is the discovery of affixes that were added to the root word to form the word under analysis. Often these affixes, especially in the case of inflexional endings, determine the use of the word in an utterance, or, at least narrow its possibilities.

(iii) Learning new words
Whenever an unknown word is encountered in

Nick Cercone is in the computing science program at Simon Fraser University, Burnaby, British Columbia.

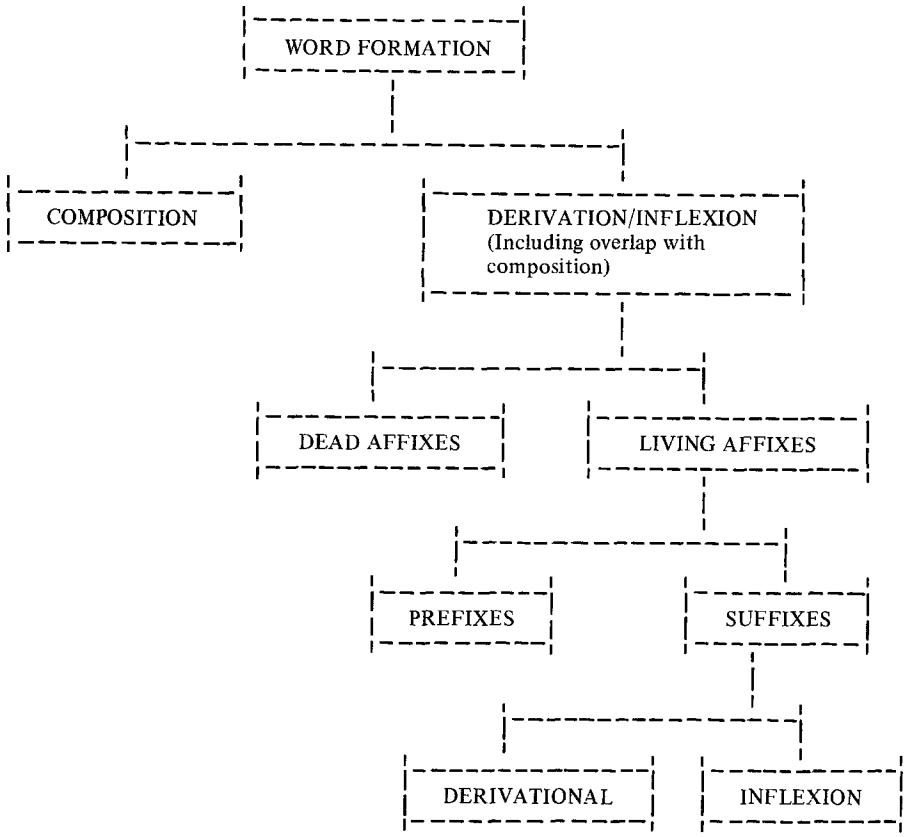


Figure 1. Short taxonomy of English word formation

text, preliminary analysis of structural and affix relationships can aid in determining the word's function in the utterance. Our ability to infer or guess meanings can only be enhanced through extensive morphological analysis.

(iv) Derivational information
Derivational affixes can affect word meaning, often in a systematic way, for example "-esque," like a; "non-," negation. A word like "booklet" can be satisfactorily understood as "little book" through morphological analysis without explicitly storing both words.

Figure 1 illustrates word formation as either (i) composition—the formation of a word by the close combination of two or more elements each of which is also a separate word, e.g., "goldsmith"; or (ii) derivation/inflexion—formation by the close combination of two or more elements only one of which can be a separate word, e.g., "kindness."

Word formation by composition generates compound words. There are compound nouns ("goldsmith," noun-noun; "blackboard," adjective-noun; "drawbridge," verb-noun), compound adjectives ("blameworthy," noun-adjective; "over-anxious," adverb-adjective), compound pronouns ("myself"), compound verbs ("overcome," adverb-verb; "daresay," verb-verb), and compound participles such as "airborne." Back formation is a special type of composition in which the second element denotes an agent or action, for example "housekeeper." Word formation also gives repetition compounds ("goody-goody," "fifty-fifty"). Nevertheless, most composite words yield little useful affix information except where there is overlap with derivation.

Inflexion and derivation are related, but while inflexion modifies a word (book-books-book's), derivation can result in the formation of a different word (kind-kindness).[1] Prefixes in English are always derivational. Suffixes may be derivational or inflexional, so we must distinguish between the two kinds.

The computer program which performs morpho-

logical analysis treats prefix-root-suffix relationships. Affixes are classified into two groups—living or dead. Examples of the dead group include: for-, "forgive," "forget"; with-, "withhold"; and -ant, "servant"; -le, "handle." The program is restricted to living affixes.

There are approximately 75 English prefixes. Most of the living ones amongst these are analyzed and include the following: a-, ante-, anti-, arch-, auto-, be-, bi-, co-, counter-, de-, dis-, em-, en-, ex-, extra-, fore-, hyper-, in-, inter-, mal-, mis-, non-, post-, pre-, pro-, re-, semi-, sub-, super-, trans-, ultra-, and un-. Since some living prefixes occur infrequently, they are stored as separate lexical items along with root words.

Usually the morphological analyzer can easily remove prefixes leaving just the root plus suffixes. For this reason, the major part of the program is devoted to the analysis of derivational and inflexional suffixes. Suffix analysis can help determine the word's part of speech. For example, a word with suffix "-est" can be an adjective, one with "-ist" would be a noun. Furthermore, we can determine the part of speech by examining the root, since inflexional suffixes do not change the part of speech. Inflexional suffixes usually come last in a word and do not "pile up." Since an analysis of regularly inflected words would always yield the root plus the inflexional suffix then inflected forms need not be stored separately. Winograd (1972) has shown this for many cases. On the other hand, roots with derivational suffixes are not always related to one particular syntactic class, e.g., "-ful," adjective – "forgetful," noun – "handful." Derivational suffixes can "pile up," as in "fertilizers" whose two suffixes "-ize" and "-er," do not close the word, followed by the inflexional suffix "-s" which does.[2] Because of this phenomenon, when suffix removal does not expose the root form, the analysis must procede recursively and must take into account the entire derivation up to that point of analysis. This is necessary to analyze words like "relationship" or "patronizingly." The suffixes in the program include: -able, -ible, -age, -al, -ance, -ation, -cy, -dom, -ed, -en, -er, -est, -ful, -ing, -ish, -ist, -ity, -ize, -less, -like, -ly, -ment, -ness, -ous, -s, -'s, -s', -ship, -some, -ster, -ward, -way, -wise. This list is far from exhaustive.

The basic algorithm for performing morphological analysis is given in Cercone (1974). Listing 1 is a sample output of that analyzer operating with a lexical structure constructed according to the syntax rules given in Table 4. The algorithm was implemented using a converted MACLISP modified to run on an IBM 360/67 under the Michigan Terminal System [MTS]. The STEM routine takes one argument: the English word to be analyzed. STEM returns a list containing four elements: (1) the original word; (2) the prefixes; (3) the root form; and (4) a list of suffixes. Whenever the analysis yields a root form from the original word with no prefix or suffix, the word NIL appears, corresponding to the second and/or fourth argument. In the bicolumnar form below, the left column entries show STEM routine invocations and entries in the right show the result of the analysis.

Typically words have multiple meanings and multiple forms. As humans, we have a remarkable ability correctly to recognize and parse contiguous words in utterances. It is desirable, in the interpretive phase of understanding, to identify the functions and meanings of words. Morphological analysis can aid this phase. For example, consider the multiple forms of the lexical item "drink." If the form "drinking" appears in an utterance, it can be regarded as a participle, a noun, or an adjective. The form "drinkings" can only be regarded as a noun. Listing 2 shows how the interpretive phase uses a morphological analyzer to identify a word's function. The lexical entry for "drink" appears in Figure A.2 of Appendix A. The CLASS routine (see Appendix B) extracts only the relevant portion of the lexical entry based on the morphology of its argument, i.e., some form of "drink" in this example.

Learning the meanings of new or unfamiliar words is not considered at this time, although information obtained through morphological analysis would be valuable in doing so. It is simply more economical in terms of processing time to store unfamiliar words or morphological rarities explicitly than to include programs for their analysis.

3. The Organization of the Lexicon

By way of introduction, let me point out that the next two subsections are distinguished for explanatory purposes only and there is no intent to dichotomize syntax and semantics. It is a rather touchy issue to decide whether some of the features and category items in the next section are syntactic at all.

3.1 Lexical Categories and Syntactic Features. Lexical items are items of vocabulary; usually, but not necessarily, words. Traditionally (the Aristotelian view), they are said to have both lexical (material) and grammatical (formal) meaning. This distinction between meanings can best be expressed in terms of open or closed classes (sets of alternatives) as explained below.

```
# R NEW:MACLISP
# 21:50.18
=(RESTORE  'CHKPT)                    =NIL
=(STEM   'INTEREST)                   =(INTEREST NIL INTEREST NIL)
=(STEM   'INTERESTED)                 =(INTERESTED NIL INTEREST (ED) )
=(STEM   'INTERESTING)                =(INTERESTING NIL INTEREST (ING) )
=(STEM   'INTERESTINGLY)              =(INTERESTINGLY NIL INTEREST (ING LY) )
=(STEM   'BASHES)                     =(BASHES NIL BASH (S) )
=(STEM   'BATHES)                     =(BATHES NIL BATHE (S) )
=(STEM   'BATHS)                      =(BATHS NIL BATH (S) )
=(STEM   'LEANING)                    =(LEANING NIL LEAN (ING) )
=(STEM   'LEAVING)                    =(LEAVING NIL LEAVE (ING) )
=(STEM   'DENTED)                     =(DENTED NIL DENT (ED) )
=(STEM   'DANCED)                     =(DANCED NIL DANCE (ED) )
=(STEM   'KISSES)                     =(KISSES NIL KISS (S) )
=(STEM   'CURVED)                     =(CURVED NIL CURVE (ED) )
=(STEM   'CURLED)                     =(CURLED NIL CURL (ED) )
=(STEM   'ROTTING)                    =(ROTTING NIL ROT (ING) )
=(STEM   'ROLLING)                    =(ROLLING NIL ROLL (ING) )
=(STEM   'PLAYED)                     =(PLAYED NIL PLAY (ED) )
=(STEM   'PLIED)                      =(PLIED NIL PLY (ED) )
=(STEM   'REALEST)                    =(REALEST NIL REAL (EST) )
=(STEM   'PALEST)                     =(PALEST NIL PALE (EST) )
=(STEM   'KNIVES)                     =(KNIVES NIL KNIFE (S) )
=(STEM   'PRETTILY)                   =(PRETTILY NIL PRETTY (LY) )
=(STEM   'NOBLY)                      =(NOBLY NIL NOBLE (LY) )
=(STEM   'PATRONIZINGLY)              =(PATRONIZINGLY NIL PATRON (IZE ING LY) )
=(STEM   'RELIABLE)                   =(RELIABLE NIL RELY (ABLE) )
=(STEM   'ACCESSIBLE)                 =(ACCESSIBLE NIL ACCESS (IBLE) )
=(STEM   'ACREAGE)                    =(ACREAGE NIL ACRE (AGE) )
=(STEM   'MILAGE)                     =(MILAGE NIL MILE (AGE )
=(STEM   'STOPPAGE)                   =(STOPPAGE NIL STOP (AGE) )
=(STEM   'CULTURAL)                   =(CULTURAL NIL CULTURE (AL) )
=(STEM   'RIDDANCE)                   =(RIDDANCE NIL RID (ANCE) )
=(STEM   'OPERATION)                  =(OPERATION NIL OPERATE (ATION) )
=(STEM   'STARVATION)                 =(STARVATION NIL STARVE (ATION) )
=(STEM   'ACCURACY)                   =(ACCURACY NIL ACCURATE (CY) )
=(STEM   'CONSTANCY)                  =(CONSTANCY NIL CONSTANT (CY) )
=(STEM   'CAPTAINCY)                  =(CAPTAINCY NIL CAPTAIN (CY) )
=(STEM   'DUKEDOM)                    =(DUKEDOM NIL DUKE (DOM) )
=(STEM   'HANDFUL)                    =(HANDFUL NIL HAND (FUL) )
=(STEM   'PATRIOTISM)                 =(PATRIOTISM NIL PATRIOT (ISM) )
=(STEM   'SOCIALIST)                  =(SOCIALIST NIL SOCIAL (IST) )
=(STEM   'VISIBILITY)                 =(VISIBILITY NIL VISIBLE (ITY) )
=(STEM   'SENTIMENTALITY)             =(SENTIMENTALITY NIL SENTIMENT (AL ITY) )
=(STEM   'CIVILIZE)                   =(CIVILIZE NIL CIVIL (IZE) )
=(STEM   'PENNILESS)                  =(PENNILESS NIL PENNY (LESS) )
=(STEM   'RESTLESS)                   =(RESTLESS NIL REST (LESS) )
=(STEM   'CHILDLIKE)                  =(CHILDLIKE NIL CHILD (LIKE) )
=(STEM   'ARGUMENT)                   =(ARGUMENT NIL ARGUE (MENT) )
=(STEM   'SHIPMENT)                   =(SHIPMENT NIL SHIPMENT NIL)
=(STEM   'DRUNKENNESS)                =(DRUNKENNESS NIL DRUNK (EN NESS) )
=(STEM   'GOODNESS)                   =(GOODNESS NIL GOOD (NESS) )
=(STEM   'WICKEDNESS)                 =(WICKEDNESS NIL WICKED (NESS) )
=(STEM   'NERVOUS)                    =(NERVOUS NIL NERVE (OUS) )
=(STEM   'ASLEEP)                     =(ASLEEP A SLEEP NIL)
=(STEM   'ANTEROOM)                   =(ANTEROOM ANTE ROOM NIL)
=(STEM   'ANTICHRIST)                 =(ANTICHRIST ANTI CHRIST NIL)
=(STEM   'ARCHBISHOP)                 =(ARCHBISHOP ARCH BISHOP NIL)
=(STEM   'AUTOBIOGRAPHY)              =(AUTOBIOGRAPHY AUTO BIOGRAPHY NIL)
=(STEM   'BEMOAN)                     =(BEMOAN BE MOAN NIL)
=(STEM   'BIANNUAL)                   =(BIANNUAL BI ANNUAL NIL)
=(STEM   'COUNTERACT)                 =(COUNTERACT COUNTER ACT NIL)
```

Listing 1. Output from morphological analysis

(Listing 1—continued)

```
=(STEM  'DECODE)                =(DECODE DE CODE NIL)
=(STEM  'ENDANGER)              =(ENDANGER EN DANGER NIL)
=(STEM  'EMBED)                 =(EMBED NIL EMBED NIL)
=(STEM  'HYPERACTIVE)           =(HYPERACTIVE HYPER ACT (IVE) )
=(STEM  'IMMORAL)               =(IMMORAL IM MORAL NIL)
=(STEM  'INTERRELATIONSHIP)     =(INTERRELATIONSHIP INTER RELATE (ATION SHIP) )
=(STEM  'MISCONDUCT)            =(MISCONDUCT MIS CONDUCT NIL)
=(STEM  'NONSTOP)               =(NONSTOP NON STOP NIL)
=(STEM  'POSTWAR)               =(POSTWAR POST WAR NIL)
=(STEM  'RECONSIDER)            =(RECONSIDER RE CONSIDER NIL)
=(STEM  'SUBAWARENESS)          =(SUBAWARENESS SUB AWARE (NESS) )
=(STEM  'SUPERMARKET)           =(SUPERMARKET SUPER MARKET (NIL)
=(STEM  'ULTRACONSERVATION)     =(ULTRACONSERVATION ULTRA CONSERVE (ATION) )
=(STEM  'UNNECESSARY)           =(UNNECESSARY UN NECESSARY NIL)
=(STEM  'UNREST)                =(UNREST UN REST NIL)
=(STEM  'COEDUCATION)           =(COEDUCATION CO EDUCATE (ATION) )
=(STEM  'COOPERATIONAL)         =(COOPERATIONAL CO OPERATE (ATION AL) )
=(STEM  'DEHUMANIZE)            =(DEHUMANIZE DE HUMAN (IZE) )
=(STEM  'INEQUALITY)            =(INEQUALITY IN EQUAL (ITY) )
=(STEM  'REELIGIBILITY)         =(REELIGIBILITY RE ELIGIBLE (ITY) )
=(STEM  'LOUDLY)                =(LOUDLY NIL LOUD (LY) )
=(STEM  'LENGTHWAYS)            =(LENGTHWAYS NIL LENGTH (WAY S) )
=(STEM  'HOMEWARDS)             =(HOMEWARDS NIL HOME WARD S) )
=(STEM  'NONLOUDLY)             =(NONLOUDLY NON LOUD (LY) )
=(STEM  'BEER)                  =(BEER NIL BEER NIL)
=(STEM  'MURDER)                =(MURDER NIL MURDER NIL)
=(STEM  'OTHER)                 =(OTHER NIL OTHER NIL)
=(STEM  'ARABESQUE)             =(ARABESQUE NIL ARAB (ESQUE) )
=(STEM  'REALIZE)               =(REALIZE NIL REAL (IZE) )
=(STEM  'GROTESQUE)             =(GROTESQUE NIL GROTESQUE NIL)
=(STEM  'NONAGENARIAN)          =(NONAGENARIAN NIL NONAGENARIAN NIL)
=(STEM  'NONALIGNMENT)          =(NONALIGNMENT NON ALIGN (MENT) )
=(MTS)
```

Lexical categories are properties associated with lexical items used in parsing. Through categories, the representation of an appropriate lexical item can be selected from the lexicon. Categories are identified with classes (or parts of speech) rather than expressions of a particular syntagmatic relation among items in an utterance. Generally, classes are separated into open and closed classes. Characteristically, closed classes have a strictly limited membership which cannot be increased by adding new formations or loanwords (words which have been incorporated by one language from another language). The significance of closed-class items is best expressed by their grammatical function. In contrast, open classes have a large, flexibly increasing membership. The meaning of open class words is best expressed through synonyms. The difference between the classes represents a mixture of criteria, both statistical (the number of forms in a class), and diachronic (concern with the way in which language changes over time).

The lexical categories for English shown in Table 1 adapt categories used by Woods et al. (1972) and Winograd (1972). The lexical items used in this research have been classified according to class and feature (features shown in Tables 2 and 3). Some of the decisions made in this classification were arbitrary, especially those pertaining to whether a word or group of words should form a new class or be given a new feature within a class. However, the classification scheme is used to aid, not constrain, parsing; detailed concern about this type of arbitrariness is unwarranted. Please note that the problem with the use of coordinators, which can link words, phrases, clauses, or sentences, has not been addressed. Any adequate solution to this problem might entail substantial changes in the interpretation or assignment of closed categories (and syntactic features).

The syntactic features which can be attached to the various lexical items are shown in Table 2 (open category) and Table 3 (closed category) and explained in detail below. These features are necessary to insure formal agreement in person, number, or tense between two or more lexical items, or parts of sentences.

Most of these terms are used in their ordinary sense. The special labels are as follows: PERS

```
#  R NEW:MACLISP
#  22:39.27
=  (RESTORE  'CHKPT)
=        NIL
=
=  (CLASS  'DRINK)
=        (N (NS ( (0 0) ( /*DRINK2) ) ( (0 0) (/*DRINK4) ) ) A (PRES
=        ( (0 0)  (/*DRINK1 P1 P2) ) ( (0 0) (/*DRINK1A P1 P2) )
=        ( (0 0)  (/*DRINK3 P1 P2) ) ) )
=
=  (CLASS  'DRINKS)
=        (N (NP ( (0 0) (/*DRINK2) ) ( (0 0) (/*DRINK4) ) ) A (PRES
=        TPS ( (0 0) (/*DRINK1 P1 P2) ) ( (0 0) (/*DRINK1A P1 P2)
=        ) ( (0 0) (/*DRINK3 P1 P2) ) ) )
=
=  (CLASS  'DRINKER)
=        (N (PERS ( (0 0) (/*DRINK5) ) ) )
=
=  (CLASS  'DRINKERS)
=        (N (NP PERS ( (0 0) (/*DRINK5) ) ) )
=
=  (CLASS  'DRINKING)
=        (N (NS ( (0 0) (/*DRINK6) ) ) A (PART ( (0 0) (/*DRINK1
=        P1 P2) ) ( (0 0) (/*DRINK1A P1 P2) ) ( (0 0) (/*DRINK3
=        P1 P2) ) ) NM (ADJ CLASF ( (0 0) (/*DRINK2 P1 P2) ) ( (0
=         0) (/*DRINK4 P1 P2) ) ( (0 0) (/*DRINK5 P1 P2) ) ( (0
=        0) (/*DRINK6 P1 P2) ) ) )
=
=  (CLASS  'DRINKINGS)
=        (N (NP NS ( (0 0) (/*DRINK6) ) ) )
=
=  (CLASS  'DRINKABLE)
=        (N (NS ( (0 0) (/*DRINK2) ) ( (0 0) (/*DRINK4) ) ) NM (ADJ
=        CLASF ( (0 0) (/*DRINK2 P1 P2) ) ( (0 0) (/*DRINK4 P1
=        P2) ) ( (0 0) (/*DRINK5 P1 P2) ) ( (0 0) (/*DRINK6 P1 P2)
=        ) ) )
=
=  (CLASS  'DRINKABLES)
=        (N (NP NS 9 (0 0) (/*DRINK2) ) ( (0 0) (/*DRINK4) ) ) )
=
=  (CLASS  'DRINKWISE)
=        (AM (AT3 ( (0 0) (/*DRINK1 KIND) ) ( (0 0) (/*DRINK 1A
=        KIND) ) ) )
=
=  (CLASS  'DRINKLIKE)
=        (NM (ADJ ( (0 0) (/*DRINK2 P1 P2) ) ( (0 0) (/*DRINK4
=        P1 P2) ) ( (0 0) (/*DRINK5 P1 P2) ) ( (0 0) (/*DRINK6 P1
=        P2) ) ) )
=
=  (CLASS  'DRINKETTE)
=        (N (DIM ( (0 0) (/*DRINK2) ) ( (0 0) (/*DRINK4) ) ) )
=  (CLASS  'DRINKETTES)
=        (N (NP DIM ( (0 0) (/*DRINK2) ) ( (0 0) (/*DRINK4) ) ) )
=
=  (CLASS  'DRINKIE)
=        (N (DIM ( (0 0) (/*DRINK2) ) ( (0 0) (/*DRINK4) ) ) )
=
=  (MTS)
```

Listing 2. Relevant selection from lexical entry

OPEN CATEGORIES

N . . . . . nominal, typically either a noun (man, airplane, city) or a proper noun (John, Canada)
A . . . . . action, typically a verb (walk, throw, fly)
NM. . . . . nominal modifier, typically an adjective (tall, happy)
AM. . . . . action modifier, typically an adverb (quickly, suddenly)

CLOSED CATEGORIES

CONJ	=	conjunction (and, or, but)
BIND	=	binder (before, if)
PREP	=	preposition (to, for, over)
PRO	=	pronoun (I, you, they)
DET	=	determiner (the, a, those)
ORD	=	ordinal (first, second, last, final)
NEG	=	negative (not)
COMP	=	comparative (more, less, greater)
OP	=	operation (plus, times)
QWORD	=	question nominal (who, what, why)
QNTFR	=	quantifier (some, any, none)
PRT	=	particle (knock "out")
NUM	=	number (one, two, three)
INTJ	=	interjection (oh)

Table 1. Lexical categories

indicates a personal nominal (e.g., employee); DIM indicates a diminutive (e.g., booklet). POSS indicates possession, as in "John's." The time features differ if the nominal is a time word (e.g., day, year–TIME) or indicates a relative time (e.g., yesterday–FTIME). The feature AUX indicates an auxiliary (i.e., a verb form used in forming the tenses, moods, and voices of other verbs). Included in the auxiliaries are the features BE, DO, HAVE, WILL, and MODAL,[3] which help determine constituents of action phrases.

Classifiers may also be nominals, as in Winograd's (1972) example, "water meter cover adjustment screw." The type features attached to AM's are specific to adverbial modifiers. In part, adverbial modifiers are based on Zadeh's (1972) work, especially the adverbial modifiers with features "AT1" and "AT2." Features classify adverbs according to how they operate in an utterance. The feature AT1 is attached to adverbs that act on single fuzzy sets as in "John was VERY decent"[4] where "very" raises the criteria of all aspects that contribute to decency. The feature AT2 applies when the adverb operates on a predicate in the following way: In "John was ESSENTIALLY decent," "essentially" accentuates those aspects of decency which are most crucial to its possession and de-emphasizes those features which are less crucial. The features AT11 and AT22 are similar to AT1 and AT2. They indicate more context dependence; the effect of AM's with these features is partially determined by their proximity to the verb they modify, for example the adverb "slightly" as well as some sentence adverbials. The feature AT3 applies to predicate limiting adverbs such as "emotionally" and "healthwise." Manner adverbs, e.g., quickly, quietly, etc., have the feature AT4 attached. Whenever a word acts as an adverb of degree it is given the feature AT5, as in "I was DEAD tired." Finally AT6 is the applicable feature for modal adverbs, such as "certainly," and "possibly."

3.2 Meaning Representations for Word Concepts. Associated with open-class category words are meaning representations: one for each sense of the word. The structure of a meaning representation is

OPEN CATEGORIES

N's

NS	singular	TIME	time
NP	plural	FTIME	functional time
COLL	collective	PERS	personal
POSS	possessive	DIM	diminutive

A's

AUX	auxiliary	BE	be
WILL	future	DO	do
HAVE	have	MODAL	modality
TRANS	transitive	ITRNS	intransitive
PART	participle	IREG	irregular
PRES	present	PAST	past
INF	infinitive	TPS	3rd person

NM's

ADJ	adjective
COM	comparative
SUP	superlative
CLASF	classifier

AM's

AT1	adverb type one
AT11	adverb type one one
AT2	adverb type two
AT22	adverb type two two
AT3	adverb type three
AT4	adverb type four
AT5	adverb type five
AT6	adverb type six
AAA	adverb modifying another adverb or adjective
AA	adverb modifying an action
AP	adverb modifying a preposition or prepositional phrase
ADT	adverb specifying definite time
AIT	adverb specifying indefinite time
AL	adverb specifying location
AJ	adverbial adjunct

Table 2. Syntactic features (open categories)

CONJ	BIND	PREP	NUM
ORD	OP	PRT	NEG
COMP	QWORD	INTJ	

PRO's

NP	plural	COLL	collective
NS	singular	POSS	possessive
REL	relative	PERS	personal
DEM	demonstrative	DEF	definite
INDEF	indefinite	SUB	subject
OBJ	object		

DET's

DEF	definite	INDEF	indefinite
NP	plural	COLL	collective
NS	singular	DEM	demonstrative
QDET	question		

QNTFR's

NE	negative	NS	singular
NONUM	no number	NP	plural
COLL	collective		

Table 3. Syntactic features (closed categories)

based on the semantic network notation developed by Schubert (1974). Pragmatic and semantic information are included in a meaning representation for words.

Figures 2 through 7 show networks that illustrate some of the main senses of the word "drink," concentrating on its action aspects. For illustrative purposes Figures 2, 4 and 7 are divided into a pragmatic section and a semantic section. The pragmatic section includes the template(s) that guides the parse of the utterance and two lists: the first contains propositions that represent the implications that are likely to be needed for the comprehension of subsequent text; and the second contains propositions representing critical implications that we expect to match in the surface structure. In Figure 2 this first list is (P3) and the second list is (P1,P2). The semantic section contains the network that represents the meaning of the word sense. Figures 3, 5, and 6 show various nominal senses of the word "drink."

Notice that Figures 2, 4, and 7 all have the notion of "change in containment location" in common. This corresponds to a "general concept" that subsumes not only differing senses of "drink," but also other more specific concepts as well, like "eating" or "receiving an enema." This observation has led to the following consideration.

When creating the meaning representations (as extended semantic networks) for concepts, it is desirable to avoid the duplication of propositions in storage. If we extract more general concepts from the specific concepts that they subsume (totally or in part), we can avoid duplication by associating the common propositions with the more general concept. In a sense the work of both Schank (1972) and Wilks (1973) support the contention that the meaning of a concept is best represented by predications at the highest level of generality that adequately explain the term's meaning. Thus we extract from "drinking" (and "eating," etc.) the structure shown in Figure 8. We might reasonably label the concept expressed by this structure "ingest." It is important to note, however, that while Schank and Wilks might conclude that "ingesting" is a primitive action, I consider it a general concept. This applies to all primitive actions put forward of Schank and Wilks. Examination of

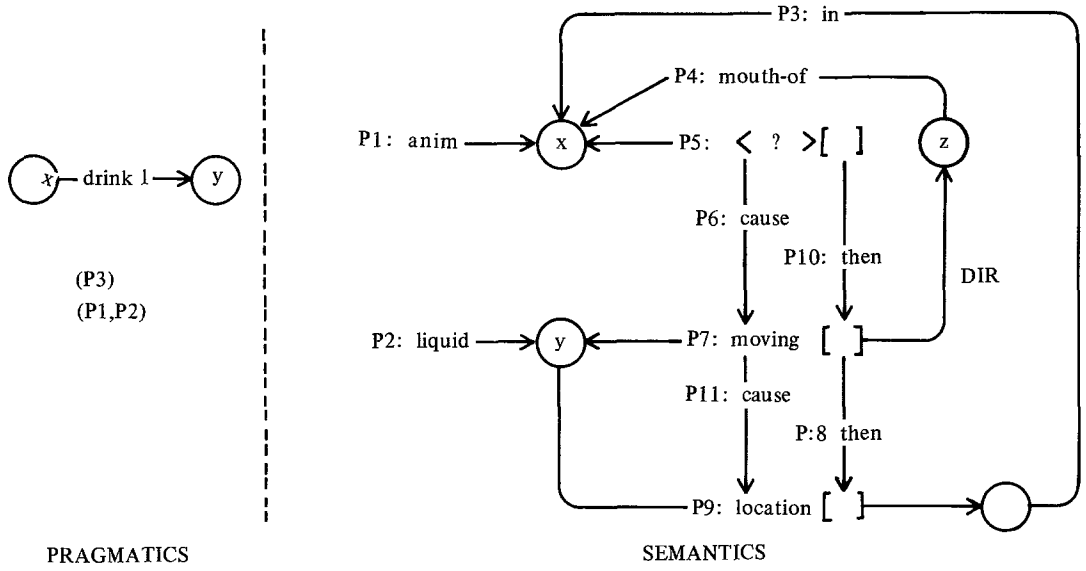


Figure 2.
"(John) 'drinks' (water)"
"(Mary) 'drinks' (prune juice)"

Figure 8 shows clearly that ingesting is "not a primitive action" but one whose meaning is expressed in terms of causes, motion, time, and other concepts.

At this point the original representations for the various action senses of "drink," i.e., Figures 2, 4, and 7, can be replaced with more simplified diagrams based on the general concept "ingest" (Figure 8). In similar fashion Figure 10 diagrams one meaning of "eating," again based on the general concept "ingest."

The key to effective use of the meaning representation for comprehension centers on developing propositions with arguments that we expect to match in the surface utterance. The lexical item for "drink" would contain, among other things, pointers to a list of the arguments that we expect to match with words in the text and are most frequently needed for comprehension. At times, however, other propositions may be required for comprehension. The word sense illustrated in Figure 2 shows that we expect, in an utterance about drinking, an anim(x) and a liquid(y), propositions P1 and P2. But the question can be posed, "What is the effect of John's drinking?" To answer this question entails a further investigation of other propositions in the network, especially the first list of implications. Although it is implicit in the semantic structure, we make explicit in the pragmatic structure the inference that "x - drink - y" necessarily implies that it causes y's location to be "in" x at some time after x initiates the drinking action. Of course, since this implication is common to all senses of "drink" (and eats, inhales, etc.) it is

Figure 3.
"(John is a) 'drinker' "

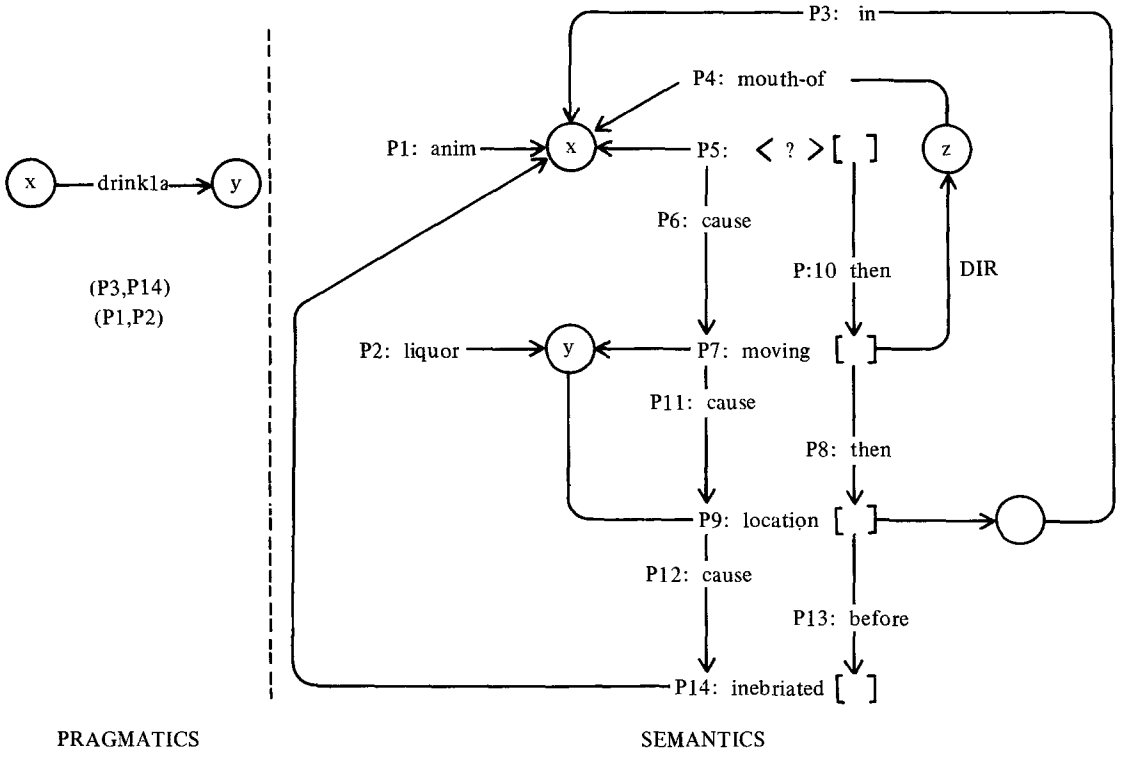


Figure 4.
"(John) 'drinks' (whiskey)"
"(John) 'drinks' "
"(Mary has a) 'drinking' (problem)"
"(Mary) 'drinks' (a lot)"

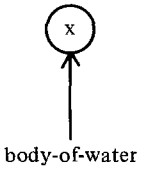


Figure 5.
"(Throw John into the) 'drink' "

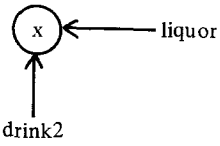


Figure 6.
"(John is drinking a) 'drink' "

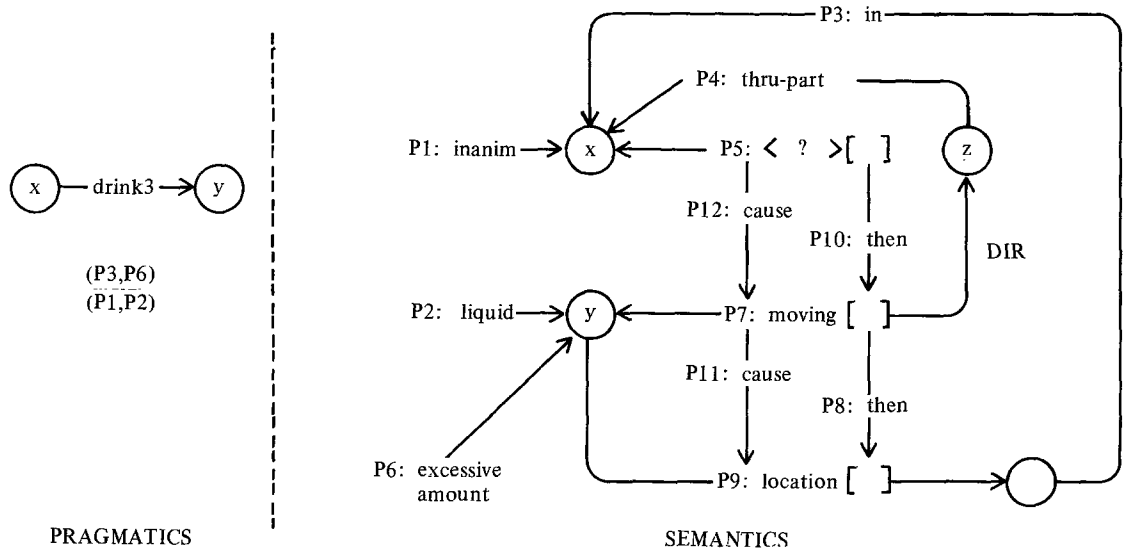


Figure 7.
"(My car) 'drinks' (gasoline)"
"(The donut) 'drinks' (coffee)"

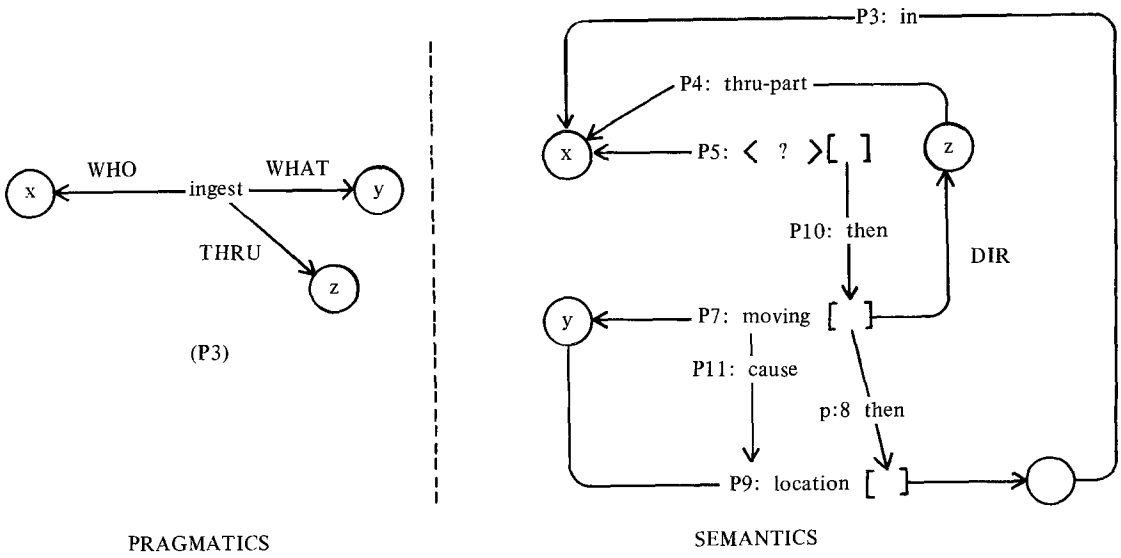


Figure 8.
"ingest"

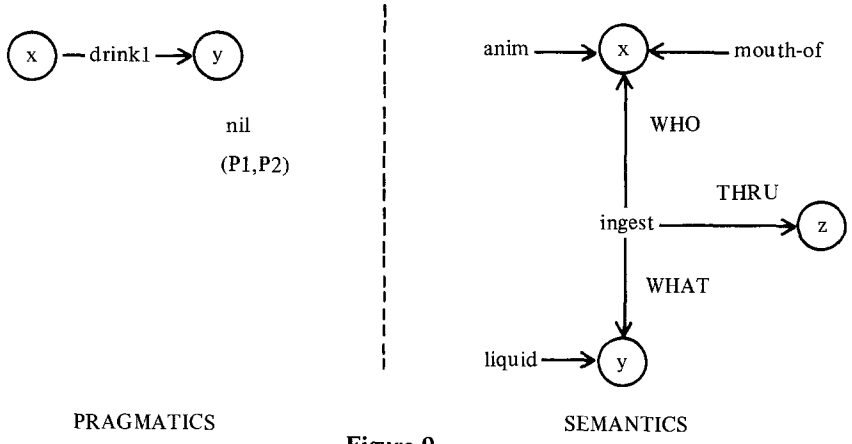


Figure 9.
"(John) 'drinks' (water)"

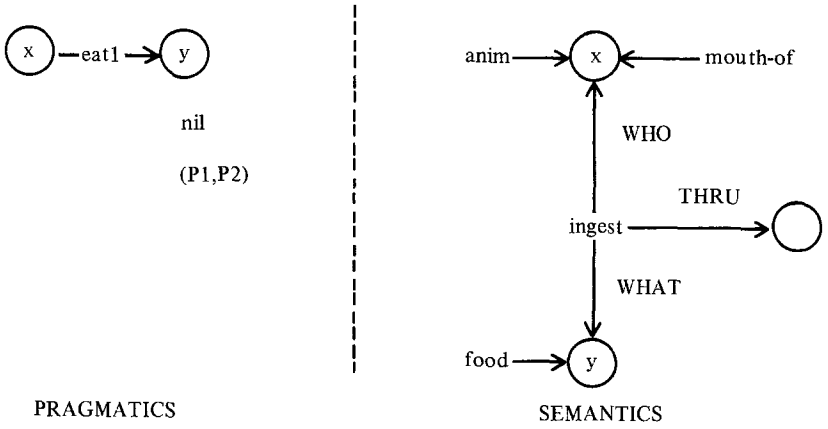


Figure 10.
"(John) 'eats' (cake)"

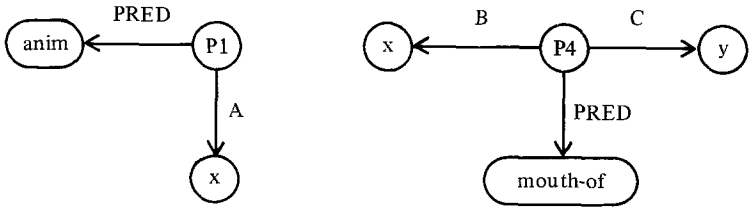


Figure 11.

abstracted into the same general concept "ingest" as well, as shown in Figure 8.

. The semantic structure for "drinks" is represented as properties attached to each word sense. The main properties include ARGS, the list containing arguments in the word sense; IMPLICS, a list of implications; the propositions P1, P2, etc., that relate the arguments to predicates that make up the network explicating the given word sense; and templates of the form

arg1 arg2 . . . argi WORD argi+1 . . . argn.

The implications make the most commonly used inferences part of the representation of a concept. The propositions, for example P1 and P4 shown in Figure 2 are, in turn, represented as shown in Figure 11. See Appendix A for sample lexical entries, in particular the entry for "drink."

Many advantages accrue by representing meaning formulas in this way. First, unlike Wilks' (1973) meaning formulas, the representation is suggestive of the meaning of a word. I see no justification for (binary) lexical decomposition trees as meaning representations for words since such trees neither suggest the type of processing required nor the propositions they encode.[5]

A second and major advantage is that the meaning

⟨lexical entry⟩	: : =	(⟨root⟩ ⟨meaning *⟩)
⟨root⟩	: : =	root of word given meaning
⟨meaning⟩	: : =	⟨lexical category⟩ ⟨category value⟩ ⟨synonym⟩ ⟨antonym⟩ ⟨compound⟩ ⟨idiom⟩ ⟨abbrev⟩
⟨lexical category⟩	: : =	⟨open category⟩ ¦ ⟨closed category⟩
⟨open category⟩	: : =	N ¦ A ¦ NM ¦ AM
⟨closed category⟩	: : =	CONJ ¦ PREP ¦ . . .
⟨category value⟩	: : =	⟨ ⟨root feature list⟩ ⟨word sense formula *⟩ ⟩*
⟨synonym⟩	: : =	SYN ⟨synonym value⟩ * ¦ ⟨lambda⟩
⟨antonym⟩	: : =	ANT ⟨antonym value⟩ * ¦ ⟨lambda⟩
⟨idiom⟩	: : =	ID ⟨idiom value⟩ * ¦ ⟨lambda⟩
⟨abbrev⟩	: : =	ABB ⟨abbrev value⟩ * ¦ ⟨lambda⟩
⟨compound⟩	: : =	COMPD ⟨compound value⟩ * ¦ ⟨lambda⟩
⟨synonym value⟩	: : =	a synonym for the root
⟨antonym value⟩	: : =	an antonym for the root
⟨idiom value⟩	: : =	the idiom or slang expression involving the root
⟨abbrev value⟩	: : =	abbreviation for the root
⟨compound value⟩	: : =	(⟨tree⟩ *)
⟨tree⟩	: : =	(⟨word⟩ ⟨result⟩ ⟨tree⟩ *)
⟨result⟩	: : =	⟨word⟩ ¦ ⟨lambda⟩
⟨word⟩	: : =	any word
⟨lambda⟩	: : =	
⟨root feature list⟩	: : =	(⟨morph code⟩⟨root feature*⟩) *
⟨word sense formula⟩	: : =	(the construction of semantic units and/or concepts that express the correct word sense l a function to be applied)
⟨morph code⟩	: : =	-ING ¦ -ED ¦ . . . ¦ ⟨lambda⟩
⟨root feature⟩	: : =	AT1 ¦ DEF ¦ BIND ¦ . . .

Table 4. Syntax for lexical items

```
# R NEW:MACLISP
# 20:23.08
= (RESTORE 'CHKPT)
=       NIL
=
= (FIND 'SOME OCAT)
=       NIL
=
= (FIND 'DRINK OCAT)
=       ( ( (N ( (NIL NS) (S NP) (ABLE NS) (ETTE DIM) (IE DIM)
=       ) ( ( (0 0) (/*DRINK2) ) ( ( 0 0) (/*DRINK4) ) ) ( (ING
=       NS) ) ( ( (0 0) (/*DRINK6) ) ) ( (ER PERS) (EER PERS)
=       (IST PERS) ) ( ( (0 0) (/*DRINK5) ) ) (SYN DRAFT POTATION
=       BEVERAGE LIQUOR) (ID BOOZE HOOCH MOONSHINE) ) (A
=       ( (NIL PRES) (S PRES TPS) (ING PART) ) ( ( (0 0) (/*DRINK1
=       P1 P2) ) ( (0 0) (/*DRINK1A P1 P2) ) ( (0 0) (/*DRINK3
=       P1 P2) ) ) (SYN CONSUME SWALLOW IMBIBE GUZZLE TOAST)
=        (ID SWIG SOP/-UP) ) (AM ( (WISE AT3) (WAYS AT3) )
=       ( ( (0 0) (/*DRINK1 KIND) ) ( (0 0) (/*DRINK1A KIND)
=       ) ) ) (NM (ABLE ADJ CLASF) (ING ADJ CLASF) (LIKE
=       ADJ) ) ( ( (0 0) (/*DRINK2 P1 P2) ) ( (0 0) (/*DRINK4
=       P1 P2) ) ( (0 0) (/*DRINK5 P1 P2) ) (0 0) (/*DRINK6
=       P1 P2) ) ) ) ) )
=
= (FIND 'SEXUAL OCAT)
=       NIL
=
= (FIND 'SOME ADVERB)
=       NIL
=
= (FIND 'DRINK ADVERB)
=       NIL
=
= (FIND 'SEXUAL ADVERB)
=       ( ( (AM ( (LY AT3 AAA AA) ) ( ( (0 0) (/*SEXUAL1) ) ) ) ) )
=
= (COMBINE OCAT ADVERB '*NIL)
=       NIL
=
= (FIND 'SEXUAL OCAT)
=       ( ( (AM ( (LY AT3 AAA AA) ) ( ( (0 0) (/*SEXUAL1) ) ) ) ) )
=
= (FIND 'SOME OCAT)
=       NIL
= (FIND 'DRINK OCAT)
=       ( ( (N ( (NIL NS) (S NP) (ABLE NS) (ETTE DIM) (IE DIM)
=       ) ( ( (0 0) (/*DRINK2) ) (/*DRINK4) ) ) ( (ING
=       NS) ) ( ( (0 0) (/*DRINK6) ) ) ( (ER PERS) (EER PERS)
=       (IST PERS) ) ( ( (0 0) (/*DRINK5) ) ) (SYN DRAFT POTATION
=       BEVERAGE LIQUOR) (ID BOOZE HOOCH MOONSHINE) ) (A
=       ( (NIL PRES) (S PRES TPS) (ING PART) ) ( ( (0 0) (/*DRINK1
=       P1 P2) ) ( (0 0) (/*DRINK1A P1 P2) ) ( (0 0) (/*DRINK3
=       P1 P2) ) ) (SYN CONSUME SWALLOW IMBIBE GUZZLE TOAST)
=        (ID SWIG SOP/-UP) ) (AM ( (WISE AT3) (WAYS AT3) )
=       ( ( (0 0) (/*DRINK1 KIND) ) ( (0 0) (/*DRINK1A KIND)
=       ) ) ) (NM ( (ABLE ADJ CLASF) (ING ADJ CLASF) (LIKE
=       ADJ) ) ( ( (0 0) (/*DRINK2 P1 P2) ) ( (0 0) (/*DRINK4
=       P1 P2) ) ( (0 0) (/*DRINK5 P1 P2) ) ( (0 0) (/*DRINK6
=       P1 P2) ) ) ) ) )
=
= (DICTADD OCAT '(S O M E) '*
=   '( (PRO (INDEF NS NP) ) (QNTRF (COLL NS NP NONUM) ) ) )
=       NIL
=
= (FIND 'SOME OCAT)
=       ( ( (PRO (INDEF NS NP) ) (QNTRF (COLL NS NP NONUM) ) ) )
=
= (MTS)
```

Listing 3. Lexical manipulation and maintenance

representation for a word is not required to be in terms of "primitives." Rather, each predicate in the propositions that form the network representing the word's meaning can be represented in an analogous manner. In particular the notion of a "cause" seems no more "primitive" than "drink." This method of representing word meanings enhances the representational schema for comprehension, since any amount of detail can be in the meaning representations by adding propositions to the networks.

Third, inference mechanisms, heuristic processing algorithms, and superimposed knowledge-organizing schemas can be incorporated using this representation for word meanings as easily as in any other representation. Incomplete information in surface text can be inferred, when necessary, directly from the meaning representation, in some cases as a missing argument. This type of meaning representation for lexical items is further explained in Cercone (1975a). For a brief sketch of parsing (without an explicitly stated grammar) based on this representation, see Appendix C.

4. Formal Specification of Lexical Items

Table 4 shows the grammar by which lexical items are entered in the dictionary. The notation used is basically the Backus Naur Form (BNF) metalanguage with the addition of the Kleene * operator. The metalinguistic characters include brackets ⟨ ⟩, Kleene operator *, the form ::=, and the bar |. Brackets surround phrase-class names which stand for sets of entities. The form ::= can be read as "is of the form." The bar denotes alternation, one form or the other. And the * defines an arbitrarily repeatable (zero or more) constituent when surrounded by brackets; otherwise, it defines an arbitrarily repeatable (one or more constituent; e.g., ⟨a*⟩ means zero or more a's, while a* or ⟨a⟩* means one or more a's.

In Appendix A, examples of closed and open category items are shown as they exist in the lexicon (Cercone, 1975a). They were constructed according to the syntax rules shown in Table 4.

5. Lexical Manipulation and Maintenance

In order to enable the rapid retrieval of relevant lexical information, a scheme was developed that exploits the way tree structures are stored in LISP. The root form is a binary branching tree that suggests a search method similar to a binary search. Letters in the query word serve as an index to a subset of lexical entries which contain the letters in corresponding positions. For example, in "drink," the "d" would be used to locate the lexical items beginning with "d." All other lexical items would not be considered further. The letter "r" would locate all items that begin with "dr" and so on until the word is found. In this way the number of searches needed to locate a lexical item is directly proportional to the number of letters and the size of the lexicon. This is easily done in LISP.

This lexical structure was designed for a small (300 words) dictionary used with an experimental program designed to create extended semantic-network meaning-representations for various utterances. The program, with relatively few heuristics, creates network structures on the average of about three seconds of CPU time per sentence (simple sentences, active voice only). This is accomplished with an "interpretor only" LISP system running under the Michigan Terminal System [MTS] on an IBM 360/67 computer. Ninety-five percent of this CPU time is devoted to non-lexical referencing operations. Experiments are being designed to gather exhaustive statistics to determine the running times of lexical manipulations (insertion, deletion, searching, etc.) for different size lexicons with various types of organizations (alphabetic, letter-frequency oriented, use-frequency oriented, etc.).

Listing 3 is a sample output which shows, in the following order, a search through dictionaries for lexical items, the merging of two dictionaries, a search for the same items in the merged version, an addition to the merged dictionary, and finally a search for the newly added lexical item. The FIND routine has two arguments: the first is the word to be found and the second specifies the dictionary to be searched.

The algorithms for manipulating and maintaining lexical items as shown in Listing 3 are given in Appendix B.

6. Conclusions

In this paper the construction of a lexicon, as well as the manipulation and maintenance of lexical items, has been explained. This lexicon has been used in an experimental program, see Cercone (1975a), that was designed to create semantic structures from utterances for the ultimate purpose of understanding natural language. This lexicon has proved to be significant to the experimental program because of the ease with which lexical items can be manipulated and maintained. Since the lexicon has a uniform structure, the routines which access and manipulate lexical information are relatively simple to understand and use.

Acknowledgments

I would like to thank Rici Liknaitzky for his ideas and programming aid, especially with the lexical maintenance routines. I am indebted to Len Schubert, Jeff Sampson, and Carol Murchison for their careful reading and suggestions. I would also like to thank the reviewers, especially Don Ross of the University of Minnesota, for their invaluable suggestions and insight. Part of this work was supported by the National Research Council of Canada, grant A4309.

Appendix A

The lexicon is organized as a general list structure comprised of many similar structures. The first element is a list of all root forms beginning with the letter A, the second, those beginning with B, etc. Within each list element a similar list organization is imposed. Each meaning-sense of a word contains a pointer to its corresponding meaning-representation (a proposition-based semantic network, see section 3.2 above and Cercone, 1975b). "Drinking," "drinkable," and "drinklike" all have pointers to *DRINK2, *DRINK4, *DRINK5, and *DRINK6 as possible meanings within the utterance in which they appear. The following are sample entries, first from the closed category (Figure A1) and then from the open category (Figure A2) lexicon.

```
(B(E(F(O(R(E(*(BIND () (*BEFORE1))
                (PREP () (*BEFORE2))
                ((AM (AIT)) (*BEFORE3))) ))))
    (H(I(N(D(*(PREP () (*BEHIND))) ))))
    (L(O(W(*(PREP () (*BELOW1))
             (AM ((AP AA)) (*BELOW2))) )))
    (N(E(A(T(H(*(PREP () (*BENEATH1))
                 (AM ((AP AA)) (*BENEATH2))) ))))
     (S(I(D(E(* (PREP () (*BESIDE))) )))))
  (O(T(H(*(QNTRF ((NP COLL)) (*BOTH1))
          (PRO ((INDEF)) (*BOTH2)) (AM () (*BOTH3))) )))
  (U(T(* (BIND () (*BUT1)) (AM () (*BUT2))) ))
  (Y(* (PREP () (*BY1)) (PRT () (*BY2))) ))
(D(O(W(N(*(PREP () (*DOWN1)) (PRT () (*DOWN2))) ))))
(E(A(C(H(* (QNTFR ((NS)) (*EACH1)))))
          (PRO ((INDEF NS COLL)) (*EACH2))) )))
  (I(T(H(E(R(*(QNTRF ((NS NP)) (*EITHER1))
              (PRO ((INDEF NS NP)) (*EITHER2))) ))))
    (G(H(T(*(QNTRF ((NP)) (*EIGHT1)) (NUM () (*EIGHT2)))
           (H(*(ORD () (*EIGHTH))) )))))
  (L(S(E(* (AM () (*ELSE))) )))
  (V(E(R(Y(* (QNTFR ((NS)) (*EVERY)))
           (O(N(E(* (PRO ((INDEF)) (*EVERYONE))) )))
           (T(H(I)N)G)* (PRO ((INDEF NS)) (*EVERYTHING)))))))))))
  (X(C(E(P(T(*(PREP () (*EXCEPT)) (CONJ () (*EXCEPT2))) ))))))
(F(E(W(* (QNTRF ((NONUM NP COLL)) (*FEW)))
       (E(R(*(QNTRF ((NONUM NP COLL)) (*FEW))) ))))
  (I(F(T(H(*(ORD () (*FIFTH))) )))
    (R(S(T(*(ORD () (*FIRST))) )))
    (V(E(* (QNTFR ((NP)) (*FIVE1)) (NUM () (*FIVE2))))))
  (O(R(* (PREP () (*FOR1)) (CONJ () (*FOR2))) )
    (U(R(*(QNTRF ((NP)) (*FOUR1)) (NUM () (*FOUR2)))
         (T(H(*(ORD () (*FOURTH))) )))))
  (R(O(M(*(PREP () (*FROM))) ))))
```

Figure A.1. Closed category lexical entries

```
(D(R(A(N(K(*(A ((NIL IREG PAST))
                (((O O) (*DRINK1 P1 P2))
                 ((O O) (*DRINK1A P1 P2))
                 ((O O) (*DRINK3 P1 P2))) ) ))))
   (I(N(K(*(N ((NIL NS)(S NP)(ABLE NS)(ETTE DIM)(IE DIM))
              (((O O) (*DRINK2))
               ((O O) (*DRINK4)))
              ((ING NS))
              (((O O) (*DRINK6)))
              ((ER PERS)(EER PERS)(IST PERS))
              (((O O) (*DRINK5)))
              (SYN DRAFT POTATION BEVERAGE LIQUOR)
              (ID BOOZE HOOCH MOONSHINE) )
          (A ((NIL PRES)(S PRES TPS)(ING PART))
              (((O O) (*DRINK1 P1 P2))
               ((O O) (*DRINK1A P1 P2))
               ((O O) (*DRINK3 P1 P2)))
              (SYN CONSUME SWALLOW IMBIBE GUZZLE TOAST)
              (ID SWIG SOP-UP) )
          (AM ((WISE AT3)(WAYS AT3))
              (((O O( (*DRINK1 KIND))
               ((0 0) (*DRINK1A KIND))) )
          (NM ((ABLE ADJ CLASF)(ING ADJ CLASF)(LIKE ADJ))
              (((O O) (*DRINK2 P1 P2))
               ((O O) (*DRINK4 P1 P2))
               ((O O) (*DRINK5 P1 P2))
               ((O O) (*DRINK6 P1 P2))) ) ))))
   (U(N(K(*(N ((NIL NS)(S NP)) (((O O) (*DRUNK1))) )
          (A ((NIL IREG PART))
              (((O O) (*DRINK1 P1 P2))
               ((O O) (*DRINK1A P1 P2))
               ((O O) (*DRINK3 P1 P2))) ) ))))))
(E(A(T(*(N ((S NP)(IE DIM))
           (((O O) (*EAT3)))
           ((ER PERS)(EER PERS))
           (((O O) (*EAT3)))
           (SYN FOOD)
           (ID MUNCHIES GRUB FULLERS GRUMBLIES) )
      (A ((NIL PRES)(S PRES TPS)(ING PART))
           (((O O) (*EAT1 P1 P2))
            ((O O) (*EAT2 P1 P2)))
           (SYN CONSUME DEVOUR FEED FARE ERODE WEAR)
           (ID GOBBLE) )
      (NM ((ABLE ADJ CLASF)(ING ADJ CLASF))
           (((O O) (*EAT3))) ) )
   (E(N(* (A ((NIL IREG PART))
              (((O O) (*EAT1 P1 P2))
               ((O O) (*EAT2 P1 P2))) ) )))))
```

Figure A.2. Open category lexical entries

Appendix B

Algorithms for the routines shown in Listing 2 and Listing 3 are presented as a brief description followed by its LISP code.[6]

The CLASS routine has one argument—the word to be classified. If the word does not appear as a lexical entry, the message "I do not know the word" appears followed by the word. If the word appears in the closed-category lexicon, the entry's meaning, as appearing in CLOSCAT, is returned. Otherwise the relevant portions of the entry in the open category lexicon (OCAT) are extracted and returned (using the routine FINDER), based on morphology.

```
(DEFPROP CLASS
 (LAMBDA (WORD)
  (PROG (W S A SL LEX)
   (SETQ W (STEM WORD))
  (RETURN
   (COND
    ((NOW W) (PRINT '(I DO NOT KNOW THE WORD)) (PRINT WORD))
    ((SETQ A  (CHK  (EXPLODC  (CADDR W)) CLOSCAT)) (CAAR A))
    (T (SETQ A (CAR (SETQ SL (REVERSE (CADDDR W)))))
       (SETQ LEX (CAR (CHK (EXPLODC (CADDR W)) OCAT)))
       (COND
        (SL (COND
             ((NOT (EQ A 'S))
              (MAPCAN '(LAMBDA (X) (FINDER X A)) LES))
             ((CDR SL)
              (SETQ A (FINDER (CAR LEX) (CADR SL)))
              (LIST (CAR A)(CONS 'NP (CAR (CDR A)))) )
             (T (MAPCAN '(LAMBDA (X) (FINDER X A)) LEX)) ))
        (T (MAPCAN '(LAMBDA (X) (FINDER X NIL)) LEX)))))) ))
EXPR)

(DEFPROP FINDER
 (LAMBDA (CAT SUF
  (PROG (ANS X)
   (DO I (CDR CAT) (CDDR I)
        (OR (NULL I) (EQ 'SYN (CAAR I)) )
        (COND ((SETQ X (DO J (CAR I) (CDR J)
                             (OR (NULL J) (EQ SUF (CARR J)))))
               (SETQ ANS (APPEND (APPEND (CDAR X) (CADR I)) ANS)))))
   (RETURN (COND (ANS (LIST (CAR CAT) ANS)))) ))
EXPR)
```

The FIND routine has two arguments: the first is the word and the second is the dictionary in which to search. The searching algorithm has been described in Section 5.

```
(DEFPROP FIND
 (LAMBDA (W D)
  (PROG ()
   (COND
    (D (COND
        (W (DO J D (CDR J) (NULL J)
           (COND ((EQ (CAR W) (CAAR J))
                  (RETURN (CHK (CDR W) (CDAR J)))) )))
       ((EQ (CAAR D) '*) (RETURN (LIST (CDAR D))))
       (T (RETURN NIL))) )) ))
EXPR)
```

To add items to an existing lexicon, the routines DICTADD, ADD, and MUNG are used. DICTADD has four arguments. The first specifies the lexicon to which the second argument is to be added. The third specifies the flag, i.e., the character to be inserted after the letters of the lexical entry to designate the end of the root and the beginning of the meaning field; the fourth argument specifies the meaning field. ADD is invoked from DICTADD and does the actual addition by searching for the proper position and invoking MUNG to ready the item for the addition. MUNG is invoked from ADD to consolidate the parts of the lexical entry that become the single lexicon entry.

```
(DEFPROP DICTADD
 (LAMBDA (DICT WORD FLAG PROP)
  (PROG (TDICT)
        (SETQ TDICT (CONS NIL DICT))
        (ADD TDICT WORD FLAG PROP)
        (RETURN (CDR TDICT))))
EXPR)

(DEFPROP ADD
 (LAMBDA (DICT WORD FLAG PROP)
  (COND ((NULL WORD)
         (COND ((EQ FLAG (CAADR DICT))
                (NCONC (CADR DICT) PROP))
               ((RPLACD DICT (CONS (LIST FLAG PROP)
                                   (CDR DICT))))))
        ((PROG NIL
         (DO J (CDR DICT) (CDR J) (NULL J)
          (COND ((EQ (CAR WORD) (CAAR J))
                 (RETURN (ADD (CAR J) (CDR WORD) FLAG PROP))))))
        ((NCONC DICT (LIST (MUNG WORD FLAG PROP)))))
EXPR)

(DEFPROP MUNG
 (LAMBDA (WORD FLAG PROP)
  (COND ((NULL WORD) (CONS FLAG PROP))
        ((CONS (CAR WORD)
               (LIST (MUNG (CDR WORD) FLAG PROP))))))
EXPR)
```

To combine lexicons and merge items with the same root into one lexicon, the COMBINE routine is used. COMBINE has four arguments; the first two specify the old and new dictionaries. The old dictionary is combined to the new one, so the first argument names the combined dictionary. The old dictionary is not destroyed and may still be used. The third argument is the flag (as in DICTADD). The fourth argument, specified as NIL, is used internally, since the COMBINE routine is recursive, for building up the new dictionary. COMBINE uses the ADD routine.

```
(DEFPROP COMBINE
 (LAMBDA (ODICT NDICT FLAG SOFAR)
  (MAPC
   'LAMBDA (X)
    (COND
     ((EQ (CAR X) FLAG)
      (ADD (CONS () ODICT)(REVERSE SOFAR) FLAG (CDR X)))
     ((COMBINE ODICT (CDR X) FLAG (CONS (CAR X) SOFAR)))))
  NDICT))
EXPR)
```

Although not appearing in any of the above listings, the routines DICL, DICLIST, JUSTN, and JUSTNAMES can be used to list lexicons neatly. DICLIST and JUSTNAMES have one argument, the name of a dictionary. The former lists the contents of the dictionary exactly as they appear in storage, while the latter gives just the root forms of the lexical items. DICLIST invokes DICL and JUSTNAMES invokes JUSTN in analogous manners, i.e., to print the listing.

```
(DEFPROP DICLIST (LAMBDA (DIC) (DICL DIC '* NIL)) EXPR)

(DEFPROP JUSTN (LAMBDA (DIC) (JUSTNAMES DIC '* NIL)) EXPR)

(DEFPROP DICL
 (LAMBDA (NDICT FLAG SOFAR)
         (MAPC '(LAMBDA (X)
```

```
                    (COND ((EQ (CAR X) FLAG)
                           (PRINT (LIST (REVERSE SOFAR)
                                        FLAG
                                        (CDR X)))
((                         ((DICL
                                  (CDR X)
                                  FLAG
                                  (CONS (CAR X) SOFAR)))))
              NDICT))
EXPR)

(DEFPROP JUSTNAMES
 (LAMBDA (NDICT FLAG SOFAR)
         (MAPC '(LAMBDA (X)
                 (COND ((EQ (CAR X) FLAG)
                        (PRINT (IMPLODE (CAR (LIST
                         (REVERSE SOFAR) FLAG (CDR X))))))
                       ((JUSTNAMES (CDR X) FLAG (CONS
                                                  (CAR X)
                                                  SOFAR)))))
               NDICT))
EXPR)
```

Appendix C

The following discussion explains how English is parsed in an experimental program that uses the lexical structure. A semantic structure expressing a particular utterance is formed according to simple structural rules. The central role of verbs is acknowledged and preferred semantic categories for the subjects and objects of verbs guide each choice in the creation of meaning structures. Word sense disambiguation for verbs, modifiers, and nominals follows naturally in this approach, vide Cercone (1975a). Extensive trial and error searches are eliminated since the interpretation takes on a "slot and filler" character. The approach to interpretation is almost completely semantically oriented and syntax is used only when meaning-analysis fails.

Initial Classification

Initially the text is read (either in discourse mode or from an external file for longer text) and broken into clauses (at present this process is very unsophisticated). Each clause is then "classified" in the following manner. Words are morphologically analyzed and, based on that analysis, are classified to determine all of their possible syntactic functions. For example, the form "drinks" of the root word "drink" can only be used nominally or as an action. The root form is located in the lexicon and using affix information from the morphological analysis, all of the possibilities for the word are extracted. When all words in the clause are classified, the next phase, parsing, begins.

Parsing

Traditionally, the purpose of parsing sentences has been to output syntactic trees. These trees served as input to semantic routines charged with the generation of meaning structures. Winograd (1972) and Woods (1970) tried, with some degree of success, to integrate the two processes and have each guide the other. Schank (1972) and Wilks (1973) have stressed that syntactic processing was secondary to meaning analysis and should be necessary only when the resolution of ambiguity by meaning analysis alone had failed. Their parsing phase is almost completely semantically oriented. One important by-product in the method to be described is the detection of the correct "sense" of nominals and actions and, although not yet implemented, modifiers as well (I am restricting utterances to active voice).

The parsing proceeds as follows. Words in a classified clause are scanned from left to right in search of a suitable candidate for an action. Once found, the sentence is separated into

((FIRST PART) (ACTION CANDIDATE) (SECOND PART)).

The action candidate contains, among other things, a list of possible action "senses" that this particular root form may have. These senses are ordered by a scheme, albeit a very superficial scheme, to be described later. Associated with word senses are templates as described in Cercone (1975a). For example the sense *GIVE1 of the root form "give" has a template

X GIVE Y Z

and an alternative (ALTERN) template

X GIVE Z TO Y.

The template is used to guide the parsing. In this example X, Y and Z are variables representing the arguments of the predicate "give" that we expect to find in the surface utterance, in the given order. If an argument is not present in the utterance, the implication template can be used to infer arguments. More detailed information concerning the arguments is obtained by examining the network propositions; for the sense of "give" in question, those which involve the arguments. Thus X would represent an ANIMATE nominal capable of "giving."

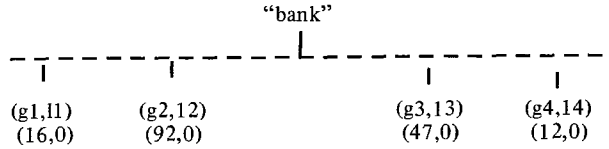
This is similar to what Schank does when parsing in conceptual dependency theory. If the words in the surface utterance do not satisfy the constraints for arguments, one of four reasons is likely. First, alternate syntactic constructions could exist. Second, a different "sense" of the action is "correct." Third, the particular action-candidate is not the valid action of the clause. Finally, some other reason, like slang expressions or a metaphor might be the cause.

Whenever arguments fail to satisfy a predicate, a search for alternative implication templates begins. If this fails then the list of senses for the root form is further examined. If other senses of the action candidate exist, they are examined further to see if arguments in the surface utterance match variables in the template. This procedure is repeated until the correct sense of the action candidate is found or the list of senses is exhausted. If the sense list is exhausted, scanning continues in the surface clause for another suitable action candidate and the process is repeated.

The matching of predicates' arguments in surface text to variables in implication templates includes finding the correct sense of nominals and modifiers as well. The sentence "A drinker drinks many drinks" has as the second argument of the predicate "drinks" the word "drinks." Possible nominal senses for that "drinks" include an alcoholic beverage, a body of water (throw John into the drink), or a thirst quencher. Thus, if the first sense of a nominal fails as argument, all other senses must be examined before deciding not to accept it as argument. This reasoning applies with respect to modifiers in a similar but not identical fashion. For instance, a "yellow cake" is a type of cake much like a chocolate cake, whereas a "yellow car" is something that is yellow and something that is a car. Using these methods, sentences such as "A 'drinker' 'drinks' many 'drinks' " and "The pilot 'banked' his plane near the river 'bank' over the 'bank' that he 'banks' on for good 'banking' service" present little difficulty.

Morphological analysis is important since only those forms that can authentically be considered as actions need be examined. In the example, "A drinker drinks many drinks" morphological analysis eliminates "drinker" immediately as an action candidate. Thus, we are quickly able to get a right choice.

Both Schank and Wilks used their intuition to set up respective meaning representations. The way that they defined and used semantic "primitives" are one example. One way in which my intuition has shaped the experimental program can be shown with the following superficial scheme for choosing word senses.



Associated with each sense of a word are g1's and l1's which denote frequency counts for "global" and "local" usage of the ith meaning sense of the word. Whenever a term is encountered, the local frequency counts are first examined to see if any context has been established in the dialogue thus far. They are all zero in the example, so no context has been established; then the global frequency counts are examined. Accordingly the second sense is selected as the most likely candidate. If it fails, then the third, first, and fourth senses would be selected in that order. Suppose that the third sense turns out to be correct. The local frequency count is set to one, and, whenever the term "bank" is encountered, the third sense will be selected first and its local

frequency count will be incremented by one (if it is the correct sense). This would continue until the third sense fails to be correct. At this point we would examine the second, first, and fourth senses until we arrive at the correct meaning sense (i.e., the ith term). The l3 is added to g3, li is set to one (non zero), l3 is reset to zero, and the ith meaning sense is selected whenever the term "bank" is encountered.

The list of modifiers found in the clause, further classified as to function, and associated with correct predicate arguments they modify, is also given as part of the parsing phase.

Once the parsing phase has been completed, the meaning representation is built for the clause, and that structure is integrated into the semantic network, vide Schubert (1974). The first step involves building an intermediate structure based only on the predicate of the clause and its arguments. After this structure is created, it may be altered to accommodate other information detected in the parsing phase. This information includes mainly modifiers (only some adjectival modifiers are now analyzed, however adverbial and quantificational are planned).

NOTES

1. A word with either a different syntactic class from the original, a different meaning, or both.
2. Suffixes may "pile-up" to about three or four in number (e.g., normalizers) whereas prefixes are normally single. When suffixes do "pile-up," their order is fixed and we can take advantage of this fact.
3. They include auxiliaries of periphrasis, which assist in expressing the interrogative, negative, and emphatic forms of speech, viz. "do" ("did"); auxiliaries of tense, "have," "be," "shall," "will"; of mood, "may," "should," "would"; of voice, "be"; of predication (i.e., verbs of incomplete predication which require a verbal complement), "can," "must," "ought," "need," also "shall," "will," "may," when not auxiliaries of tense or mood. (OED s.v. Auxiliary, B. Sb., 3.)
4. The type features have all been placed under the category AM because of the nature of action-modifying adverbs; however, as type one adverbs show, this is not always the case.
5. Much of Wilks' representation of meaning in formulas is based on lexical decomposition trees developed by Lakoff (1972). Those representations have foundations in the "generative semantics" school of thought. The argument concerning the "correct" theory of grammar between advocates of transformational syntax on the one hand and generative semanticists on the other continues. An excellent critique of both avenues is presented in Bartsch and Vennemann (1972), pages 6-28. Much of the material they review has been reprinted in Davidson and Harmon (1972), see especially Parsons, Montague, and Lakoff.
6. Excluded from this Appendix is the STEM routine which has been described in Cercone (1974).

REFERENCES

Bartsch, R., and T. Vennemann (1972). SEMANTIC STRUCTURES, Athenaum Verlag, Frankfurt, Germany.

Cercone, N. (1975a). "Representing Natural Language in Extended Semantic Networks," Department of Computing Science, TR75-11, University of Alberta, Edmonton, Alberta.

Cercone, N. (1975b). "The Nature and Computational Use of a Meaning Representation for Word Concepts," American Journal of Computational Linguistics, Volume 2, AJCL Microfiche 34, 64-81.

Cercone, N. (1974). "Computer Analysis of English Word Formation," Technical Report TR74-6, Department of Computing Science, University of Alberta, Edmonton, Alberta.

Cercone, N., and L. K. Schubert (1974). "Toward a State-Based Conceptual Representation," Department of Computing Science, TR74-19, University of Alberta, Edmonton, Alberta. (Also IJCAI-IV, pp. 83-90.)

Davidson, D., and G. Harman, eds. (1972). SEMANTICS OF NATURAL LANGUAGE, D. Reidel Publishing Company, Boston, Massachusetts.

Heny, F. W. (1973). "Sentence and Predicate Modifiers in English," in SYNTAX AND SEMANTICS, vol. 2, J. Kimball ed., Seminar Press, New York, New York, 217-245.

Lakoff, G. (1973). "Hedges: A Study in Meaning Criteria and the Logic of Fuzzy Concepts," Journal of Philosophical Logic, v 2, pp. 458-508.

Lakoff, G. (1972). "Linguistics and Natural Logic," in SEMANTICS OF NATURAL LANGUAGE, D. Davidson and G. Harman eds. D. Reidel Publishing Company, Boston, Massachusetts.

Montague, R. (1972). "Pragmatics and Intensional Logic," in SEMANTICS OF NATURAL LANGUAGE, D. Davidson and G. Harman eds. D. Reidel Publishing Company, Boston, Massachusetts.

Moon, D. (1974). "MACLISP Reference Manual," Project MAC–M.I.T., Cambridge, Massachusetts.

Parsons, T. (1972). "Some Problems Concerning the Logic of Grammatical Modifiers," in SEMANTICS OF NATURAL LANGUAGE, D. Davidson and G. Harman eds. D. Reidel Publishing Company, Boston, Massachusetts, 127-141.

Quillian, M. (1968). "Semantic Memory," in SEMANTIC INFORMATION PROCESSING, M. Minsky ed. MIT Press, Cambridge Massachusetts, 227-270.

Quine, W. (1960). WORD AND OBJECT, MIT Press, Cambridge, Massachusetts.

Russell, B. (1923). "Vagueness," Australasian Journal of Philosophy, 1, 84-92.

Schank, R. (1974). "Adverbs of Belief," Lingua, v 33, North Holland Publishing Company, 45-67.

Schank, R. (1972). "Conceptual Dependency: A Theory of Natural Language Understanding," Cognitive

Psychology, v 3, 552-631.
Schubert, L. (1974). "On the Expressive Adequacy of Semantic Networks," TR74-18, Department of Computing Science, University of Alberta, Edmonton, Alberta. (Also AI, 7, 2, 163-198.)
Wilks, Y. (1973). "Preference Semantics," Stanford AI Project, Memo AIM-206, Stanford University, Stanford, California.
Winograd, T. (1972). UNDERSTANDING NATURAL LANGUAGE, Academic Press, New York, New York.
Woods, W., R. Kaplan, and B. Nash-Webber (1972). "The Lunar Sciences Natural Language Information System: Final Report," Bolt Beranek and Newman Inc., Cambridge, Massachusetts.
Zadeh, L. (1972). "A Fuzzy-Set-Theoretic Interpretation of Linguistic Hedges," Journal of Cybernetics, v 2, n 3, 4-34.

The following IBM Reports are available on request to IBM Corporation, Armonk, N.Y.

"Data Entry of Chinese and Kanji Characters" no. 5249, edited by E. F. Yhap. A keystroke system with 37 keys, upper and lower shift, is proposed for data entry of Chinese and Kanji characters into computer systems. This data entry method has been applied to the 881 Kanji characters which are prescribed as a minimum requirement for the six elementary grades in Japanese schools by the Japanese Ministry of Education. The resulting average number of keystrokes for this set of 881 characters is just a little under 4.2 keystrokes per character. Reasonably high rates of character input are therefore expected to be achievable (60 cpm or better). Other advantages claimed (but not yet tested) for this data entry method are ease of operator training, and lack of operator mental fatigue.

"An Organization for a Dictionary of Senses" no. 5548, edited by Dick H. Fredericksen. This paper describes a lexical organization in which "senses" are represented in their own right, along with "words" and "phrases," by distinct data items. The objective of the scheme is to facilitate recognition and employment of synonyms and stock phrases by programs which process natural language. Besides presenting the proposed organization, the paper characterizes the lexical "senses" which result.

"On Natural Language Based Query Systems" no. 5577, edited by Stanley R. Petrick. Some of the arguments which have been given both for and against the use of natural languages in question-answering (QA) systems are discussed. Several QA systems are evaluated in assessing the current level of QA system development. Finally, certain pervasive difficulties which have arisen in developing natural language based QA systems are identified, and the approach which has been taken to overcome them in the REQUEST System is described.

"The Request System" no. 5604, edited by Warren J. Plath. REQUEST is an experimental Restricted English QUESTion-answering system which is currently capable of analyzing and answering a variety of English questions, spanning a significant range of syntactic complexity, with respect to a small *Fortune-500*-type data base. The long-range objective of this work is to explore the possibility of providing non-programmers with a convenient and powerful means of accessing information in formatted data bases without having to learn a formal query language. In order to address the somewhat conflicting requirements of understandability for the machine and maximum naturalness for the user, the REQUEST System employs a language processing approach featuring: (1) the use of restricted English; (2) a two-phase, compiler-like organization; and (3) linguistic analysis based on a transformational grammar. The present paper explores the motivation for this approach in some detail and also describes the organization, operation, and current status of the system.