Two

Semantic Networks: Their Computation and Use for Understanding English Sentences

R. F. Simmons
The University of Texas

In the world of computer science, networks are mathematical and computational structures composed of sets of nodes connected by directed arcs. A semantic network purports to represent concepts expressed by natural-language words and phrases as nodes connected to other such concepts by a particular set of arcs called semantic relations. Primitive concepts in this system of semantic networks are word-sense meanings. Primitive semantic relations are those that the verb of a sentence has with its subject, object, and prepositional phrase arguments in addition to those that underlie common lexical, classificational and modificational relations. A complete statement of semantic relations would include all those relations that would be required in the total classification of a natural language vocabulary.

We consider the theory and model of semantic nets to be a computational theory of superficial verbal understanding in humans. We conceive semantic nodes as representing human verbal concept structures and semantic relations connecting two such structures as representing the linguistic processes of thought that are used to combine them into natural-language descriptions of events. Some psycholinguistic evidence supports this theory (Quillian 1968, Rumelhart and Norman, 1971, Collins and Quillian, 1971); but a long period of research will be necessary before enough facts are available to accept or reject it as valid and useful psychological theory.

We are on much stronger ground when we treat semantic networks as a computational linguistic theory of structures and processing operations required for computer understanding of natural language. The nodes model lexical concepts and the semantic relations represent a combination of processes that are useful or necessary for analyzing English strings, for paraphrastic transformations, for question-answering operations, and for generating meaningful English sentences. Semantic nets are simple—even elegant—structures for representing aspects of meaning of English strings in a convenient computational form that supports useful language-processing operations on computers.

As linguistic theory, semantic nets offer a convenient formalism for representing such ideas as "deep structure", "underlying semantic structure", etc. The content of the structure represented in semantic nets depends on the conventions of the linguistic theory that is adopted. Our semantic networks will be seen to reflect a linguistic theory of deep case structures originated by Fillmore (1968) and further developed by Celce-Murcia (1971). The processes undertaken on the nets to generate language strings provide a theory of how language can be generated from underlying semantic structures. Computational processes for analyzing language into semantic nets provide a precise description of a theory of how some aspects of sentence meaning can be understood as a well-defined semantic system. The term "understanding" is given precise operational meaning through the programs that recognize or generate paraphrases and answer questions. The extent of the understanding is measurable by the ease or difficulty of the question-answering tasks, the size of vocabulary, and the efficacy of the system in handling complexities and subtleties of English structure.

When backed up by working programs, computational theories introduce a measure of logical rigor into the soft-sciences of linguistics and psychology. A minimally satisfactory computational theory of language requires that some set of natural language strings be generated and understood in terms of formal elements of that theory such as lexical structures, grammars, and semantic representations. A working set of computer programs that carry out recognition, paraphrase, question-answering, and language generation tasks proves the consistency and demonstrates the degree of completeness of the theory.

Despite logical rigor, computational theories may be weak or powerful in terms of the amount of language phenomena they account for; they may be elegant or cumbersome; they may be alien or closely related to human thought processes as we think we understand them; they may be in or out of fashion with respect to psychology, linguistics, or computer science. Ideally, they complement purely linguistic or psychological theories by formulating and testing precise descriptions of the structures and processes described more loosely in the theory. In natural language systems, computational theories have been taken beyond the bounds ordinarily set by linguists, psychologists, or logicians to develop an interdisciplinary theory of verbal communication based on conceptual structures underlying language; lexical, syntactic, semantic operations for recognizing and generating English and useful psychological

c networks as a computa-
ɔns required for computer
ical concepts and the se-
ɪt are useful or necessary
ons, for question-answer-
:ences. Semantic nets are
s of meaning of English
·eful language-processing

rmalism for representing
:cture", etc. The content
the conventions of the
·vill be seen to reflect a
:nore (1968) and further
·n on the nets to generate
nerated from underlying
language into semantic
·pects of sentence mean-
e ter ɪnderstanding"
·at recognize or generate
ɪtanding is measurable
size of vocabulary, and
ties of English structure.
eories introduce a mea-
ɪychology. A minimally
·ome set of natural lan-
elements of that theory
ɪtations. A working set
e, question-answering,
ɜmonstrates the degree

k or powerful in terms
ɪey may be elegant or
ɪ thought processes as
shion with respect to
· complement purely
g precise descriptions
the theory. In natural
. beyond the bounds
·p an interdisciplinary
tures underlying lan-
nd generating English

strings; and logical and mathematical operations for determining the equivalence of two or more semantic structures.

The theory and model of semantic nets presented in this chapter is still incomplete: limited in its present development to single sentences, truncated at a certain conceptual depth, unspecified with regard to many of the complex phenomena of English, and unexplored with respect to other languages. In its favor, it encompasses such major subtasks of the verbal communication process as the generation and recognition of English strings and their understanding in terms of limited capability to answer questions and to generate and recognize paraphrases. As modelled in a working set of *LISP* 1.5 programs it is precisely stated, internally consistent, and potentially useful to guide further research and for various applications to information retrieval, computer aided instruction, and other language processing operations.

An Abstract Model of Communication

The main human use of language is for one person to communicate feelings and ideas to other people. The simplest model of this communication process is diagrammed in Figure 2.1.

Thus simply shown, the diagram is largely vacuous with respect to meaning. If we develop a model of what is meant by "ideas and feelings", another for "language," and a set of functions to map language onto ideas and ideas onto language, we then have at least a mathematical theory of the communicative use of language. Semantic network structures form the model of ideas. A syntactic and semantic description (i.e., a grammar and a lexicon) of allowable ordering rules of words and phrases to make acceptable English sentences is an important aspect of the model of

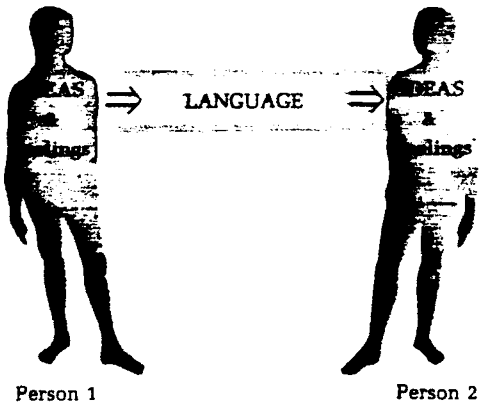


Figure 2.1 Diagram of Communication Process. (Note: The symbol "⇒" is read as "maps onto.")

language. Equally important are the rules for mapping words, phrases, and sentences into the semantic net structure of the model of ideas and for mapping the ideas into language strings.

If the model of ideas is also to be used to represent the processes of rational thought, then it must be able to represent one idea as a consequence of another or of a set of other ideas. For example, "tiger" implies "mammal." This is one essential feature of problem-solving, theorem-proving, and question-answering behaviors. It also is the basis for recognizing that two sentences are paraphrases that (from some point of view) mean essentially the same thing. This feature of the model is carried in implicational rules and functions that map one semantic structure into another.

The ideas may be mapped into language forms other than English or other natural languages. We can define a language to describe a structured sequence of actions and a mapping function from ideas into that language. The behavior of a robot hand in selecting and stacking blocks, for example, has been described in this volume by Winograd as a language composed of permitted sequences of simple operators as Grasp(x), Move(x,y) Put(x,y), etc. Semantic representations of such imperative sentences as "Put the large green pyramid on top of the blue block" are mapped into strings of this operator language which are then interpreted by a computer (in complex ways) to result in the appropriate actions by a (simulated) mechanical hand.

The content of visual representations can also be seen as a language string o edging, cornering, and shading elements. This string is mapped onto a semantic structure of images that has been represented in semantic net form by Clowes (197 and Preparata (1970). Presumably there is a language to describe internal organ responses, such as feelings, and mapping functions that show correspondences between semantic net representations of ideas and feelings.

The mappings into ideas of events presented visually, as verbal strings, of a structure of organic reactions, or of a series of actions can all be represented linguistically in terms of a grammar and a lexicon that transform a language string into a semantic representation that is taken as a model of underlying ideas. The semantic representation of these events can be mapped out into any appropriate language using the corresponding grammar and lexicon of that language.

Ideally we hypothesize one central cognitive structure of semantic net form into which perceptions of speech, vision, action, and feeling can map, and from which can be generated speech, physical actions, hallucinations, feelings, and other thoughts. So far, however, we have only studied semantic nets to represent a class of English sentences.

At a very abstract level this model of communication can be simply represented as three mappings:

M1 (language, ideas)
M2 (ideas, ideas)
M3 (ideas, language)

This abstract statement provides only an illusion of simplicity, since the processes M1, M2, and M3 are incredibly complicated. Learning them is a major occupation of humans for most of their lives. Analyzing and programming them involves much of the content of linguistics, psychology, logic, computational linguistics, and other sciences depending on the nature of the ideas that are studied.

The mappings M1 and M3 are in a complex inverse relation. For a given pair of language string and idea, L I, if M1(L) $\Rightarrow$ I, then M3(I) $\Rightarrow$ L′ such that M1(L′) $\Rightarrow$ I. In other words, a given semantic structure, I, that is derived from a language string, L, will generate another language string, L′, which is either identical to L or a paraphrase of L and whose semantic structure is analyzed back into I. In this theory, L and L′ are not restricted to strings from the same language or the same modality (i.e., speech, vision, feeling, etc.).

The mapping, M2, of ideas onto other ideas clearly encompasses many ideational processes. Perhaps the lowest level is simple association where one structure can be substituted for another if they have an element in common. Thus the ideas, "I saw a tree" and "trees grow" are related by the identical concept, "tree". Mappings can be in terms of paths from one idea to another; e.g., "a tree is a plant" that could be described as Superset(tree) $\Rightarrow$ plant. Vastly more complex mappings are commonly used for detecting paraphrase or answering questions such as:

Quest: Did Napoleon lose the battle of Waterloo?
Ans: Wellington defeated Napoleon at Waterloo.

The detailed statement of this mapping is a complex relation between the concepts "lose" and "defeat" which is stated later in this chapter.

This abstract model of communication proposes that there is a single cognitive representation of ideas, whether they originated as visual, auditory, or tactile perceptions or whether they were derived from verbal descriptions in English, French, or Hindustani. At the present level of development of semantic network representations of meaning, emphasis has been concentrated on English sentences. The structures presented in this chapter are shown to be largely sufficient to account for understanding at the level of answering factual questions and forming verbal paraphrases. Schank presents a deeper level of ideational representation in Chapter 5 and Winograd shows a level of ideational representation (not in semantic network form) that is deep enough to mediate between language and action in the robot's world of blocks.

Linguistic Structure of Semantic Nets

A sentence is a string of ambiguous word symbols that implies a complex structure of underlying concepts. A semantic analysis of a sentence transforms this string into a structure of unambiguous concept nodes interconnected by explicit semantic relations. The concept nodes in this system of semantic nets are taken as lexical word-sense meanings and the semantic relations are variously deep case relations that connect nominal concepts to verbs, and conjunctive, modifier, quantifier, and classifier relations that connect and specify concepts.

Deep Case Structure of Verbs: Our semantic representations of sentence meanings are based partially on a linguistic theory of deep case structures as developed by Celce-Murcia (1972) deriving from earlier work by Filmore (1968) and Thompson (1971). In its essence, this theory provides for each sentence an underlying structure of a verb, its modality, and its nominal agruments. A phrase structure grammar can be used to describe this underlying structure as follows:

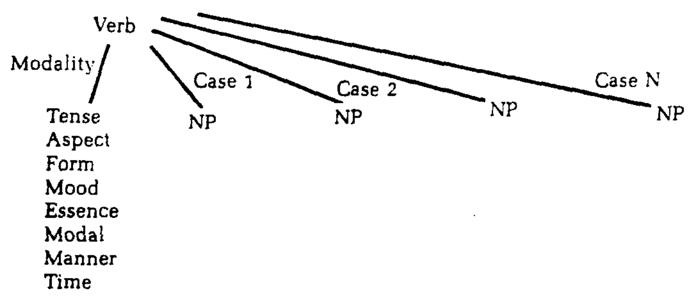
S ⟶ Modality + Proposition
Modality ⟶ Tense, Aspect, Form, Mood, Modal, Manner, Time
Proposition ⟶ Vb + (CASEARGUMENT) *
Vb ⟶ run, walk, break, etc.
CASEARGUMENT ⟶ CASERELATION + [NP|S]
NP ⟶ (prep) + (DET) + (ADJ) * + (N) + N + (S|NP)
CASERELATION ⟶ CASUALACTANT, THEME, LOCUS, SOURCE, GOAL.

Box 2.1a shows a tree diagram of case structure and 2.1b expands the definitions of the elements comprising the modality.

The modality carries detailed information concerning tense, form, truth value, manner, time, and syntactic form of the sentence. It can be seen in a later section to

---

Box 2.1 Syntactic Form of a Case Structure for a Sentence, with (a) being the General Case Structure for a Sentence and (b) the Possible Values of the Modality.



Modality

Tense: Present, Past, Future   Voice
Aspect: Perfect, Imperfect
Form: Simple, Emphatic, Progressive
Mood: Declarative, Interrogative, Imperative
Essence: Positive, Negative, Indeterminate
Modal: may, can, must
Manner: Adverbial
Time: Adverbial

serve as a blueprint for generating a particular syntactic form of sentence from a semantic proposition. The Proposition is a verb that dominates a set of noun-phrase or sentence arguments, each of which is in a definite, named, case relation to the verb.

Celce-Murcia argues convincingly that all arguments of the verb can be classified as members of five deep case relations; namely, Causal Actant, Theme, Locus, Source and Goal. In a related case structure system Chafe (1970) prefers several different case names including Agent, Patient, Benefactor, etc., as proposed by Filmore and other case structure theorists. We have chosen to follow the naming conventions suggested by Celce-Murcia. A simple sentence such as "Mary wore a sweater" gives the following propositional structure:

Wear: LOCUS Mary, THEME a sweater.

A more complicated example, "John broke the window with a hammer", has the following propositional structure:

Break: CA1 John, Theme the window, CA2 a hammer.

This example shows that two Causal Actants (CA1, CA2) may be present in a single sentence. The sentence, "An ape is an animal" can be interpreted as having two themes, as follows:

Be: T1 an ape, T2 an animal.

Two loci can be seen in "Mary wore a sweater in the park".

A fair degree of familiarity with this and other systems of case-structure naming conventions is required before people come to agreement in assigning names to propositional arguments. At this early period in the development of the theory, it is quite possible that other naming conventions will be generally adopted and that more objective criteria will be developed for identifying the role of arguments in a proposition.

Verbs are assigned to classes called paradigms in accordance with the way their deep case relations are allowed to be ordered in surface strings. For example, "break" belongs to the ergative paradigm that allows the following patterns of surface strings for the active voice:

John broke the window with the hammer.
John broke the window.
The hammer broke the window.
The window broke.

Each of these variations is generated with argument ordering and deletion operations from the following propositional structure:

Break: CA1 John, T the window, CA2 a hammer.

The process of generating such sentences requires that the modality be specified and an appropriate surface ordering rule be selected. The modality for the above set is as follows:

MODALITY: TENSE Past, VOICE Active, FORM Simple,
ESSENCE Positive, MOOD Declarative.

Unspecified values for Aspect, Modal, Manner, and Time indicate null representations in the surface string. The selection of a paradigmatic ordering rule depends on very complex considerations such as choice of subject emphasis, deletions of arguments because of context, embedding environment, etc. Paradigmatic ordering rules for the ergative class verb are as follows:

```
(CA1,VACT,THEME,CA2)
(CA2,VACT,THEME)
(THEME, VACT)
(THEME, VPAS, CA1,CA2)
(THEME, VPAS, CA2,CA1)
(THEME, VPAS, CA1)
(THEME, VPAS, CA2)
(THEME, VPAS)
```

If the Modality is marked for Emphatic, Progressive, Imperative, or Interrogative, the choice and ordering of elements of the verb string and of nominal arguments will differ within the application of rules such as the above. Details of this generation process are presented in a later discussion.

As in Chomsky's transformational theory, this proposes a deep structure underlying each embedded sentence, but the deep case structure can meet our requirement that the semantic analysis of a sentence result in a structure of unambiguous concepts connected by explicit semantic relations. Unambiguous concepts are provided by the selection with various contextual restrictions of particular word-sense meanings that map onto the lexical choices. The small set of case designators name specific semantic relations between the verb and its arguments. The transformational deep structure, in contrast, provides only syntactic relations to connect the elements of a structure.

The Celce-Murcia theory also suggests that what we have seen as varied sense meanings of a verb can now be accounted for as varied implications of a given event-class that the verb designates, under the differing circumstances signified by different choices of semantic classes of arguments. This notion is rather difficult to understand at first reading. For example, the verb, "run" is taken to designate one event class—that of rapid motion—in all of the following environments:

John ran to school.
John ran a machine.

The machine ran.
The brook ran.

This verb belongs to a reflexive-deletion paradigm where the theme is deleted if it corresponds to the CA1. Thus the propositional structure of the first example is as follows:

Run: CA1 John, T John, Goal to school.

During the event, the Theme incurs rapid motion with the instruments of motion associated with that Theme, namely legs and feet. Similarly, in the running of a machine or of a brook, the Themes, "machine" and "brook" incur the rapid motion with their customary instruments; respectively, motors and gravity. The first two examples specify "John" as the animate Causal Actant, while in the latter two the causal actants are unspecified. The result is that the semantic definition of "run" is informally approximated by the following:

Run:	THEME	(incurs rapid motion)
	CA1	(animate instigator)
	CA2	(instrumental cause of motion)
	GOAL	(condition of cessation of motion)

The development of this line of thought for numerous verbs offers an attractive area of research in the implicational meanings in language. The present level of understanding of semantic net structures achieves syntactic simplicity and computational advantages from expecting a single meaning for a verb (excepting homographs), but is not yet deep enough to use this form of definition in question answering and other applications.

The theory is also consistent with recent linguistic suggestions (Jacobs and Rosenbaum, 1968) that adjectives be treated similarly to verbs. In deep case structure, an adjective can be represented as a verb with the Modality marked Adjective, and a one argument proposition. Thus, "a red dress" might receive the following structure:

Red: MODALITY...Adjective, THEME a dress.

Similarly, a prepositional phrase such as "the book on the table" might be expressed:

Be: MODALITY...NP, THEME the book, LOCUS on the table.

Nominalized verbs such as "defeat" in "Napoleon's defeat at Waterloo" might be represented as:

Defeat: MODALITY...NP, THEME Napoleon, LOCUS at Waterloo.

The nesting of embedded sentences in this theory has been explored by Celce-Murcia (1972) who shows that a structure such as shown in Box 2.2 can be used.

These are all attractive but still incompletely developed aspects of the theory of deep case structures that have influenced our conventions for semantic network representations of sentence meanings. Thus far we have adopted the case structure representation for verbs and their arguments, and the use of paradigm classes and embedding conventions for verbs. We do not yet treat adjectives and noun phrases in this manner, although it will probably be advisable to do so as we begin to deal with the task of answering difficult questions.

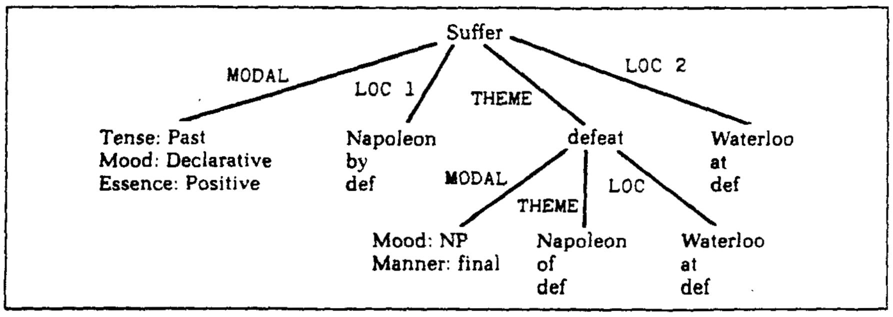
Semantic Representations for Case Arguments: This subsection develops the conventions used in semantic networks for representing the meanings of words in general, nouns, NPs, adjectives, adverbs, prepositional phrases, conjunctions, etc.

Words: The word is a convenient unit in dealing with printed text in that it is easily distinguishable and is used as a basis for ordering the lexicon. If we think of the lexicon as a set of word-sense meanings each composed of syntactic and semantic data, then each meaning can be taken as a concept or an idea. Each meaning in the lexicon maps onto one or more character strings or words, and each word represents one or more meanings in the lexicon. Each meaning in the lexicon is designated by a number prefixed by L, for example, L57, L1072, etc.

The contextual meaning of a word in a semantic network is represented by a term such as Ci, Cj, etc., where i and j are unique numbers. This term is connected by the TOKen relation to a particular word-sense meaning. The primary function of a word in context is to refer to a particular sense meaning in order to make that meaning available for use in understanding the events described by a sentence. An example of this basic semantic structure is:

C1 TOK apple

Box 2.2 Proposed Deep Case Analysis of "Napoleon suffered final defeat at Waterloo."



In this expression "apple" is printed for human communication to represent some node such as L23 which is the designator of a lexical structure containing the syntactic category, noun, a set of features such as NBR-singular, SHAPE-spherical, COLOR-red, PRINTIMAGE-apple, THEME*-eat, etc. These features are consulted for syntactic and semantic operations by parsers, generators, and question-answering systems.

Inflectional Suffixes and Auxiliaries: Singular and Plural forms and tense and agreement markings are usually carried as suffixual elements of the word. They may be discovered either by direct lookup of the full word form in the lexicon, or by a suffix stripping logic such as that described by Winograd in Chapter 4. Every noun is characterized in a semantic net with the relation NBR whose values are Singular, Plural or both. Thus for "apples", the net shows,

C1 TOK apple, NBR Pl

A DETerminer relation is also required on noun structures and it is discussed in a later paragraph.

Suffixes and auxiliaries provide much of the information required in the Modality structure. An example sentence with a simple proposition and a very complex modality will illustrate the way the modality information is signified.

Could John have been courting Mary falsely last year?

The semantic structure for this sentence is shown next.

C1 TOK Court, CA1 (John), THEME (Mary), MODALITY C2.

C2 TENSE Past, VOICE Active, FORM Progressive, ASPECT Perfect, MOOD Interrogative, MODAL Can, MANNER (Falsely), TIME (last year) ESSENCE Indeterminate

Each C-node is a set of semantic relations whose values may be; constants such as Past, Active, etc., lexical items, or other C-nodes. The parenthesized elements of the above example structure are shorthand notations that show that another structure not germane to the discussion is actually required to represent the semantic structure of the parenthesized value.

The details of obtaining this structure for the Modality will become apparent in Section V where the programs for computing it from surface strings are presented. For the moment a few explanatory remarks will suffice. "Could" signifies TENSE-Past, MODAL-Can, and by its introductory position, MOOD-Interrogative. The phrase, "have been courting" signifies FORM-Progressive, ASPECT-Perfect, and VOICE-Active. ESSENCE refers to Truth or Falsity of the statement—which as a question is indeterminate. "Falsely" and "last year" are respectively MANNER and TIME adverbials, which in the present conventions of this system are carried on the Modality. The information required to form these particular relations and values is obtained during parsing with grammar rules and the contents of the lexicon. Detailed computational analysis of the way in which verb strings signify the specific values of the modality have been described by Simmons and Slocum (1972) and Winograd (1972).

Determination and Quantification: In English every noun is associated with a determiner, either explicitly with words such as "this," "these," "some," "a," "an," "seven," "all," "the," etc., or implicitly where the absence of a determiner is interpreted as "most" or "all" (as in "bears eat fish"). In our semantic treatment we distinguish four semantic relations, DET, COUNT, NEG and QUANTifier or Q. The values of DET are definite, indefinite, or general. COUNT has as values a number or the meanings signified by "many," "few," "most," etc. NEG takes only the value "none." QUANT has values such as "some," "all," "every," etc. COUNT, QUANT, and NEG are not marked unless they are explicitly signified in the sentence. No claim is made that this is either a complete or completely satisfactory scheme for analyzing the truly vast complexity of determination of English nouns; it is instead a starting point which must be modified as further research reveals more details of this semantic structure.

One very important aspect of determination can hardly be discussed within the framework of a single sentence. When a noun has a definite determiner, it refers to a concept that has been mentioned previously, to something in the nearby environment or to a well-known class of events. Our relation DET with the value definite signifies this (respective) anaphoric, deictic or generic usage; just which usage is implied and to what it refers requires that DET be operated as a function to examine the textual environment. The manner in which this can be accomplished is suggested by Baranofski (1970).

The following examples illustrate our conventions for representing determination in semantic networks:

All seven windows
C1 TOK window, NBR Plural, DET Def., COUNT 7, Q All.
Some windows
C1 TOK window, NBR Plural, DET Indef, Q Some.
No window
C1 TOK window, NBR Sing, DET Generic, NEG none.

Combinations of these semantic relations signify various logical and numerical quantifiers. In our present uses of semantic nets for generating paraphrases and answering fact questions at the paraphrastic level, we have found it necessary to deal only superficially with logical quantifiers. These become of critical importance in more difficult questions and in verbal problem solving.

Adjectival Modification: Whether it occurs in a predicate or noun modifier position, we represent adjectival modification in the same semantic form. The two strings:

the barn is red
the red barn

each receive the following semantic representation:

C1 TOK barn, NBR Sing, DET def, MOD C2.
C2 TOK red, DEG Pos.

The semantic relation DEGree takes as values Positive, Comparative, or Superlative. If the value is positive, there is one noun argument for the adjective; comparative requires two, and superlative more than two.

The relation MOD is in fact a temporary expedient that serves only to indicate an adjectival modification. The linguistic structure of adjectives is almost as complicated as that of verbs. The meaning of a given adjective is relative depending on context. For example, the sentence "a large ant is smaller than a tiny elephant" shows that associated with the meanings of "ant" and "elephant", there must be a characteristic size value which is further specifiable by a size adjective. "A large ant" thus means something like "a large tiny-ant" while "a tiny elephant" indicates "a tiny large-elephant." Semantic relations underlying MOD include SIZE, SHAPE, COLOR, etc., which Schank (1969) suggests are characteristic attributes of nouns. The function of the adjective is apparently to modify the characteristic attribute of the noun as for example, "yellow brick" changes the characteristic reddish color associated with "brick" to the value "yellow".

The comparative and superlative uses of adjectives introduce a whole range of complex sentence structures which have been treated carefully in a dissertation by Celce-Murcia (1972). Our treatment of adjectives in semantic nets is only sufficient at the moment for dealing with the simplest cases. Future developments will probably require the adoption of a structure similar to that now used for verbs.

Adverbial Modification: A previous example showed that adverbs are values of such Modality relations as Manner and Time. The fact that they can also modify adjectives offers further motivation for treating adjectives as having a structure similar to verbs. Since linguists presently understand so little about the semantic behavior of adverbs, we again adopt the expedient of a gross relation, VMOD, in our computational models. This relation can be further specified as, MANNER, TIME, FREQUENCY, INTENSITY, etc., depending on the semantic class of the adverb.

Conjunction: In addition to the frequent occurrences of "or" and "and," many common adverbial conjunctions or sentence connectors are used in English. These include words such as "since," "because," "thus," "before," "after," etc. Our representation of these important terms in semantic structures is to form a TOKen structure followed by a list of arguments, as illustrated below.

C1 TOK (any conjunction), ARGS C2, C3, C4...

The conjoined elements may be words, phrases or sentences. The meaning of conjunctions enters deeply into paraphrase and question-answering tasks, and they are used frequently to order sentences in time, causation, etc. Much detailed knowledge of the meaning of particular conjunctions is recorded in style books and dictionaries, but little formalization of this knowledge has so far been developed. Once again we are in the position of preserving a lexical indicator in the semantic net structure with little corresponding understanding of the lexical structure to which it refers.

The verbs Have and Be: Since these two verbs have noun phrase transformations, we choose to represent them in semantic networks as nominal structures. A few examples for "is" illustrate the applicable conventions:

The girl is beautiful.
C1 TOK girl, DET Def, NBR S, MOD (beautiful).

The girl is a mother.
C1 TOK girl, DET Def, NBR S, SUP (mother).

The girl is in the chair.
C1 TOK girl, DET Def, NBR S, LOC C2
C2 TOK chair, DET Def, NBR S, PREP in

The first example is treated as an adjectival MODification even though it occurs in predicate form. The second shows that the concept associated with "girl" is a subclass of that associated with "mother". The third shows the same structure as would have been derived from the noun phrase, "the girl in the chair".

Examples for "have" are as follows:

Mary has long fingers.
Mary has money.
Mary has fun.

The three semantic relations expressed here are respectively, HASPART, POSSess, and ASSOCiated. They are also signified by the apostrophe in such forms as "Mary's fingers", "Mary's money" and "Mary's fun". These alternate forms are assigned the same semantic structure as those with the verb expressed. The next example shows the treatment of "have" with a prepositional phrase.

Mary has fun in the park.
C1 TOK Mary, DET Def, NBR S, ASSOC (fun), LOC C2.
C2 TOK park, DET Def, NBR S, PREP in.

Eventually the theory of deep case structures may require that various forms of nominal modification should always be dominated by a verb and its modality. If we were to adopt this convention for "is" and "have" and their nominal forms, the following examples would result:

Mary is in the park.
C1 TOK Be, MODALITY...TENSE Present, THEME(Mary), LOCUS C2.
C2 TOK park, DET Def, NBR S, PREP in.

Mary has fun in the park.
C1 TOK Have, MODALITY...TENSE Present, THEME(Mary), LOCUS C2.
C2 TOK park, DET Def, NBR S, PREP in.

The structures immediately above would result whether "is" or "have" are present or deleted. It is not clear at this time whether these case structure conventions applied to nominal structures will simplify the computational structure and improve paraphrase and question-answering capabilities of the model. One apparent advantage is that paraphrase transformations might always be expressed in a formalism referring to case relation arguments. A disadvantage is that the syntactic depth of the constructions would be increased.

Additional discussion of conventions for expressing semantic structures found in English sentences, some definition of lexical structure, and the development of inverse relations and their use for representing embedded sentence structures can be found in Simmons (1970b).

Computational and Logical Structure of Semantic Nets

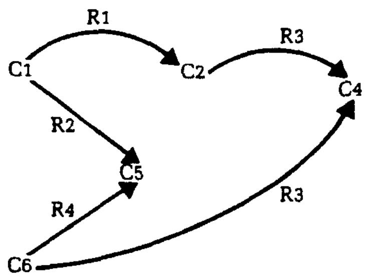
In addition to their linguistic form, semantic nets have a computational representation, a logical structure, and a conceptual content. No one of these aspects has been completely explored although enough knowledge of each has been obtained to make interesting computer programs for experimenting with the process of understanding verbally expressed ideas.

Computational Representation: To say that a structure is a network implies only that it has nodes and connections between them and that there are no restrictions such as those that exist in a tree where a daughter node may not have a direct connection to a sister or grandparent. When we add the modifier "semantic" to form "semantic network," we introduce a notion of content, i.e., a semantic network is a structure that contains meanings of language arranged in network. A semantic net generally contains *concept nodes* interconnected by *semantic relations.* Primitive verbal concepts are lexical meanings that map onto the character strings of words. Every concept is a node that has a set of relations to other concept nodes.

Figure 2.2 shows graphic and list representations of networks. The simplicity of these structures makes them ideal for computer representation as attribute-value lists or lists of triples, with the subsequent advantage of easy accessibility for processing operations. A network is defined according to the following:

Network := Node*
Node := Atom + Relationset, terminal Constant
Atom := Ci, Li (a number prefixed with L or C)
Relationset := Relation + Node
Relation := member of a list of semantic relations
Terminal Constant := character string

The asterisk signifies one or more repetitions of the marked element. The comma represents "or," the + "and." Terminal constants include English words as well as such examples as "noun," "sing," "Active," "Past," etc., which are values of lexical relations.



(C1(R1 C2)	C1 R1 C2
(R2 C5))	C1 R2 C5
(C2(R3 C4))	C2 R3 C4
(C6(R3 C4))	C6 R3 C4
(C6(R4 C5))	C6 R4 C5

Figure 2.2 Representation of Networks. (a) Abstract Network as a Directed Graph. (b) Attribute-value List and Triples Representation.

From this definition, a semantic network is an interconnected set of nodes. A node is simply a term such as Ci or Li. Its relationset encodes the information it represents. The meaning of any node is an ordering of the rest of the nodes of network with which it is related. Assuming a richly interconnected network, the complete meaning of any particular node may involve every other node in the system. This feature of semantic networks was discussed at length by Quillian (1968) who showed that human subjects when asked repeatedly to define the words in their definitions of words, could continue the process indefinitely.

Semantic relations are viewed computationally as functions and procedures. In our present realizations these relations are largely undefined as procedures although in a previous paper (Simmons and Slocum, 1972) we showed how they could be defined as generation functions that would produce an appropriate syntactic structure corresponding to each semantic relation and its arguments.

In our present development, such relations as THEME, CAUSAL ACTANT, etc., can be perceived dimly as procedures which in some cases will change the contextual definitional structure to reflect the action of a verb. Thus, THEME(John, run) as a procedure might be expected to apply the characteristic of fast motion involving legs and feet to the ordinary structure defining John. Similarly, CA1(run, John) might be expected to add the information that John instigated the motion; and GOAL(run, store) must add some terminal condition to the motion implied by "run." A most interesting and potentially rewarding research task is to develop this idea computationally.

Logical Structure: The semantic network representation of sentences is also a logical system. A semantic net is a set of triples, (A R B) where A and B are nodes and R is a semantic relation. For example, (Break THEME Window) from an earlier example is one such triple; (Window DET Def) is another. Nodes are required to be elements of a set of unambiguous symbols—usually entries in the lexicon. Semantic relations are required to be defined or definable relations such as the following list:

1. Connectives OR, NOT, SINCE, BUT, AND, IMPLY, etc.
2. Deep Case Relations CA1, CA2, THEME, SOURCE GOAL, LOC
3. Modality Relations TIME, MANNER, MOOD, ASPECT, etc.
4. Attributive Relations MOD, POSSESSIVE, HASPART, ASSOC, SIZE, SHAPE, etc.
5. Quantitative Relations Q, NBR, DET, COUNT
6. Token substitution TOK
7. Set Relations SUP, SUB, EQ, PARTOF, etc.

These relations can be defined extensionally in a given system by listing the arguments that are acceptable. They can be defined intensionally by indicating properties that are required on their arguments. For example, in (A CA1 B), CA1 for Causal Actant 1, requires that B be animate, that it be the instigator of A, and that A belong to a class of action verbs that can accept an agent.

We can also apply a set-theoretic interpretation to semantic network structures. Each node is taken as the name of a set of processes and each relational arc is a restriction on the sets signified by the nodes it connects. Let us consider the propositional structure for the following example:

John broke the window with a hammer.
C1 TOK break, CA1 C2, THEME C3, CA2 C4
C2 TOK John, DET Def, NBR S
C3 TOK window, DET Def, NBR S.
C4 TOK hammer, DET Indef, NBR S, PREP with.

The hearer knows of a set of events that he calls "breakings." These include the breakings of glasses, of windows, of crime rings, of news stories, of horses and of hearts. The net C1 restricts these breakings to only those in which John is an instigator, a particular window received the action, and a hammer was the instrument. The subnets further specify a particular man named John, a definite member of the class named windows, and some one hammer. The modality further specifies the event in terms of time of occurrence, truth value, etc.

The relationset of C1 can thus be viewed as a conjoined set of binary predicates that restrict the application of the concept named by "break" to a particular subclass of events each of which is more or less precisely specified by the values of such relations as DET and NBR.

Logical aspects of semantic net structures are developed more formally by Sandewall (1970, 1969), Palme (1971) and Simmons and Bruce (1971). Sandewall's development is of particular interest in showing conventions for insuring that the representation will be in a first order calculus and in providing fairly explicit methods for axiomatizing meanings of words. Palme has demonstrated how these techniques can be used in a semantic net-based question-answering system. Simmons and Bruce showed an algorithm for translating from semantic net structure notation into a fairly standard form of first order predicate calculus.

The most significant consequence of defining semantic networks as a logical system is to make the techniques and results of research in automatic theorem proving easily transferable to problems of question answering and problem solving in semantic nets. It has been apparent for some time that the question-answering and paraphrase problems of natural language processing are closely related to the more abstract problem of proving logical and mathematical theorems. (See Green and Raphael 1968, and Simmons 1970a.) For the shallow level of answering factual questions or recognizing paraphrases, little use of theorem-proving logic is required. For more difficult questions and verbal statements of such problems as, "the missionaries and cannibals" or "the monkey and the bananas," problem-solving and theorem-proving techniques must be used.

Conceptual Level: The conceptual level of semantic net structures has been carefully limited to that of word-sense meanings connected by semantic relations that are frequently very closely related to corresponding syntactic relations. Is there any satisfactory rationale for selecting this level rather than the semantically deeper levels chosen by Schank (Chapter 5) or Rumelhart and Norman (1971)?

The depth of a syntactic or semantic structure can be defined as proportional to the extent to which it accounts for a set of strings which are, from some point of view, paraphrases of each other. If we define syntactic paraphrases as those strings which differ only in inflectional suffixes, certain forms of deletion (as of "have," "be," and "of" prepositions) and in the ordering of lexical selections; then a minimal depth of semantic structure would be shown by a structure which was the same for strings that are syntactic paraphrases of each other. Deeper semantic levels would provide identical structures for paraphrase sets of strings that have differing choices of content words and may vary in syntactic form.

We consider the following set of sentences as syntactic paraphrases:

John broke the window with a hammer.
The window was broken by John with a hammer.
The window was broken with a hammer by John.
It was a hammer with which John broke the window.

Each of these sentences can be generated from or analyzed into the following propositional structure:

C1 TOK break, CA1(John), THEME (the window), CA2 (with a hammer).

The variation in the Modality structure as to active and passive voice, and the choice of different argument-ordering rules from the verb paradigm account for the different syntactic forms of these sentences.

We consider the following two sentences to be semantic paraphrases of each other—i.e., they are very close in meaning and describe the same event using different words and syntactic forms.

John bought the boat from Mary.
Mary sold the boat to John.

A semantic structure deep enough to represent the common meaning of these two sentences is the following:

```
C1 TOK and, ARGS C2, C3.
C2 TOK transfer, SOURCE(John) GOAL(Mary), THEME (money)
C3 TOK transfer, SOURCE(Mary) GOAL(John), THEME(boat).
```

This structure—with appropriate variations in modality—can account for both syntactic and semantic paraphrases of the two sentences and is consequently deeper than one that accounts only for the syntactic paraphrases of either. It also makes explicit the implied fact that "buy" and "sell" involve a transfer of money and of ownership in opposite directions. This is analogous to the depth of structure used by Schank in Chapter 5.

The shallower structure of semantic nets described in this chapter is shown below:

```
C1 TOK buy, SOURCE(Mary), GOAL(John), THEME(boat).
C2 TOK sell, SOURCE(Mary), GOAL(John), THEME(boat).
```

In order for the present system to determine that the two structures are semantic paraphrases, it is necessary to have a paraphrase rule such as the following connecting "buy" and "sell":

```
R01 (BUY (S-S)(G-G)(T-T) SELL)
```

This rule simply means that the TOKen of C1 may be rewritten as SELL and that no change in the values of the arguments is indicated—that is, the value of SOURCE remains Mary, the GOAL, John, and the THEME, boat. Differing generation rules for "buy" and "sell" result in the reordering of the case arguments in the surface string. The rule can be expanded to introduce a CA2–MONEY, if desired and thus, through a semantic transformation, account for the same facts as the deeper structure previously illustrated. The formulation and use of such rules is developed at some length in a later section.

It is probable that the deeper structure forms a more satisfactory psychological model of conceptual structure as well as one that will answer questions more economically. The argument for the shallower structure is that it neatly defines a distinction between syntactic and semantic transformations at the level of lexical choice and at least for the moment offers a definable reference level in the confused area of generative semantics.

The Computation of Semantic Nets from English Strings

An English string can be transformed automatically into semantic structures such as those shown in the previous section with the aid of a program that consults a lexicon and a grammar. We use a variant* of a system developed by Woods (1970) called an "Augmented Finite State Transition Network" (henceforward, AFSTN) which interprets a grammar—shown graphically as a transition network—as a program to transform an English string into a semantic network. The same system with different grammars is also used as a basis for generating English strings from the semantic networks and for embedding an algorithm that answers questions. These latter two applications will be discussed in sections immediately following this one. In this section we will briefly describe the operation of the AFSTN system and show a grammar for translating from a small class of English strings into semantic nets.

The Woods AFSTN System: Simple phrase structure grammars can be represented in the form of state transition networks. An example grammar is shown below:

NP → (ART) + (ADJ ∗) + N + (PP ∗)
PP → PREP + NP
S → NP + (AUX) + VP
S → AUX + NP + VP
VP → V + (NP) + (PP∗)

Figure 2.3 shows the augmented finite state network that represents this grammar. The grammar shown above is in context-free, phrase structure format. It uses the conventions that parentheses indicate optionality and an asterisk shows that one or more repetitions of the phrases are allowed. In the graph of Figure 2.3 the nodes or states are shown as circles with labels such as "S," "NP," "VP," "q7" etc., and the arcs or paths are labelled by phrase names such as "NP," "PP," "VP," or by word-class names such as "Aux," "V," "Prep," etc. Some states such as q4, q6, q8, and q10 are specially marked with the symbol, "T" to show that a phrase or sentence can end at that node. The grammar can be seen to be recursive in that such paths as "ADJ" and "PP" form loops. It has subgraphs, such as NP, VP, and PP, which are also the names of arcs.

The figure shows only syntactic category information associated with the arcs, but each arc may in fact have an associated set of conditions to be met and operations to be performed as control passes from state to state. In this fashion, an AFSTN augments the ordinary state transition network by allowing a program to occur at each arc.

The reader can imagine a scanner that looks at each word in such a sentence as "The merry widow danced a jig," and examines its word class under the control of the "S" net. The first arc examined is the one labelled NP, which causes a transfer to the net, NP, where the category, Article, corresponds to the name of the first arc

*Programmed by D. Matuszek and J. Slocum at Univ. of Texas following Woods' description.

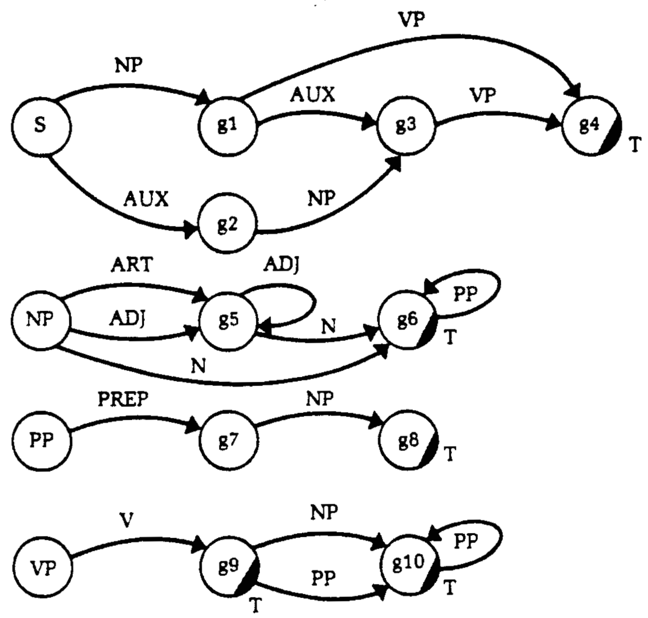


Figure 2.3 An Augmented Finite State Transition Network for a Simple Grammar.

and allows transition to state q5. The next word, "merry" is category Adjective which allows transition over the loop labelled adj to state q5. "Widow" allows transition of the arc, N, to state q6 where the noun phrase has been satisfied, and is POPped to achieve transition of the arc labelled NP in net S. This takes the system to state q1 where transition of the VP arc will successfully complete the scan of the sentence. By operating the programs associated with each arc, a structure of the sentence may be created in any form designed by the programmer. Thus transformational deep structures result from one such set of programs written by Woods, and semantic network structures result from the programs described in the following paragraphs.

For complete understanding of the following example programs, the reader will find it helpful to study Woods' careful description of the structure and operation of his AFSTN system (Woods 1970).

Analysis of a Noun Phrase: The following program recognizes and transforms simple noun phrases into semantic structures.

```
(S(Push NP T
  (SETR  SUBJ  *)
  (TO  Q10) ))
```

```
(NP(CAT ART T
   (MAKEPR (QUOTE DET)(GETF DET))
   (SETR NBR (GETF NBR))
   (TO N2))
   (TST ADJ T
   (SETR NBR OK)
   (JUMP N2) ))
(N2(CAT ADJ T
   (SETR ADJ (PUT(GENSYMC)(QUOTE TOK) *))
   (TO N3))
   (TST N T
   (SETR NBR OK)
   (JUMP N3)))
(N3(CAT NOUN (AGREE(GETR NBR)(GETF NBR))
   (ADDPR (QUOTE MOD)(GETR ADJ))
   (ADDPR (QUOTE NBR)(GETF NBR))
   (ADDPR (QUOTE TOK)  *)   (JUMP N4) ))
(N4(POP(PUTPRL(GENSYMC)(GETR MLIST))T))
```

A graph of this program is shown in Figure 2.4 and an explanation of its flow ano effects is shown in Table 2.1. The figure shows the major test above the arc and th operations to be performed below each arc. The table shows the condition of the sta. (*) register usually containing the word under the scanner except when a POP is to occur when it contains the results to be passed back up to the control level of the last PUSH. The flow of program operations is numbered, the main result is listed in the next column, and the last column shows the registers or structures that contain the result.

If we enter the program with the phrase, "a merry widow" the system scans the first element, "a" and enters the network at S. (S(PUSH NP T) has the effect of transferring control to the network node labelled NP. At this node, (CAT ART) means that a procedure called CAT looks at the dictionary entry for what is in the * register to discover if it is an article. Since * contains "a", the CAT function returns true and the operations associated with that arc are undertaken. (Note in Figure 2.3 that if we were considering the phrase "old dowager", CAT ART would have failed and TST ADJ would have transferred control to N2 without moving the scanner —by using JUMP instead of TO.)

The first operation, line 3 in Table 2.1, is (MAKEPR (QUOTE DET) (GETF DET)). GETF DET gets the value of the feature DET from the lexicon for the element in the * register. This value for "a" is "indefinite." MAKEPR puts the pair (DET INDEF) in a register called MLIST.

The next operation, line 4, is (SETR NBR (GETF NBR)). This operation gets the value of NBR for "a" which is SINGULAR, and puts it into the register named NBR. The terminal operation (TO N2) transfers control to N2, setting the * register to "merry" the next element in the phrase.

ation of its flow and
ibove the arc and the
condition of the star
ept when a POP is to
ontrol level of the last
result is listed in the
uctures that contain

the system scans the
T) has the effect of
is node, (CAT-ART)
ry for what is in the
the CAT function re-
aken. (Note in Figure
AT ART would have
t moving the scanner

DET) (GETF DET)).
or the element in the
pair (DET INDEF) in

is operation gets the
register named NBR.
ing the * register to

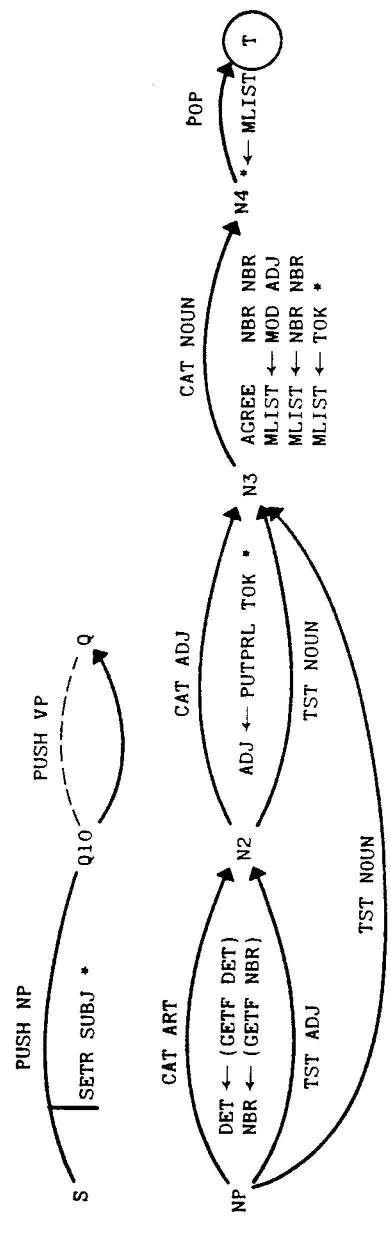


Figure 2.4 Graph of Semantic Transformation of a Simple Noun Phrase.

Table 2.1 Program Flow for Analysis of "a merry widow."

Scanner *register	Line #	Flow of program operations	Result	Location of result
a	1	(S(PUSH NP T)	TRANSFER TO NP	
	2	(NP(CAT ART)	TRUE Conditional	
	3	(MAKPR(QUOTE DET)(GETF DET))	(DET INDEF)	MLIST
	4	(SETR NBR(GETF NBR))	SINGULAR	NBR
merry	5	(TO N2))	Transfer to N2, Move scanner to "merry"	
	6	(N2(CAT ADJ) T)	TRUE conditional	
	7	(SETR ADJ(PUT(GENSYMC)(QUOTE TOK)*))	(C1(TOK MERRY)), ADJ ← C1	ADJ and Property List
widow	8	(TO N3))	Transfer to N3, move scanner to "widow"	
	9	(N3(CAT NOUN)	TRUE conditional	
	10	(AGREE(GETR NBR)(GETF NBR))	(AGREE Singular Singular), TRUE Condition	
	11	(ADDPR(QUOTE MOD)(GETR ADJ))	(MOD C1)	MLIST
	12	(ADDPR(QUOTE NBR)(GETF NBR))	(NBR Singular)	MLIST
	13	(ADDPR(QUOTE TOK) *)	(TOK widow)	MLIST
	14	(JUMP N4))		
	15	(N4(POP(PUTPRL(GENSYMC)(GETR MLIST))T))	(C2(TOK widow)(DET INDEF) (NBR Singular)(MOD C1))	*and Property List
	16	(SETR SUBJ *)	(C2 etc)	SUBJ

At N2 the first test is (CAT ADJ) which is found True for "merry." The main operation undertaken here, is to create a property list structure, (C1 (TOK merry)), by using the functions, GENSYMC which creates the symbol C1 and PUTPRL which makes the property list for C1 and returns C1 as its value to be used by SETR which places C1 in the register, ADJ. The next operation in N2 is (TO N3) which sets the scanner and * register to the next element in the string, namely "widow", and transfers control to node N3. Again it can be noticed – in Figure 2.4 – that if a phrase without an adjective had been analyzed, the test, (CAT ADJ) would have failed and (TST NOUN) would have succeeded, causing a (JUMP N3) without moving the scanner.

At N3 (line 9) two tests are called for; first (CAT NOUN) second, (AGREE(GETR NBR)(GETF NBR)). Previously nodes have had a CAT test or the dummy TST each followed by the sumbol T in place of a second conditional. The symbol T has meant that the second conditional was automatically taken as TRUE. Here, however, the second conditional tests for agreement in number between the noun and any article that it may have (and the second conditional is evaluated first by the system).

The register NBR has been set in line 4 to the value SINGULAR and (GETR NBR) retrieves this value. Since "widow" is singular, (GETF NBR) returns this value. Thus the condition reduces to (AGREE SINGULAR SINGULAR) which evaluates to TRUE. At this point additional semantic agreement tests are usually introduced to select word-sense meanings, but to maintain simplicity of exposition they are omitted in this example.

Since the two conditions of N3 have been met, some operations are now undertaken to form a semantic structure for the phrase. These are ADDPR functions which create the structure shown in the result column for lines 11-13. At line 14, the terminal operation (JUMP N4) transfers control without moving the scanner. N4 provides an unconditional POP using PUTPRL and GENSYMC to create the structure shown in the result column. POP assigns the value, C2 to the * register and returns control to the place where (PUSH NP) occurred – i.e., line 1.

At this point we can notice that a PUSH arc is also a conditional that returns True if the next element or elements in the string being scanned form the phrase which is the argument of PUSH. (PUSH VP, PUSH PP, etc.). The PUSH NP being true in this case, its operations set a register called SUBJ to the value of the * register – i.e., C2 –, move the scanner and transfer control to Q10. C2 is the name of a property list structure containing the following semantic structure for the phrase:

```
(C2(TOK WIDOW)(NBR SINGULAR)(DET INDEF)(MOD C1))
(C1(TOK MERRY))
```

It should be noticed that the resulting semantic structure corresponds to conventions described in the preceding section.

Analysis of a Verb Phrase: As a result of making the noun phrase of the preceding example, control was returned to the top level, the (PUSH NP) arc was successfully completed, C2 assigned to the SUBJ register, and control was passed to node Q10. At Q10 we have a choice of two arcs; either (POP *) or (PUSH VP). Assuming, now, that our sentence had continued as follows: "A merry widow had been dancing a jig," then the scanner would contain "had" and the arc, (PUSH VP) would be attempted.

A portion of the VP network is shown in Figure 2.5 and program corresponding to it is listed in Appendix Table 1. This part of the network has the purpose of determining the modality (i.e., NUMBER, TENSE, VOICE, FORM, ASPECT & MOOD) and constructing the semantic form of the verb. This figure shows fairly completely the tests (above the arc) and the operations (below the arc) that are required. The functions that make structure pairs are indicated by a left-pointing arrow, ←; those such as LIFTR that send arguments up to the next level by a vertical arrow, †. To maintain simplicity of exposition, some paths in the network are left incomplete where they concern modal, emphatic and future auxiliaries.

We will follow the example sentence through the graphed network of Figure 2.5 leaving the interested reader to consult the program for complete statements. Control is at VP with the scanner containing "had." (CAT AUX (GETF BE)) fails since "had"

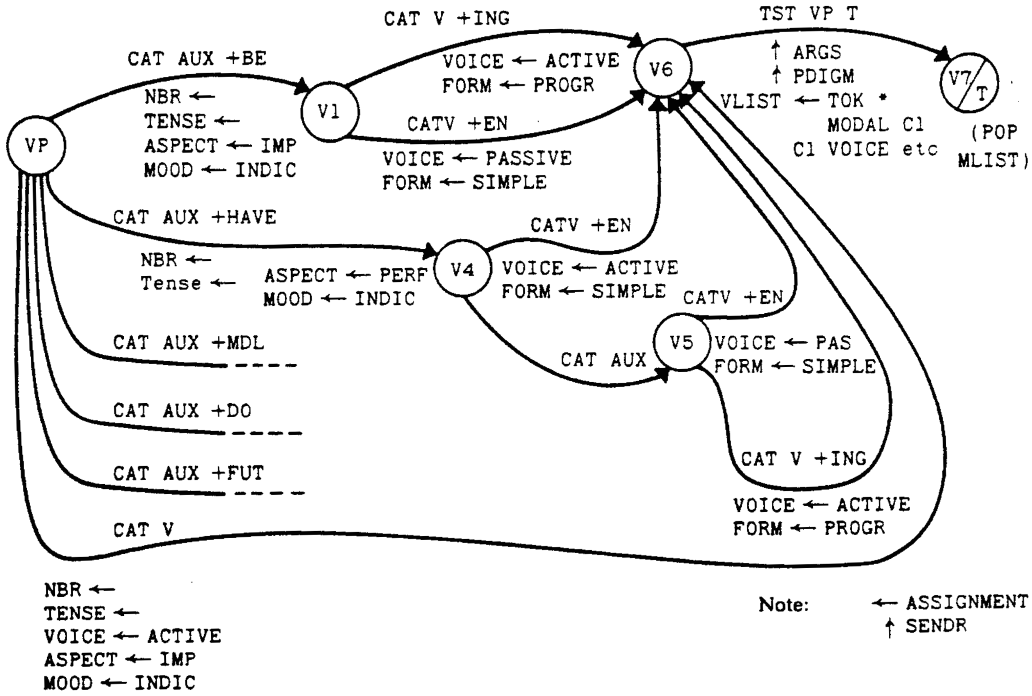


Figure 2.5 Partial VP Net for Computing MODALITY and Verb Structure. Note: ← ASSIGNMENT; † SENDR.

is not a form of "to be." (CAT AUX (GEFT HAVE)) succeeds since "had" has the feature HAVE. The operations indicated are to create in register MLIST the pairs (NBR 3PPL)(TENSE PAST)(ASPECT PERFECT) and (MOOD INDIC). MAKEPR first cleans the register MLIST, then adds a pair, while ADDPR simply adds a pair to MLIST. Thus MAKEPR is called first on entry to a level to insure emptying MLIST at that level of any previous contents such as those inserted in the last example. The scanner is advanced to "been" and control is passed to V4 by the terminal actions, (TO V4).

Since "been" has the category "aux" it fails the CAT V test, but passes the (CAT AUX) and control is passed to V5 with the scanner at "dancing." Notice that no additional semantic structure is assigned here because the "been" is only part of the indicator for either a passive or progressive verb form. The first path tests for (CAT V) and (GETF +EN); since "dancing" is not an "en" form it fails this test and the next arc is attempted. This one tests for (CAT V) and (GETF +ING) which succeeds because the verb is a progressive or +ING form. At this point the pairs (VOICE ACTIVE) & (FORM PROGRESSIVE) are added to MLIST by using ADDPR. Control is passed without advancing the scanner by using (JUMP V6).

V6 is an unconditional arc in that (TST VP T) always evaluates TRUE. (TST is a null test so the argument VP is only for a human reader, and T is the second conditional which evaluates to TRUE). At V6 we now have the elements of the verb structure in the MLIST and must create the appropriate semantic structure and send it and other information back up to the top level of the sentence. LIFTR is a function that sends information up a level, and it is used here to send the content of the verb features PDIGM and ARGS to the next higher level in the sentence where they will be used to continue the example in the next subsection of the paper. In traversing nodes, VP V4, and V5 we accumulated the modality structure for the verb as follows:

```
((NBR 3PPL)(TENSE PAST)(ASPECT PERFECT)(MOOD INDIC)(VOICE
ACTIVE)(FORM PROGRESSIVE))
```

Here PUTPRL is used to form a property list headed by the result of operating GENSYMC, namely C3; and the pair (MODAL C3) is put onto a clean MLIST. The pair (TOK dance) is added to MLIST, the scanner is advanced to "a" and control is passed to V7. V7 is an unconditional POP that transfers the contents of the MLIST in the * register and sends it back up to Q10 where (PUSH VP) occurred.

Analysis of the Sentence: Figure 2.6 shows a portion of the top level of a sentence network and Appendix Table 2 shows the corresponding portion of program. The figure shows some incomplete paths, indicated by dotted lines, to suggest additional grammar that is not required for the present example.

As we returned to Q10, the * register contained the following content:

```
((TOK DANCE)(MODAL C3))
```

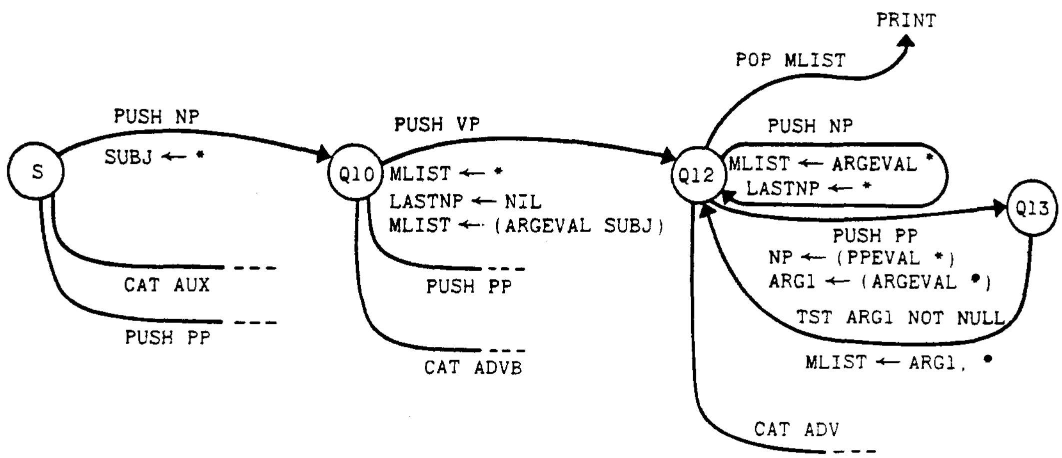


Figure 2.6 Network for Top-level Analysis of a Sentence.

This is put in MLIST by (SETR MLIST *) and a register called LASTNP is set to NIL (for later use). At this point we are ready to determine what semantic relatio holds between the subject and the verb. A function called ARGEVAL takes the nou phrase in question, consults the lexical entry for the verb and determines whethe the NP is a causal actant, source, locus, theme, etc., and returns either NIL or a deep case structure name (or names) as a value. The relevant lexical information for "dance" for this purpose was sent up earlier from the NP net into the registers PDIGM and ARGS whose contents are as follows:

```
(PDIGM(CASES(CA1 LOCUS) THEME LOCUS2))
      (SUBJ(CA1 LOCUS))(OBJ THEME)(PREP(WITH LOCUS2)))
(ARGS (LOCUS ANIMATE)(THEME dance)(LOCUS2 ANIMATE))
```

In evaluating SUBJ, ARGEVAL must first obtain the head noun with a (GET SUBJ (QUOTE TOK)) which returns the lexical node for "widow". Consultation of the registers ARGS and PDIGM then shows that the noun in SUBJECT position for VOICE-ACTIVE must be a CA1 and a LOCUS. (If voice had been passive, ARGEVAL would have read SUBJECT as OBJECT). The data in ARGS shows that CA1 and LOCUS for this verb must be marked animate. The noun "widow" is so marked in the lexicon so ARGEVAL returns (CA1, LOCUS) as the value of the relation between "widow" and "dance."

In a similar manner when this function is later called with "jig," it will discover that this noun is an OBJECT and marked with "dance" and so return THEME. If presented with a phrase such as "with John," it recognizes by the preposition "with" and the animate marker on "John" that it is dealing with a LOCUS2.

In this example the function ADDPR is then called with arguments as follows: (ADDPR (CA1 LOCUS)(GETR SUBJ)). The result is to put two pairs on the MLIST as follows:

```
((CA1 C2)(LOCUS C2))
```

The scanner is advanced and control passed to node, Q12 by the instruction, (TO Q12). The first arc leaving Q12 is (PUSH NP T). Since the phrase following the verb is "a jig," the push to NP returns with the * register containing, C4 whose property list is as follows:

```
(C4(TOK JIG)(NBR SINGULAR)(DET INDEF))
```

ARGEVAL of C4 returns THEME and ADDPR adds to the MLIST the pair (THEME C4). At this point the terminal action (TO Q12) advances the scanner and passes control to Q12 again. But the sentence ended with "jig" so the * register is set to NIL and the PUSH NP and PUSH PP arcs fail. The arc (POP etc.) T) is unconditional so it succeeds in building the final structure for the sentence and passing up the node C5 whose property list is as follows:

```
(C5(TOK DANCE)(MODAL C3)(CA1 C2)(LOCUS C2)(THEME C4))
```

The complete expansion of C5 gives the following semantic structure for the sentence:

```
C5 TOK DANCE    C3 NBR 3PP1            C2 TOK WIDOW
   MODAL C3        TENSE PAST             NBR SINGULAR
   CA1   C2        ASPECT PERFECT         DET INDEF
   LOCUS C2        MOOD INDIC             MOD C1
   THEME C4        VOICE ACTIVE
                   FORM PROGRESSIVE
C1 TOK MERRY    C4 TOK JIG
                   NBR SINGULAR
                   DET INDEF
```

Had the sentence continued with a prepositional phrase such as in ". . . danced a jig with John", the PP arc of Figure 2.6 would have operated, and the additional structure (LOCUS2 C5)(C5(TOK John)(NBR SINGULAR)(DET DEF)) would have been added.

The semantic net developed for the "merry widow" sentence is in fact a tree. As additional sentences in a discourse are analyzed, they will refer to nodes in earlier structures and the tree of the single sentence becomes part of a larger network. Elements of the sentence tree are also inter-connected by paths through the lexicon. Thus what we see in this analysis of the sentence is an explicit structure of unambiguous lexical references. It is the surface tip of an iceberg with great depths of inter-relationship in the data contained in the lexicon but not shown here as part of the analysis of the sentence. We claim that what is shown is the shallowest level of semantic structure.

Generating English Sentences from Semantic Nets

A basic function called GEN is central to the process of generation. This function takes a list as its argument. The list contains the name of a structure from which a sentence is to be generated followed by a series of constraints on the modality. For example, if we wish to generate a question from the sentence "A merry widow danced a jig," the call to *GEN* would be written as follows:

```
GEN((C5 ACTIVE, INTERROG, (QUERY JIG)))
```

This call is designed to generate, "What did the merry widow dance?"

GEN calls first a function that gets the modal structure of C5 and rewrites th values specified in the call. After this has been accomplished, another functio., PATTERN, is called to select one of the verb paradigm patterns associated with the verb "dance." The paradigm for generation is selected by discovering which one fits the case arguments of the semantic structure. In this example, the following paradigm is selected:

```
((SUBJ (CA1-LOCUS))(OBJ THEME))
```

The register, SNTC, is then set to the list,

```
(CA1-LOCUS VACT THEME)
```

It is this list that will be scanned and presented in the * register to control the generation sequence of the sentence. At this point, GEN turns over control to the generation grammar, R, with the call, (RETURN(PUSH R)). The POP from R will cause the sentence that has been generated to be printed out as the value of the function GEN.

The generation grammar-program, R will be explained by first showing the top level of control flow then by looking at the generation of the NPs and VP. Appendix Table 3 shows the grammar for the top level and Figure 2.7 presents it as a network that forms the basis for the explanation.

ɔf a larger network. Ele-
ɔs through the lexicon.
licit structure of unam-
rg with great depths of
ɔt shown here as part of
is the shallowest level

ıeration. This function
structure from which a
:s on the modality. For
ence "A merry widow

v danc..."
.25 and rewrites those
ɪed, another function,
ns associated with the
overing which one fits
ie, the following para-

ter to control the gen-
control to the genera-
ƆP from R will cause
value of the function

first showing the top
Ps and VP. Appendix
esents it as a network

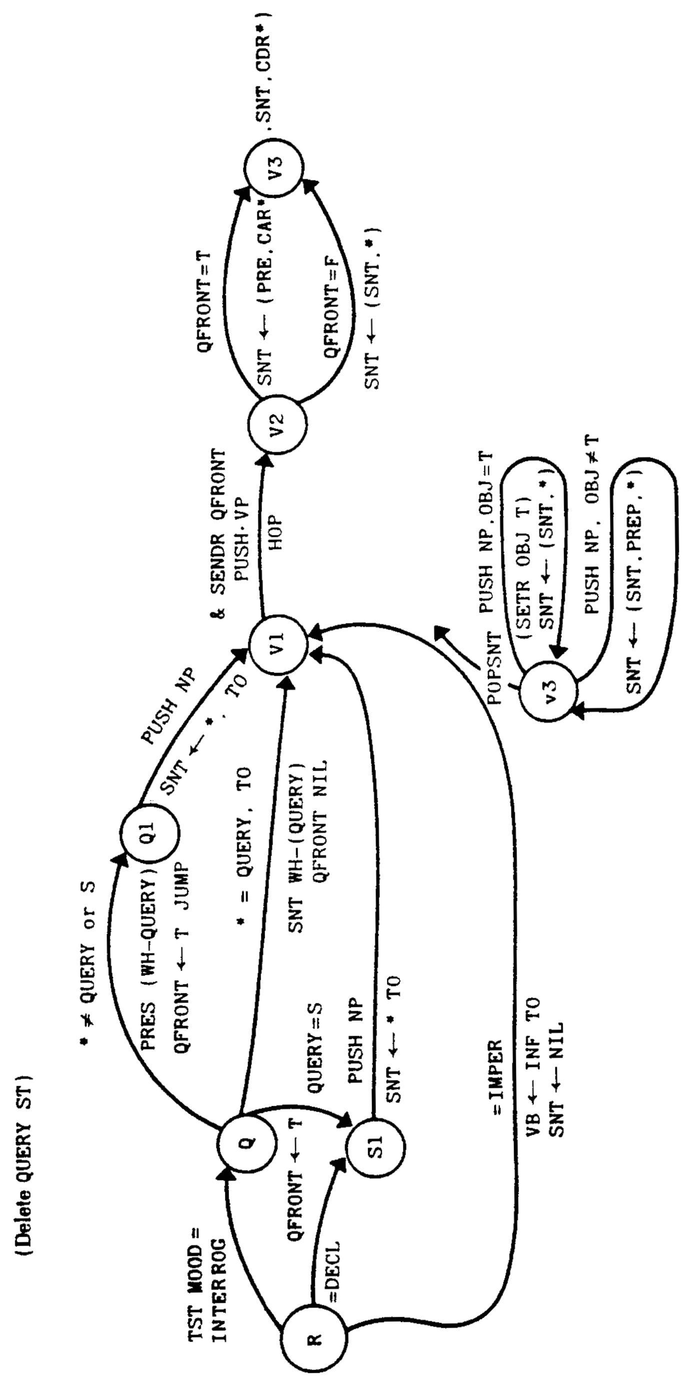


Figure 2.7 AFSTN Top-level Sentence Generation.

The semantic structure for the sentence after modification by GEN appears as in Table 2.2 below.

Table 2.2 Semantic Structure for Generation.

C5 TOK DANCE MODAL C3 CA1-LOCUS C2 THEME C4	C3 NBR 3PPL TENSE PAST ASPECT IMPERF MOOD INTERROG VOICE ACTIVE FORM SIMPLE QUERY THEME	
C1 TOK MERRY	C2 TOK WIDOW NBR SINGULAR DET INDEF MOD C1	C4 TOK JIG NBR SINGULAR DET INDEF

The register SNTC at the time of the PUSH to R contains (CA1–LOCUS, VACT, THEME) and the * register contains CA1–LOCUS.

Figure 2.7 shows that at node R the MOOD value of C3 is examined to determin whether it is Interrogative, Imperative, or Declarative. Since it is marked INTERROG control is jumped to State Q. Arcs leaving Q test to determine the form of the questio by examining the QUERY arc in C3. Since the value of QUERY is THEME and the * register contains CA1–LOCUS a query-fronting transformation on the questioned element will be required. This is signified by setting the register QFRONT to T. The question word is then generated by calling the function WH- with THEME as its argument. This function computes the question word "What" for the THEME Value "jig" of the structure C5. Control is then JUMPed to node Q1 leaving the * unchanged. At Q1 we PUSH NP, which results in the generation of "a merry widow" which is POPped back up in the * register. Register SNT is set to this phrase and control is passed to V1 with * set to the next element of the control string, VACT.

At V1 a PUSH VP is tried and on successful completion it returns in the * register, "did dance." The "did" was generated because the register QFRONT was set to T; otherwise "danced" would have been the value. We JUMP to V2 where QFRONT is again tested. Since the value is T, the register SNT is set to the sequence, (PRE, (CAR *), SNT, (CDR *)), whose values are respectively, What, did, the merry widow, dance. Since there are no arguments in C5 that are now unaccounted for, the transfer to V3 results in a (POP SNT) where SNT contains the generated question, "WHAT DID THE MERRY WIDOW DANCE."

In passing we can note that at node Q, if the value of QUERY on the MODAL structure had been CA1–LOCUS, the contents of * would have matched it, QFRONT would have been set to F, and the question generated would have been, "WHO danced a jig". If the value of QUERY had been S, the question would have been "DID A MERRY WIDOW DANCE A JIG".

Generating Simple NPs: Figure 2.8 shows a grammar network for dealing with noun phrases containing only a determiner, an adjective string, and a noun. Generalizations of this net to include further modifications by nouns and prepositional phrases simply require extension of the grammar in the form described. More complex embeddings of relative clauses etc., will require continued study particularly with reference to appropriate sequencing and limitations of depth.

On the PUSH NP of the previous example (Figure 2.7) the * register contains CA1–LOCUS and there has been a (SENDR ST (GET ST *)). The effect of this latter operation has been to make the structure C2 available at the NP level. An expansion of C2 is as follows:

```
C2 TOK WIDOW         C1 TOK MERRY
   DET INDEF
   NBR SINGULAR
   MOD C1
```

At node NP there are three TST arcs to examine the determiner and choose an article. The test that is made for an indefinite article is as follows:

```
(TST DEF (EQ(GET(GETR ST)(QUOTE DET))(QUOTE INDEF))
```

This test gets the value of DET (which is INDEF) from C2 and matches it against the value INDEF. Since the TST condition returns T, the register SNT is set to the value (A) and control is jumped to node N1.

At N1 the arcs test for the presence of adjectives with the following expression:

```
(TST ADJ (SETR ADJ (GET (GETR ST)(QUOTE MOD)))
```

As a result in the present example, register ADJ is set to C1. The graph notation ADJ ← MOD shows this consequence, and control is JUMPED to N2. If the (GET ST MOD) had returned NIL signifying no MOD relation on the structure, ADJ would have been set to NIL and the condition on the TST arc would have failed allowing the next arc (TST NOADJ T) to cause a JUMP to N3.

At N2 the test is made to determine whether there is one adjective, (ATOM (GETR ADJ)) = T, or more if the predicate fails. Since the value of ADJ is the atom C1, there is only one modifier. The notation in the figure:

SNT ← SNT + (GETLEX ADJ NIL)

is a shorthand for the following expression in the actual grammar:

```
(SETR SNT(APPEND(GETR SNT)(LIST(GETLEX (GETR ADJ) NIL))))
```

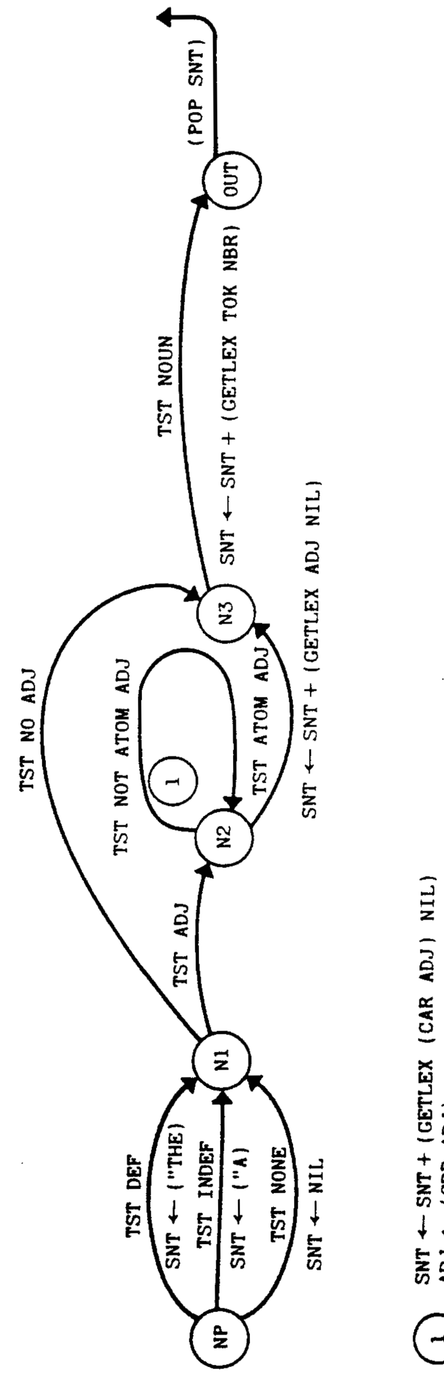


Figure 2.8 NP Generation Net.

The function (GETLEX A B) takes a structure name and a morphological attribute—such as NBR, TENSE, and so forth—and returns the word form. In this case GETLEX returns MERRY as its value. SNT is reset to the concatenation (i.e., APPEND) of its old value to the list (MERRY) making its current value, (A MERRY). Control is then JUMPED to N3.

If register ADJ had contained a list of values, GETLEX would have been called with (CAR(GETR ADJ)). . . , ADJ would have been set to (CDR (GETR ADJ)), and control JUMPED to N2 to loop through the list of adjective modifiers.

At N3 in this net the noun head of the structure is developed by the call (GETLEX (GETR ST) NBR) which returns WIDOW, the singular form. SNT is then reset to A MERRY WIDOW and control is JUMPED to OUT where (POP SNT T) puts this phrase in the * register and returns control to the higher node—Q1 in this example—which called it.

Generating Verb Strings: Figure 2.9 shows the net representation of the grammar-program for generating verb forms. The upper part of the figure shows the (PUSH VP) with its conditions. These are the sending down to the next lower level of the Modal structure, the TOKen of the verb and the register QFRONT. The VP subnet will use this information to generate a verb string according to the data in the Modal structure, and its successful POP will return the verb string that has been generated in the * register.

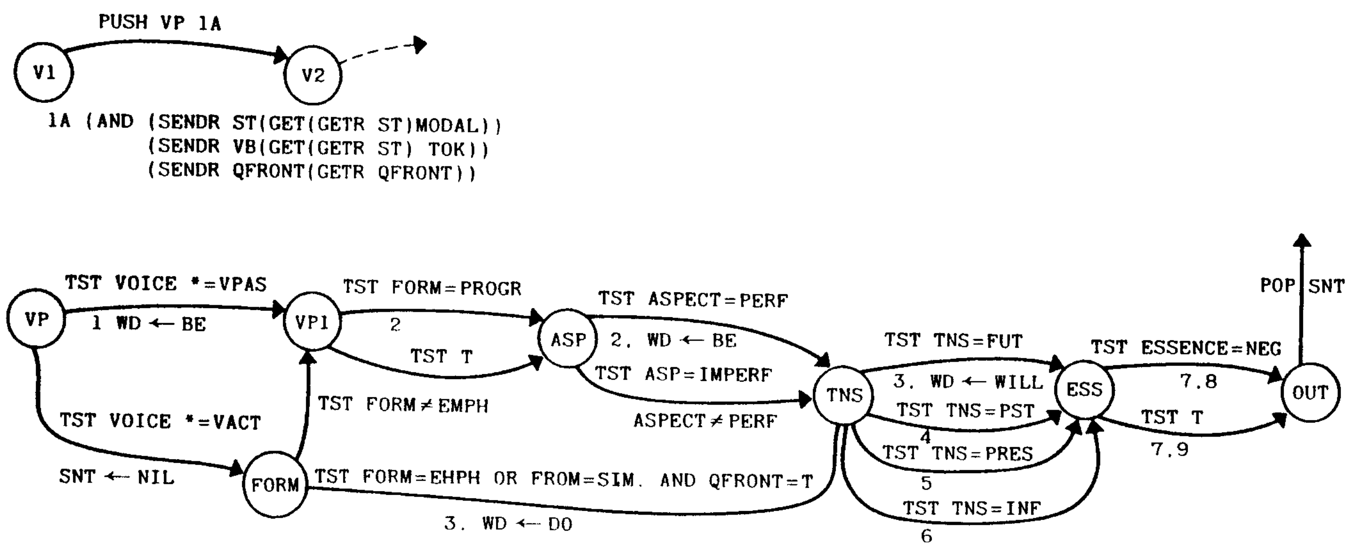
At the PUSH to VP the * register contains either VACT or VPAS from scanning the generation pattern. The two arcs leaving VP begin to generate the verb string in one of these two forms. Under the arc, is a number referring to the operations listed in the lower part of the figure, which actually construct the elements of the string. In our example, the * register contains VACT. The operation on this arc is to set the register SNT to NIL in order to clear it. Control is JUMPED to node FORM where the FORM attribute on the Modal structure is found. Since FORM has the value SIMPLE (Table 2.2), QFRONT is T, and ASPECT is IMPERF, operation #3 is performed to set SNT to the value returned by

```
(LEXWD (GETR WD)(QUOTE INF))
```

LEXWD takes as arguments a word token and a morphological signal; like GETLEX, it returns a word form—in this case, DANCE. The second operation on this arc is to set the register WD to the value DO—introducing an auxiliary to be fronted for the question form. In the case of a PROGRESSIVE or a PERFECT form, other arcs—from VP1 or ASP—would introduce an auxiliary verb, BE or HAVE, which in the case of a question could be fronted.

After these operations, control is jumped to TNS where the value of the attribute TENSE on the Modal structure is examined. The operations associated with these arcs will produce a tensed English verb form for whatever is in the register WD. In the present example WD contains DO and the value of TENSE is PAST so LEXWD returns DID. If a simple declarative present sentence were being generated, WD would still

Figure 2.8 NP Generation Net.



1 SNT ← (LEXWD WD,:EN)
2 SNT ← (LEXWD WD,:ING)+SNT
3 SNT ← (LEXWD WD,INF)
4 WD ← (LEXWD WD,PST)
5 WD ← (LEXWD WD,PRES)
6 WD ← (LEXWD WD,INF)
7 WD ← (Agree WD NBR)
8 SNT ← (WD + NOT + SNT)
9 SNT ← (WD + SNT)

Figure 2.9 Net Representation for Generating Verb Forms.

contain the form of the verb sent down from V1 and the verb string generated would be a simple verb in present form such as DANCE.

Control is then jumped to node ESS where the form in WD is made to agree in NBR with the subject, a NOT is inserted for a negative, and SNT is set to the concatenation of the value of WD and SNT by:

```
(SETR SNT(CONS(GETR WD)(GETR SNT)))
```

Control is JUMPED to OUT where the contents of SNT are POPped to the calling level in the * register.

Answering Questions with Semantic Nets

So far the semantic net structures have been shown to preserve the meanings expressed by a phrase or a sentence at a level such that syntactic paraphrases are represented by a canonical semantic structure – one that differs only in such sentence design features as are represented in the modality. This level of structure is well-suited to generating syntactic variations as needed to embed sentences in varying environments without changing the intended meaning, but it falls short of what is required for question-answering applications.

The following two sentences would usually be judged to carry the same* meaning; particularly, if one is a question, the other would be selected as an answer.

1. Wellington defeated Napoleon at the Battle of Waterloo.
2. Bonaparte lost the Battle of Waterloo to the Duke of Wellington.

These two examples have differing semantic structures because of the different lexical choices that have been made for the concepts WIN–LOSE–DEFEAT, NAPOLEON–NAPOLEON I–NAPOLEON BONAPARTE–BONAPARTE, and WELLINGTON–THE DUKE OF WELLINGTON–THE IRON DUKE.

Earlier it was mentioned that deeper semantic structures can be devised such that the two examples above might have the same semantic or conceptual representation, but that our present approach was deliberately fixed at the definable level where unambiguous lexical concepts – i.e., word sense descriptions – are related by explicit semantic relations. This choice of level requires an additional mechanism of paraphrase rules in order to account for paraphrase resulting from different lexical choices. In studying the process of answering questions from text, it is apparent that a deeper structure will be more economical of computation, but that paraphrase rules will probably continue to be required.

*"same" is taken to mean "equivalent with respect to a purpose."

```
1 SNT ← (LEXWD WD, :EN)
2 SNT ← (LEXWD WD, :ING)+SNT
3 SNT ← (LEXWD WD, INF)
4 WD ← (LEXWD WD, PST)
5 WD ← (LEXWD WD, PRES)
6 WD ← (LEXWD WD INF)
7 WD ← (Agree WD NBR)
8 SNT ← (WD + NOT + SNT)
9 SNT ← (WD + SNT)
```

Figure 2.9 Net Representation for Generating Verb Forms.

Paraphrase rules to account for the two example sentences above can be expressed quite simply. First, let us show an abbreviated representation of the two semantic structures:

```
DEFEAT; C1 WELLINGTON T NAPOLEON, L BATTLE OF WATERLOO
LOSE; S BONAPARTE, T BATTLE OF WATERLOO, G DUKE OF WELLINGTON
```

The abbreviations for deep case relations decode as follows: C1-Causal Actant 1, T-Theme, L-Locus, S-Source, G-Goal. Some paraphrase rules associated with LOSE are shown below:

```
Rule 1  (LOSE(S-S)(T-T)(G-G) WIN)
Rule 2  (LOSE(C-G)(T-S)(L-T) DEFEAT)
Rule 3  (LOSE(L-S)(T-DEFEAT)(G-G)(L-T) SUFFER)
```

If we seek to transform the second semantic structure into the first, rule R2 applies since it connects LOSE and DEFEAT. The rule is interpreted to have the following effect:

```
LOSE; S BONAPARTE            DEFEAT; C DOW
      T BOW          ⇒               T BONAPARTE
      G DOW                          L BOW
```

An interpreter given the relevant rule and the structure headed by LOSE, does the following:

1. Begin a copy of the structure.
2. Write the new TOK value as DEFEAT.
3. Write a semantic relation C and set its value to the old value of G (i.e., Duke of Wellington).
4. Write a relation T and set its value to the old value of S.
5. Write a relation L and set it to the old value of T.

If we were now to generate an active declarative sentence from the transformed or new structure we would get:

The Duke of Wellington defeated Bonaparte at the Battle of Waterloo.

The rule is symmetric, so if, reading from right to left, we applied it to the first sentence structure headed by DEFEAT, we could have transformed into a structure that would generate:

Napoleon lost the Battle of Waterloo to Wellington.

Thus rule R2 accounts for a fairly complex paraphrase relation between LOSE and DEFEAT. If we are to demonstrate that the two example sentences are completely equivalent in terms of this semantic system, we must also have rules such as the following:

```
R4  (NAPOLEON-BONAPARTE-NAPOLEON+I-NAPOLEON+BONAPARTE)
R5  (WELLINGTON(PMOD-TOK)(PREP-OF)(DET-DEF) DUKE)
R6  (WELLINGTON(MOD-IRON)(DET-DEF) DUKE)
```

Rule R4 is a simple substitution rule that is interpreted as meaning that any instance of one of the terms can be substituted for any instance of another. This is a relatively rare form of perfect synonymy of names. Rules R5 and R6 are a more common case in which a word is transformed into a phrase that has the same meaning. The same interpreter is used to transform the structure WELLINGTON into DUKE, PMOD WELLINGTON, PREP OF, DET DEF. The rule R5 is still a symmetric rule but with an important difference from the previous example. Since DEF and OF are not semantic relations, they must be values and the interpreter takes them as conditions on the structure headed by DUKE. Thus, THIS DUKE OF WELLINGTON or THE DUKE OF WELLINGTON will transform into WELLINGTON, whereas A DUKE AT WELLINGTON fails as does THE DUKE OF WINDSOR, etc.

The result of applying the rules illustrated to the semantic structure of either of the two sentences is to take it into an exact match with the other. The rules that have been illustrated are quite simple and they require very few conditions for their application. Other rules may be very complex with different conditions applying depending on the direction in which the rule is to be applied.

Another pair of example sentences will show a higher degree of complexity:

Napoleon commanded the troops that lost the battle.
Napoleon lost the battle.

The abbreviated structures for these two sentences follow:

```
COMMAND, C1 NAPOLEON, T TROOPS
LOSE, S TROOPS, T BATTLE

LOSE S NAPOLEON, T BATTLE
```

A rule to show the paraphrase relation between these two structures must show that in certain circumstances a combination of two sentences implies a third. Such a rule could be written as follows for application to COMMAND and LOSE;

```
R7  (COMMAND [(T(1st) = C(2nd))(TOK(2nd) = LOSE)]
             (S-C(1st))   (T-T)  LOSE)
```

The elements in the square brackets are conditions to be met by the structure to which the rule is to be applied. They say that the Theme of the first sentence must correspond to the Source argument of the second, and that the Token of the second is limited to LOSE. The remainder of the expression is the transformation which produces the desired structure.

A rule of this degree of complexity is no longer obviously symmetric, and several new notational conventions have been introduced. These vastly increase the complexity required of the interpreter of rules. It becomes apparent that rules for generating paraphrase are at least as complicated as those required for analysis and generation of sentences. Once again the Woods AFSTN interpreter can be used, this time with states as rule names and paraphrase rules written in the form of conditions and operations.

If we assume the * register contains the name of the structure to be examined and that a function, GETA, with a semantic relation as an argument returns the value associated with that relation for the * structure, the following arcs illustrate a mode for writing paraphrase transformations in the AFSTN:

```
(R7(TST  COMMAND-LOSE(AND(EQ(GETA TOK)(QUOTE COMMAND))
                         (SETR 2NDVB(GET(GETA T)S*))
                         (EQ(GET(GETR 2NDVB)TOK)(QUOTE LOSE)))
     (MAKEPR TOK(QUOTE LOSE))
     (ADDPR  S (GETA C))
     (ADDPR  T (GET(GETR 2NDVB T))
     (JUMP OUT)))
(OUT(POP(PUTPRL(QUOTE QT)(GETR MLIST)) T))
```

The number of conditions and operations on this arc reveal the complexity of rule R7. Essentially, the condition is an ANDed set of tests to determine that the Token of the structure under the scanner is COMMAND, that the value of its Theme argument is the Source argument for a verb it dominates (S* is the backlink from TROOPS to LOSE), and that the dominated verb is LOSE. If all these conditions are met, then MAKEPR and ADDPR construct a new list of arguments and values on a register called MLIST. At the jump to OUT, PUTPRL makes a property list—i.e., a new semantic structure—with the name QT, which is then POPped in the * register.

The point to be emphasized is that paraphrase rules of arbitrary complexity can be used if interpreted by the AFSTN system. On the other hand, if only simple rules such as R1-R6 are required, a simpler translating function will prove much more efficient. The question of efficiency for a given purpose is central to the design of a question-answering algorithm for it has a great deal of computation to accomplish.

A Question-Answering Algorithm: It is from paraphrase rules such as those just described that a text-based question-answering system derives much of its power. But more than this is required. If we assume that a data base of semantic structures representing sentence meanings has been accumulated, then it is first necessary to select a set of structures that appear relevant to the question. One measure of rele-

met by the structure to
the first sentence must
he Token of the second
: transformation which

symmetric, and several
astly increase the com-
urent that rules for gen-
uired for analysis and
preter can be used, this
i the form of conditions

ucture to be examined
iment returns the value
g arcs illustrate a mode

OMMAND))
:)S*))
K)(QUOTE LOSE)))

'eal the complexity of
ermine that the Token
of its Theme argument
klink from TROOFS to
iditions are met, then
ies on a register called
–i.e., a new semantic
register.
itrary complexity can
d, if only simple rules
ill prove much more
itral to the design of a
tation to accomplish.
les such as those just
s much of its power.
of semantic structures
t is first necessary to
One measure of rele-

vance is the number of lexical concepts in common between the proposed answer and the question. This can be obtained by a simple algorithm that orders the candidates according to the number of Token values they have in common with the question. Such a function is called CANDS. It takes the name of the question structure as an argument, goes to the lexicon to obtain a list of structures that contain each content word used in the question, and orders these in terms of their match with the question.

The task of CANDS introduces a new lexical requirement, that each lexical entry contain a list of the semantic structures in which it is used. The structures to be indexed for each word are the sentences (or larger discourse units) in which each occurs. In terms of a previous example, the words Napoleon, Wellington, Defeat, Battle, Waterloo all occur in structure C1, and Bonaparte, lose, Battle, Waterloo, Duke, and Wellington occur in C2. Thus, for this example, Wellington and Waterloo have as values for the U/I (used/in) attribute, C1, C2; while Napoleon has U/I C1 and Bonaparte has U/I C2. If we ask the question,

Did Napoleon win the Battle of Waterloo?

We will discover that there are four content words in the question, three of which occur in C1 and two in C2. The ordering that CANDS returns for these candidate-answer structures is thus, C1, C2.

The task of the question-answering function, called ANSWER, is now to match the question structure against each candidate. The first step is to determine if the token of the head verb of the question matches the head verb of the candidate. If there is no direct match, there may be a paraphrase rule that applies that can transform the question structure into that of the candidate answer. But how is the relevant rule, if it exists, to be located? We have gained much experience with question-answering and theorem-proving experimentation and know that the cost of finding and applying such transformations is very high indeed.

Additional lexical structure helps to simplify the problem. If each entry in the lexicon indexes the rules that transform it to another entry–i.e., the paraphrase rules–then the task becomes manageable. The following fragments of lexicon for some of the words in example sentences, C1 and C2, show the indexing method.

```
(LOSE(IMPLY(WIN R1)(DEFEAT R2)(SUFFER R3)))
(DEFEAT(IMPLYBY (LOSE (R2,R3))))
(WELLINGTON (IMPLY (DUKE (R5,R6))))
(NAPOLEON (IMPLY ((NAPOLEON, BONAPARTE, ETC...)R4)))
```

Thus a lexical entry shows that LOSE will paraphrase to DEFEAT by rule R2 (shown on p. 100).

A function named PATHS takes two arguments such as LOSE and DEFEAT. It examines the lexical structures associated with the two words (actually word-sense addresses) to discover if there is a rule connecting the two. If not, it takes the set that the first word will transform into and calls itself recursively to see if any member of that set transforms into the second word; that failing, it takes the set of words the second transforms into to determine if one of these goes into words derived from the first. It stops either when a match is found or when an arbitrary depth of search has been reached. If successful, it returns an ordered list of rule names that the interpreter function can use to translate from the first word to the second.

PATHS is the function that discovers whether one word can be transformed into another. It is an important timesaving device that uses a directed search to avoid an exhaustive exploration of the network of paraphrase rules.

When such a path has been found, a function called TRANSLATE is used to transform a copy of the question structure into the form of the candidate answer. This is the interpreter function that has been discussed previously; if the rules to be used are of simple form, TRANSLATE can be a simple function to interpret them; if the rules are complex, TRANSLATE can push to a paraphrase grammar that is interpreted by the AFSTN system. In either case TRANSLATE returns the name of the question structure as its value.

The function MATCH1 is the control function for matching the Tokens of two semantic structures. It also examines quantifiers and modalities to determine if quantificational relationships are satisfied and if tense and negation relationships are matched. It has the additional task of examining the question's semantic structur to determine if the relation QWD is present and satisfied.

An example will show more clearly what MATCH1 does in the context of its ca by the ANSWER function.

What man lost a battle?

The semantic structure would be as follows:

```
Q1 TOK LOSE, S Q2, T Q3
Q2 TOK MAN QWD WHAT
Q3 TOK BATTLE, DET INDEF
```

(ANSWER Q1) calls (CANDS Q1) which returns the ordered list (C2,C1) as candidate answering structures. MATCH1 is then called for application to the first of these, C2, in the following fashion:

```
(MATCH1 Q1 C2)
```

MATCH1 embeds the following call:

```
(TRANSLATE (PATHS LOSE LOSE))
```

No transformation is required and MATCH1 returns Q1 unchanged, thus signalling that the head of Q1 matches the head of C2. MATCH1 is itself embedded by a function MATCH which attempts to see that the structure that is the value of each semantic relation of the question matches the structure that is the value of each semantic relation in the candidate answer. What it does is to call

```
(MATCH1 (GET Q1 S)(GET C2 S)) which means
     (MATCH1 Q2 C21)
```

where C21 is the structure

```
C21 TOK BONAPARTE
    DET DEF
```

and Q2 is:

```
Q2 TOK MAN
   QWD WHAT
```

When the paraphrase rule (BONAPARTE IMPLY MAN) is found, by PATHS, the translation gives Q2′ TOK BONAPARTE, and MATCH1 looks then at the QWD relation and puts BONAPARTE in a register called QANS whose value will be returned by ANSWER if the rest of the question is accounted for by the candidate structure.

In a later call, (MATCH1 BATTLE BATTLE), this function will compare the determiners and find that INDEF in the question is encompassed by DEF in the candidate. Eventually the match is found to be complete and BONAPARTE is returned as the answer to "What man lost a battle?" LISP definitions of the functions ANSWER, MATCH, MATCH1, and other related functions are included in the appendix to this chapter. The complexities of traversing two semantic structures can be understood from studying these functions, but because of their deeply recursive nature, further verbal description will not be attempted.

Concluding Discussion

Three topics of critical importance to the computation and use of semantic structures have only been lightly touched on in this chapter. Lexical structure, semantic disambiguation, and the translation from semantic structure to procedural language will each be discussed briefly in this section, and then a short concluding summary will close the chapter.

Lexical Structure: The form of a lexical entry is the same as that for any other semantic structure—a node associated with a set of relational arcs connecting it to other nodes. The nodes are word-sense meanings or constants, and the arcs are semantic relations of various types. Some of these are indicators of paraphrase transformations such as SUPerset, SUBset, $Rule_i$, etc. Some are morphological to show the meanings of various endings, such as Present or Past Participle, Future, Singular, Plural, etc. Some relate the word-sense to its syntactic word-class. Additional relations, such as Print Image and Used/In, map word-sense meanings onto words and data statements, respectively.

If the system is to be used for mapping from English into another natural language or into a procedural language, additional semantic relations must be encoded into the dictionary for these purposes and used by grammar programs that can accomplish these tasks. As yet we have not attempted a careful description of lexical content as it is dependent on the uses of a language processing system. Each new task typically requires use of some standard lexical features but adds its own unique requirement.

Semantic Disambiguation: The relevant lexical information for this task is in the form of semantic classes or markers and selection restrictions associated with each lexical entry. The information is used in the parsing grammar in a manner similar to that illustrated for testing syntactic agreements. Such an approach is minimally satisfactory for analyzing a carefully controlled subset of English; but as Bolinger (1965) has argued, disambiguation may require consultation of any aspect of knowledge that the listener may have. A generally satisfactory scheme for semantic disambiguation has not yet been developed but will probably require complex conditions and consultation with the whole context of a discourse. This area is suggested as a profitable one for computational linguistic research.

Translation to Procedural Languages: The semantic network structures for sentences have been defined at the level of deep case structures. The question arose as to whether this is properly called a syntactic or semantic level. We defined transformations that do not change the choice of lexical entries as syntactic and those that do as semantic, thus forming a fairly clear distinction between the concepts "syntactic paraphrase" and "semantic paraphrase." From these notions it is immediately apparent that any transformations into other languages, natural or procedural, are semantic in nature. The structure on which syntactic and semantic transformation both operate is called a semantic structure and defined as a set of unambiguous references to word-sense meanings connected by explicit, definable semantic relations.

Woods and Winograd have each shown how a procedural semantics—a system of semantic transformations—can be used to operate on sentence structures to transform them into commands in a procedural language. Both of these researchers are concerned with objects in a data base that are designated by noun-phrase descriptions and each embeds the identifying elements of the noun phrase in a retrieval command to discover particular named elements of data such as AA-57 or the red block named B3. This appears to be the first level of procedural language transformation—the discovery of the particular data objects identified by a noun phrase. Winograd's system most clearly includes a deeper level of procedural language in its use of Microplanner to assemble a program of actions in the command language that drives the simulated robot hand.

For example, the sentence, "Place the red block on the blue block" first retrieves an object name such as B3 corresponding to the *noun phrase*, "the red block" and similarly, B4 for "the blue block." The sentence now has a semantic structure equivalent to the following:

Place: Mood Imper, T B3, On B4

A transformation associated with the verb, "place" transforms this into a goal statement roughly as follows:

ThGoal: (ON B3, B4)

A Microplanner *program expands this state description into a series of* commands that will achieve the desired state and passes this as a program for the interpreter of the command language to run, and so physically to accomplish the goal.

In this example we can see three levels of procedural transformation: first, the identification of referents of the NPs; second, transformation of the sentence structure into a desired goal state; and third, the assembly of a command language program by Microplanner to achieve the desired goal state. The resulting command language statement is a representation of the pragmatic meaning of the English statement, and the dynamic interpretation of the command language statements results in changes in the world of blocks and is an operational definition of the "meaning" of the sentence.

The semantic structure of a sentence can thus be seen to be simply a first stage representation of meaning which can be operated on by various semantic transformations to produce paraphrases *or translations into* other languages including procedural ones. Schank's conceptual structures and Winograd's goal structures can both be seen as modelling deeper levels of thought that are signified by semantic structures of verbal meanings. Transformation from either the semantic structure or these deeper structures into procedural languages, models the human process *of generating actions* in the world under the control of thought processes which also correspond to verbal expressions.

Summary: This chapter has described a consistent approach to the derivation and manipulation of representations of verbal meaning for a subset of English sentence structures. The subset treated is a small one and we have undoubtedly ignored more English forms than we have accounted for. We have, however, described a process for mapping from English into a semantic level, from that level back into English, and procedures for discovering equivalence relations between different semantic structures. This is a theory and a model of superficial aspects of verbal communication and one that fits naturally into those systems which model the deeper forms of thought required for problem solving and the accomplishment of non-verbal actions.

These theories and models of language understanding offer a very rich area for continued research. Computational research is needed to improve data representations and algorithms required by the models and to provide additional systems such as the Woods AFSTN and PLANNER to simplify the programming tasks. A great deal of linguistic research is needed to expand the range of natural language constructions for which syntactic and semantic conventions can be agreed on. Psychological research is necessary to determine how closely these theories and models account for experimentally determined facts of human verbal memory and human language understanding and generation skills. Finally, there is need for hardware development of computers with gigantic memories, multiple processors, and command languages at the levels now exemplified by LISP and PLANNER.

Acknowledgments

The ideas and programs that I have described in this chapter, which was prepared with the support of the National Science Foundation, Grant GJ 509X, were developed conjointly with my students and research associates as a team effort. Each of the following persons contributed significantly: Robert Amsler, Sharon Baronofsky, Bertram Bruce, David Matuszek, and Jonathan Slocum. I take full responsibility, however, for any inaccuracies and inadequacies in what I have presented here.

Appendix to Chapter Two

Question-Answering Algorithm — *J. Slocum*

```
   (ANSWER (LAMBDA (QST)(PROG (CANDS QANS)
          (SETQ CANDS (CAND QST))
AGAIN     (COND((NULL CANDS)(RETURN NIL))
            ((MATCH(MATCH1 QST(CAR CANDS))(CAR CANDS))
                                              (RETURN QANS))
            (SETQ CANDS(CDR CANDS))
            (GO AGAIN) )))

   (MATCH(LAMBDA(QT ST)
      (COND((OR(NULL QT)(NULL ST)) NIL)
            ((MATCH2 (INDICATORS QT)) T)
            (T NIL) )))

   (MATCH2 (LAMBDA (INDS)
      (COND ((NULL INDS) T)
          ((MATCH(MATCH1 (GET QT (CAR INDS))(GET ST(CAR INDS)))
                 (GET ST (CAR INDS)))(MATCH2 (CDR INDS)))
          (T NIL)  )))

   (MATCH1 (LAMBDA (QT ST)
     (COND((NOT(DETMTCH(GET QT DET)(GET ST DET))) NIL)
       ((NOT(MODMTCH(GET QT MODAL)(GET ST MODAL))) NIL)
       ((QWDTEST QT ST) NIL)
       (T (TRANSLATE(PATHS(GET QT TOK)(GET ST TOK))))  )))

   (QWDTEST(LAMBDA(QT ST)
       (AND(GET QT QWD)(SETQ QANS ST) NIL) ))
```

Table A.1 Program for VP.

```
(Q10(PUSH VP T

(VP(CAT AUX(GETF BE)
     (MAKEPR (QUOTE NBR)(GETF NBR))
     (ADDPR (QUOTE TENSE)(GETF TENSE))
     (ADDPR (QUOTE ASPECT)(QUOTE IMPERF))
     (ADDPR (QUOTE MOOD)(QUOTE INDIC))
     (TO V1))

  (CAT AUX(GETF(HAV)
     (MAKEPR (QUOTE NBR)(GETF NBR))
     (ADDPR (QUOTE TENSE)(GETF TENSE))
     (ADDPR (QUOTE ASPECT)(QUOTE PERF))
     (ADDPR (QUOTE MOOD)(QUOTE INDIC))
     (TO V4))

  (CAT V T
     (MAKEPR (QUOTE NBR)(GETF NBR))
     (ADDPR (QUOTE TENSE)(GETF TENSE))
     (ADDPR (QUOTE VOICE)(QUOTE ACTIVE))
     (ADDPR (QUOTE ASPECT)(QUOTE IMPERF))
     (ADDPR (QUOTE MOOD)(QUOTE INDIC))
     (JUMP V6)))

(V1(CAT V (GETF +ING)
     (ADDPR (QUOTE VOICE)(QUOTE ACTIVE))
     (ADDPR (QUOTE FORM)(QUOTE PROGRESSIVE))
     (JUMP V6))

  (CAT V (GETF +EN)
     (ADDPR (QUOTE VOICE)(QUOTE PASSIVE))
     (ADDPR (QUOTE FORM)(QUOTE SIMPLE))
     (JUMP V6) ))

(V4(CAT V(GETF +EN)
     (ADDPR (QUOTE VOICE)(QUOTE ACTIVE))
     (ADDPR (QUOTE FORM)(QUOTE SIMPLE))
     (JUMP V6))
     (CAT AUX T
     (TO V5)))

(V5(TST VP T
     (LIFTR ARGS (GETF ARGS))
     (LIFTR PDIGM (GETF PDIGM))
     (MAKEPR (QUOTE MODAL)(PUTPRL(GENSYMC)(GETR MLIST)))
     (ADDPR (QUOTE TOK) *)
     (TO V7) ))

(V7(POP (GETR MLIST)T))
```

Table A.2 Program for Top-level Analysis of a Sentence.

```
(S(PUSH NP  T
    (SETR SUBJ *)
    (TO Q10))
    (CAT AUX  T
       -
       -
       -
    (PUSH PP  T
       -
       -
       -
          ))

(Q10(PUSH VP T
    (SETR MLIST *)
    (SETR LASTNP NIL)
    (ADDPR (ARGEVAL(GETR SUBJ))(GETR SUBJ))
    (TO Q12) ))

(Q12(PUSH NP T
    (ADDPR (ARGEVAL *) *)
    (SETR LASTNP *)
    (TO Q12))
    (PUSH PP T
    (PPEVAL * (GETR LASTNP))
    (SETR ARG1 (ARGEVAL *))
    (JUMP Q13))
    (POP(PUTPRL(GENSYMC)(GETR MLIST))T))

(Q13(TST ARG1 (NOT (NULL(GETR ARG1))))
    (ADDPR (GETR ARG1) *)
    (TO Q12) ))
```

Table A.3 Top-level Sentence Generation Net.

```
(R(TEST INTER
         (EQ"INTER (GET(SETR MODE1(GET(GETR ST) * ))"MOOD)
    (JUMP Q))
  (TST IMPER(EQ"IMPER(GET MODE1"MOOD))
    (TO V1))
  (TST DECL T (JUMP S1)))
(S1(PUSH NP T
    (SETR SNT *)
    (TO V1)))
 (Q(TST QUERY(EQ * (GET(GETR MODE1)"QUERY))
     (SETR SNT (WH- *))
     (SETR QFRONT ())
     (TO V1))
   (TST SQUERY(EQ"S(GET(GETR MODE1)"QUERY))
     (SETR QFRONT T)
     (JUMP S1))
   (TST OTHER T
     (SETR SNT(WH-(GET(GETR MODE1)"QUERY)))
     (SETR QFRONT T)(PUT ST(GET(GETR MODE1)"QUERY()))
     (JUMP Q1)))
(Q1(PUSH NP T
    (CONC(GETR SNT) * )
    (TO V1))
  (POP(PRING"Q1)T))
(V1(PUSH VP(AND(SENDR WD(GET(GETR ST)"TOK))
               (SENDR ST(GETR MODE1)))
    (HOP V2)
(V2(TST QFRONT(EQ(GETR QFRONT)NIL)
    (CONC(GETR SNT) * )
  (TST NO T(CONS(LIST(CAR * )(GETR SNT))(CONC(GETR ST)(CDR *))
```

Table A.4 NP Generation Net.

```
(NP(TST DEF(EQ(GET(GETR ST)(QUOTE DET))(QUOTE DEF))
       (SETR SNT (QUOTE THE))
       (JUMP N2))
   (TST INDEF(EQ(GET(GETR ST)(QUOTE DET))(QUOTE INDEF))
       (SETR SNT(QUOTE A))
       (JUMP N2)))

(N2(TST ADJ (SETR ADJ(GET(GETR ST)(QUOTE MOD))
       (JUMP N3)
   (TST NO ADJ T
       (JUMP N4)

(N3(TST ONE ADJ (AND(ATOM(GETR ADJ))(NOT(NULL(GETR ADJ))))
       (APPEND(GETR SNT)(GETLEX(GETR ADJ))NIC))
       (JUMP N4))
   (TST MORE T
       (APPEND(GETR SNT)(GETLEX(CAR(GETR ADJ))NIC))
       (SETR ADJ(CDR(GETR ADJ))
       (JUMP N3))

(N4(TST NOUN T
       (APPEND(GETR SNT)(GETLEX(GET(GETR ST)TOK)
                                (GET(GETR ST)NBR)))
       (JUMP OUT)

(OUT(POP SNT T)
```