A B S T R A C T

This thesis develops a theory and a computer model of learning based on English text. The model is experimentally implemented as a computer program, called CLET (Computer Learning from English Text), which achieves the learning of elementary arithmetic from an ordinary fourth-grade textbook. CLET takes all of its input from unmodified sentences appearing in this book. It performs syntactic, semantic, and discourse level analyses of the input material. CLET must then induce the general algorithms from the examples presented to it. It builds up, automatically, a program to perform the required operations. CLET then solves elementary arithmetic problems using the program it has itself constructed.

Logic, deductions, and procedural power have been heavily emphasized in previous approaches to computer understanding of natural language. These earlier systems had many shortcomings which prevented them from being able to learn directly from English texts. The hypothesis asserted here is that these difficulties cannot be solved by slightly increasing the sophistication of earlier methods. A more complete linguistic analysis, of the sort carried out in CLET, is required.

CLET does not attempt to provide a psychological model of a child's learning behavior. On the other hand, its capabilities go far beyond the simple numeric adjustment of a predetermined model. It emphasizes semantic structure as well as elaborating procedures that analyze coherent discourse. CLET can be said to learn because it "understands" and makes inferences from connected text.

Learning is one of the most remarkable aspects of human intelligence. By exploring this process on computers, we hope to go one step further in the quest for artificial intelligence.

Acknowledgements.

I am indebted to Professor L. Stephen Coles for his guidance in this thesis and his criticisms at the various stages of its development. I would also like to express here my gratitude to all those friends who encouraged me throughout this work by their thought provoking discussions, or simply their presence.

This work was supported in part by the Air Force Office of Research, under contract AFOSR-F44620-71-C-0087.

TABLE OF CONTENTS

NOTATION

Double quotation marks ["] are used for standard quoting. Single quotation marks ['] introduce a new word, or more often, signal the unconventional use of a word or expression.

In the examples, a sentence or node-structure is starred [*] if it would be rejected, whatever the reason. A star in parentheses [(*)], however, indicates a sentence which would be rejected on a first attempt, but could be eventually understood under forced interpretations.

Transformation of some syntactic or semantic construct into another is indicated by an arrow [--►]. On the other hand, semantic implication [==⟩] is one of the basic relations in the memory network. It is introduced in section 3.1.3.

To improve readability, sentences or relations will often be written in a shorter form where node-types and noun-verb relationships are not spelled out unless necessary to avoid confusion. Thus,

(benefactive-action-process-V:give agent-N:Sue
beneficiary-N:Ed patient-N:(19 candies))

might be written:

(give Sue Ed (19 candies)).

If prepositions are present, they are simply written with a colon [:] in front of the noun they modify. Thus,

(put John candy in: car).

For the same reason, internal structures will often keep instantiation implicit. Thus, in the above sentence, "Sue" ("Ed") appears in fact as an instance of "girl" ("boy") whose name is "Sue" ("Ed"). Both "19" and "candies" refer to instances of the concepts "19" and "candy."

If the intention is to describe the relations involving a particular noun which is the current focus of interest, this noun is written outside the relation(s), and its place is indicated by an asterisk [*]:

```
26 :(has-as-parts   *   (2   tens))
   !
   !(has-as-parts   *   (6   ones))
```

C H A P T E R 1
I N T R O D U C T I O N

1.1 GENERAL DESCRIPTION.

The objectives of this thesis are twofold: to advance the state of the art of computer natural language processing, and to achieve more insight in the problem of modeling human learning. We would like to stress our belief, from a philosophical standpoint, that computers will demonstrate a definite step forward in intelligence only when they can understand and learn from a human-like language. This does not have to be an actual language, but should compare satisfactorily with the level of complexity and information content of ordinary human languages. The importance of language in thought processes is a controversial matter. But this is not the point: it remains undisputably easier for a human to be told things rather than having to rediscover them for himself. The same should apply for computer systems. Whatever inferential capabilities they may possess, it will always be to their advantage to share others' experiences.

This thesis departs from previous work in that our program does not expect as input author-composed sentences or problem statements but actual textbook sentences from the body of the text. The book which was chosen is "Seeing Through Arithmetic," 4th grade (published by Scott, Foresman & Co. [Hartung & al. 1967]), and the excerpts selected concern addition and subtraction of integers (pp. 54-69).

During the preliminary work of problem definition, we looked for a textbook that would explain arithmetic operations as a clearly stated set of rules. The extensive efforts in this search led to the following, somewhat surprising result: nowadays, young American grade-school children are never told how to perform addition or subtraction in a general way. They are supposed to infer the general algorithms from

examples. Thus actual texts are usually composed of a series of short illustrated 'stories.' Each story describes an example of execution of the addition or the subtraction algorithms.

CLET (Computer Learning from English Text) is a large program: 60 pages of Snobol, and 125 pages of Fortran and assembler code. Its initial knowledge of arithmetic is identical to that expected from a child studying the same textbook. It knows that a number can be represented by ones, tens, etc. It also knows that there can be more than one such representation (decomposition of one ten into ten ones, regrouping, etc.). It knows the subtraction tables, and also how to add 2 or even 3 digits. (This is in the form of pre-compiled subroutines.) It analyzes the various examples, ignoring the pictures which accompany the text (average processing time per sentence: 3.0 seconds; total time for learning addition and subtraction: 11 minutes). It "learns" the general algorithms by analyzing the flow of control in each case.

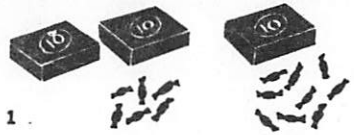
By using the "number line," the child, and CLET, are supposed to know already that 9 + 7 + 5 = 21. But they have yet to learn how to solve 16 + 5 or 9 + 12. The important point then is to determine what steps to perform, when, and how to "carry" or "borrow."

The following pages display successively two particular examples from the text, followed by the internal structures representing the algorithm which is eventually induced by the system (after 8 selected examples), and finally by an author-generated English re-statement of these structures. The figures are given here without explanation. The structures are discussed at length throughout the thesis. At this point, they are merely shown to give the reader a better feeling of the complexity that underlies even such simple algorithms as addition. The reader is invited to examine and compare carefully the original and the final versions.

1 Computation

Addition (two-digit numerals)

See



1

A Ed had 26 candies. Sue gave him 19 more. Then Ed had how many candies?

$26+19=t.$

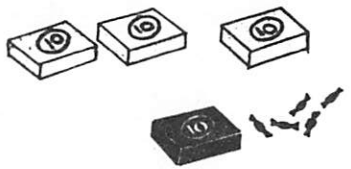
You are to find the sum of 26 and 19.

2

Put the 9 candies with the 6 candies. There are 15 candies.

$$\begin{array}{r} 26 \\ 19 \\ \hline \end{array}$$

Add the ones. There are 15 ones.



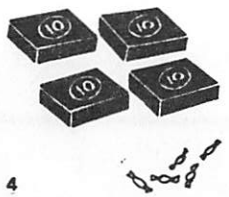
3

Put 10 of the 15 candies into a box. 5 candies are not in boxes of 10.

$$\begin{array}{r} {\scriptstyle 1} \\ 26 \\ 19 \\ \hline 5 \end{array}$$

15 ones are 1 ten 5 ones. Write 5 in the ones' place of the answer to show there are 5 ones.

Write 1 above the 2 in the tens' place to show there is one more ten.



4

Put the boxes of 10 candies together. There are 4 boxes of 10 candies.

$$\begin{array}{r} {\scriptstyle 1} \\ 26 \\ 19 \\ \hline 45 \end{array}$$

Add the tens. There are 4 tens.

Write 4 in the tens' place of the answer to show there are 4 tens.

5

There are 45 candies in all.

$26+19=45.$

Then Ed had 45 candies.

think B David sold 23 tickets, Mark sold 46 tickets, and Jim sold 85 tickets. Altogether the boys sold how many tickets?

$23+46+85=k.$

You are to find the sum of 23, 46, and 85.

$$\begin{array}{r} 23 \\ 46 \\ 85 \\ \hline \end{array}$$

What do you add first? How many ones are there?

$$\begin{array}{r} {\scriptstyle 1} \\ 23 \\ 46 \\ 85 \\ \hline 4 \end{array}$$

Think of 14 ones as 1 ten ■ ones.

Why is 4 written in the ones' place of the answer?

Why is 1 written above the 2 in the tens' place?

$$\begin{array}{r} {\scriptstyle 1} \\ 23 \\ 46 \\ 85 \\ \hline 4 \end{array}$$

What do you add next? How many tens are there?

$$\begin{array}{r} {\scriptstyle 1} \\ 23 \\ 46 \\ 85 \\ \hline 154 \end{array}$$

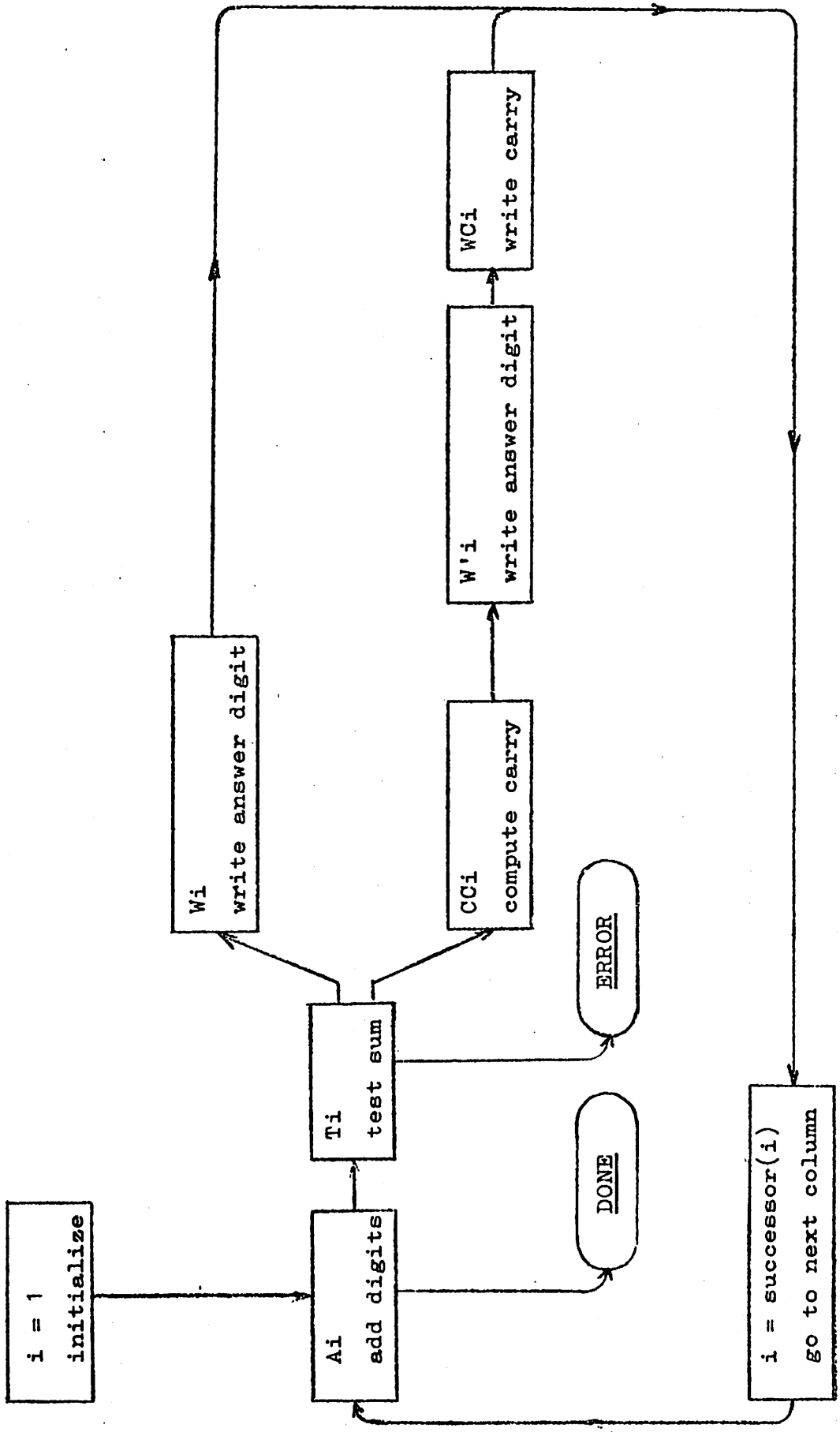
Think of 15 tens as 1 hundred ■ tens.

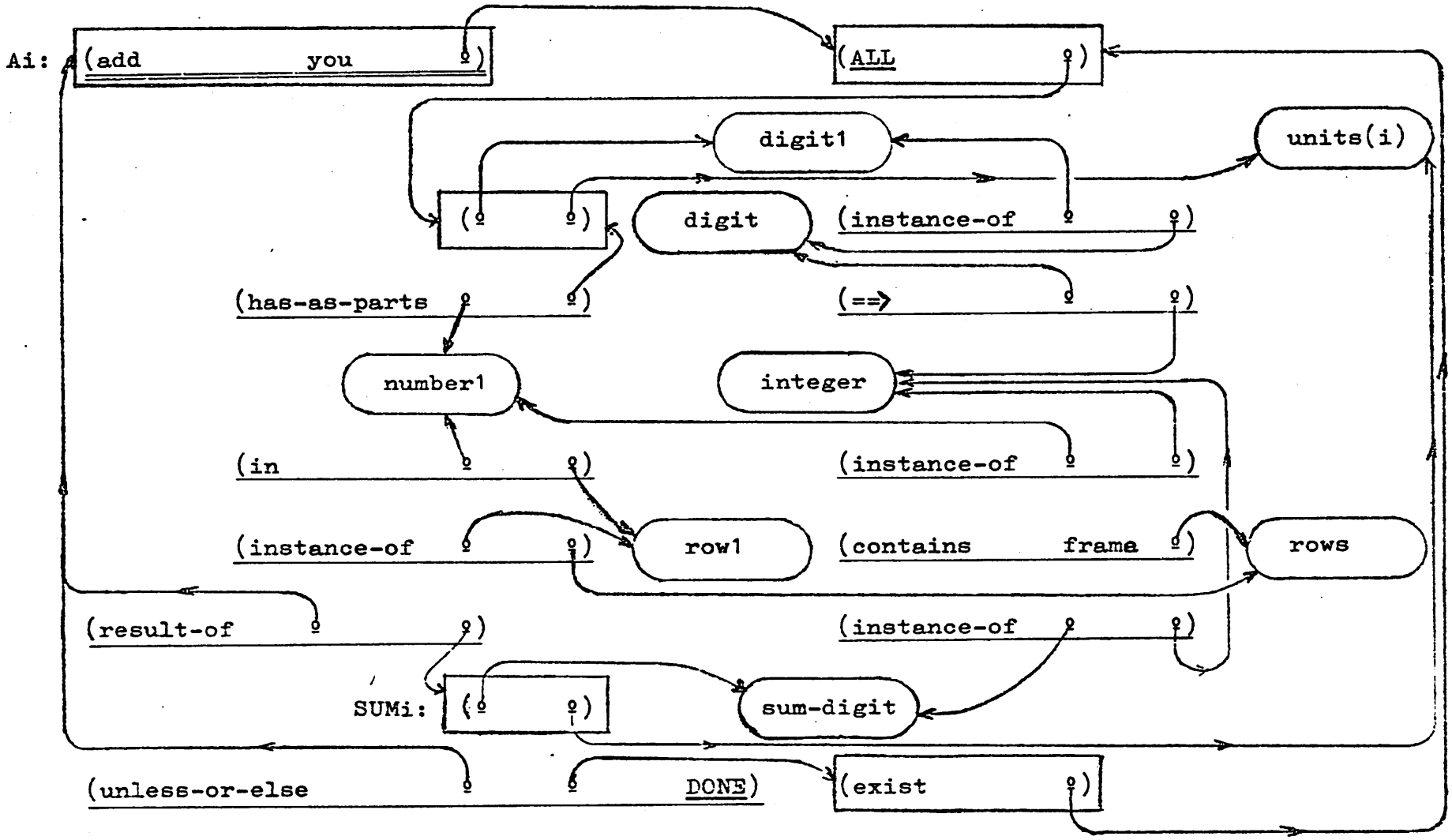
Why is 5 written in the tens' place of the answer?

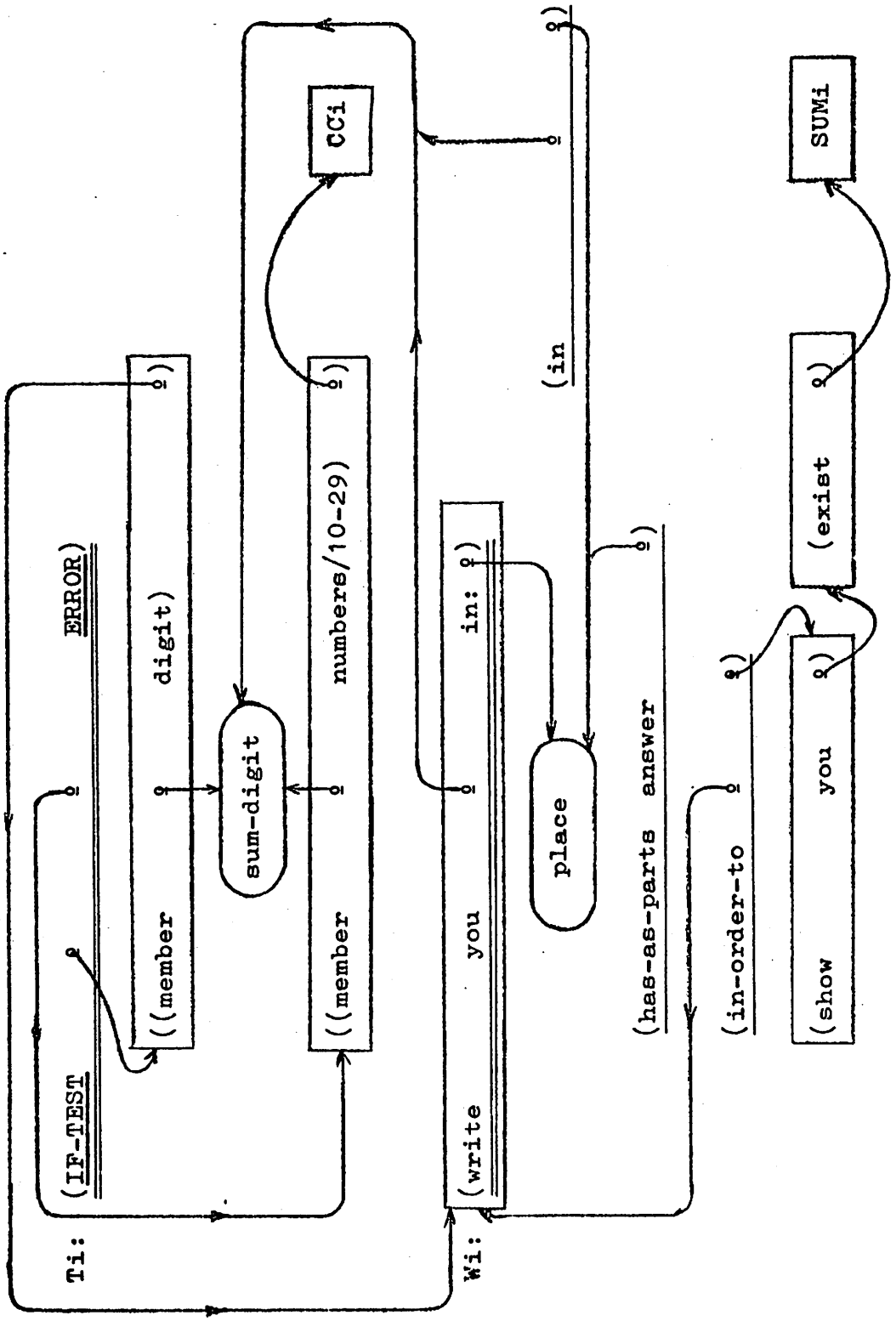
Why is 1 written in the hundreds' place of the answer?

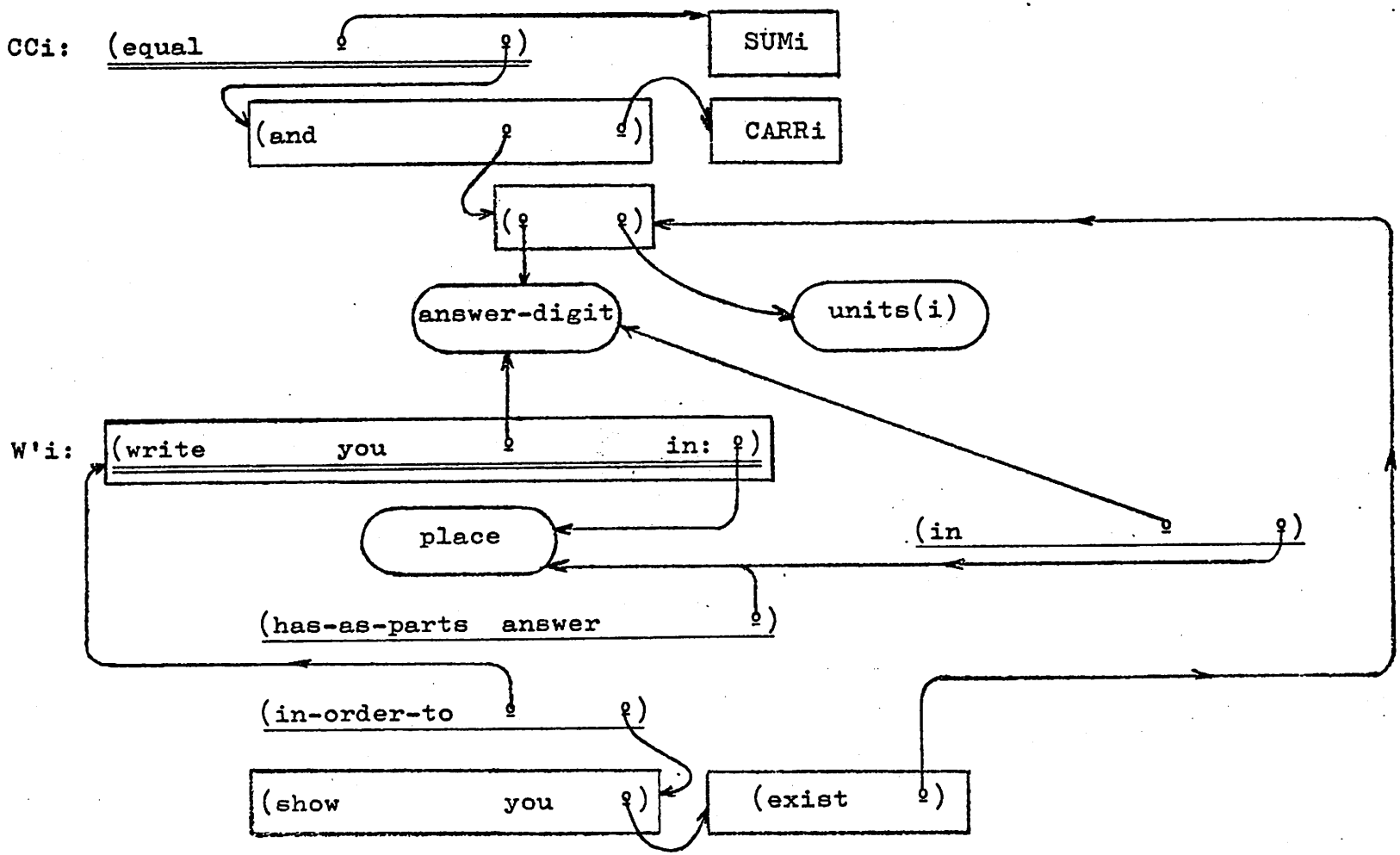
$23+46+85=k.$

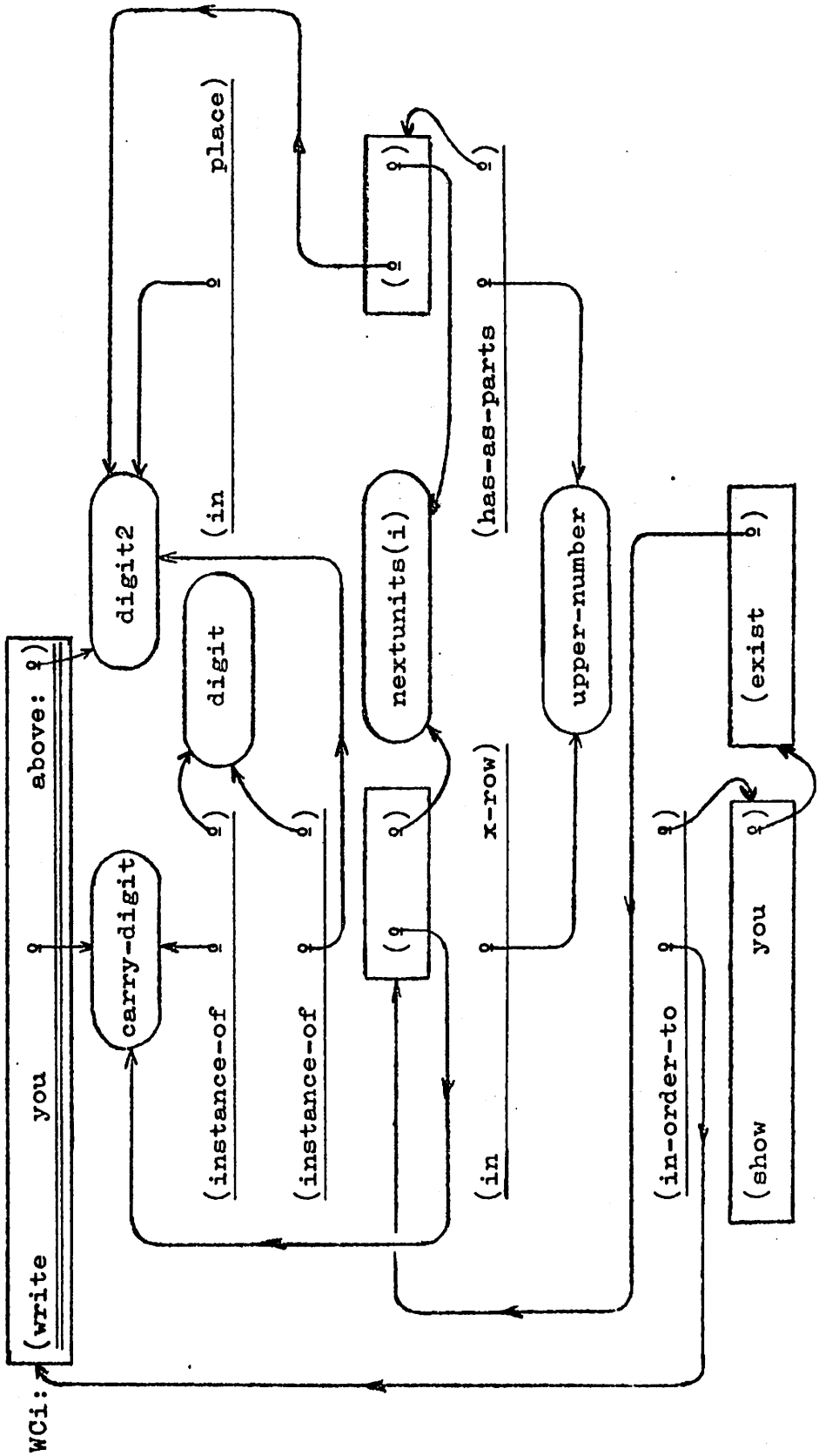
Altogether the boys sold ■ tickets.


i = 1
initialize
Ai
add digits
Ti
test sum
ERROR
DONE
Wi
write answer digit
CCi
compute carry
W'i
write answer digit
WCi
write carry
i = successor(i)
go to next column


Ai:
(add you)
(ALL)
digit1
units(i)
digit
(instance-of)
(has-as-parts)
(==>)
number1
integer
(in)
(instance-of)
(instance-of)
row1
(contains frame)
rows
(result-of)
(instance-of)
SUMi:
sum-digit
(unless-or-else DONE)
(exist)


Ti:
(IF-TEST
ERROR)
((member
digit)
sum-digit
((member
numbers/10-29)
CCi
Wi:
(write
you
in:)
place
(has-as-parts answer
(in-order-to
(in
(exist
SUMi
(show
you


CCi:
(equal º º)
SUMi
(and º º)
CARRi
(º º)
answer-digit
units(i)
W'i:
(write you º in: º)
place
(in º º)
(has-as-parts answer º)
(in-order-to º º)
(show you º)
(exist º)


WCi:
(write you ♀ above: ♀)
digit2
(in ♀ place)
carry-digit
(instance-of ♀)
digit
(instance-of ♀)
(♀ ♀)
nextunits(i)
(♀ ♀)
(has-as-parts ♀)
(in ♀ x-row)
upper-number
(in-order-to ♀ ♀)
(exist ♀)
(show you ♀)

Throughout the description which follows, 'units(i)' will refer to a special node whose value varies with i: units(1)="ones", units(2)="tens", etc. Similarly for 'nextunits(i)' which is the same as 'units(i+1).'

GENERAL-ADD ALGORITHM:

[For i = 1, 2, 3, 4:

1. Test the existence of 'units(i),' which are digits, and which are part-of numbers written in the operand rows of the picture.
   If not present, we are DONE, the result is the number in the answer row.
   If present, add them, obtaining SUMi units(i).

2. If SUMi is a digit ($0 \leqslant SUMi \leqslant 9$), write it in the units(i)' place of the answer in-order-to show that there exist SUMi units(i).

3. If SUMi is an integer between 10 and 29:
   a. SUMi units(i) equal CARRi = Tens-of(SUMi) and Ones-of(SUMi) units(i).
      Write this last digit in the units(i)' place of the answer in-order-to show that there exist that many units(i).
   b. Write CARRi above the digit in the nextunits(i)' place of the upper-number to show that there exists that many more nextunits(i).]

Extensive processing was necessary to generate the induced algorithm from a set of examples. In fact, the program is written in fairly distinct modules. Each of these will be discussed in detail in later chapters. Chapter 2 describes the syntax analyzer, which is a transformational bottom-up recognizer using a simple base-component and rather complex transformations. The semantic analysis is built around a Quillian-like memory network [Quillian 1966] which is described in Chapter 3. In this same chapter, the "Sentence-Matching" scheme is applied to various aspects of sentence analysis, starting with the most elementary sentence type, and building up more complex forms. Chapter 4 takes up at the next level of discourse analysis: within each example-story, sentences may have anaphoric references; questions may be asked; each story may have its own emphasis, but the final goal imposes contextual selectivity. Eventually, inductive learning merges the various stories together.

Finally, Chapter 5 discusses the results obtained, points out limitations and suggests extensions for future research. This is done in the light of a certain number of evaluation criteria which are discussed in the remainder of this chapter, where they are also applied towards a critical review of previous research in the field.

1.2 BACKGROUND.

Before attempting an evaluation of previous research, it seems necessary to emphasize the difficulties involved in the evaluation process. There would be few difficulties, of course, if there were a recognized linear ordering of intelligence problems: each system would then presumably improve on the previous ones by solving difficulties of a higher level (or solving the same ones more elegantly) There is nothing like the Chomsky hierarchy of grammars [Chomsky 1959] for semantics or problem solving, and it is not clear whether it will ever be possible to define one. Thus any evaluation relies on the belief that we, humans, have an innate or educated feeling for these levels of semantic complexity, even though we cannot

express them. The early history of Artificial Intelligence would seem to contradict such a belief. Successes (e.g., chess), and difficulties (e.g., automatic translation, children stories) were not the expected ones. Even today, these matters are highly controversial. Each person has his own conception of what is important, where the difficulties lie, in which direction to work, etc. One can only hope to share most of one's hypotheses with others.

A by-product of this difficulty in measurement is the fact that no one can state exactly the capabilities and limitations of a system. Thus, samples of performance are shown, with the hope that the reader can infer the subset of linguistic/thought processes which are handled. Such a procedure depends heavily on how clearly the distinction is drawn between those aspects of the samples that are actually treated and those that are not.

Expectedly, the following set of criteria is also representative of our bias:

a. Syntax. Systems are expected to perform more analysis than simple keyword look-up.

b. Semantic processing. By this, we refer to the often ignored level of study dealing with basic sentence analysis (semantic, not syntactic), influence of previous knowledge on the understanding process, anaphoric references, context, and discourse organization.

c. Semantic structure. This is the domain of operation of the semantic processing mentioned in (b). Of particular importance here are the amount of information gained through the structural organization itself, and the uniformity of representation of old and new information (e.g., in view of feedback.)

d. Deduction. Logical complexity of the problems that can be handled.

e. Inductive inference. Induction through understanding. Influence of structural organization. Specifically, learning new actions and incorporating them in the old repertoire.

After a short historical introduction, the various systems will be reviewed on the basis of the above criteria. The chapter will then conclude with a

brief summary outlining the good points that inspired our approach and the deficiencies we attempted to overcome.

After the earlier systems to be discussed below, there was a growing awareness among researchers of the importance of extensive linguistic analysis. The timing is quite significant here: the publication of Chomsky's "Aspects" in 1965 [Chomsky 1965] is an important milestone. Wide agreement appeared over the necessity of introducing elaborate syntactic and semantic components. One essential point is made: "understanding" is nothing more than restructuring the input into some internal representation (Chomsky's "deep structures") which can be manipulated for the purpose of answering questions, making inferences, etc. The main controversy is over specifying how "deep" one ought to go, what is the precise nature of those deep structures, and of the accompanying analysis procedures which map input strings into these structures. After 1965, several theses seem to indicate major splits: in particular, [Quillian 1966], [Coles 1967], and [Woods 1968]. Each of these seems to lead to an independent line of research with sharply differing viewpoints. Thus appear the 'structuralists,' the 'logicians,' and the 'proceduralists.' Each side is critically reviewed in turn below. Meanwhile, the reader may have been surprised at the importance given to linguistics. This is partly due to the fact that this thesis is centered on learning from an English text. But it also reveals our belief that a good structural description and understanding of the material are central to the learning process. The literature on learning systems is finally reviewed in section 1.2.5.

1.2.1 Early Systems.

These are cited essentially for their historical interest: the very early BASEBALL system [Green & al. 1961], Lindsay's SAD-SAM about family relationships [Lindsay 1964], Raphael's SIR question-answering system [Raphael 1964] (see below 1.2.3), and finally Bobrow's high school STUDENT program [Bobrow 1964]. Somewhat later but really belonging to this same broad category are Weizenbaum's simulated dialog with a psychiatrist: ELIZA [Weizenbaum 1966] and Simmons' text based indexed system PROTOSYNTHEX I [Simmons 1966]. All of these systems have been analyzed and criticized at length in

the literature. They all perform an extremely limited, ad-hoc analysis of the input.

1.2.2 Structural Systems.

Several systems belong to this category: Simmons' revised versions of PROTOSYNTHEX [Simmons 1968], Thompson's DEACON [Thompson 1966], Kellogg's CONVERSE [Kellogg 1968], Quillian's Teachable Language Comprehender, a continuation of his original thesis [Quillian 1966, 1969], and finally, Schank's "conceptual" parsing theory [Schank 1970, 1971].

All of these systems have one common feature: they place emphasis upon the internal representation into which English input is to be transformed. Information is thus organized in the frame of a network or "semantic memory." Differences between the various systems are relatively unimportant. Their advantage over many later systems is precisely this common emphasis on a problem which has blocked the others. It is significant in this respect to notice the ever wider acceptance of the necessity of good structural descriptions (see, e.g., [Minsky 1969], [Winston 1970].) On the whole, structural systems seem best prepared to deal with really general semantic problems. They have the important advantage over more formal systems of preserving the natural ambiguity and low specificity of human languages. They also integrate new information with background knowledge very naturally.

One difficulty in the objective evaluation of structurally oriented systems is the small power actually implemented in them. Their inferential component is very limited. Their results often appear unimpressive. Of course, power cannot be the only criterion: Bobrow's STUDENT could solve high school algebra problems that SIR could not. Was that the point? At least now, we can confidently say: No. On the other hand, it is true that none of these systems has been pursued to the point of convincingly showing the advantages underlying their scheme. Indeed, Quillian's original system had no syntax to speak of. He himself, and very recently a different group [McCalla & al. 1972], have attempted to introduce a more elaborate syntactic component. Nevertheless, the semantic processes implemented remain rudimentary.

Schank's approach is, in our opinion, among the most promising today. Indeed, our thesis shares with it much of its basic philosophy. Schank illustrates his ideas with some examples that are much beyond the scope of other systems. (Of course, one may question his system's ability at handling examples that are readily accepted by others.) He shows most clearly how a semantic store can be used to introduce the pragmatic idea of "normality" as part of the normal parsing procedure. Moreover, he points out the advantages of "predictive" analysis: the input is essentially "expected" instead of being passively received for analysis. On the other hand, in spite of its many advantages, this system can be criticized along several lines. In particular, the author draws no clear distinction between what is already accomplished and what still belongs to the domain of expectations. Criticisms of the analysis on the sentence level would require long discussions. Overall however, the system seems to lack structural flexibility for effective discourse processing. In fact, the organization of the semantic memory has not been focused upon until very recently [Schank 1972]. Processing of connected text and inductive inference are beyond its scope.

Within their restricted claims, structural systems have introduced many essential ideas. We believe that these constitute excellent starting elements. But they need to be integrated into a more complete framework.

1.2.3 Logic-Based Systems.

Raphael's thesis (mentioned above) could be better discussed here: in spite of its deficiencies, it had many interesting features. It in fact paved the way to the 'logic viewpoint.' On the other hand, Coles' system [Coles 1967, 1968, 1969] was the first one to perform elaborate syntactic analysis. Furthermore, it featured direct interactions between the understanding process and a "real-word" context provided by a pictorial scene on a CRT screen. Here, what is being talked about is represented internally in terms of fully quantified predicate calculus. Previous knowledge is represented as axioms, new information is a theorem to be proved or disproved. Sentence acceptance or rejection becomes a purely formalized inferential problem. Green uses this same approach with minor variations [Green & al. 1968, 1969].

Logic-based systems make use of Robinson's resolution algorithm [Robinson 1965] which is a "complete uniform proof procedure" for the first-order predicate calculus. An interesting discussion appears in Winograd ([Winograd 1971], pp. 231-232). One inconvenience is the impossibility of directing the proof procedure. Thus, such systems are quite successful as long as the universe of discourse is kept within limits. They become terribly inefficient when dealing with everyday contexts which are extremely varied but involve little of the heavy apparatus required for theorem-proving.

For example, elementary quantification is quite frequent in ordinary discourse (it appears in every sentence in the inflection of nouns and verbs). Humans get by rather well with it, but they might find it very hard to follow an argument relying on a subtle interplay of mixed quantifiers. Similarly, theorem proving helps in solving problems with many levels of "indirectness" in the pursuit of a goal. Again this is desirable, specially if the system is to behave with some insight into long term consequences of various courses of action. It is also true that humans have obvious deficiencies in this respect. But on the whole, no system behaves quite as intelligently as a human does.

Thus, an elaborate proof procedure can provide enormous help towards an intelligent system. It can and should participate in semantic analysis itself. But it cannot replace some of the basic linguistic processes. Significantly, Coles himself concludes by suggesting the need for more structuring of semantic information in the vein of Quillian's work ([Coles 1967], pp. 111-113) and has already moved somewhat in that direction ([Coles 1972a, b]).

1.2.4 Procedurally Oriented Systems.

Somewhat later, two systems appeared which were organized around a procedure-based scheme: information was actually represented by procedures ([Woods 1968], [Winograd 1971]). This has one main advantage: each procedure has potentially the power of a Turing Machine. In order to handle some of the more complex information processes, such power was often needed in

other systems. Instead of superimposing somewhat ad-hoc procedures which do not always fit the original model, the theory here takes general procedures as its basis.

As such, the flexibility of the procedural framework can be criticized as slightly deceiving: organizing one's information into neatly conceived, separate procedures is only possible for very restricted domains. The claim here is that one could apply Winograd's own arguments about syntax to semantics: He concedes that grammars cannot remain "perspicuous" when the English subset becomes substantially larger ([Winograd 1971], p. 203). By analogy, it would seem that the procedural organization would gradually lose its simplicity as the universe of discourse increases in structural complexity.

The procedural framework seems less important than other contributions of these systems: (1) several components actually researched and implemented, (2) introduction of general principles used in writing the procedures. Thus, Winograd's system touched on a number of previously unexplored problems: discourse semantics, language generation, etc. On the other hand, the deductive process is modified in an information dependent manner, thus avoiding the uniformity problem of the logic-based systems.

One difficulty in evaluation is that the Turing-Machine-power argument can always be applied: any algorithm (past or future) can be considered as a sub-component within the procedural framework. This merely postpones facing the problem! On the other hand, procedural approaches have tended to have the same kind of deficiencies as logic-based systems. In essence, they misunderstand basic semantics. It is as if syntax was followed by inference with nothing in between. Such approaches would probably misplace the responsibility of finding the difference between the following sentences:

"How many pounds did John weigh?" (1)

"How many packages did John weigh?" (2)

In a strict sense, this is neither syntax, nor inference. The distinction is purely semantic. Moreover, some systems might simply ignore such differences.

Several such basic problems of semantics are treated very lightly, if at all: e.g., semantically meaningful noun-verb relationships ([Chafe 1970]), recovering underlying information by association or

otherwise, and in general, integrating old and new information into a uniform framework. The main point is that higher level processes such as discourse analysis, learning by induction, even deductive reasoning will remain limited as long as the elementary pieces have not been worked out more deeply. One cannot keep building on shaky ground.

1.2.5 Inductive and "Learning" Systems.

"Learning" has long been a focus for research: its solution has the fascinating appeal of providing the basis for bootstrapping computers into higher levels of intelligence. The task is, however, very difficult and easily misunderstood. Essentially two main lines of approach can be distinguished, both of which are examined below.

The so-called "adaptive" or "self-organizing" systems perform induction as a result of repeated elementary experiments in a trial-and-error fashion. The learning problem is then reduced to the mathematical convergence of some vector. The most famous example of this family of devices is the "perceptron." While it is clear that for all learning processes, one must specify some sort of convergence, it rapidly appeared that this aspect of the problem was rather secondary compared with the semantic problem of description underlying it. Thus, we will not study this type of device in detail; instead, we refer the reader to the excellent study of Perceptrons by Minsky and Papert [Minsky & al. 1969], which also includes an extensive annotated bibliography on the subject.

More semantically oriented systems appeared later. Undoubtedly, researchers were faced with the fact that learning is closely associated with understanding. If we are to take Webster's definition: "to learn is to gain knowledge or understanding of or skill in by study, instruction or experience." (underlining mine) Among the first papers, one must cite McCarthy's "The advice-taker" and "Situations, actions, and causal laws" [McCarthy 1959, 1963]. These essentially emphasized the importance of the problem and directed it away from the blind perceptron approach. Later, induction was studied in more detail for restricted subject areas by Evans for his geometric analogy problems [Evans 1963]. Some years later, several papers were presented at the International Joint Conference on Artificial Intelligence in

Washington, D.C. (1969). Among these were Colby's simulation of belief systems [Colby & al. 1968, 1969a, 1969b] and Abelson's approach to the pragmatic analysis of situations [Abelson 1969]. Becker takes up the problem of analogy again [Becker 1969] but attacks it within a more general context than the others. His approach seems most interesting, but it suffers again from the lack of structure in the organization of information. Thus, assume that induction is to be performed over sentences of the following form:

"Write 3 in the ones' place of the answer." (3)

where the "3" appears successively as any of the 10 digits depending upon the particular example. Becker's program would eventually replace the varying element by a "dummy variable" x which is left completely unspecified: it could be a digit, a number, a house, or just any noun. This great loss of information places obvious limitations on such a system.

Most recently, Winston published a thesis on "learning structural descriptions from examples" [Winston 1970]. His system constructs elementary concepts such as "house," "arch," from visual input of scenes composed of cubes, wedges, etc. One paradigm (which was already latent in previous work) is strongly emphasized: "Good descriptive methods are of central importance in this work" (p. 6). Thus a great deal of the study is devoted to the analysis of visual scenes, bringing out the "important" relations, such as "left-of," "on-top-of," "big." Clearly, there are difficult problems associated with the derivation of such abstract relations from a non-structured digitized array representing a picture. The main criticism is that the establishment of these relations, the order in which they are examined and their exact nature are <u>within</u> the program. One might not see much difference between this approach and one that consists of specifying these relations as program interpreted data. The advantage in the latter approach is that it leads naturally toward a self-modifying program. Winston recognizes the lack of feedback in his overall flow-chart (p. 252). But, there is no uniform notation for old and learned concepts: this is exactly the reason for the impossibility of getting feedback. Absence of feedback precludes bootstrapping, and again limits the system's capabilities to the restricted, original universe of study.

On a different level, Winston seems to argue that there is no major difference between "learning to do" and "learning to recognize" (p. 126). One would certainly agree that both can be "understood in terms of processes that construct and manipulate

descriptions." But the exact specifications of these manipulations can be quite different depending on the goal pursued. Hence, problems of "node induction" (of the kind mentioned above in connection with Becker's system) and inferring the appropriate flow of control from examples of algorithm execution hardly bear any relationship to each other in practice (see Chapter 4.)

Finally, one should mention the recent efforts developed at S.R.I. to incorporate a learning component within their robot system [Fikes & al. 1971, 1972]. The learning part proper is still rather elementary: mostly transformation of constants into variable parameters. However, its smooth interaction with the rest of the system is quite impressive. After having solved a certain problem once, the system is able to recognize that a new problem is sufficiently similar, and solve it by analogy with a great saving in computing time. Thus, it demonstrates quite well the impact one may expect from the introduction of learning. Presently, the learning component is being redesigned to take advantage of the new QA4 theorem-proving system. It will certainly be of interest to see the new developments that this new environment will allow.

1.2.6 Summary.

Here is a brief summary of the preceding review. The criteria defined at the beginning of section 1.2 will be here referred to as (a) to (e). Except for the earlier systems, the syntactic component (a) has sufficient power to deal at least with the sentences on hand. Thus all approaches seem equally defensible. The structuralists emphasized the organization of semantic memory (c), while the logicians stressed deductive inference (d). Procedure-oriented systems had substantially improved semantic processing (b) and deductive (d) components. "Learning" systems featured inductive schemes (e) with varied levels of elaboration. Both of the last two groups lacked uniform structural organization (c). In conjunction with their other limitations, this seriously impaired their ability to generalize to unrestricted subjects and profit from acquired knowledge.

Let us finally note that it was not our intention to be exhaustive in this survey, but rather

to bring out the salient points relevant to this thesis. For a broad overview of progress in Computational Linguistics, the reader must be referred to the two surveys by Simmons [Simmons 1965, 1970].

CHAPTER 2
SYNTACTIC ANALYSIS

The syntactic module is similar to a transformational analyzer [Chomsky 1957, 1965] in its overall organization. On the other hand, it bears close resemblance to the more recent Transition Network Grammars [Woods 1970] in its emphasis on analysis, rather than synthesis of language. The original formulation of transformational syntax was basically generative. One important consequence is that transformations are applied to structures which have been fully developed through the base component grammar. Attempting to apply transformations to the original input string presents major difficulties. The input has no obvious structure, hence no information with which to guide the application of transformations. These, in turn become overly complicated or simply insufficient. Using the generative grammar with little modification for top-down parsing leads, however, to a "combinatorial explosion" (see [Petrick 1965], [Woods 1970]). Faced with this and other problems, researchers have come to admit that "grammars will not be as perspicuous as we might hope" ([Winograd 1971], p. 203). Indeed, our study of coordination (see later in this Chapter, section 2.4) is a good example of the complexities that arise: deletion rules (identity reduction) interfere with this transformation, making the analysis much more difficult than the generative approach seems to have led some to believe ([Petrick & al. 1969], [Winograd 1971], p. 204).

The output expected from the syntactic module is just a rough grammatical surface structure organized into a sentence tree, handed down to subsequent modules for further processing. This may seem surprising at first, but it is essentially due to the existence of certain problems, such as mixed quantification and pronominalization, that require the surface structure to be kept almost intact. Within the syntactic component itself, several reasons (e.g., coordination transformation) lead to similar requirements. In particular, the original word order is needed for later processing (this includes passive inversions and

others). Thus, at best one would need to carry both surface and deep structures in parallel till the end. Furthermore, given the present embryonic state of linguistic research, it is conceptually clearer to separate syntax from semantics as much as possible. (This subject is interestingly discussed in [Coles 1967], pp. 29-33.) In particular, the correct 'attachment' of prepositional phrases, which requires semantic analysis, is simply postponed by using uncommitted structures; these are then further analyzed by the next module (see next chapter, section 3.3). Incidentally, this reduces the number of possible constructions considerably, and greatly contributes to the simplicity of the syntax module.

In this system, the analysis is handled by three fairly distinct components: an initial phase of minor transformations at the lexical level, a formal base grammar, and a transformational component, all interacting under the supervision of a top-level 'executive' responsible for the distribution of the work, and, in general, of the major decision-making.

Mnemonics used in this chapter for syntactic categories are relatively common. For more details, see Appendix 4 on the base grammar.

2.1 INITIAL PHASE.

Each sentence is first submitted to a set of initial transformations which analyze each word and attach to it syntactic categories and "features" as appropriate. The word is looked up in a dictionary:

a) If it is there, we retrieve the alternative parts of speech that can be assigned to it. Most words have only one assignment, but some homographs may have as many as 5: e.g., "left" may be the past or past participle of the verb "leave", it can also be an adjective, an adverb, or a noun, as shown in the following examples:

"John had left his house already." (1)
"He only had two dollars left in his pocket." (2)
"Find the first left parenthesis." (3)
"Turn left." (4)
"The conservatives united against the left." (5)

b) If the word is not listed as such in the dictionary, a few inflectional transformations are tried until a root is found which is listed in the dictionary:

- the word may start with a capital letter simply because it is at the beginning of the sentence.
- it may be composite (e.g. "ice-skating") in which case the group of words gets the same syntactic category(-ies) as its last word.
- it may be the inflected form of a noun or a verb. All of the simpler (algorithmic) transformations are implemented in this fashion taking advantage of the Snobol pattern matching and replacement facilities. Thus,

```
WORD    'IES' RPOS(0)  =  'Y'
```

transforms "candies" into "candy."

- etc.

Details will not be mentioned here: the dictionary is described in detail in Appendix 2 and initial transformations appear in Appendix 3. A few notes, however, are in order at this point:

- The validity of the transformation is not completely checked. In many cases, it was felt that nothing more could be learned by being more exhaustive. More importantly, it seems sometimes pointless to reject words when the intended inflection is obvious; e.g., "candys," "candies," "wraping," "wrapping," ... will all be accepted.
- When the inflection is irregular, in the sense that there is no corresponding rule of any generality, all different forms of the word will be listed in the dictionary: e.g., "buy", "bought"; "child", "children".
- In some cases, inflections may help with homography. This does not happen with s-type inflections: "plants, flies, ..." are still noun-verb mixes, but "planted, flying, ..." can only be verbs.

c) If all inflectional transformations fail, the word is taken by default to be a proper name and parsing continues. This is purely for the sake of programming completeness. Difficulties arise precisely when the text being analyzed is attempting to make the program learn a new word (otherwise, the concern is of no use in practice). This problem was not studied to any extent. Without underestimating the importance of this matter, we simply chose to concentrate on some of the many other problems of language. An approach to the syntactic problems involved is the use of morphology to determine the likely syntactic categories of the unknown word. e.g.,

[adjective] 'ness' : Noun
---- 'age' : usually Noun
---- 'able' : Adjective

This is the approach usually taken by projects attempting the use of limited dictionaries.

d) A few words will trigger special transformations corresponding to idiomatic peculiarities of English which are best handled during this initial phase.
e.g., if "all" is preceded by "in" and not followed by an adjective or a common noun, we recognize the adverbial idiom "in all" meaning "altogether" as in: "In all Jane and Mary collected 15 butterflies." (6)

e) Finally, global transformations are sometimes 'prepared' at this stage. If the sentence is an interrogative (ends with "?"), the parser described in the next section will be 'warned' by setting a special flag. Another flag will signal the presence of coordination conjunctions in the sentence. These flags are also used by the executive in deciding which particular constructs are allowable and thus which transformations should be made possible.

2.2 BASE COMPONENT.

This is basically a bottom-to-top parser corresponding to a very simple SLR(1) grammar. It is hard to over-stress the simplicity of grammar that one can achieve by surrounding it with the appropriate components. Of course, one could put all the burden on the peripheral components; the point is that simplicity can be achieved while keeping every component very 'natural'. SLR(1) languages stand as follows in the scale of complexity:
[finite-state - most of programming languages - SLR(1) - LR(1) - LR(k) - Context-free - Context sensitive - etc.]
As may be expected, these grammars allow fast parsings. On the other hand, the existence of an efficient compiler-compiler makes it quite easy to modify them: this partly answers the traditional dilemma between fast bottom-to-top and easily modifiable (modular) top-down parsing. (It may interest the reader to know that an operational parser for a 60-production grammar takes 5.5 seconds of CDC6400 CPU-time to be produced.) More details about these grammars and the parsing

algorithm appear in the original papers by Earley [Earley 1969, 1970]). The specific grammar used appears in Appendix 4. It includes 63 productions, 17 terminal symbols, and 27 non-terminals.

Of course, SLR(1) languages are non-ambiguous, but this turned out to be unimportant: The only true cases of structural ambiguity were due to higher level processes such as complex transformations. One should point out here a rather common confusion: some ambiguities are purely semantic, as, for example, the different meanings of the preposition "of." These need not concern us here insofar as they do not affect the parse structure. (Note here another reason to separate syntax from semantics.) Hence, the base component was designed to yield the single most likely parsing of each phrase. One immediate consequence is that homographs must be completely resolved, i.e., reduced to a unique syntactic category, as soon as the scanner reaches them during its unique left-to-right pass. This problem is handled according to the following rules:

a) Eliminate those syntactic categories that are not acceptable at this particular stage of parsing (making allowances for non-standard constructions whenever special transformation flags are set). For example,
"The plant is beautiful." : After "the", a noun/verb homograph such as "plant" must be in its noun form.
b) Make use of some simple 'context-sensitive' rules for further reduction. These are precisely the ad-hoc rules that specify a certain 'most-likely' choice even when several categories would be compatible with the state of the parser at that point. For example,
An adjective/adverb combination (e.g. "last") will be considered an adjective if followed by an adjective, noun, or adjective/noun, and as an adverb otherwise:
"Last week, I was at home." (7)
"What is the last funny movie in town?" (8)
"What do you add last?" (9)

c) The verb and auxiliaries are also handled at this stage by a set of functions regrouped under the heading "verb transformation." This takes care of all variations in verb inflection, combined with the possible presence of adverbs interspersed in the middle. Note that this could not be made during the initial phase. Hence, consider:
"How many pennies did you have left in your pocket?" (10)

If "have left" had been initially reduced to a simple verb, there would be no way to separate the individual

elements and recover the pairing "did-have" which is obvious when (10) is presented in its declarative form:

"You had so-many pennies left in your pocket." (11)

This example will be explained in more detail with the interrogative transformation.

Finally, the parser will jump to a specified 'semantic routine' (this designation is unfortunately standard in the literature) each time a production is recognized. This is where the strategy employed resembles most the one used by Woods: after the failure of many attempts at neatly formalizing the syntax of natural languages, it seems inevitable to get to the idea of surrounding a simple base by a set of free format modules (in [Woods 1970], any Lisp function; similarly here with Snobol). In our case, these routines essentially fulfill the role of checking the various feature agreement rules:

- number and person of subject and verb.
- case of pronouns (e.g. "he" vs. "him")
- verb construction (number of complements, etc.)
- semantic features of verb and related nouns, of nouns and attached relative pronouns, etc.

(Only a partial implementation of the last two points was attempted within the syntactic component; see 2.5)

Here is an example of application of the subject-verb agreement rules:

"sheep": number=S/P, animate=+, ...
"were running": past, prog., number-person=S2/P,
subject:animate=+; number-of-compl.=0, ...
"The sheep were running": successful agreement,
features get further specified as follows:
"sheep": number=P, ...
"were running": number-person=P3, ...

2.3 BASIC TRANSFORMATIONS.

Basic transformations, together with coordination, constitute the transformational component. Because of the particular bottom-to-top environment of this parser, their formal description presents difficulties which are discussed below. However, informal presentation can easily convey the basic idea behind each of them.

2.3.1 Interrogative Transformations.

When the special flag is set for interrogatives, it may still be that the sentence does not require a transformation, as in:

"How many pine cones were left in the basket?" (12)

This is so whenever the unknown element involves the subject of the main verb in the sentence. In other cases, inversions do occur and they can be classified in two main categories. The corresponding transformations are described below in a notation inspired by Snobol. Exclamation marks ("!") separate alternatives, and the period (".") indicates the assignment of whatever part of the sentence matched the pattern on the left side to the variable on the right. '' denotes the empty string and "ADVPS" a string of adverbial phrases or subordinate clauses:

```
IT1:  ((ADVPS ! '')) . X1  (Cop . X2)  (NP . X3)
        ((Adj ! NP) . X4)  (Remainder . X5)  -->
        X1   X3   X2   X4   X5
```

e.g.,

"In any case, are you happy now?" (13)

```
 ------------ --- --- ----- ---
       1       2   3    4    5
```

yields:

"In any case, you are happy now." (14)

```
IT2:  ((ADVPS ! '') . X1)  (NP . X2)  (Aux . X3)
        (NP . X4)  (V . X5)  (Remainder . X6)  -->
        X1  X4  VT(X3 X5)  X2  LOVT(X3 X5)  X6
```

(VT(x) denotes an analysis of the string x in an attempt to extract a verb, the leftover from the string yielding LOVT(x).) e.g.,

"How many puppets has she been selling in the last two quarters?" (15a)

yields:

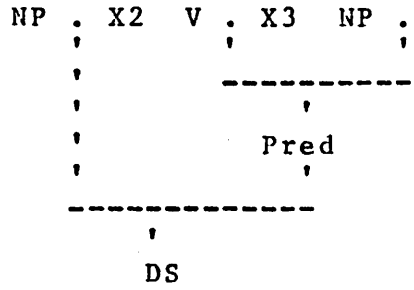
"She has been selling how many puppets in the last two quarters." (15b)

IT3: ((ADVPS ! '') . X1) (Aux . X3) (NP . X4)
(V . X5) (Remainder . X6) -->
X1 X4 VT(X3 X5) X6
(Here LOVT(X3 X5) must be empty.)
e.g.,
"Has he been sick?" (16a)
yields:
"He has been sick?" (16b)

2.3.2 Formulation of Transformations.

However, the above notations may be misleading in their implication of how the analysis actually takes place. Implementation of transformations in the particular environment of a bottom-to-top analyzer needs special formulation. IT2 was chosen as an example to demonstrate the mechanism in some detail, in the case where X1 is an empty string, as for (10) above:

a) Deciding for the application of IT2:



So far, the parser has found a "declarative sentence" DS (where "subject" and "verb" do not necessarily agree) and hits a word or string of words recognized as a verbal form. The following conditions must hold (it is understood that a record is kept of the current level of recursion to avoid confusion between main clauses and sub-clauses):
- the interrogative flag is set (presence of "?").
- there is only one "complement" to the verb: X4.
- the verb was in fact just an auxiliary or a modal: X3.
- X3 can be the start of a verb completed by at least some of the words in X5.

b) The transformation itself is a straightforward tree manipulation except for the reconstitution of the verb: VT(X3 X5). Checks are made to insure the agreement of the new elements of the declarative sentence.

c) Analysis is resumed with the parser scanning the left-over from X5, if any, or X6. The state of the parser is reset to correspond to finishing the analysis of the noun-phrase X2 as a complement of the new verb in the new declarative which is expected.

The description of the state of the parser can be made much more specific and rigorous by indicating the position of the scanner (indicated below by "#") relative to the various productions of the grammar which are involved at the time. Thus for example, in our grammar, (c) above would be:

```
[S]    ::=  ([Advps] ! '') # [DS]
[DS]   ::=  [NP] # [Pred]
[Pred] ::=  [V] # [NP]
[NP]   ::=  [SNP] # ('' ! [Defadj])
```

where SNP is a simple noun-phrase (e.g., "five pennies") and Defadj is what we call a deferred adjectival phrase (like "left in your pocket").
This method certainly improves the formalism, but it is even more grammar-dependent. Thus, as long as one remains vague, one retains the advantage of being able to call on the natural grammar shared by all native speakers of the language.

2.3.3 Other Transformations.

Several other transformations deal with the following:
- Non-standard nominal adverbial phrases (e.g. "last week", "home", etc.)
- "That" deletion (e.g. "I know [that] the man will go.")
- Referent deletion (e.g. "Ed had 3 candies. Sue gave him 5 more [candies].") In this case, the syntax phase will simply insert a dummy noun which will be fully reconstructed later, while analyzing references.

These are all described in more detail in Appendix 5.

2.4 COORDINATION.

The treatment of coordination is one of the most complex problems of language analysis. In particular, it is closely related to many aspects of syntax and semantics. So far, the approach taken by most linguists has been a generative approach of analysis by synthesis. Much of the recent linguistic work on the subject appears in [Reibel & al. 1969]. On the computational side, the most recent work ([Petrick & al. 1969], pp. 223-233) is also characteristic of this approach. We would like to present here an analytic approach to the problem. Even though, in theory, there should be no difference, it is our feeling that this inverse approach may shed more light on the problem.

2.4.1 Introduction.

It has been recognized that coordination occurs in essentially two modes, sentential and phrasal, as illustrated by the following examples:

a) Purely sentential:
"John and Mary know the answer." (17)
(17) has the underlying structures:
"John knows the answer." (17a)
"Mary knows the answer." (17b)
or, equivalently,
"Both John and Mary know the answer." (17c)
The following, however, would not be derived from (17) above:
"John and Mary together [only as a group] know the answer." (17d)

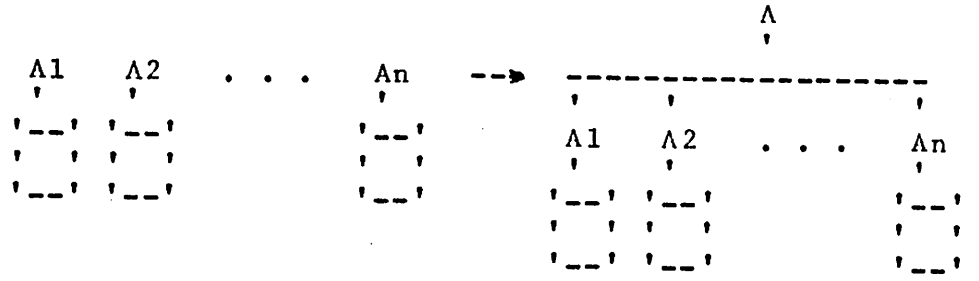
b) Purely phrasal (local):
"You are to find the sum of 23 and 19." (18)
where the coordination joins "23" and "19" into a numeric "simplex". Clearly, one could not derive the following underlying structures for (18):
"You are to find the sum of 23." (18a)
"You are to find the sum of 19." (18b)

c) Mixed, ambiguous mode (this is very common):
"Mary and John went to the movie." (19)
"Stones and bricks make strong walls." (20)
with the obvious ambiguity (phrasal vs. sentential).

We would like to emphasize the fact that in all cases, the coordination may be superficially (one might almost say syntactically) considered as phrasal or local. Indeed, this will now be stated as a principle:

"Syntax allows for only one mode of coordination, namely as simplexes. The actual nature of these can (only) be determined semantically."

The corresponding transformation can then be expressed as a first approximation as follows:



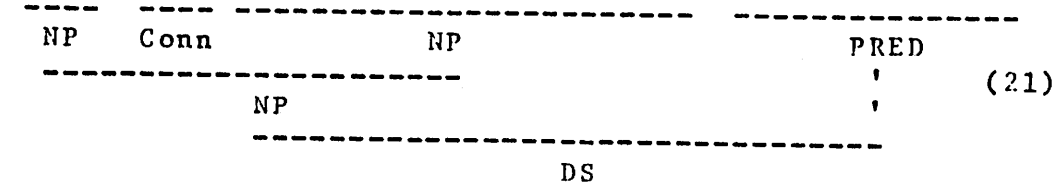
where A1 = A2 = ... = An = A (identity of roots) and all subtrees whose roots are the Ai's are identical (identity of subtrees).

This formulation corresponds closely to the analysis by synthesis approach. The refinements needed are discussed in the remainder of this section.

2.4.2 Basic Approach.

Coordination may occur with fairly different subtrees as demonstrated in the following examples:

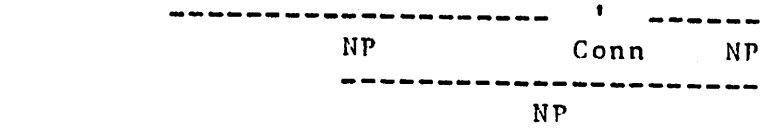
"John, like his pretty little sister, lived in Paris."



(21)

and

"There are 3 boxes of 10 shells and 11 shells."

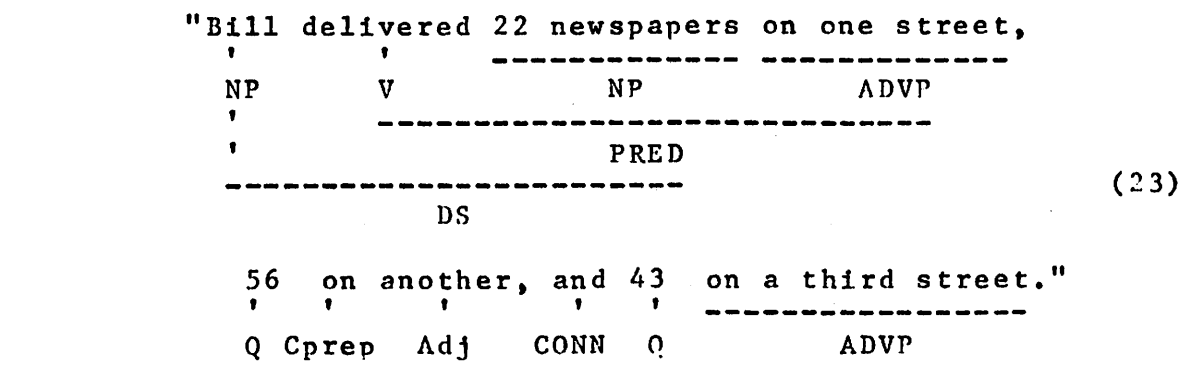


(22)

Thus what matters most in allowing coordination is the identity of roots of the coordinated subtrees rather

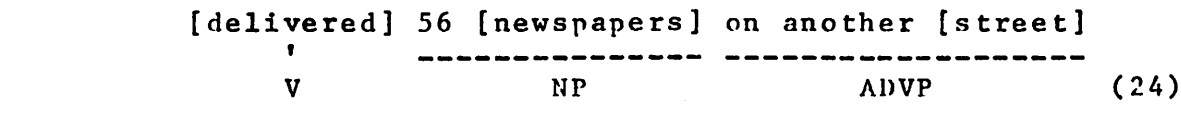
than the similarity of these subtrees.

More importantly, because of deletion phenomena, it may happen that the situation becomes unsymmetric. There may be a leading subtree which is fully structured and which will be called hereafter the 'Majtree', and on the other hand, pieces of a subtree ('Mintree') from which a few elements have been deleted by the coordination process. Of course, one Majtree may lead several Mintrees. An example will help to illustrate:

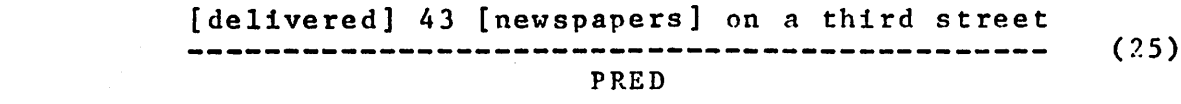


(23)

Here the Majtree is the predicate (PRED) of the complete declarative sentence on the top, the Mintrees do not appear in full in the surface structure. They are restored in (24) and (25) below:



(24)



(25)

Hence, the Mintrees will usually appear as a sequence of disconnected subtrees whose roots are in order BETA1, BETA2, ..., BETAp. (The roots BETAi's are singled out rather than the entire subtrees BETATi's for the reasons discussed above.)
e.g., in (8), p = 3, BETA1 = Q, BETA2 = Cprep, BETA3 = Adj; in (9), p = 2, BETA1 = Q, BETA2 = ADVP.

The problem of analyzing such sentences is first the determination of the Majtree and then the reconstruction of the Mintrees. The Majtree, whose root

will be called GAMMA, must normally include a set of subtrees whose roots (ALPHAi's) are such that: ALPHAi = BETAi for i = 1, 2, ..., p. The idea then is to traverse GAMMA and when a match ALPHAj is found for BETAj, to skip the subtree dominated by ALPHAj (which is assumed to correspond to BETATj) and start looking for a root ALPHA(j+1) to match BETA(j+1).

We will now present an algorithm to perform this search on a given tree GAMMA. We first need to define a function Prenextskip(T) which returns the node which would follow T in pre-order if the subtree dominated by T was not present. Prenextskip(T) fails if T dominates all the nodes which would appear after it in a normal pre-order traversal. Also, Prenext(T) returns the successor of T in pre-order and fails if none exists. These functions are trivial tree operations which are described here for the sake of completeness:

Prenext(T) = if T is not a leaf then the leftmost son of T else Prenextskip(T).

Prenextskip(T):

1. Set S := T.
2. If S is the original root GAMMA of the whole subtree being traversed, then return failure.
3. Else if S has a right sibling, then return this sibling.
4. Else set S := Father(S) and go to step 2.

The basic algorithm follows:

Algorithm A1:

1. Set PHI := GAMMA; j := 1.
2. If Root(PHI)=BETAj then go to step 4.
3. Else PHI := Prenext(PHI); if Prenext fails, then Return failure, else go to step 2.
4. If j=p, then (all BETA's matched) Return success.
5. Else j := j+1; PHI := Prenextskip(PHI); if Prenextskip fails, then Return failure, else go to step 2.

This basic algorithm leaves several problems

unsolved: choice of Majtree (i.e. choice of GAMMA) and validity of the coordination discovered, interpretation of primary failures, and reconstruction of the Mintree. These are discussed in the next two sub-sections.

2.4.3 Choice of Majtree and Validity Checking.

The choice of GAMMA depends essentially on a 'structural juxtaposition' rule which we will attempt to illustrate with examples:

"The clerk prepared 5 boxes of 10 candies and 3 candies." (26)

The basic algorithm (A1) applied with GAMMA as the root of the complete declarative sentence on the left would yield 3 possible matches, with p = 1, ALPHA1 = BETA1 = NP:

(the clerk) & (3 candies) (26a)
(5 boxes of 10 candies) & (3 candies) (26b)
(10 candies) & (3 candies) (26c)

It seems clear that even though (26b) is the only good interpretation, (26b) and (26c) can only be distinguished on semantic grounds. They are both syntactically correct, whereas (26a) simply does not sound grammatical. Similarly consider:

"John and Mary ate dinner at the restaurant." (27)

which would yield:

(John) & (Mary) (27a)
(John) & (dinner) (27b)
(John) & (the restaurant) (27c)

Again, (27a) is the only acceptable interpretation. This time, it is also the only syntactically valid coordination. Finally:

"The director and the manager of the company were arguing." (28)

might be structured as:

(the director) & (the manager of the company) (28a)

(e.g., the text might have mentioned a bank director coming to a company which is going bankrupt.)

or:

((the director) & (the manager)) of the company (28b)

but obviously not as:

(the director) & (the company) (28c)

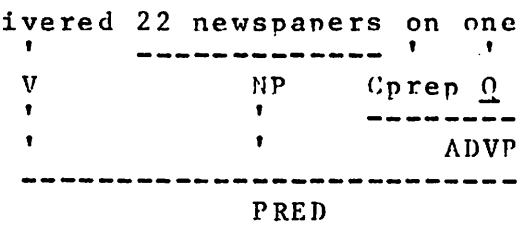
With the following definition:

"A coordination is called a <u>left</u> (<u>right</u>) coordination if the Mintrees appear on the left (right) of the Majtree; informally: if the coordination conjunction occurs before (after) the main verb.

the above examples seem to lead to a rule of the following form:

"In a left (right) coordination, a criterion of validity is that ALPHAl (ALPHAp), as defined earlier, should not have any left (right) sibling."

However, consideration of instances where coordination is accompanied by deletion phenomena shows that the rule above is only an approximation: it would apply exactly if the deleted words were restored. Hence, looking back at sentence (23):



(29)

Here p = 3, ALPHA3 = Q ("one" - this does match BETA3 = Adj as discussed later) and ALPHA3 has a right sibling. If "street" were restored in the Mintree to the right, ALPHAp = ADVP would be on the right edge of the Majtree. Thus we conclude with the following rule, more general and at the same time somewhat more symmetric than above:

(R1): A left (right) coordination is not valid unless the root GAMMA of the minimal Majtree does not have any left (right) sibling.

This rule calls for the obvious definition:

Def. The minimal Majtree is the subtree which dominates all the ALPHAi's found by the basic algorithm (A1) and such that no smaller subtree (or equivalently none of its sons) has the same property.

Now, (R1) clearly suggests the following:

(R2): In a left coordination, start with GAMMA as the NP root of the "subject" of the sentence (which, in this case, must be declarative or interrogative) and try to reduce the Majtree by following the sequence of left sons. In a right coordination, GAMMA should begin as the PRED root of the "predicate" of the sentence. Reduction can then be attempted by following the rightmost sons.

(We should point out here that we have not encountered instances where the Majtree could not be considered as a well formed syntactic tree according to some 'natural' grammar. Hence there seems to be no problem in choosing GAMMA as stated above. However, there is no guarantee that this should always be the case. Related matters are discussed in 2.4.7 at the end.)

Now, consider the sentence:

"Time flies like an arrow." (30)

This is clearly a correct coordination. Still, if (R2) is applied as stated, (30) will be considered a right coordination, GAMMA will be chosen to be the PRED node, and the basic algorithm (A1) will fail. Thus:

(R3): If normal application of (R2) leads to a failure and we do not even get a partial match (see below), then try again after setting GAMMA to the DS root of the whole sentence. Of course, the reduction step for minimizing the Majtree cannot apply if this rule was needed.

Finally, the coordinated structures may be surrounded by other subtrees which are not directly involved in the transformation. In the following:

"David drove John and Mary to the movie." (31)

the final Adverbial Phrase "to the movie" is clearly irrelevant to the coordination. However, the outcome of the base component parsing will be: p = 2, BETA1 = NP, BETA2 = ADVP. (We assume a grammar which does not connect these two subtrees into one NP as would be the case with, say, "the man from Texas"). In this case, the basic algorithm (A1) will fail again after having matched "John" and "Mary" (ALPHA1 = BETA1 = NP). What is needed is simply dropping BETA2 from the process. Of course, the same problem may occur with left coordination as in:

"In their car, John and Mary were watching the movie." (32)

Thus, we conclude with the following rule for partial match:

(R4): Right coordination. If failure has occured in algorithm (A1) and, on exit, j = q where $1<q<p$, then try to match ALPHA and BETA elements up to q only and to resume overall parsing. This can only be successful if BETA(q+1), ..., BETAp is a valid syntactic continuation of the Majtree.

Left coordination. If failure has occurred in algorithm (A1), then try again with some subsequence BETA(q+1), ..., BETAp where 1<q<p (in fact try successively q = 2, ..., p), from the original sequence of BETA's. If this is possible, try to resume overall parsing. This can only be successful if BETA1, BETA2, ..., BETAq is a valid syntactic predecessor to the Majtree.

(This rule is not fully satisfactory in that it precludes early execution of the coordination transformation: no attempt is made at resuming normal parsing once coordination is complete. However, determining precisely when this happens is not a straightforward matter, especially in those cases where deletions occurred.)

2.4.4 Reconstruction of the Coordinated Subtree.

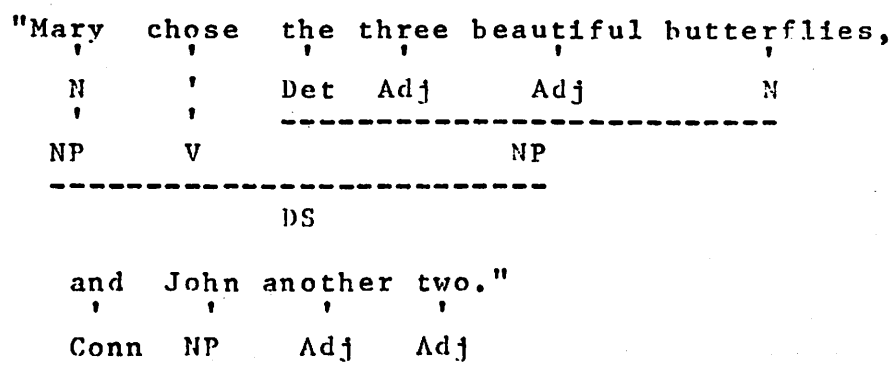
Superficially, this may appear as a trivial problem, and it is true that the main idea is simple:

(R5): In the Majtree as determined by (R1) and (R2), replace the subtrees whose roots are the matching ALPHA-nodes by the corresponding subtrees whose roots are the BETA's. This yields a subtree which has a root DELTA = GAMMA, and we do this for all Mintrees in turn (of course, for each Mintree, there will be a different set of ALPHA's from GAMMA, but all restored Mintrees will have the same root.) Coordination becomes then similar to the original formulation as in the generative approach:

SC ::= SC [, SC , ... ,] Conn SC

where SC is any syntactic category and Conn any connective. The commas are actually required, or could be a repetition of Conn's, even though this latter form is definitely non-standard.

Again, the rule above deserves more careful study. Consider, for example, the following sentence (which is accepted by the rules stated so far):



(33)

Now, in (33), did John simply choose two butterflies or did he actually get beautiful ones? One would tend to say that (33) is ambiguous in this respect. However, there seems to be no natural way to analyze this particular phenomenon. It is some sort of 'serial' or 'linear' ambiguity, more subtle than the structural sort studied below. In particular, native speakers will often find this ambiguity more difficult to resolve. This problem was not investigated further.

2.4.5 Ambiguity.

As is usual in language processing, ambiguity is the source of most difficulties. In this case, neglecting ambiguities of the kind mentioned in the previous section, the obvious source of ambiguity is the non-uniqueness of possible matching sequences in the basic algorithm (A1). We will first describe a modified algorithm to handle this. (A2) tries to find a sequence ALPHAi, ..., ALPHAp matching the corresponding BETA's, starting from a given i: Match(i). A separate entry point, Rematch(p), will attempt to find a new sequence by trying a different ALPHAp, else a different ALPHA(p-1), and so on down to ALPHAi.
In the description of algorithms, we use the notation "(* ... *)" to indicate comments.

Algorithm A2:

***Entry: Match(i).

1. Set PHI := GAMMA; j := i; r := i; go to step 5.

***Entry: Rematch(p).

2. Set r := p.

3. Set j := r; PHI := ALPHAr.

4. Set PHI := Prenext(PHI,GAMMA)
(*Note here the use of Prenext to compute the usual Prenext with the added ability to specify the limits of the subtree to be explored as the

```
    2nd argument*)
    if Prenext fails, then go to step 6.

5.  if Root(PHI)=BETAj
    then set ALPHAj := PHI;
        if j=p,
        then Return success;
        else j := j+1; PHI := Prenextskip(PHI,GAMMA),
            if Prenextskip fails
            then go to step 6;
            else go to step 5;
    else go to step 4 (*try match at next node*).

(*When the end of the tree is reached, the current
match ALPHAr cannot be continued to completion. A
rematch is tried with the next lower r*)
6.  if r=i,
    then Return failure (*can only change ALPHAi*)
    else set r := r-1; go to step 3.
```

If the sentence is ambiguous, several structures will be derived using (A2). The problem then is to evaluate the plausibility of each parse-tree. What one would really like to say is that a coordination is 'better' than another if the corresponding ALPHA's and BETA's are, in the former, more 'similar' in some sense than in the latter. It is important to realize that the way the basic algorithm was set up (compare the roots of the remainder of the Mintree to possible matches in the Majtree) implied a certain bias (or a hypothesis) with respect to the question of similarity which concerns us here. However, this is obviously not enough to eliminate all ambiguities. The consequence, anyhow, is that the main source of information for resolving ambiguities lies now in the comparison of the corresponding subtrees for similarity.

Since no particular measure seems forced upon us, we chose to use a very rough measure; normally, there are not so many ambiguous interpretations of a particular sentence, so that any reasonable measure would do. One could use the following:

(R6): Plausibility = -[product, for i = 1, 2, ..., p: |(Nonod(ALPHAi) - Nonod(BETAi))| + 1] where Nonod(T) is the number of nodes in the tree rooted at T.

This rule assigns plausibilities as expected, but the result is not necessarily semantically best. Consider again sentence (26):

"The clerk prepared 5 boxes of 10 candies and 3 candies." (26)

which was studied earlier. (R6) assigns a higher plausibility to (26c):

(10 candies) & (3 candies) (26c)

than to (26b):

(5 boxes of 10 candies) & (3 candies) (26b)

and indeed (26c) is syntactically preferable to (26b). This justifies why ambiguities are better resolved later, in the semantic phases, rather than just on the basis of syntactic plausibility. One might use more refined comparison procedures for establishing similarity: syntactic and semantic features are obvious candidates. These still appear insufficient for examples such as (27) above. Thus no simple rule permits rejection of all obvious cases of illegal coordination. The transformational component will be simply expected to pave the way for more elaborate semantic analysis (see Chapter 3).

2.4.6 Final Algorithm

We have seen that this basic routine must be topped by other rules discussed above: (R1) to (R6). It is important to specify the precise order in which each rule is applied and how the rules may interact. This is the final global algorithm:

Algorithm A3: (*EPS is the number of solutions*)

1. Get mode of coordination and start with GAMMA as specified by (R2); set EPS := 0.

2. Try to find a match as in (A2) above: Match(1). If this fails, then go to step 4.

3. Else check for validity as in (R1). If solution is valid, then go to step 6, else try a rematch: Rematch(p). If rematch possible, then go to step 3.

4. (*If above fails, look for "partial match"*) Unless EPS=0, go to step 6. Otherwise, check for partial match and for the possibility of fitting the remaining BETA's in the sentence structure, as in (R4).

(*This may make repetitive use of (A2) both for Match and Rematch, and (R1) for validity checking*) If this step succeeds, then go to step 6.

5. (*If all of the above steps failed, look for "boundary crossing"*)
Unless EPS=0, go to step 6. Otherwise, set GAMMA := Father(GAMMA) and try again the same loop as in steps 2 and 3. If this fails again, then declare ***ERROR***.

6. (*One solution found*)
Set EPS := EPS+1. Substitute BETA's for ALPHA's in GAMMA as in (R5). Evaluate plausibility as in (R6). Store results.

7. Retry: look for possible ambiguities. This implies returning to whichever point jumped to step 6, i.e., step 3, 4, or 5.

8. Up to 3 'best' matches are retained. Perform the actual coordination of the fully restored subtrees and replace in original sentence tree. Check new feature agreement rules that apply and reset flags as necessary (if there were 2 coordinated subjects forming together a plural NP, subject-verb agreement would presumably fail on the first pass where only the closest NP may have been detected as the subject.)

9. (*Note that left coordination may be accompanied by right mode too*)
Check for simultaneous left and right coordinations. If necessary, reset mode and restart at step 1.

Finally, when ambiguity does exist, the semantic module will choose the right interpretation by considering the various parsings, starting with the most 'plausible' structure; it will go down the scale, rejecting any instance that does not make sense within the context of the discourse, until it finds a structure that it can accept (if there is one).

2.4.7 Coordination and Grammars.

It is clear from the description of the algorithms in the preceding sections that their processing is extremely sensitive to the grammar used to describe the syntax of the language: some grammars might even fail to allow coordinaton in instances where the coordination would have been perfectly justifiable (this would be the case if some structures which are 'normally' considered similar, are assigned distinct syntactic categories). One could conclude that the grammar must be set up in such a way as to allow all 'accepted' coordinations and only these. However, we would prefer the slightly different idea of introducing equivalence classes between syntactic categories (denoted SC) and to modify the previous algorithms to read ALPHAi .EQV. BETAi instead of ALPHAi = BETAi.

e.g., "from three houses" & "from another" can be coordinated if SC("three") .EQV. SC("another").

This solution is preferable since it is sometimes helpful to introduce syntactic categories with no 'natural' justification, just for the sake of simplifying the parsing process.

On the other hand, there is a more subtle question which must be raised in connection with this aspect of the problem. Consider sentence (33) once more:

"Mary chose the three beautiful butterflies,

and John another two." (33)

Suppose the productions for noun-phrases are as follows:

```
NP (Noun Phrase) ::= (Det)+  (Q)+  (Adj)*  Noun
Det (Determiner) ::= the ! another
Q (Quantifier)   ::= two ! three
Adj (Adjective)  ::= beautiful
Noun             ::= butterflies
```

where '' is the null string, X+ indicates 0 or 1 occurrences of X, and X* indicates 0 or more occurrences of X.

Then algorithm (A3) will not detect any ambiguity: "another" will match "the" and "two" will match "three", the substitution rule (R5) will restore the Mintree uniquely as follows:

"John chose another two beautiful

butterflies." (34)

Suppose, on the other hand, the grammar is as follows:

```
NP (Noun Phrase)              ::= (Nint!'') Noun
Nint (Noun Introducer)        ::= Det NMS ! NMS ! Det
NMS (Noun Modifier string)    ::= NM (NMS!'')
NM  (Noun Modifier)           ::= Adj ! Q
Det (Determiner)              ::= the ! another
Q (Quantifier)                ::= two ! three
Adj (Adjective)               ::= beautiful
Noun                          ::= butterflies
```

then ambiguity can be detected if we elaborate on the substitution rule. We now get:

```
ALPHA2 = Nint   ("the three beautiful")
BETA2  = Nint   ("another two")
```

Hence, one could say that if corresponding subtrees have a different number of terminals in their derivations, this may be the source of ambiguity. The problem is obviously quite complex and no satisfactory treatment is known at this point.

To summarize, this section explored the problems related to the analysis of coordination. It presents a solution which is able to handle a number of cases, including the many different examples mentioned in the text. Of course, many points are left unanswered; more work is needed in this area.

2.5 EXAMPLE.

Sentence (23) below is taken from the original text (see Appendix 1, p. **157**, example B). The previous section discussed some aspects of its analysis that are connected with coordination. In this section, the reader can get an overall view of the parser by following this sentence throughout the process of syntax analysis.

On input:

"Bill delivered 22 newspapers on one street,

56 on another, and 43 on a third street." (23)

After the first phase, the words are stored in an array as shown below. Features are indicated by mnemonics. In fact, they are normal Snobol strings. Thus:

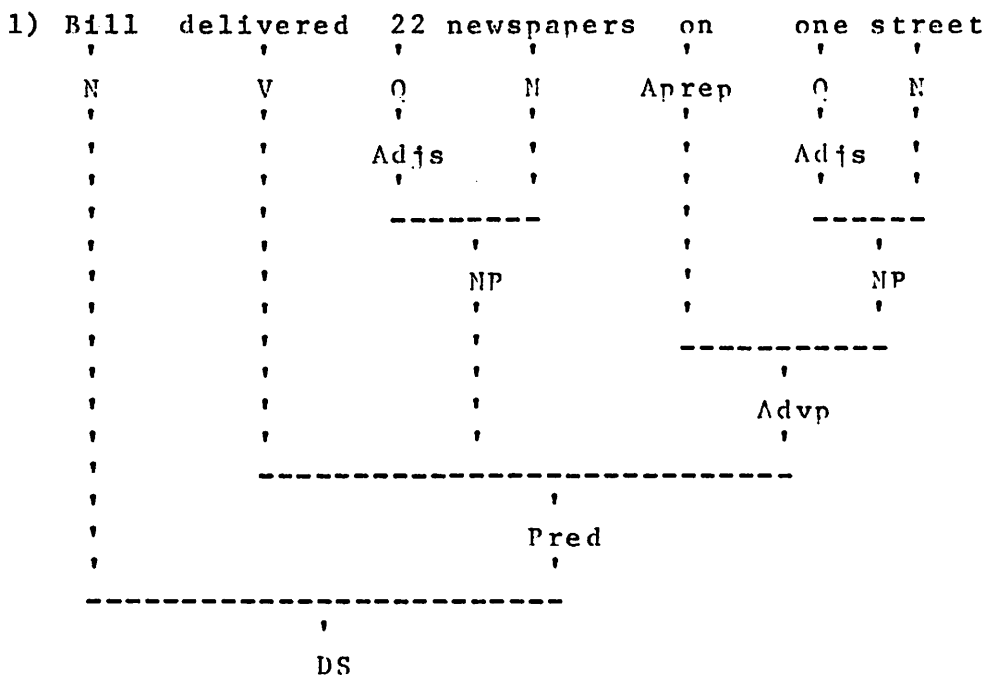
```
P-name,male  : ,Gen=M,Com=-,Cnt=-,Anm=+,Hum=+,Abst=-,
Past         : ,Form=2/3,Tens=Pa,
C-name       : ,Gen=N,Com=+,Cnt=+,Anm=-,Hum=-,Abst=-,
Plural       : ,Num=P,
```

where Gen is for gender, Com = common, Cnt = count, etc.

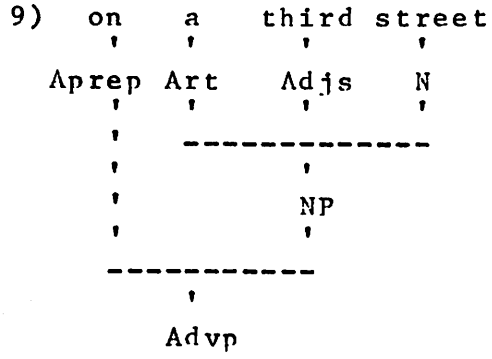
	Root	Features	Syntactic Category
1:	Bill	P-name,male	Pn
2:	deliver	past	V
3:	***2 homographs for word***		(O.N.)
3-1:	22	-	O
3-2:	22	C-name	N
4:	newspaper	C-name,plural	N
5:	on	-	Cprep
6:	***2 homographs for word***		(O.N.)
6-1:	1	-	O
6-2:	1	C-name	N
7:	street	C-name	N
8:	,	-	Np-b
9:	***2 homographs for word***		(O.N.)
9-1:	56	-	O
9-2:	56	C-name	N
10:	on	-	Cprep
11:	another	-	Adj
12:	,	-	Np-b
13:	and	-	Conn
14:	***2 homographs for word***		(O.N.)
14-1:	43	-	O
14-2:	43	C-name	N
15:	on	-	Cprep
16:	a	-	Nint (Art)
17:	third	-	Adj
18:	street	C-name	N
19:	.	-	end-marker

Also Coord.flag is set 'ON'.

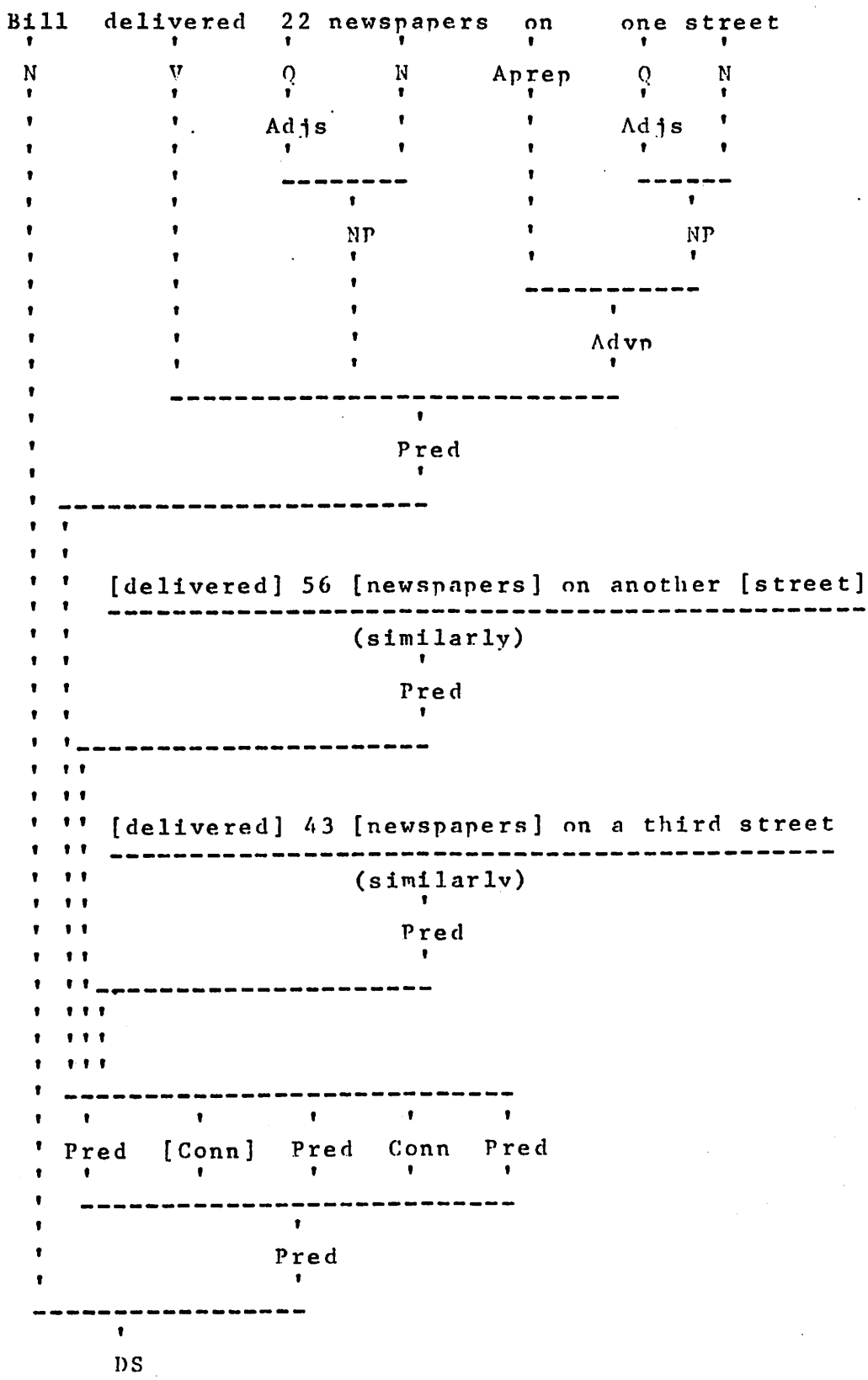
Parsing according to the base component grammar is then started. Many failures occur but no rejection is ordered. The coordination flag commands resuming the parse, each time starting a new subtree; analysis continues in this interrupted fashion. After the sentence is completely exhausted, Coord.flag is still 'ON', Cmode is also 'ON' indicating that failures did occur. Thus a transformation is necessary. A sequence of 9 subtrees is handed to the transformational component:



2) , ---------- Np-b
3) 56 --------- Q
4) on --------- Aprep
5) another ---- Adj
6) , ---------- Np-b
7) and -------- Conn
8) 43 --------- Q



The coordination module finally yields the following structure where three complete predicates are coordinated into one (the "[]" indicate the words restored by the analysis):


Bill delivered 22 newspapers on one street
N V Q N Aprep Q N
Adjs Adjs
NP NP
Advp
Pred
[delivered] 56 [newspapers] on another [street]
(similarly)
Pred
[delivered] 43 [newspapers] on a third street
(similarly)
Pred
Pred [Conn] Pred Conn Pred
Pred
DS

2.6 DISCUSSION.

Before concluding this chapter, it is important to discuss various weaknesses and strengths of the syntax analyzer.

First, the handling of homographs is not always adequate. Because the basic parser could not carry several parse-trees simultaneously (i.e., there is no backtracking), it was necessary to resolve the homograph as soon as the scanner reached the word involved. It was pointed out that this did not create problems with the sentences on hand, but the simple rules used are insufficient to handle those cases where necessary look-ahead is too long. Consider:

"The general commands the army." (35)

and

"The general commands are simple." (36)

(note that "general" is Noun/Adjective and "commands" is Verb/Noun).

With a full context-free parser, one could carry two parsings until one of them gets rejected by either "the" or "are." Our analyzer would simply recognize its inability at handling either sentence. (One could elaborate on the context-sensitive rules for resolution, but the approach above is much more natural because the rules needed are not intrinsically context sensitive.) Incidentally, there seems to be a misconception concerning the type of grammar needed for an 'ideal' base component. It is sometimes argued that since natural languages involve a great deal of context, the syntax must be context sensitive. This is not clear, and one never seems to need more than a context free grammar as a base. On the other hand, context sensitive grammars do not have the power of transformations. Thus, they would not contribute to the simplification of the total parser in any way.

Other problems arise in connection with the question of feature agreement because, again, decisions are made too early. Thus:

"The garden swarms with bees." (37)

would be rejected before realizing that the actual agent of "swarm" is "bees." Such inverted constructions are not always easy. However,

"How many apples am I holding?" (38)
would be handled correctly by this analyzer: initial subject-verb disagreement would be recorded and inversion expected. Only if the inversion does not take place would the disagreement flag command rejection. Hence:

*"How many apples an in this basket?" (39)
and

*"How many apples am you holding?" (40)
would be rightly rejected.

In general, the solution to these problems is simply to delay the decisions until the last moment even at the expense of sometimes doing unnecessary work, analyzing the rest of a long sentence when the beginning is already wrong. Incidentally, the human approach seems to be to delay decisions up to a point where one decides that nothing new seems to be coming up from the sentence which could remedy the disagreement. On the other hand, it is interesting to note that it is precisely the final disagreement that will trigger the exploration of the possibility of inverted constructions, leading eventually to the correct analysis of sentences like (37) above.

This approach is very much in line with the philosophy underlying the whole design of this parser. The 'principle' could be roughly stated as follows: "Proceed as usual unless you fail; then, and only then, look for odd cases." The main advantage here is that additions to the grammar do not increase the complexity of the inner core. As far as efficiency is concerned, this is a desirable property (parse time of simple sentences remains unaffected). Moreover, such an approach is conceptually 'cleaner' because it provides more incentives for keeping various aspects of language analysis in distinct modules. In this system, the executive has precisely the role of directing the parser according to the failures or successes of the various modules.

One final point is that the implementation of semantic features at the syntactic level was considered as just a heuristic for parsing. We feel that these belong to semantic processing and the next chapter will show precisely how to handle them in a more general fashion.

Of course, more weaknesses are present which simply illustrate the fact that syntactic analysis is still an open field of investigation, and this was not

the main area of our research anyway. What was needed (and accomplished) was a simple, fast parser which could handle a sufficiently large subset of English to allow the analysis of unmodified text from an actual textbook. The use of bottom-to-top transformational recognition allowed the achievement of parse times ranging from 0.1 second for simple sentences up to 2.0 seconds for complex sentences such as (23) above, using the Snobol interpreter on the CDC6400.

CHAPTER 3
SENTENCE SEMANTICS

3.1 SEMANTIC MEMORY NETWORK.

As was pointed out repeatedly (see, in particular, [Quillian 1966]), the structure of semantic memory is a very important part of the general problem of semantics. Of course, one can supplement any type of structure with the appropriate machinery toward some particular goal. An adequate structure is, however, highly desirable. It is conceptually easier to grasp; it is psychologically more natural; and also, though perhaps secondarily, it is a more efficient framework.

Essentially, the memory is structured as a graph where the nodes belong to one of the following types, which we first present informally:

1) Elementary nodes: these are verbs and nouns.
e.g., "give", "add"; "man", "John."

2) Relational nodes: these represent combinations of primitive nodes. As a whole they may represent general nominal or verbal nodes, or various sentence types expressing facts or actions or, generally, any relation. They are further subdivided into four sub-classes:

a) Sentence nodes.
e.g., "Sue gave Ed 19 candies."

b) Composite verbal nodes.
e.g., "drive slowly"

c) Composite nominal node. Disjunction or coordination of nouns, and/or Boolean combination of their defining properties.
e.g., "a human or an animal," "a dog or a cat," "a man and a woman," "a head and a body," "a white shirt with blue stripes," "a tall, blond girl."

d) Function nodes.
e.g., "the color of that door."

3) Complex nodes: these express what would best be called 'stories' in the sense of a related set of facts. Two examples follow:

First story -- adding two 2-digit numbers:
Add the ones; check for the result being greater than ten; if so, hold a carry to the tens' place; write the ones' digit of the answer. Add the tens, write the result in the tens' place, and possibly the hundreds' place of the answer.

Second story -- eating a meal at the restaurant:
Ask the waiter for the menu, make your choice for a meal according to hunger, taste, budget, ... For each dish, process food into manageable mouthfuls, bring the food to your mouth, chew, etc.

The above presentation was by necessity very informal: memory structure is in fact a complex problem. The remainder of this section discusses memory organization in greater detail, as well as giving some of the motivations that directed its design.

3.1.1 Elementary Nodes.

There has been much discussion about the exact meaning of the concepts of "verb" and "noun." The real difficulty stems from the fact that these concepts are possibly the only true primitives in our semantic structure. This implies that we can only describe them through examples: verbs are states, processes, etc. Nouns are persons, things, etc.

The definition that has been most often proposed for "noun" is "subject of discourse." This has the inconvenience of restricting nouns to their surface materializations through human speech. On the other hand, it is superior to such definitions as "object" in that it includes abstract words like "beauty," "color," "mathematics." However, it seems that abstract nouns are indeed linguistic artifacts. The answer to the question "What is beauty?" comes inevitably in the following vein: "Through education or maturation, people develop a taste through which they feel that something is more or less beautiful..." In short, "beauty" is derived from "beautiful." Similar facts apply to the concept of "color." It has been

established through careful experimentation that color words (blue, green, etc.) only cover a rather small part of the 2-dimensional plane of hue vs. luminosity. This presses us to use the noun "color" as a linguistic support of our expression in sentences like: "This object has a strange yellowish color." It is also for reasons of convenience that abstract nouns do appear as special nodes in this semantic memory. They designate an 'axis' that regroups a set of related properties (cardinality of a set, ordinal rank, dimensions, etc.) Indeed, problems of comparisons and antonomy (see section 3.4) lead to the use of abstract noun nodes whether or not a corresponding word naturally exists for them in English.

Verbs are central to sentences and, in general, to the whole language. Of course, nouns are equally essential to our frame of conceptualization. One does not think of "blue," "tightening," or "give" as entities by themselves but rather of "something being blue," "something undergoing tightening," or "someone giving something to someone." The sentence is the basic concept in our mind. That nouns are, in a way, secondary to verbs stems from the fact that nouns revolve around verbs and take existence through them. A noun is an agent of an action, or an object of a state, process, or action, etc. Also, sentences always include a verb while there may be a variable number of nouns. On the other hand, arguing whether the noun "human" comes from the adjectival state verb "to be human" or vice-versa is reminiscent of the chicken and egg dilemma. In cases where both the verbal property and the corresponding noun exist, we will arbitrarily derive the latter from the former. Finally, derivational structures are not frozen but may dynamically change through learning. This adds to the difficulty of precisely defining these primitives.

3.1.2 Elementary Node Classifications.

Following Chafe's approach [Chafe 1970], we will identify four main classes of verbs:

1) 'states': the name is self-explanatory. Most often, they appear in surface structure as adjectives, possibly accompanied by the copulative "be" (e.g., "The rope is tight", "The flower is red"). One reason why they have this distinct status from other verbs is probably that it helps the speaker to form reasonably understandable sentences. Hence, contrast "The big fat

turkey" with "The turkey which is big and which is fat."

2) 'processes' roughly answer the question "What happened to N?" where N denotes a noun whose state is presumably undergoing change. e.g., "The rope tightened."

3) 'actions' answer the question "What did N do?" where N denotes the agent of some action, as in "John is singing" or "Jim added two numbers."

4) 'action-processes' finally are just a combination of the previous two: e.g., "David opened a box of candies." Their existence merely shows the compatibility of the two previous categories. However, it is important to mention them as they constitute, together with states, the large majority of verbs.

Various attributes introduce further subdivisions within each verb category. Thus, a state verb can be 'completable;' it then requires a complement to determine its meaning more precisely (e.g., "Jim weighs 150 pounds"). It can also be 'locative' when it expresses the location of some object (e.g., "The candy is in the box"). An action or action-process can be 'benefactive' when it involves a 'beneficiary' in the usual sense (e.g., "Sue gave Ed 19 candies"). There are many other categories but their description is not essential to this thesis; more details can be found in [Chafe 1970].

The ordinary verbs described above are those which form the basis for sentential structures. They are usually identified as "predicative verbs." Other categories are used in forming verbal or nominal structures. As it will become clearer below, adverbs, quantifiers, and conjunctions coordinating nouns can also be 'verbs.' Thus, we will stretch the definition of verbs to be "any element around which a combination of nodes may be built." All these verbs have different functions of course, since they do not result in the same kind of structure. The reason for grouping them together under the same heading is that a unified analysis procedure can be applied in all cases as will be shown later.

Nouns are essentially defined through properties, which are mostly related to verbs:

"A boy is a young, male human." (1a)

"A transformation is the result of someone transforming something." (1b)

Because of this close relationship, nouns may often be used as state verbs:

"John is a man; he is a pilot; he has a lot of courage." (2a)

or as the nominalized form of the corresponding verb:

"The transformation of the line into a circle did not seem natural." (2b)

In the latter case, the deep presence of the verb "transform" is the sole justification for the preposition "into." (The verb is brought out by the analysis as discussed in 3.3.2.) For the sentences in (2a), we will postulate the existence of the appropriate verb nodes associated with the right properties. The nouns here represent the class of objects having these properties.

Insofar as nouns are defined by their properties, their most natural expression is through Boolean combinations of these. Problems arise in particular from disjunction. One of these is discussed in section 3.4.2. Another one centers on the distinction between disjunction of properties and disjunction of nouns. One tends to think differently of "blue or black sweater" and "boys or girls." The main difference is in the number of 'heads' of noun phrases. This is a purely syntactic concept. Note that "boys or girls" is very similar to "male or female children" (except, maybe, in the strangeness of "male child," but more generally, in the emphasis: with a common noun phrase head, the similarity between the elements of the disjunction is stressed; otherwise, it is only coincidental.) We chose to assign the same meaning to what may be formalized in full generality as: "an instance of a property or an instance of another" and "an instance of either one property or some other."

Coordination of nouns by "and" does not bring in such problems as there is no possible confusion with any Boolean combination of properties. "John and Mary" is neither "John or Mary" nor "a person who is simultaneously John and Mary."

A final noun-phrase type to be covered here is the one examplified by the following: "the color of apples," "the speed of the flight." These phrases have common characteristics. They are headed by a word playing the role of a function (or operation). This is followed by "of" and finally the parameters. This "of," which we will call "functional of," is to be opposed to "of" used to indicate contents, part, nature, etc. Section 3.3.2 shows how these other uses imply a whole sentence or structure. Functional nodes only reflect the use of mathematical, physical, or linguistic functions. This last distinction merely emphasizes the numeric or alphabetic nature of parameters and/or results. Mathematical functions are numeric in both parameters and result, hence "sum," "difference," etc. Physical functions have a numeric result, e.g. "weight," "pressure," etc. Linguistic functions are purely non-numeric, e.g., "color." For this latter kind of function, the header word is precisely the abstract noun that was referred to before as heading a group of related properties. This construction allows reference to the whole group whenever the actual verb/property which applies to the noun is unknown or is left unspecified for any reason. When the 'position' on the axis is made more explicit, a transformation resolves the functional node in the appropriate manner:

"The color of the apple is red." --- (red :color apple:) --- (red apple) (3a)

This transformation can be very powerful. The following examples presuppose the discussion of adverbs and compatibility for full understanding (sections 3.3.6 and 3.4.1), but the notation is sufficiently suggestive:

"The speed of the flight was moderate." --- (moderate :speed flight:) --- ((moderate speed) flight) (3b)

"The flight was moderately fast." --- ((moderate fast) flight) (3c)

Note that the traditional "features" of the transformational systems are not singled out in our

classification of nouns. The "+/- animate," "+/- human," "+/- unique," and "count/mass" systems are not distinguished from other noun properties. (This is discussed further in 3.2.4 below.) Only one classification system is used: 'abstract'/'concrete.' This distinction is connected with the analysis of some particular compound noun structures, as discussed above. It is also important for the problem of 'compatibility' (see section 3.4).

3.1.3 Relational Nodes.

Relational nodes group two or more nodes, one of which is distinguished as the verb. Component nodes can be elementary, or recursively any type of node. A relation is normally written as follows:

(V X1 ... Xn)

where V is the verb of the relation, X1 through Xn are nouns or verbs (see below) and n is at least 1.

Sentence nodes are the most familiar type of relation. They combine a verb and a variable number of nouns each of which stands in a particular relationship with the verb. One might distinguish sentence nodes by the number of nouns included. But this is only a superficial aspect of a sentence. Semantic significance lies rather in the noun-verb relations. These are extensively discussed by Chafe, whose classification will be used here (see [Chafe 1970]). Some examples are presented below. For each category or sub-category of verbs, the set of required nouns follows the "+++" mark. These nouns are associated with semantic relations rather than syntactic positions. Thus, 'patient' roughly indicates the noun whose state is being talked about in the sentence; syntactically, it can be the subject of a state or process verb, or the complement of an action or action-process verb, etc. Other names are self-explanatory, or have been presented above in 3.1.2.

- V:state +++ patient-N
  e.g., "The apple is red."
- V:state,locative +++ patient-N, location-N.
  e.g., "The candy is in the box."
- V:action,process,benefactive +++ agent-N, patient-N, beneficiary-N.
  e.g., "Bill sold 12 stamps to Mr. Clark."

Other types of relational nodes appear in connection with the study of adverbial phrases. One distinction is important here: Some adverbial phrases

modify nouns; they act like verbs and make up a full sentence. In other cases, adverbs as well as adverbial phrases operate exclusively on verbs or on sentences. Thus:

"the man in the car" = "the man" + "the man is in the car" = man--(in * car) (4a)

"John spoke slowly." = ((slow speak) John) (4b)

The main point is that adverbs and adverbial prepositions are essentially verbs. Some of them have identical spellings when operating on nouns and when acting on verbs or sentences. Sometimes the words are different, restricting the usage of each. Thus:

"John is well." (5a)
"He spoke well." (5b)
"Jim is slow." (6a)
"He speaks slowly." (6b)

Therefore, as far as structure is concerned, we have two possibilities. In some cases (predicative use as in (4a), (5a) or (6a)), the verb may form an ordinary sentence with either a noun or a sentence node (which is equivalent to a noun). Otherwise (attributive use as in (4b), (5b) or (6b)), a different kind of sentence appears: these will be called 'verbal sentences' because they include several verbs, and more importantly, because they constitute as a whole an entity which is functionally equivalent to a simple verb. Note that some adverbs, such as "very," can only appear in the second form, while temporal adverbial phrases only take the first one. For example,

*"Jim is very." (7a)
*"John yesterday spoke." (7b)

One of the most basic relations for structuring memory is that of semantic implication, denoted by ==], which is very similar to logical implication or inclusion. It is partly inspired by the relation used in Becker's study of induction [Becker 1969], even though there are some important differences. As for all primitive concepts, its complete definition must be through its use in the thesis (see, in particular, sections 3.2 and 3.4). Note however that one of the unique features of this theory is the uniform treatment of relations. A priori, no particular relation is singled out, and the only reason for considering some of them as more basic than others is the specific role that they play in the semantic analysis (e.g., many verb descriptions refer to the node "animate"). Apart from this, one should emphasize that there is no essential difference between the operators --

==], instance-of, properties, or, and, etc.
and the features --
animate, abstract, process, state, etc.
or the predicates --
give, blue, in, tired, drink, etc.
This uniformity must be stressed. Its somewhat non-standard character makes it appear rather confusing upon initial presentation. This subject is further discussed later in this chapter; see 3.2.4.

On the other hand, the memory is entirely interconnected through sentence-nodes or relations, or more complex nodes. In fact, the meaning of a concept is precisely defined through its connections in the memory network. We often represent sentences by themselves, but the reader must keep in mind that they are closely bound to the whole memory. Hence,

"John is hot." (8a)

might be simply written:

(hot John) (8b)

But one should remember the following relations that are implicit:

(==] hot state-verb)
(==] hot relative)
(==] John boy)
(==] boy male)
(==] Bill boy)
(give Bill Jane box)
etc.

If all connections are followed through, the list would include the entire semantic network.

3.1.4 Complex Nodes.

Several sentences may be joined together into a complex node which can be described as an algorithm, a story, or a situation in the sense of Becker [Becker 1969]. Its structure is essentially determined by the relative timings associated with its component sentences and the various relations linking the nodes that appear in it.

a) Time. Time is one of the most difficult problems in current linguistic research. Its importance for the description of algorithms is obvious. Yet it is a central concept in English. This is reflected in the fact that English verbs always carry some sort of inflection (present, past, perfective, etc.) which serves as a time reference. It has been pointed out, however, that this is not universally true.

> The Wintu's language does not "force" him to distinguish past, present, and future. He must, however, choose verb forms that automatically indicate the sort of evidence that lies behind his statements. If the speaker is describing an event within his field of vision, the verb would be conjugated in one manner. If the speaker's assertion is based on hearsay, a different verb form would be required. Still another form would be used when the speaker comments on a predictable and recurrent event by asserting, for example, that "The chief is hunting" (based on knowledge that he regularly hunts at this time). ([Manis 1968], p. 90)

This was not meant to deny the importance of time. It can be argued that one's confidence in an event is mainly involved in association with learning whereas time is present in all physical events. Rather, the quotation suggests that both time and degree of confidence are central to human behavior. In CLET, both are attached in some way to every sentence.

One representation of time references might use a unique, absolute time axis (relativistic theories are not relevant here). Actions might then appear as an interval (or crudely, a point) on this axis. One problem then is the ordering imposed on events taking place in independent 'stories.' If "Jim pushed a light switch in his home at 6:15 P.M. on October 19, 1950" and "The whole city of Boston fell into darkness at 6:16 P.M. on October 19, 1950," one can conclude that the first event is a predecessor of the second one. The whole point is whether one really has a deep consciousness of this fact, or whether it is even desirable that such awareness be explicit. We believe that it will only be so if special circumstances lead one to associate both events, e.g., the latter event may turn out to be a direct consequence of the former. Hundreds of people turn their lights on and off every minute in a large city, and it would not naturally occur to anyone to keep track of events that follow or precede such switchings.

A second example will help illustrate a slightly different aspect of this problem. While someone is reciting the Star Spangled Banner, if he is interrupted and asked to repeat the fifth word before the one he said last, he will usually need to start over from the beginning.

Humans seem therefore to restrict their attention to time relations between logically connected events. Indeed, the main difference between random and intelligent behavior is precisely that, in intelligent behavior, actions are carried out in a certain order, corresponding to their logical connections (as defined by the goal to be achieved). In some 'ideal' (imaginary) environment, it could be the case that time structures are constantly derived from logical structures. Most often however, humans seem to behave differently. Actions are performed as they have been described; even though they correspond to some logical order, the full logic behind them often escapes their agent. This distinction is important in practice. It accounts for the routine way in which most of our actions are performed, and the savings achieved thereby. Indeed, reordering rules within a complex node for the sake of efficiency corresponds closely to the idea of habit forming. In some cases, one does not need to think anymore, one simply 'follows' an algorithm. (A tag could indicate when the complex node had been 'assimilated.') Incidentally, this seems connected with the difference that people feel between programming in LISP and Fortran or some other sequentially oriented language. A pure LISP program is perhaps the best example of sequences of actions being entirely determined by their logical structure as reflected by the recursively interconnected function definitions. In Fortran, the steps follow each other in their expected order of execution: f(g(h(x))) becomes x1=h(x); x2=g(x1); f(x2). The main point is that an example of operation performance (including the examples in the text selected for this thesis) are more readily observed as a sequence rather than a LISP nest. In fact, the internal algorithms derived by CLET's inductive component are organized sequentially.

Semantic memory is not organized along one infinite time line. Within stories, events are associated relative to each other. Thus the body of a complex node is essentially relocatable on all four dimensions of space-time. This is quite obvious for the algorithms describing the performance of some action-verb. Nevertheless, and this is particularly true in the case of real stories or situations, there usually exists an absolute setting which applies generally to the whole body of the node. This is implemented through the use of absolute situational references (time and place) appearing once as a header for each complex node.

b) Constraints. A complex node may also include a list of constraints: in some cases, a relation involving some of the component nodes within the node body seems important even though it is not explicitly stated in the text. This could be some noun-verb relation (even across sentence boundaries), or more often, a relation between different nouns. Thus in the sentence,

"Write 2 in the tens' place of the answer to show there are 2 tens." (9a)

"2" occurs twice, but depending on the current knowledge of the listener and the context, it may not be really clear whether the identity is intentional, as in (9a), or coincidental, as in (9b) below:

"11 tens are 1 hundred and 1 ten." (9b)

Using the same node for both occurrences is perhaps too great a commitment to one interpretation. The resulting loss of information may prevent successful learning since structural similarity is an essential factor in analogy formation. It seems that the best way to account for this kind of fuzzy relation is through the use of an external constraint. In this particular case, it would have the form:

(identical N1(=2) N2(=2))

In general, a constraint will be a sentence node.

In addition, each sentence within a complex node has a 'confidence level' attached to it. This applies equally well to the situational references, to the sentences within the node body, and to those in the list of constraints. Clearly, in the kind of dynamic situation implied by learning, probabilistic aspects play an important role, and the confidence levels just mentioned reflect a higher level of this phenomenon. Their values lie in the interval [0,1]: 0 denotes total unreliability, in which case the sentence can be neglected or entirely deleted, while 1 denotes complete reliability.

3.2 "SENTENCE MATCHING."

The "sentence matching" scheme, later denoted SM-scheme, is central to this view of analysis. To avoid confusion, it will first be presented in connection with a simple, idealized sentence skeleton in which all inflections and fuzziness inherent in the words are neglected. To be concrete, let us consider the following example:

"The dog is white." (10a)

where we ignore for the moment the reference implied by the definiteness of "dog," the present tense of the copula, and the non-absolute character of "white" as a description. The elementary structure corresponding to (10a) is:

V:white, subject-N:the-dog (10b)

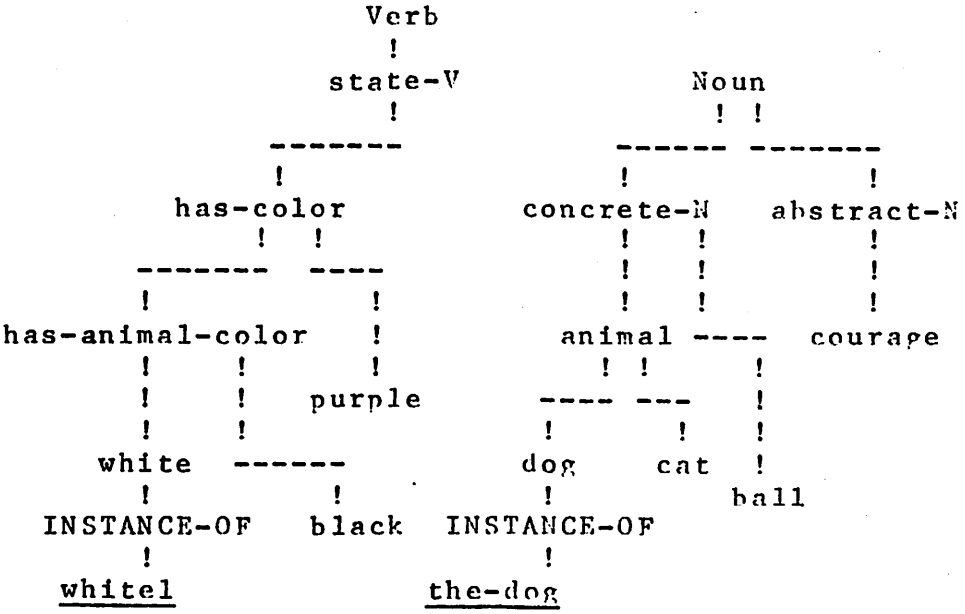
In fact both words are part of a chain of semantic implications which can be diagramed as:

white = Vn ==> V(n-1) ==> ... ==> V0 = V (11a)

the-dog = Nm ==> N(m-1) ==> ... ==> N0 = N (11b)

(which is a short way of writing the set of relations representing (11a-b) in the standard format)

The memory could actually contain the following structures (Note: an interrupted line from a concept to the one above indicates a "==>" relation):



On the other hand, there may be some additional sentence nodes:

(has-color concrete-N) (12a)
(has-animal-color animal) (12b)
(whitel the-dog) (12c)
(We will usually assume the first two to be in memory.)

The SM-scheme involves two parts, 'verb pivot' and 'noun pivot,' which will be illustrated with examples.

3.2.1 Verb Pivot.

The analysis starts with a search of the memory by going up the V-chain, checking for the existence of a sentence node containing a verb Vi for some i in [0,n]. The search stops as soon as one is found. Thus, Vi is chosen to be closest to the verb in the sentence being analyzed. (As defined earlier, the top node, V0, is the general verb node V; here, i is chosen the largest possible.) The sentence node expresses a property of Vi and specifies a sentence construction. Three cases may be distinguished:
a) If no such Vi is found, the sentence is rejected. This is really a trivial case. It should never happen in practice as it implies that the memory is essentially empty.
b) Vi does exist, but there is no "match" with the input sentence; i.e., there is no sentence node of the form:
(Vi Nj)
for some Vi and Nj belonging to the initial chains. Rather, the sentence specification is:
(Vi N')
where N' represents a set which excludes the actual noun. (This matter of 'incompatibility' will be discussed at length in section 3.4. As a rough approximation, one might think of N' as simply not belonging to the N-chain.) In this case, the sentence is again rejected. For example, if the memory structures are assumed as above, the following sentence would be rejected:
*"Courage is purple." (13)
c) In the case of the sentence at hand (10a), this upward search will be successful, the matching pair being (12b): i = 3, j = 2, (V3:has-animal-color N2:animal). However, unless the matching sentence is identical to the one being analyzed, the process is clearly not finished. General rules may be overriden by particular restrictions. More precisely:
(Vi Nj) presupposes (Vr Ns) for $r \leq i$, $s \leq j$. It also implies (Vi Ns) for $s > j$. But it does not specify anything a priori about the possibility of (Vr Ns) for $r > i$ (and any s).

This is the reason for the next part of the process.

3.2.2 Noun Pivot.

The analysis continues with an examination of the properties attached to the noun Nj, seeking a node which may restrict the sentence (Vi Nj). In fact, the search goes down the N-chain from Nj, looking for a node:

(V' Ns) (14)

where m ⩾ s ⩾ j and V' is a verb which is 'incompatible' with the one in the input sentence (again, see section 3.4).

If all properties of Nr are compatible with the sentence under analysis, N(r+1) is examined, and so on. This process stops in either of the following ways:

a) Some 'rejection pair' such as (14) is actually found and the sentence is refused. For instance, (12c) above, which says that the particular dog of the discourse is white, would determine rejection of:

*"The dog is black." (15a)

Also, if we admit that "purple" does not belong to animal colors, (12b) would command refusal of:

*"The dog is purple." (15b)

Note that the matching pair for (15b) would be one level higher: (12a).

b) If the whole N-chain has been examined without obtaining a contradiction, the sentence is accepted. This would include the original sentence (10a) but also both of the following:

"The cat is white." (16a)

"The ball is purple." (16b)

This basic exposition calls for several comments. One problem is the graph-like organization of ==⟩ chains. Thus in practice, when going up or down during the SM-process, one may encounter several branches leaving a single node in either direction. This poses no theoretical difficulty, however, since it suffices to follow all branches and adjust decisions accordingly. The net result is merely a slow-down in response time. Two other points require discussion: (1) evaluation of semantic deviance and (2) epistemological adequacy. These will be explored below.

3.2.3 Semantic Deviance.

It is important to observe the symmetry between the two parts of the SM-scheme: In the first phase, we go up the V-chain, checking for the compatibility of the noun with the nouns in the sentences of the memory network. In the second, we go down the N-chain, checking similarly for the verb. The rejections happening at each stage correspond roughly to the traditional distinction between "non-sensical" and "non-factual." The former are in some sense more deviant than the latter. This simply confirms the central role of the verb. The primary criterion is for nouns to conform with the verb; then only comes the reciprocal check. However, the distinction is not that sharp. The essential difference seems to lie rather in the 'level' at which rejection occurs. Compare the sentences that have been analyzed above:

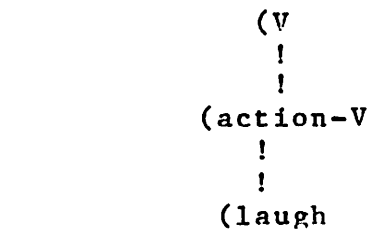
*"Courage is purple." (13)
*"The dog is purple." (15b)
*"The dog is black." (15a)

The first is rejected on the upward check; the last two are rejected during downward check, (15b) being rejected before (15a). Most readers would agree that (13) definitely contradicts semantic rules, while (15a) contradicts facts. However, the case of (15b) is not so clear-cut. It is precisely for this reason that we wish to depart from a traditional transformational approach and simply state that all three sentences are to be rejected on similar grounds, though at different levels. The fact that in our implementation (13) would already be rejected by the syntactic module is purely coincidental with the use of semantic features as an aid to parsing. In a unified implementation where all modules, in particular the syntactic and semantic analyzers, are treated as interacting coroutines, (13) will be rejected in very much the same fashion as (15a) or (15b). The degrees of deviance vary simply because the sentences that determine rejection are situated at different 'distances' from the top V and N nodes. Indeed, the indexing of the nodes in the ==> chains above was effectively done so as to reflect this idea of level. Nevertheless, one cannot define an absolute scale of 'semantic deviance':

- Memory is essentially dynamic and levels are constantly liable to change. Even the higher nodes (low indices) are not fixed, as they may be effected by growing linguistic awareness. Hence:

*"The table laughs." (17)

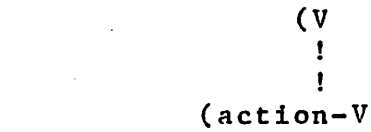
would be rejected by the following memory structure:



(18a)

with a sentence where the indices are (2,2).

A linguistically more refined memory structure might be:

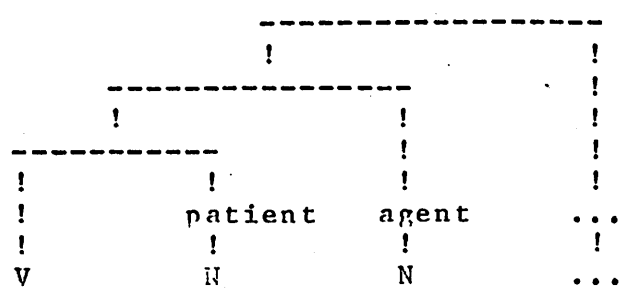


(18b)

and (17) would be rejected at level (1,1).
- Another aspect of this same problem results from variations in semantic structures between individuals. Thus the next two sentences might be treated quite differently by a biologist and a physicist:

"The human body is 40% muscle." (19)
"The human body is 99.9% vacuous." (20)

Moreover, even a fixed unique memory would still present problems:
- Memory structures are more like a graph than a tree of independent linear chains. Thus the level depends on the path chosen. In fact, one's feelings seem to derive from some combination of the different levels (such as taking the minimum or some kind of average).
- On the other hand, there is no 'natural' ordering for pairs or generally n-tuples of numbers, especially when n itself is not fixed. One could take the verb to be most important because of its centrality in the sentence, and then assign an order for nouns in agreement with their relation to the verb. In [Chafe 1970], though in a different context, the author suggests that noun-verb relations are not equally "central." Thus:



(21)

This would imply, assuming a 'standard' memory network, a greater deviance to (22) than to (23) below:

*"John frightened the table." (22)
(*)"The table frightened John." (23)
The errors are very similar except that the patient is involved in (22) whereas the agent is faulty in (23) (which naturally leads one to give, figuratively, some animate attributes to "the table").
- Other features of natural language, especially those involving fuzziness, and those concerned with procedures for recovery from primary rejections, have great relevance in this matter. They will be discussed later.

3.2.4 Epistemological Adequacy.

Another important point is that of the epistemological adequacy of this formulation. Clearly, there are many ways by which one could build a model of semantic processing. Chomsky rightly expanded on the idea of "simplicity" as one criterion for grammars ([Chomsky 1965], pp. 37-47). This was not purely a matter of syntax, but rather metatheory. In this context, there are several advantages to this scheme over one based on traditional features, even though they may appear surprisingly similar. Clearly, both rely on the idea of checking a particular instance using a more general rule based on higher level categories. The main difference is in the structural organization. Previous approaches have attached a list of features to each word. With minor variations, the entry for the verb "count" in a "lexicon" might be:

(+action, +process
(agent : +concrete, +count, +animate, +human (24a)
(patient: +count

In CLET, the verb would appear in a sentence:
(count(Verb) human count(Noun)) (24b)
where each component node would refer, through "==>" sentences, to the relevant properties.

For a verb like "drink," a representation like (24a) would require the creation of a new feature "+/- liquid," while the SM-scheme would merely imply the use of the already existing noun-node "liquid." The point is that "drink" is <u>not an exceptional verb</u>. Indeed, most words have rather particular restrictions. Using features would then lead to unbounded extensions. Eventually, lexicon entries would get filled with lengthy description lists. It seems much more natural to make use of the semantic memory, especially as one realizes that most of the required sentences like (24b) above have to be stored there anyway. This is so

because semantic processing requires knowledge of the facts with their logical connections. Thus, instead of stating:

(implies (sells x y) (has x y)) (25a)

as somewhat suggested in [Becker 1969], one might as well replace "x" and "y" with more meaningful nodes. For example, using the nouns for "human" and "concrete" respectively:

(implies (sells someone something) (has someone something)) (25b)

Woods seemed to recognize the advantages of these kinds of 'frames' and their correspondence with semantic features, but he did not develop the idea to any great extent. Indeed, the claim here is not only that the use of a structured semantic network is necessary for analysis, but also that it increases the range of semantic checks to very many different levels. The idea has far more explanatory power than earlier theories when used in connection with a variety of other problems of semantics. The remainder of this chapter will show some of these advantages more clearly.

3.3 COMPLEX SENTENCES.

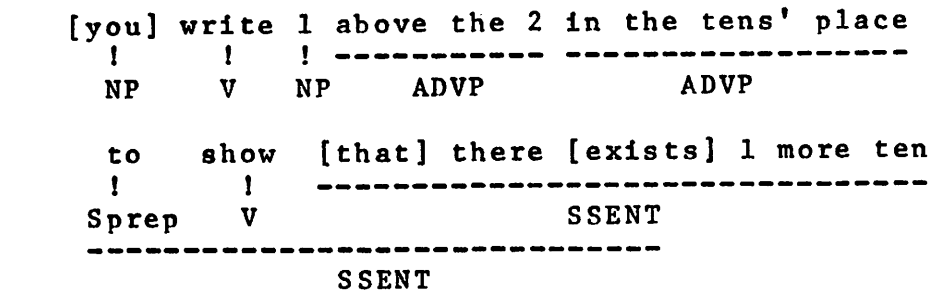
This section examines the problems involved in analyzing more complicated tree structures, in particular those sentences which include several nouns or several verbs, explicitly or implicitly. However, inflections will still be ignored throughout: tenses and modes of verbs, definiteness of nouns, etc., are discussed in section 3.5.

Along with the study of the various problems involved and besides the many examples that will be given to emphasize a particular aspect of language, one sentence will be referred to throughout:

"Write 1 above the 2 in the tens' place to show there is 1 more ten." (26a)

On entry to the semantic component, (26a) has the

following syntactic structure:



(26b)

(only essential nodes are shown here.)

3.3.1 Multiple Noun Sentences.

Application of the SM-scheme to these sentences follows essentially the same course as above. The process remains very symmetric. For each node of the sentence, a match is made to check all the other nodes for compatibility. The search for sentences is pursued upward on V-chains and downward on N-chains. However, one must state more precisely which pairs are to be compared. There is indeed a mismatch between the syntactic noun-verb relationships (e.g., subject, complement) which have been determined already for the input sentence, and the semantic ones (e.g., agent, patient) which appear in memory. The semantic analyzer cannot perform a match simply by associating each noun with any noun at random whose semantic description happens to fit well. In fact, there are fairly restricted rules governing the semantic role of each noun according to its syntactic function and the nature of the verb. For example, if the verb is a process, the subject is the patient, as in (28b-c) below, while for an action-process verb, in active mode, the subject is the agent and the object the patient, as in (28a).

Because of the potential variety of semantic noun-verb relationships, these must be stored explicitly in memory. Thus, a compact representation would have the internal relational nodes reflect the same noun order as in the expected syntactic tree structures (thus, matching by position). Some indeterminacy occurs in a few cases and finding a best match is then an important aspect of the analysis procedure. A simple example of this is the case of a verb like "weigh" which admits the following two descriptions:

(action-process-V:weigh agent-N:human
patient-N:concrete) (27a)

(completable-state-V:weigh patient-N:concrete
complement-N:weight-measure) (27b)

These allow correct analysis of the following sentences:

"Jim weighed several packages." (28a)
"Jim weighed 150 pounds." (28b)
*"A package weighed Jim." (28c)
*"A package weighs 3 inches." (28d)

Not only are these accepted or rejected as appropriate, but in each case, the exact noun-verb relationships are determined, thus providing the ability to give meaningful answers to the questions:

"What did Jim do?" (29a)
"How heavy is Jim?" (29b)

The above example is indeed crucial in showing the importance of proper relationships. Matching by position may mislead one to believe that the explicit semantic relationships are superfluous. It is hence very significant that (28a) above allows (29a) but not (29b), and vice-versa with (28b).

On the other hand, some nouns may have a preposition (or one of a set of prepositions) expected with them. Then of course, the greater information allows more freedom in the noun sequence and correspondence of nouns must be established through comparison of prepositions. This process is further complicated by the traditional problem of correct 'attachment.' One important distinction is made here between what we call 'proper prepositions' and 'adverbial prepositions.' The former are those which are associated with the construction of the verb in the sentence; the preposition determines then the specific noun-verb relation involved. The latter combine with a noun to form an adverbial phrase. These may modify the previous noun phrase or the whole sentence. They do not interfere with the internal structure. Sentence analysis revolves around the decision of whether a particular preposition is 'proper' or 'adverbial.' This is a complicated algorithm. Without going into full detail, one might mention that the decision is based on the set of proper prepositions still required or optional for the particular verb, the set of remaining prepositions in the input sentence (short look ahead only), the particular form (determiner, structure, etc.) of the previous noun-phrase, the existence of a referent for the group (previous noun phrase + candidate adverbial phrase), etc. For example, consider

the following sentence:
"Give an answer to the problem." -->

(30)

(give you (answer you problem))

The SM-scheme first attempts the analysis corresponding to *(give you answer problem) where "to" is mistaken for the preposition introducing the beneficiary of "give." This fails because the beneficiary is expected to be "human." Thus another try is required. The use of transformational reconstruction (as discussed below in 3.3.2-(b)) leads now to a correct interpretation of the preposition and successful parsing.

In the case of our main example (26a-b), the verb "write" has the following description:

(V:completable-locative-action agent-N:human
complement-N:character (31)
*optional-location-N:loc-prep + location)

This verb expects two 'pure' NP's optionally followed by a NP introduced by a "loc-prep" (which designates the set of locative prepositions). When analyzing "You write 1 (above the 2) (in the tens' place) ..." the first ADVP matches "location-N." The second is recognized as modifying "the 2" as follows: "the [Noun] [ADVP]" is an appropriate form, the "loc-prep + location-N" requirement is met already, and finally there exists indeed a "2 in the tens' place." Note the special meaning of the optional locative phrase. Every sentence can terminate with a locative; this is not part of the verb description. An explicit locative is present only when the verb can take a location as part of its own sentence structure (e.g., "write," "put," most motion verbs, etc.). In the analysis, the priorities for prepositions are as follows: (1) required in verb description, (2) optional in verb description, (3) modification of previous NP, (4) global sentence modification.

3.3.2 Noun-Noun Modification.

This type of modification occurs either through a direct N2-N1 combination ("paper flower") or through a prepositional construction NP1-Prep-NP2 ("a box of candy"). In this latter case, the preposition itself may be implicit ("tens' place," "his bank"). The main problem here is that modification implies relation, which in turn means that there must be a verb. This verb is clearly not explicit in the first case. In the second, the prepositions ("of," "by") or the construction (possessive) may correspond to so many

different meanings that they carry virtually no meaning in themselves. Thus, in all cases, the verb must be recovered from the two nouns. This point has rarely been discussed in any detail in previous work. The main reason is probably the fact that deleted words offer a challenging obstacle to non-generative approaches which do not make use of semantic memory for analysis. Generative theories need only specify the conditions and the particular form of a deletion; this is very complex. But recovering a verb is a different matter. It is not just complex; it is impossible unless the verb exists somewhere. (Note here the basic difference with deletions occurring as a result of coordination reduction. There, deleted words can be recovered by comparison of the coordinated structures.) In a noun-noun modification, the verb is sometimes completely absent:

"paper flower" = "flower made of paper" (32a)

"box of candy" = "box containing candy" (32b)

"tens' place" = "place where the digit representing the number of tens is written" (32c)

"his bank" = "bank where he is a customer" (32d)

In the simpler cases, the verb can be recovered from the noun N1. The process then becomes similar to morphological analysis. Thus, the noun may have a direct relationship to the verb with the same root:
a) agent of the verb: the modifier (N2) becomes the patient of a sentence which expresses a property of the main noun:

"traffic director" --> director : (direct * traffic) (33)

Sometimes there exists no verb in English which describes the corresponding action:

"Chancellor of the University" --> Chancellor : (act-as-chancellor-for * University) (34)

The program does not distinguish these two examples, since an elementary node does not have to correspond to one English word.

b) nominalized form of the verb: the modifier becomes the agent, patient, or location in a sentence-node which, as a whole, represents the NP:

"a move by the faculty" --> fact-of: (move agent-N:faculty something) (35a)

"wage raise" --> fact-of: (raise someone patient-N:wage) (35b)

"home call" --> fact-of: (call-at someone location-N:home) (35c)

The SM-scheme plays of course an important role in determining the particular relationship in which the modifier stands with respect to the implied verb. Also, if N2 can assume any of several relationships to the verb, there is an order of priority: patient is first, then agent, ..., whichever matches first. Explicit prepositions may impose restrictions as in (35a).

On the other hand, consider:
"the multiplication of 23 by 19" (36a)
"the product of 23 by 19" (36b)
In both cases, the reference is to the same operation. In (36a), the emphasis is on the performance of this operation, the 'fact', while in (36b), the interest is in the result of such action. Thus, the relation of the noun N1 (here "product") to the implied verb (here "multiply") is not always as direct, but the analysis remains very similar.

Such transformational uses are very frequent. This text itself is a good example. A random examination of this thesis, including this very sentence, will show that "of" is used here mostly in this fashion.

c) In cases where the above rules are unsuccessful, either because the SM-scheme results in rejection or simply because there is no particular verb which has the adequate morphological relation to N1, the missing verb may be one of a list of verbs usually associated with the preposition "of." These include: "made-of," "contain," "belong," etc. (An exhaustive list can be found in any dictionary.) They can be tried in turn, using the SM-scheme, the one(s) eventually retained corresponding to the best match(es). This rule explains examples such as (32a) and (32b) above, and all similar cases:
"the house of his father," "candy stick," "John's toy," etc. (37)

An interesting point is the use of the properties of both nouns as a heuristic to speed-up the choice. Thus "box" will favor "contain," while "cardboard" puts forward "made-of," given that the following structures

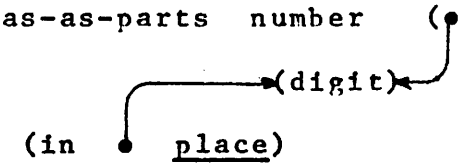
are present:

box ==> container - the class of objects -
container : (contain * concrete-object) (38)
cardboard : (made-of concrete-object *)

Indeed, it is safe to try only those verbs that are brought out by this heuristic as there does not seem to be any counter-example. One problem, not handled by the program, is precisely the correct differentiation between "N2 N1," "N1 of N2," etc. The list of verbs does appear to be identical overall, but similar restrictions on noun-verb relations as discussed in (b) above may impose a choice, or determine rejection:

"a paper cup" (39a)
*"a cup of paper" (39b)
"a piece of bread" (39c)
*"a bread piece" (39d)

d) When all of the preceding fails, the relations attached to the noun N1 are searched for a sentence or a connected chain of sentences containing a class-noun which includes N2 as a particular case. In a sense, N2 is giving more specificity to some property of N1. This is of course the most flexible rule, thus also the most delicate to handle. For example, our main example (26b) contains the expression "tens' place" which can be analyzed given the following memory configuration:



(40)

There is clearly a path between the two nouns. It includes the following nodes:

X0: place
X1: (in X2 X0)
X2: tens'-digit (41)
X3: (X2 X4)
X4: tens

A point which must be strongly emphasized is the influence of one's prior knowledge on the understanding process. Here the previously acquired structures concerning numbers directly determine the interpretation of the expression "tens' place."

Clearly the search cannot be conducted from one end only. What is needed here is a Quillian-like concept intersection process, spreading around from each noun in an attempt to find some meeting point. However, this yields a huge number of intersection

paths. While this shows that a memory network contains a great wealth of information (as pointed out in [Quillian 1966]), it is not really useful in practice. A choice must be made. One restriction limits the spreading around a node to other nodes occurring in the same sentences and to more englobing concepts, thus excluding subsets and instances. Also, a criterion that can be used to compare the various intersection paths is the 'distance' between the two concepts along each path. Simply counting the nodes proves insufficient. It seems desirable associate a distance coefficient to each relational node. This is discussed in Chapter 4, where we also define for each node an 'activation level' representing the relevance of the node to the current subject of discourse. Both points can be used here meaningfully. The length of the path is thus defined as the sum of the ratios of the distance coefficients divided by the activation levels. Comparison of the various lengths reduces the search considerably. Moreover, in practice, by associating shorter distances with the sentences underlying constructions of the type (c) above, a unified algorithm can perform the analyses for (c) (and its heuristic) and (d) simultaneously.

This process seems somewhat ad-hoc, however, and does not provide a definite solution to the problem. Moreover, some expressions are really quite ambiguous (e.g., does an "ice-cream cone" actually contain ice-cream?). Clearly there is no simple answer. One can use an obvious expediency in handling noun-noun modification. If the simpler rules (a) to (c) are not sufficient, the lengthy process of complex relation recovery (d) is temporarily postponed and an association link is created (or, if one already exists, it is updated). Resolving this link into a standard structural relation will be attempted only when necessary: when it is needed for further processing or when the two nouns have high activation levels.

3.3.3 Other Noun Modifiers.

Treatment of NP's where a noun is modified by some adjective or phrase is again essentially a process of uncovering the underlying sentence. Consider the following examples:

"red paper" (42a)
"a 13 mile long bridge" (42b)
"the box in the car" (42c)
"the boy who delivers newspapers" (42d)
"the beads left in the jar" (42e)

In these, the noun is being modified by a simple state-adjective, a completable state-adjective, an adverbial phrase, a relative clause, and a deferred adjectival phrase respectively. In each case, the noun is being further described by a sentence which can be reconstructed according to simple specific rules:

paper : (red *) (43a)
bridge : (long * (13 miles)) (43b)
box : (in * car) (43c)
boy : (deliver * newspapers) (43d)
beads : (leave somebody * in: jar) (43e)

The standard SM-scheme can then be applied to these sentences.

In the case of the example (26a), the only apparent modifiers of this class are the adjective "more" and the quantifier "one" in the expression "one more ten." Both of these are treated here as inflections. Our approach is rather unconventional in that many words traditionally classified as adjectives are considered to be generalized determiners. The intention is to separate state-verbs from all other words that are relevant, in our conception, to inflections (see section 3.5).

The main difficulty in the general analysis of modifiers is the problem of 'attachment': One must decide which elements in a sentence modify each other. When a noun phrase includes a string of adjectives and nouns:

ADJ1 ADJ2 ... ADJp N1 N2 ... Nq

deciding on the appropriate grouping may be difficult. The 'proper nesting' rule and the requirement that the adjective-noun or noun-noun pair should be meaningful is not sufficient. Too many combinations would be accepted. This phenomenon is a by-product of the following general fact: One cannot simply say of two nodes that they are either fully compatible or incompatible. There are varying degrees reflecting the relative likelihood of each combination. This explains why no one would have any difficulty in parsing uniquely the following phrase (taken from the SJCC 1971 technical program):

"(initial operational (problem oriented) (medical record) system)" (44a)

or even the next example:

"((high precision) ((low intensity) (electric current)) measuring device)" (44b)

(Parentheses were inserted to indicate the grouping.)

The solution lies in more adequately elaborate approaches to compatibility, perhaps as discussed in section 3.4.

Yet another problem is the subtle kind of structural ambiguity that appears in some phrases:

"((prestigious international contest) award)" (45a)

"(prestigious (international contest) award)" (45b)

The "prestige" associated with one noun is automatically reflected on the other one. The award is prestigious if the contest is, and vice-versa. This makes the distinction between the two parsings appear somewhat academic. These sentences have almost identical meanings, and this fact cannot be simply ignored. Each structure can be considered as deducible from the other. This alone indicates that the choice might be completely indifferent. When an adjective is equally applicable, with the same meaning, to different nouns, we arbitrarily attach it to the closer one. Note also that the similarity between (45a) and (45b) is emphasized by the same vague impression of "prestige" that surrounds both. This is undoubtedly connected with the idea of general context and node activation discussed in section 4.2.

Prepositional phrases present similar problems. Compare one's natural reactions to the following sentences:

"Time flies like an arrow." (46a)

"I threw the man in the ring." (46b)

Both sentences are somewhat ambiguous. However, the first one is resolved immediately. If alternative meanings are pointed out, the listener is usually surprised. For (46b) (which is discussed in [Quillian 1966], pp. 250-251), unless context strongly favors one of the possible meanings, the ambiguity is striking. Several interpretations are easily produced by the listener.

On the other hand, one may simply ignore the ambiguity at first. This requires a special representation, since no commitment to any particular parsing is intended. The point is specifically a problem of implementation rather than one of linguistic theory. The importance of this matter stems from the usefulness of this short-cut: effort is not diverted into the development of all possible meanings in the resolution process. Only when comprehending the remainder of the text demands it, would the ambiguity need to be resolved. Very often, this will never

happen. In CLET when ambiguity need not be resolved, the sentence will be tagged and the adverbial phrase attached to its whole structure:

#(throw I man)---(in - ring) (46c)

3.3.4 General Connectives.

Some of the remaining problems of sentence analysis are related to the treatment of subordinate clauses and coordination conjunctions. These present a much greater semantic difficulty than other linguistic problems. In particular, they operate on large units (whole sentences, paragraphs) whose nature is still not fully understood.

Subordinate conjunctions are again a form of verb. Their main characteristic is that they are only used in surface structure when the entities on which they operate are complete sentences:

S1 "to" S2 --> (in-order-to agent:S (47)
patient:S2)

as in our example:

"(Write 1 above the 2 in the tens' place) to (26a)
(show there is 1 more ten)."

Sometimes, equivalent ordinary verbs do exist whose operands are simple nouns. Thus:

S1 "because" S2 --> (because agent:S2 (48a)
patient:S1)

S1 "because of" NP2 --> (because agent:NP2 (48b)
patient:S1)

NP2 "caused" NP1 --> (cause agent:NP2 (48c)
patient:NP1)

The standard SM-scheme can again be applied except for the problem of defining the 'match' between two sentential noun forms. In fact, a sentence always represents some form of noun. It is an "action," "fact," etc. and it acquires the features of this noun when used nominally within a larger sentence. (The particular noun seems to be essentially determined by the nature of the verb, but further investigation is

needed before a fully satisfactory solution to the problem is reached.)

Coordination conjunctions are similar to subordinate ones in that they also act on non-elementary noun-nodes. In this case, not only a sentence but a full paragraph (complex node) may be an operand of the verb. Thus, one might contrast two ideas as follows:

(however agent:I1 patient:I2) (49)

where I1 and I2 can have any complexity beyond the level of a sentence. Determination of the exact boundaries of I1 and I2 within the discourse is by itself very difficult. In (49), I1 and I2 are, from all possibilities, the pair which contrasts best. An important heuristic which reduces the number of trials should rely on punctuation and, more generally, layout; these are the written substitutes for intonation, which is beyond the scope of our research.

The only cases that are handled are some of the simpler cases where the limits of coordination are clear from the syntax. Some conjunctions ("and," "or") can operate on predicates and nouns as well as sentences. When operating on predicates, coordination may be ambiguous. Here, it will be assumed that the two sentences could have appeared as two consecutive but separate entities in the text. This will imply a time sequence in the weak sense, as if the two sentences were not coordinated explicitly.

When operating on nouns, coordination establishes a compound structure. As discussed in section 2.4 above, these structures are not exactly equivalent to a merge of two similar phrases into one:

"the sum of 23, 19, and 47" (50a)

*"the sum of 23" (50b)

The combination can always be considered equivalent to a plural noun: whether the plurality of the noun is relevant or not depends on the verb. Thus:

"John and Mary own a house." (51a)

"They got married." (51b)

In (51a), the plurality of the subject (beneficiary) is purely coincidental. In (51b), the situation is exactly opposite. Not only does the verb require a plural subject, but it is even known that exactly two are needed. Similarly, in (50a) above, the verb "add" is constructed with a plural argument where the number is left unspecified. The analysis checks for this in the usual manner, as plurality is just like any other property (see section 3.5).

To be sure, the syntax is sometimes ambiguous as to which use of the conjunction is intended: at the noun or at the sentence level. Hence:

"((Jim stayed for the marriage of John and Mary) and (Steve went to marry Jane)." (52a)

*"((Jim stayed for the marriage of John) and (Mary and Steve went to marry Jane)." (52b)

Regular analysis rejects (52b).

3.3.5 Adverbs.

The general approach was presented in section 3.1 with the description of the structures involved. Clearly, when used as standard sentence predicates, adverbs behave exactly like verbs and there is nothing new to the problem. The discussion here is centered on the attributive use where adverbs are modifying verbs and forming 'verbal' sentences: in these, the adverb is a state verb whose patient is the modified verb. Semantic checks are performed again using the SM-scheme with the obvious replacement of the usual noun by the verb. Some uncommented examples will illustrate the point sufficiently. The set of nodes described in (53a-b-c-d-e) determines the analysis of the next set of sentences as shown in (54a-b-c-d) below:

(slowly patient-V: action or process) (53a)
(==> very intensity-adverb) (53b)
(intensity-adverb patient-V: relative) (53c)

(note: "relative" is an attribute of state-verbs)

"hot," "tight" ==> relative-state-V (53d)
"get-hot," "tighten" ==> process-V (53e)

will result in:

*"John is slowly hot." (54a)
"John is slowly getting hot." (54b)
"The rope is very tight." (54c)
*"The rope is very tightening." (54d)

(Note that the SM-scheme need not be really modified. The fact that a sentence is composed of a verb and several nouns is perfectly transparent to the program. In fact, in the examples above, the very nature of the node "relative" requires a verb for matching, not a noun.)

3.4 COMPATIBILITY.

Compatibility is at the very core of the sentence matching process. The need for a graded evaluation is obvious. This section shows how the necessary features can be implemented. (As was mentioned in section 3.1, nouns are defined by their properties; No particular distinction between nouns and verbs is necessary in this section.)

The general principle of contradiction can be stated very simply: different 'values' on the same 'scale' are considered contradictory. In practice, the problem is not as straightforward as the stated principle might lead one to believe. This is mainly due to the fuzziness of words, and hence of their values. The term "scale" refers to the "abstract axis" idea developed with the basic discussion of nouns (see 3.1.1 above). The whole memory network is organized along these axes, such as "color," "human-ness," etc.

Two examples will ilustrate the basic principle:
- "red" and "blue" are directly contradictory as they represent different values on the axis of "color."
- "table" and "male" imply "inanimate" and "animate" respectively. These in turn result in contradiction on the axis of "animate-ness."

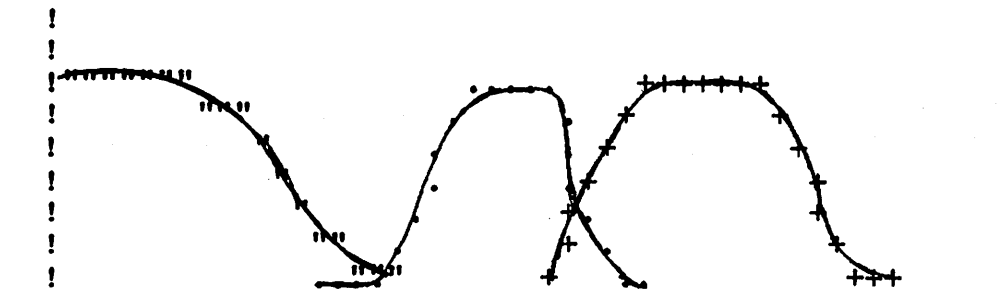
(Note: This is of course reminiscent of n-dimensional spaces. The naturally privileged set of vectors, which is here obviously a generative set, does not form a basis however, since the vectors are not mutually independent. The purpose of the compatibility evaluation component can be considered as simulating a process of projection on some virtual basis. It is not really clear at this point what this basis should be, or whether a fixed basis is even desirable.)

One problem is the great number of axes. It is extremely costly to represent each word by its components along every dimension. When there is dependence, one wishes to determine values through the

specific dependency itself in order to avoid redundancy (considerations of reliability in a large-scale implementation may justify a different approach). On the other hand, there is no point in specifying explicitly for each word the list of irrelevant dimensions due to independence. (These two points constitute important particularities and explain why a standard mathematical space-oriented approach is inadequate, even though the ideas are similar.)

3.4.1 Fuzziness.

Another difficulty is the fact that one meaning of a word is not a point in the semantic space but more like a hyper-volume. Ambiguity is not at stake here: an ambiguous word is represented by several such volumes (often with little or no overlap). This volume is rather the result of intrinsic fuzziness. Fuzziness is an essential aspect of natural languages, one which has been too often ignored. It is a sad fact that there have been few serious attempts at dealing with this problem. One of the more interesting approaches is by Zadeh)Zadeh 1970] in which he proposes to associate with each individual word a graded membership function representing its meaning. For example, for the concept "age," one might identify the following distributions:



"""" : young
... : close-to-middle-age
+++ : middle-aged

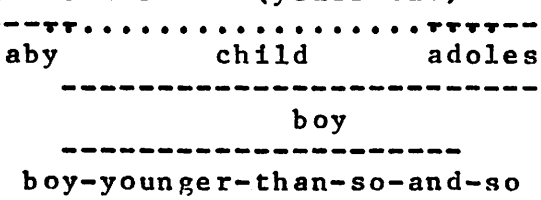
Even though such a paradigm is ultimately desirable, considerably more research is needed before attempting a broad-based implementation (it may be significant in this respect that Zadeh himself did not attempt to experiment with the concept on any full text). Considering the immediate goals of our research, it was decided to approximate Zadeh's concept through the use of intervals. The way these intervals are used in the processing reflects a simple-minded image of

probability. Very little is known about the nature of the appropriate distributions in any case. The 'compatibility coefficient' between two intervals of lengths 2a and 2b where the centers are separated by a distance d, is evaluated as:

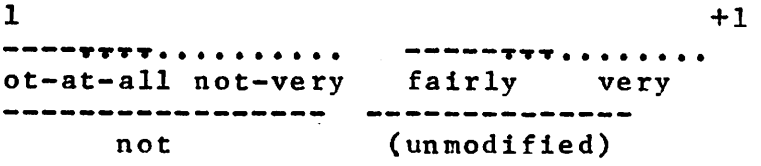
$$C = 1 - d/(a + b) \qquad (55)$$

(There is a degenerate case where the intervals are both reduced to a point, or a = b = 0. C is then either defined to be equal to 1 if both points are identical (d = 0) or a large negative number otherwise.)
Note the following features:
- the sign of C is positive when the intervals overlap and negative otherwise.
- C has no dimensions. The use of a ratio of distances provides automatic normalization of the metric over all axes.

Each axis corresponds to the range of some variable (not necessarily a lexical one), and on every axis, we permit the description of various intervals with varying degrees of precision. For example,



Note however, that the specification of real numerical values may sometimes be arbitrary. In this case, the interval)-1,+1] is used for the representation. E.g., for the adjective "courageous":



The most important use of these axes concerns relative-state and completable-action verbs, which have an elaborate range of values. Other verbs, in their pure form, have only affirmative and negative values. (Even this is subject to change through learning; e.g., is a virus an animate noun?) But all verbs can be modified by adverbs: intensity adverbs modify relative verbs along the verb's own axis. In essence, an adverb is an operator on the interval. In other cases, adverbs

introduce a new dimension along their own axis.

Modification of the interval corresponding to a verb also takes place during application of the SM-scheme. This was hinted at in section 3.2.2, but it can be made clearer at this point by using an example:

"The road is wide." (56a)

During the verb-pivot phase, "road" is matched against the patient of the unmarked word "wide" (corresponding to a very large fuzzy interval). Upon success, properties of the noun "road" are searched for the noun-pivot phase. One of these will be:

(has-road-width road) (56b)

which interferes with the previous choice by simple restriction of the range of possible values. In the minimal first example of section 3.2, "colored" and "has-animal-color" played similar roles. This "norm" phenomenon is very common in all languages. The present model permits a natural treatment of the problem.

3.4.2 Evaluation.

When matching two concepts C1 and C2, all components along all dependent axes must be determined. All ancestors through ==> relations are assembled, together with all properties (verbs) operating on the concept as a patient. These are adjectives, quantifiers, adverbs, etc., which appear in relations of the form:

(V patient:Ci)

where Ci is the concept under study. Thus, for each of C1 and C2, a semantic 'volume' is developed. On those axes where both of these volumes have an explicitly specified interval of values, compatibility coefficients are computed according to the above formula (55).

If, however, C1 has a specific value along a certain axis A, whereas C2 does not, there are two possibilities: C2 in fact covers the whole axis A (e.g., C2 = "concrete" is compatible with any value on the axes A = "human-ness" or A = "length"). Otherwise, C2 may be completely incompatible with the whole A axis (e.g., C2 = "abstract" and A is as before). This problem is characteristic of the type of hierarchical structures used here. It is implicitly assumed that a concept represents the logical conjunction of all its ancestors and other attached properties. Also siblings are meant to reflect disjunctions, but here is where the problems arise. Should one list "male" or "female"

as an explicit property of "human"? The purpose is clearly to say that "human-ness" is compatible with "sex." One might think that the very existence of the noun nodes for "man" and "woman" are a sufficient representation of this fact. In general, a first check is made to see whether there exists a concept at the intersection of both axes. This is a particular case however. Fundamental concepts such as "man" and "woman" are likely to be present in a semantic network anyway. In other cases, one might prefer a disjunctive list. A "tent" could be a "camping tent," a "nomad tent," or a "circus tent" (but not a *"processor tent"). Only when "circus tent" is encountered would a special node for it be created, the list being then used as a check of semantic acceptability.
(Note: The choice between the two possibilities can be made to dynamically change through experience: nodes would not be deleted immediately after a short text has been processed but rather on a "garbage collection" basis. If it is found that the different kinds of tents, for example, are frequently encountered in one's particular semantic universe, then one would revert to the first mode and vice-versa. This is closely connected to the differences in lexical specificity that occur between languages depending on their physical and cultural environment. Some of the most cited examples of this phenomenon are the number of Eskimo words for snow and the number of French words to describe the taste of wine.)

Compatibility coefficients are thus computed on all axes relevant to C1 and C2. The final overall value to be returned is the algebraic minimum of all coefficients. Thus one incompatibility resulting in a negative C will insure a final negative value.

At this point, one can sum up precisely all the checks performed during the analysis process:
- "Verb pivot" phase: the noun from the input sentence must be included in the class noun from the sentence descriptor.
- "Noun pivot" phase: verbs and nouns from the network sentences are compared with the corresponding nodes from the input. If they include them, the sentence is ignored; if they are included, they constitute restrictions. Otherwise, there is incompatibility.
- Reference match (see Chapter 4): the reference must include the candidate referent.

3.5 INFLECTIONS.

In prior attempts to formalize the analysis of natural languages, linguists have frequently tended to ignore inflections. It is important to realize and take advantage of the similarities that exist between two sentences which differ only by the inflections of various nouns and verbs. Previous sections of this thesis have attempted to follow this rule; indeed, this may appear sufficient as long as one processes only individual sentences. Processing the information contained in inflections is an essential step toward discourse analysis. Inflectional analysis relies not only on conventional articles, determiners and the like, but also on all relevant adjectives and adverbs ("other," "more," "together," etc.). Inflections fall into two categories: those which carry intrinsic information ('verbal inflections'), and those which simply direct the analyzer in using the appropriate nodes within the semantic memory network ('structural inflections').

3.5.1 Verbal Inflections.

An inflection may take the role of a verb in modifying a noun, or that of an adverb in a verbal sentence when modifying a verb. For instance, in the case of tense inflection with verbs, there is clearly no reason why one should make a fundamental distinction between a time adverbial phrase and a tense, since they both basically indicate the same marker. The former simply supplements the latter which is less powerful in its range of expression. Similarly, why would one make a difference between those so-called "intrinsic" properties of a noun, such as "animate," and, say, its plurality? One might suspect that the distinction was drawn because compatibility matching rarely takes place along the "number" axis (which includes "plural" as a fuzzy interval). This, de facto, diminishes the importance of this particular axis for semantic analysis. Here are some examples, however, where the inflection is crucial:

*"The Smith married the Jones." (57a)

*"Paul compared the house." (57b)

*"Jim goes yesterday to the store." (57c)

The uniform approach adopted here permits rejection to be determined by the SM-scheme as usual (see, in particular, sections 3.3.4 and 3.3.5). In short,

"present," "past," "future," "perfective," and "progressive" are 'adverbs' operating on the time axis. Modals are handled in a similar manner. "plural" is a verb, and so are all quantifiers ("many," "217," etc.). Note also that many inflections discussed in the next section ("a," "some") also carry quantity information.

3.5.2 Structural Inflections.

These derive from three basic factors: generic, definite, and dependent.

a) Generic: A verb is generic when it expresses an unending continuum of events as opposed to a single event, i.e., when the corresponding sentence expresses a characteristic of some noun as opposed to an accidental fact. A noun is generic when it refers to a class of similar nouns as opposed to a particular individual or subset of that class. Thus:

"John sings." (verb generic) (58a)
"John is a student." (generic verb: student) (58b)
"Men are mortal." (noun and verb generic) (58c)

No definite characterization of generic verbs in terms of their surface structure is known. The rule used is approximative. G is computed as a ternary predicate:

G(V) = if this is a subordinate verb and the principal verb is not generic, then "no" (this applies to adjectives, quantifiers, verbs from subordinate sentences, etc.).
if tense=past or tense=present modified by a modal, then "no."
if tense=present with perfective or progressive inflections, then "somewhat."
if tense=present with no inflections or modals, then "yes."

(This assumes the usual past tense style for most of the text. Rules would have to be elaborated for present style discourse.)

The rules for nouns are even more important in practice. A general rule is that a sentence may include a generic noun only if the verb itself is generic.

G(NP) = if there is no leading "a," "the," and no special inflectional adjective ("other," "more"), and no quantifiers, and the noun itself is not a number, and the verb is at

least "somewhat" generic,
or if there is a leading "a" and the verb is definitely generic,
or if there is a leading "any",
then "yes," NP is generic,
else "no." (special adjectives require particular processing.)

Note that the third alternative above ("any") may generate clashes with a non-generic verb:
*"Any book was on my desk." (59)

When NP is generic and simply an unmodified noun, the node for that concept is used to represent the class. If NP is in fact a more complicated structure, then, being generic, it actually designates a sub-class. A new node is created as a son of the total class node of the head of NP, with a ==] relation linking them together. When NP is not generic, it designates a particular 'instance' from the whole class. The choice of the node to be used depends of course on whether this same instance was identified beforehand in the discourse (see (b) below). If a new instance is to be created, the corresponding generic node is determined, or created, as above, and the new node is established as its son, with an "INSTANCE-OF" link between them. In the case of a previously mentioned instance, the same node is used. (Ideally, since it cannot be ascertained whether there was actually a reference, a better solution would be to create a new node and an equality sentence linking them with a certain probability. This introduces great complications and was not attempted: see section 5.2.1-(d).)

On the other hand, verb instantiation is a rather subtle concept. Here, the occurrence of a verb will be understood to refer to one principal instance node shared by all sentences. The idea is essentially that particular sentences do not determine properties of the verb in general (thus only metalanguage sentences would refer to the general node). The generic aspect of a verb is represented with other time elements or tenses. New instances of a verb are only created in the presence of adverbs. Thus a different instance is used for each occurring combination of the verb and modifying adverbs.

b) Definite: Nouns may refer to an object identified by both speaker and listener ('2-Def'), or simply by the speaker ('1-Def'), or by neither ('0-Def'). Hence:

"I am looking for the elephant." (60a)
"I am looking for an elephant; when I find it ..." (60b)

"I am looking for an elephant; when I find one ..." (60c)

These inflections are essential for the study of reference and thus of normal connected discourse. Earlier studies have too often been content with a grossly oversimplified rule such as: "the" implies definiteness, thus a reference, everything else does not. Here are a few counter-examples to justify a more elaborate approach:

"There are 15 candies. Put 10 of them into a box. 5 candies are not in boxes of 10." (61a)
(the underlining indicates a reference.)

"The dog is a domestic animal." (generic, not a reference) (61b)

"You are to find the sum of 26 and 19." (61c)
(The last sentence shows "the" used as a true determiner. It is known that there is only one sum for any two numbers. There is no intention of implying that this sum was mentioned earlier in the discourse.)

In this implementation, no distinction is made between 1-Def and 0-Def. A noun phrase designates something new (1/0-Def) or old (2-Def). It is also assumed that listener and speaker share the same classes of concepts. Thus the idea of definiteness only applies if the noun is not generic. With this primary condition, a leading "the" designates old information unless it heads a transformed construction of the type discussed above in 3.2.2-(b), such as (61c) above. In this latter case, "the" simply means that the designated object must be unique. (This is not checked for in CLET; all functions are assumed to yield one single value for given arguments.) If the non-generic noun phrase is not introduced by "the," there is some ambiguity. Looking back at (61a) above, note that "5 candies" implies an indirect reference. However, this is only due to context: removing 10 out of 15 candies leaves us with precisely 5 candies. The speaker relies on the listener's intelligence to realize that the same 5 candies are talked about. Of course, if the same expression appeared at the beginning of the paragraph, or if it simply had no referent, then it will be understood to be new. Conversely, whenever there is an equal node, an undetermined noun phrase

will be considered old information. The main rationale for such a strong rule is that discourse is essentially connected. A priori, one may assume that all sentences are related in some way. (For further discussion of reference problems, see section 4.1.)

c) Dependent: Nouns may be specified by themselves either through reference (2-Def), or through 'introduction.' In the latter case, the object is not particularly distinguished except by what is being said about it. Thus:

"I saw the cat." (62a)
"I saw a cat." (62b)

In (62a), there is a definite reference to one and only one cat, while in (62b) a new cat is being talked about. It could be any cat, except that this one is the one that I saw ("cat" is clearly not generic here).

On the other hand, the occurring noun may depend on further specification of another for its own determination. This happens mostly in 'distributive' situations where the sentence includes mixed quantification, the universal quantifier preceding an existential one. This is very dependent on the linear order of the quantified nouns in the surface structure, which is one essential reason why all syntactic transformations involving inversion of noun order (passive, interrogative) must be delayed until this phase of the analysis. (Whether the inversion is actually performed and the nouns adequatedy tagged is merely a detail of implementation.) The following structure has different meanings depending on the original order of nouns:

(give agent:a-girl patient:something
beneficiary:every-boy) (63)

A subtle problem lies in the determination of the dependency function "f" whenever there is a situation such that N1 = f(N2): f itself may or may not be known to the hearer and/or to the speaker, irrespective of the definiteness of the nouns N1 and N2. In practice, one does not seem to be very disturbed by the vagueness of the function f. At most one might be led to compare the cardinality of the sets indicated by N1 and N2 to get a rough idea of how many elements of each set are involved in each particular relation. This is sufficient for those problems revolving around simple multiplication or division operations.

Dependence phenomena are quite frequent, even though mixed quantification may not so often appear explicitly in the surface structure. In particular,

when induction is performed, the generalization may introduce quantified nouns to replace the individual objects appearing in each case. Thus, (63) above could very well be an induction from particular statements involving specific boys and girl(s). Another example must be cited because of its frequency in algorithms: whenever a loop is formed by merging similar sequences of actions, some nouns at least become dependent on the index controlling the loop. These matters will be discussed again in connection with learning.

3.6 EXAMPLE REVISITED.

In the midst of the discussion, one could only get an incomplete idea of how all the components described in the previous sections may interact in cooperation for sentence analysis. Thus it seems desirable to expand further on our main example for this chapter (26a-b):

"Write 1 above the 2 in the tens' place to show (26a)

there is 1 more ten."

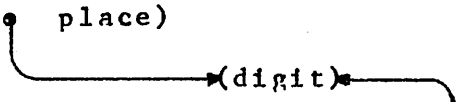
Here, the imperative verb is not generic (imperative mood being equivalent to a modal) and neither is any of the embedded subordinate verbs. Therefore, none of the nouns can be generic. Definiteness occurs throughout because previous context singles out a particular "1."

Various aspects of the analysis of the phrase: "the 2 in the tens' place" were separately discussed previously. Let us consider the analysis of the underlying sentence:

(in 2 the-tens'-place) (64a)

where "the-tens'-place" is identified to be:

(in • place)

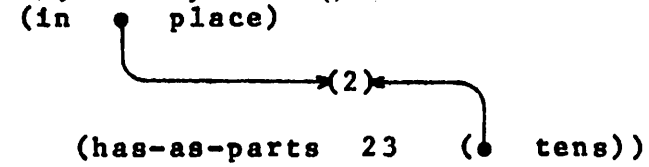


(64b)

(has-as-parts number (• tens))

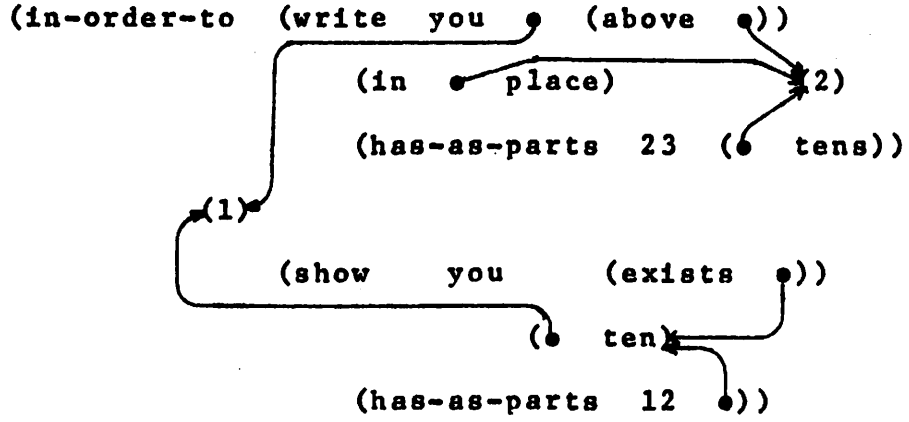
Note that both "2" and "tens' place" await further specification: the text may have mentioned several 2's beforehand, and there is a whole column in the arithmetic grid where the tens of various numbers are placed. Both nouns help specify each other through

(64a), thus yielding:



(64c)

On the other hand, the presence of "more" in "1 more ten" calls upon special processing which determines that this node is the most recent "ten" occurring in addition to the older ones. This avoids the interpretation given normally to "one ten" as an element of a group of tens that could have been mentioned previously (again, see section 4.1). Finally, previous context was the addition of the ones' digits of "23" and "19" resulting in "12" which is 1 ten and 2 ones. This provides the background for identifying "1" from (26a) with "1" in "1 ten." At the end of the analysis, the internal structure for (26a) will be:



(65)

CHAPTER 4
DISCOURSE ANALYSIS AND LEARNING

In this chapter, we present two aspects of higher level semantic analysis: generalized reference and context setting. The discussion then proceeds through deductive inference to one of the most fundamental semantic processes: learning.

4.1 REFERENCE.

If one were to single out the most characteristic feature of intelligent (and intelligible) discourse, one would most likely state "connectedness." Influence of context on our comprehension of language is tremendous. It is unfortunate that Chomskian theories on sentence structures have overwhelmingly bent linguists into forgetting reference. The whole point of language is to communicate about the real world. Yet, few researchers made any attempts to deal with this crucial problem. Coles' work [Coles 1967, 1968, 1969] is an interesting counter-example to this trend.

By "reference" we mean here any instance of that ubiquitous linguistic phenomenon of making various assumptions about the listener's prior knowledge of the world. A global form corresponds to the expected knowledge of universal facts that the listener is presumed to have before starting the conversation. The more local form involves that knowledge which the listener is supposed to have acquired from the particular dialog being conducted. The word "context" is usually restricted to this latter form. In CLET, previous knowledge and current context are all

integrated in a unique memory format. The previous chapter showed how this semantic network provides the basis for the analysis of sentences, thus performing 'global' reference. This section will concentrate on the more local aspects of reference.

4.1.1 Anaphoric Reference.

In its narrowest sense, reference is a mention of an object (the "referent") which was explicitly introduced earlier in the discourse. This is anaphoric reference. The noun may simply be repeated, or some paraphrase may be used. Thus, a combination of descriptive features hopefully permits unique identification of the implied object. The head of the noun phrase indicates a class or set which includes the particular object as a member, the other properties designate the one to be identified. One 'property' is always present, if implicit: the object has been mentioned already. The reference may designate a whole class of nouns as long as this implicit property precludes any confusion. For example, "John" may be referred to as "the boy" or "he" (note that "he" can be considered approximately equivalent to "the male human"). In addition, especially in the case of pronominalization, syntactic features come into play in determining the referent. Hence, there is no ambiguity in the following:

"John hit David. He was very angry." (1)

Here, the pronoun used as subject refers to the subject of the previous sentence. An interesting study of the numerous rules involved appears in [Baramofsky 1970]. Finally, when a plural noun is used, there may not be a unique referent. Several objects mentioned earlier can be referred to as a group. The reference is then a superset for all the intended referents. For example,

"Jane collected 23 butterflies, Sue 19, and Lucy 14. The girls ..." (or simply "they," which is equivalent to "anything plural"). (2)

4.1.2 Indirect Definite Reference.

More interesting cases are those when the reference is not as direct as above. In the following example:

"You are to find the sum of 26 and 19.

Add the ones. ..." (3)

the underlined reference could be paraphrased:

"the 6 ones from the number 26, and the 9 ones

from the number 19." (4)

Clearly, any reasonable attempt at semantically analyzing the reference should bring out the structure implied by (4). In fact, the program's knowledge of numbers and their decimal representation is reflected in its memory network. e.g., the number 26 is involved in the 3 relations indicated below:

```
26 :  ---INSTANCE-OF---  integer
      !
      !(has-as-parts  *  (6  ones))
      !
      !(has-as-parts  *  (2  tens))
```
(5)

Similar structures involve the number 19. In (3), the reference assumes some relation to previously mentioned nodes, but the relation is left implicit. In the next example:

"The coming football game is a very popular one.

The tickets are all ..." (6)

the reference is to football game tickets. Again, there is an implicit relation. Conceptually, there are great similarities between indirect references and the noun-noun modifications discussed in the previous chapter. However, there is a major additional difficulty here, since not only a path between two nodes must be recovered, but even one of the end points is unknown. Analysis cannot proceed in the same manner. The referent is here searched for among the currently 'active' nodes.

As shown in the next section, these nodes are precisely all those having some relation to the concepts explicitly mentioned in the discourse; this is very dependent on the structures present in the semantic network. Interestingly enough, it has been this author's experience to notice important differences between individuals in this respect. After letting his thoughts wander, one speaker may fully explicitate all the necessary transitional steps from the previous subject of discourse to the next. On the other extreme, he may suddenly use the referential "the" in connection with a completely new object. This

may startle the listener, unless extensive common experience has led them to have very similar semantic associations in their memories.

4.1.3 Indefinite Reference.

"In English scientific and technical articles, about 90 per cent of the sentences contain at least one anaphoric expression." [Olney & al. 1966]. But reference is not restricted to explicit manifestations such as definiteness and pronominalization. Often, an implicit set-membership relation may be involved. After a group of objects is mentioned, an unmarked occurrence of a potential member actually implies that membership does hold:

"Three girls collected butterflies. Kay had 37, Ann ..." ("Kay" = one of the "girls," and the same for "Ann") (7)

"Many people were assembled in the lounge. Suddenly, a woman started screaming." ("a woman" = one of the "people") (8)

Conversely, a factual sentence may be followed by a generic assertion concerning the whole class of objects occurring in the first sentence. In this case, of course, the membership relation is forced by the structure of the semantic memory.

Still, another phenomenon is an extension of this membership type of reference. When repeatedly within the discourse, several instances of some class of objects have the same property, later instances of the same class will automatically be assumed to possess that same property. It is exactly as if the first few instances implied a reference to a virtual sub-class having this particular property, and if later occurrences were assumed to belong to this sub-class in the same fashion as above. Hence, in a text where initially some boxes are mentioned to contain exactly 10 candies, and no other boxes are mentioned, boxes occurring later may be assumed similar to the first ones a posteriori. Normally, no check is performed. But if there is an ambiguity, such a fact may be used in resolving it. In the example below (for syntax, see Appendix 5), (9a) is rejected leaving (9b) which is the correct parsing:

*"He took 2 boxes of (10 shells and 7 shells)." (9a)

"He took (2 boxes of 10 shells) and 7 shells." (9b)

4.1.4 Example.

Finally, let us look at the references involved in a typical sequence. The following paragraph is taken from the first page of our text (see Appendix 1):

(a) You are to add 26 and 19.

(b) Add the ones.

(c) There are 15 ones.

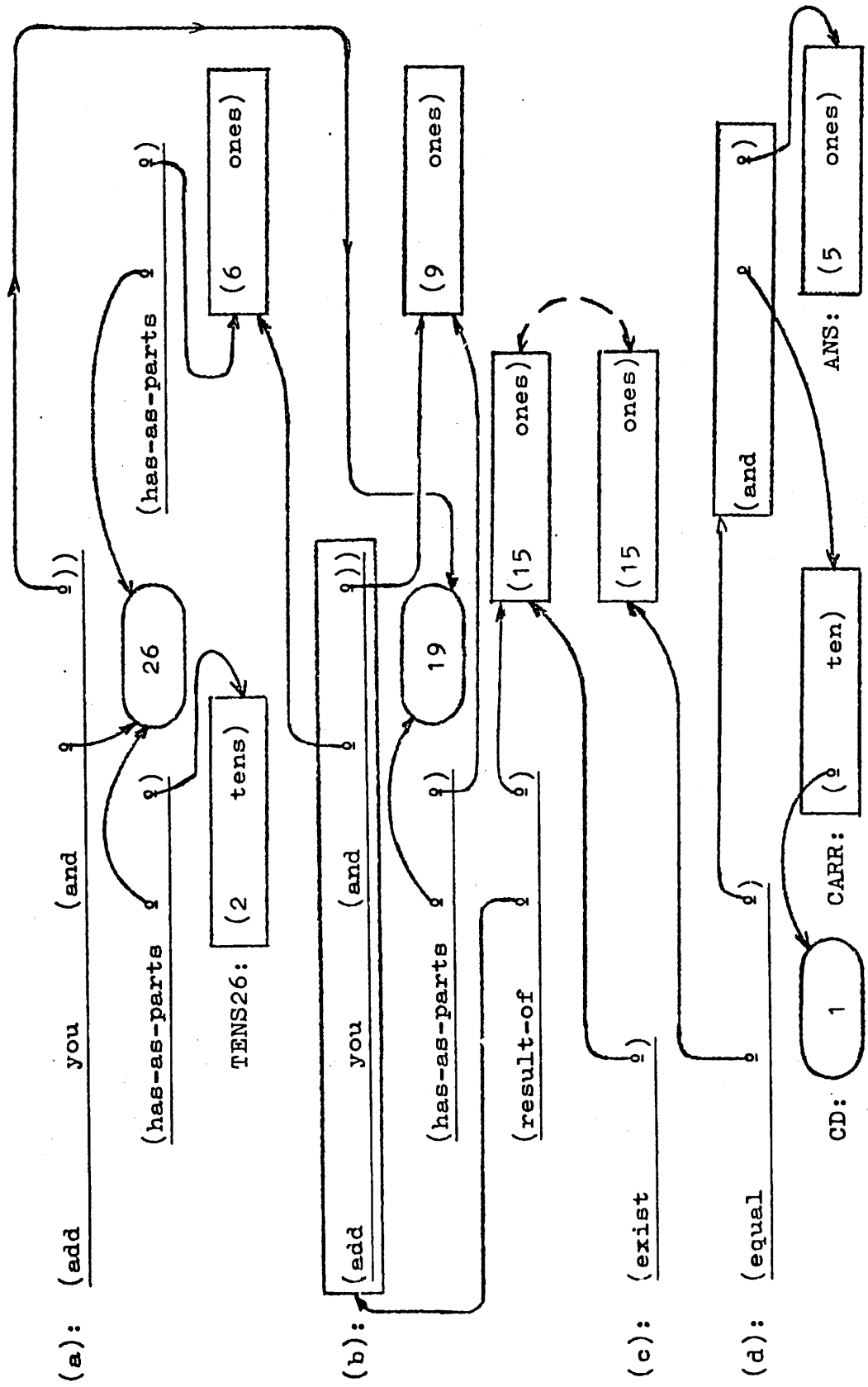
```
  1
2 6     (d) 15 ones are 1 ten and 5 ones.
                                              (10)
1 9     (e) Write 5 in the ones' place of the
---
4 5         answer to show there are 5 ones.

        (f) Write 1 above the 2 in the tens'

            place to show there is 1 more ten.
```

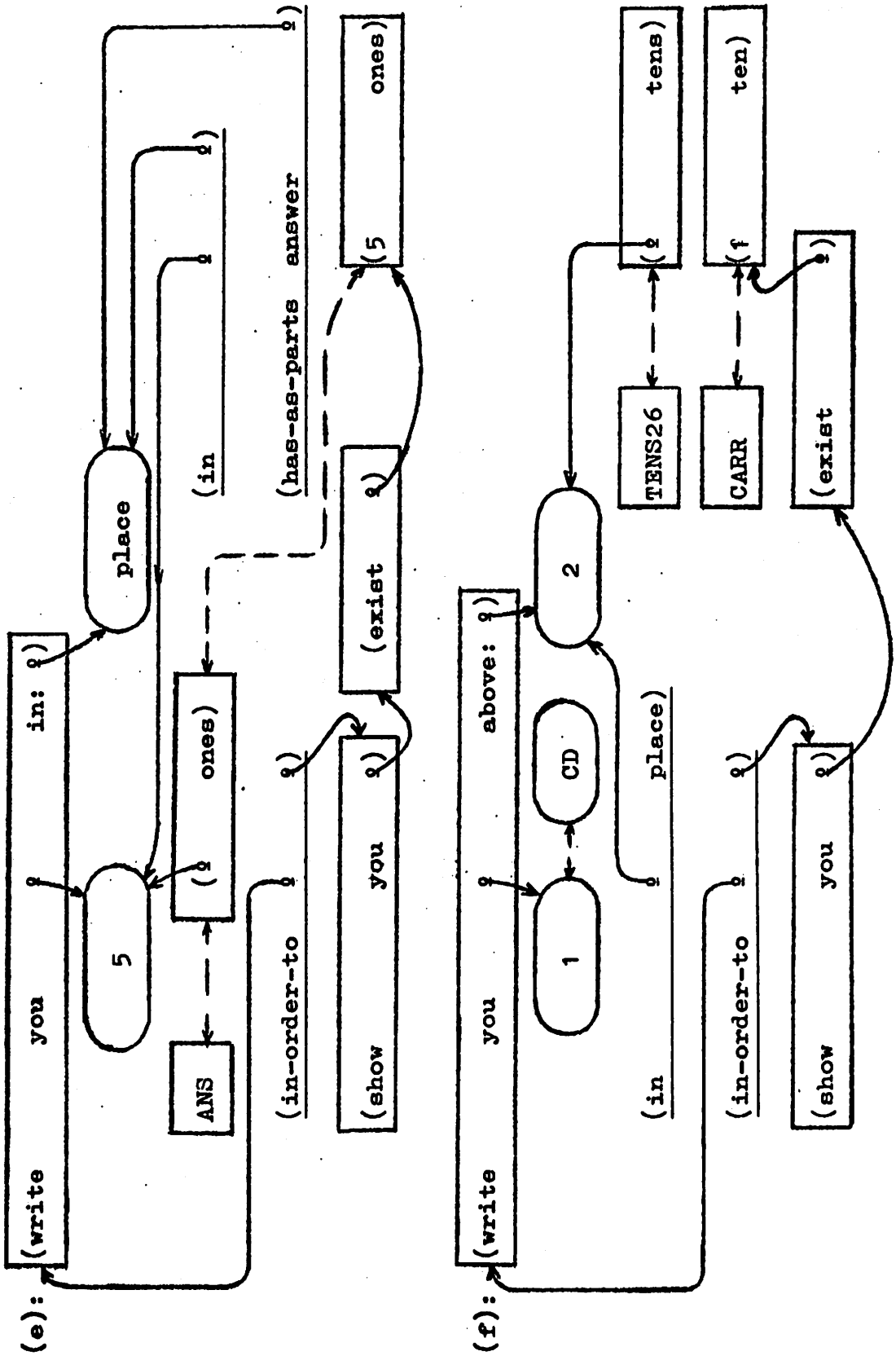
Like all imperative sentences, (a), (b), (e) and (f) contain a trivial pronominal reference to the listener. There are no other direct references. The occurrence of "the ones" in (b) was discussed above. Another indirect reference is "the answer" in (e). Implicit here is the preparation for addition which is requested in (a). The system 'knows' that it should write down the numbers in a grid, with a line below them, and it expects the answer to be formed in the bottom row. (This is implemented by using 'immediate consequence' relations as discussed later in section 4.3.) The node "answer" is thus activated from (a). A third example appears in (f). The previous chapter (3.6) showed how the analysis of "the 2 in the tens' place" brings out the fact that "2" designates "2 tens." This is the key to finding the correct referent which is part of the number 26, as for "the ones" in (b).

Other cases are ambiguous. Quantified simple nouns occur identically in different sentences but it cannot be determined from the surface structure whether the intention is to refer to the same nodes or not. One example is the "15 ones" in (c) which are the same as those understood from (b) and (d). Here the addition requested in (b) results in 15 ones. The check for their existence connected with the declarative (c) determines identity of nodes. On the other hand, some other equalities cannot be established definitely. For example, sentence (d) is generic; it expresses a general property of its nouns. But these are not

generic. One can only note the equality of the "15 ones" in (d) and in (c) as a complex node constraint. Similar arguments apply to "5 ones" in (d) and (e), "1 ten" in (d) and (f), and finally to "5" occurring twice in (e), and "1" twice in (f). These complex node constraints are eventually resolved after many examples are processed. When the same equality is noted repeatedly, the attached confidence level keeps increasing. Finally the two nodes are merged. In fact, this points out a weakness of the program. While it is true that sometimes reference cannot be established, the cases above would be recognized as true references by anyone who really understands the discourse. (But perhaps this understanding means precisely that one has definitely resolved one's complex node constraints.)

The analysis of all sentences in (10) into the standard memory format is shown in the figure below. Sentence labels (a) to (f) are repeated next to the appropriate structures. Note the great amount of implicit information which is recovered and the intricate structural organization as opposed to the linear input string.
(In the figure, interrupted lines connect nodes whose equality is noted as an external constraint. Also note that some nodes have been labeled with a name in capital letters and a ":" in front to allow for references on the second page.)


(a):
(add you (and
(has-as-parts
(has-as-parts
(6 ones)
TENS26:
(2 tens)
26
(b):
(add you (and
(has-as-parts
(9 ones)
19
(result-of
(15 ones)
(c):
(exist
(15 ones)
(d):
(equal
(and
ANS:
(5 ones)
CARR:
(ten)
CD:
1


(e):
(write you in:)
5
ANS
(ones)
place
(in)
(has-as-parts answer)
(in-order-to)
(show you)
(exist)
(5 ones)
(f):
(write you above:)
1
CD
2
(in place)
TENS26
CARR
(exist)
(tens)
(ten)
(in-order-to)
(show you)

4.2 GENERAL CONTEXT - NODE ACTIVATION.

4.2.1 Activation and Reference.

Often, the above rules of reference are implemented through the use of a "stage" where objects (players) go in and out, the current "scene" being the state of mind of the speaker/listener at any given moment. In a story-like text, there may be a first paragraph specifying the general decor (time and place) and possibly some generic facts. The beginning of the main body of the story, which marks the end of the "stage setting," is normally indicated by some restrictive time adverb (typically "One day, ...") or simply by a sentence which is clearly non-generic. Quite often in non-literary texts, the first part is completely missing. In particular, in the examples given to illustrate arithmetic operations, the story starts immediately with a non-generic past tense. No particular setting is specified. In any case, after the optional introduction, non-generic verbs become the unmarked case. Objects enter the stage the first time they are mentioned. They fade out with time unless they are mentioned again.

This stage method is but a particular aspect of the general problem of context setting in all its subtle forms. In this model, each occurrence of a word within the text results in a wave of 'activation' spreading from its node towards all its connections in the memory network. These activations are represented by a number (from 0 to 127) attached to each node, called its activation level. The mentioned node receives some value, here 32. Immediately connected nodes receive a fraction of this value, and so on. These values are added to whatever the preceding level was. On the other hand, these numbers are assumed to diminish in time. Rather than using a real clock, one can use standard sentence boundaries as a substitute time reference. At the end of each sentence processing, all activation levels are divided by some constant, e.g. 1.5. (The use of linked rings for equally active nodes helps the implementation considerably.)

'Active' nodes play an obvious role in the analysis of reference. The referent is searched for among the nodes in the same subset, the most active fit, if any, being chosen first. Otherwise, the search examines the most active nodes, choosing the first one which is compatible. The range of permissible constructions can be restricted by requiring the activation level of the candidate referent to be greater than some preset threshold. One might think that the use of a simple limited-length queue (first in/first out list) containing the recently mentioned nouns would work just as well. This is only true for the most trivial references. Activation type techniques are essential in the analysis of the more complex cases of definite reference. The connections of the memory network are responsible for indirect references. Some examples were mentioned in the previous section: numbers and their digits, setting up for addition and the expected answer, etc. Another one follows:

"The giant tumbled into the pit. When he hit the bottom, the impact ..." (11)
(note: "hit," "bottom," "impact")

4.2.2 Activation and Relevance.

More importantly, activation is fundamental to the general problem of abstracting. The higher level active nodes (repeatedly activated from 'below') determine the general topic of the discourse. This may also help in disambiguation. Between two nodes corresponding to the same word which are otherwise equally acceptable in the semantic context of the sentence, the node which is eventually chosen is the one whose supersets are more 'active' in the current discourse. A general idea of the topic under discussion may serve to give the semantic analysis a more selective orientation. In those cases where we stated earlier that some relations could be left unspecified (e.g. some noun-noun modifications), this would only apply to those constructions that do not appear particularly relevant to the main topic. The relation would be investigated further if either node's activation level is above 9 and neither is 0. In a text on arithmetic, determining the exact meaning of "of" in an expression such as "the place of tens of the answer" would be an absolute requirement. But the two relations involved in "sheets of drawing paper" may be neglected.

Node activation can be used to detect irrelevant sentences which are thrown in in the middle of a text. Rejection would then be triggered by vacuous intersection between the sentence activation sphere and the current levels. A 'relevance factor' may be defined as the average activation level of all words in the sentence. In a more immediate fashion, low relevance can be used to eliminate from an example those idiosyncracies which do not contribute to mathematical understanding. In particular, some examples are simply worked out in more detail than others. The teacher often tries to help the pupil by relating the steps in the computation to corresponding steps involving objects from the real world. However, the child's attention is specifically directed to concentrate on the mathematical steps. Thus, he focuses on the abstract entities, cardinal numbers at this level. The mathematical world is of course involved in any sentence containing quantified nouns (e.g., "Put the 9 candies with the 6 candies."). However, it is clearly much more involved when the nouns themselves are dropped and the numbers directly dealt with (e.g., "Add the 9 to the 6." or "Add the ones."). The process of node activation permits easy handling of this problem. During analysis of marginal sentences, the 'relevance factor' will result in a low value, and thus low confidence levels will be attached to these sentences. They do not become part of the induced algorithm when comparison of complete examples fails to match them. In fact, to simplify the child's understanding, the same sentences are repeated in both their real-world and abstract forms. The program takes advantage of this by simply deleting the less relevant one. Ideally, the sentences should not need to be repeated in their two forms. Rather, the program (or the child) should be able to generate the abstract relations corresponding to the real world statements. This goes much beyond the scope of our thesis.

4.2.3 Activation Specification.

Deciding what nodes should be activated is a delicate matter. There is a need for a fixed list whose nodes remain active throughout the text. This represents the title, sometimes the heroes of the story, in general whatever may justify some a priori expectation as to the subject matter of the text. On the other hand, activation occurs dynamically during processing. After each sentence, a list is set up. This must obviously include all words occurring explicitly

in the sentence. Also, if the sentence involves some implicit relations or some inference, it seems appropriate to include all the nodes that participate in the analysis process. One would like to do the same for those nodes that are involved in the induction. A node would be activated because it is 'similar' to another node in a 'similar' example. This is not possible here: the learning process is kept separate to permit an easier grasp of the whole system. On the other hand, all nodes are not equally important. This is crudely approximated by having a 'high' and a 'low' list (initial increments of 32 and 8 respectively). The first receives the top level verb and nouns of a sentence while the second receives the adjectives, the implicitly restored relations, etc.

Activation levels spread around from the initial nodes in a fashion that needs further specification. The fraction transmitted from one node to another decreases with the distance between them. But this distance cannot be taken to be simply the number of nodes in the path. For example, one would like to say that the word "mathematics" should activate many concepts of the mathematical world, at least to a depth of 3 or 4 nodes, but not too many words connected through the common superset of abstract entities. Even the above needs further elaboration since one's reactions are very different depending on the overall topic of the text: mathematics or philosophy of abstract concepts. Resorting empirically to intuition is here particularly hard since (as the previous remark shows) the additive nature of activation greatly perturbs one's impressions. One solution might consist of considering the distance of two nodes as inversely proportional to the sum of their activation levels. This makes node activation a strongly self-reinforcing process. To keep the implementation more flexibly under control, it was rather decided to associate with each link a directional (non-symmetric) distance coefficient. Hierarchical relations get greater coefficients. But these vary according to the number of connected concepts when going in that direction. Ordinary verbal sentences, basic properties represent stronger links; they receive smaller coefficients. Not all distances need be specifed explicitly of course, as default values exist for various relation types.

Finally, association links can play an important role here. Whenever a relation was not fully investigated, the dummy link created still permits the flow of activation to go through. Furthermore, such dummy links can be manually inserted to associate two ideas. (One can even give some psychological

justification to this as thousands of such weak associations probably do get established in the human mind.) This permits further control over the distribution of activation levels. Such links also allow the introduction of contextual heuristics in performing inference. Most often, one can dispense with a complete memory search and simply concentrate on linked concepts in order to reach some conclusion. For example, if one wants to know how many candies John has now, one should only need to consider how many he had at some point in time, and how many he gave or received subsequently. Association links for the verb "have" permit the avoidance of a systematic search through all events involving John.

4.3 DEDUCTIVE INFERENCE.

4.3.1 Consequence Relations.

A very primitive deductive component is part of this system. It is based on the use of 'consequence' relations specifying implications between sentences. The following example was already implicitly assumed when presenting the internal structures derived for the analysis of the sentences in (10) above (see section 4.1.4):

```
(consequence
  (write somebody something prep: somewhere) (12)
  (prep something somewhere))
```

In (12), it is intended that the same nodes occur in the "write" sentence and in the locative sentence. When something is written somewhere, one can conclude trivially that it eventually is in that place. Specifically, in the example of the previous section, when 5 is written in "the ones' place of the answer," it is known that 5 is in the ones' place, i.e., that 5 ones are part of the answer.

Inference is practically essential for answering questions. Our system can only handle elementary "WH-" and "yes/no" type questions. WH- questions require filling some "semantic gap" in a sentence. This could be a noun node (e.g., "who," "what") or a verb node (e.g., "how-many"). In the case of "why," a whole implicit sentence constitutes the gap:

"Why [S]?" --> (reason-for WH- [S]) (13)

On the other hand, yes/no questions are answered by testing the truth value of some proposition.

In the simplest cases, the question may refer directly to past information stored in the semantic memory. Finding an answer is then purely a matter of retrieving the appropriate sentence in full. More often, some deductive process is called for. For example, the first two relations sketched below (14ab) allow the answers as indicated in (14c):

(consequence (gives x y z) (has y z)) (14a)

(consequence (and (has x y) (has x z)) (has x (and y z))) (14b)

"Ed had 26 candies. Sue gave him 19 more. Then Ed had how many candies?" Answer before learning addition, "26 candies and 19 candies;" afterwards "45 candies." (14c)

4.3.2 'Deductive Explosion.'

One problem is to decide in general what deductions should be carried out and when. It would be obviously impractical to look for all possible conclusions every time a sentence has been processed. Each event, no matter how apparently shallow, usually results in very many consequences, most of which will be irrelevant to the purpose of the discussion: e.g., from (14c) above, one can at least deduce all of the following:

Sue had at least 19 candies.
She handed them to Ed.
She had an arm.
She grasped the candies.
She had some reason to give them. (14d)

On the other hand, one would like to retain the ability to reject a sentence on the grounds that it is contradictory with the consequences of past information, or vice-versa. Indeed, some of this is performed by the SM-scheme as discussed earlier. However, this was restricted to rather direct contradictions.

Those difficulties do not simply arise from defects in our deductive component. It is not clear whether it is at all possible, even in theory, to perform all the necessary processing. It is significant in this regard that even humans are more or less short-sighted. We never have 100% realization of all consequences of each event. (Otherwise, every vote would probably bring unanimity!) What seems best for a general purpose program is to pursue the consequences up to some small depth, as a function of the importance of the matter on hand. (Some heuristics similar to those presently used for game playing could be developed. Here again, concentrating on areas of high activation levels may restrict the breadth of the processing.)

Our implementation may appear ad-hoc; but it is based on introspection. Some relations seem to be marked in our minds as being more 'immediate' in some sense than others. Thus, from "Sue gave Ed 19 candies," one seems to first conclude that Ed gets them, as indicated in (14a). Only then do other conclusions come to mind, such as those listed in (14d), again in some order. In practice, the semantic network contains some special relations "IMM-PROP," "IMM-CONS" (immediate property or consequence) which lead the system directly to perform some additional processing.

"IMM-PROP" indicates an immediate development for an elementary node. For example, as soon as a number is reached in the analysis of a sentence, an "IMM-PROP" relation directs the analyzer into developing specific "has-as-parts" relations between the number and its digits representing the ones, tens, etc. "IMM-CONS" relations indicate deductions to be performed as soon as a full sentence is analyzed.

In most cases, the action to be performed consists of simply copying the consequent relation and replacing the general nodes by the specific occurrences from the text. In general, this instantiation problem may create difficulties in the determination of the corresponding pairs. However, in our restricted model, no such difficulties arise. The same corresponding pairs determined in the matching process of the SM-scheme are used for the replacement. A simple example is a restatement in full of (14a) above:

(15)

Sometimes, the immediate action is rather complex: a full algorithm is needed to specify it. Instead of using very complex nodes, one makes use of pre-compiled routines. This is the case for the setting up of numbers in a standard frame in preparation for an arithmetic operation.

Finally, the problem of 'explosion' of consequences referred to above is simply non-existent here because of the limited information in this particular network. Few consequence relations are present.

4.3.3 Deduction and Induction.

Deduction and induction are not independent. An interesting class of questions are precisely those whose answer relies on the learning process. The program is then expected to answer questions within a problem by analogy with similar questions or statements which occurred during previous examples. If the two questions are identical, the answer is a by-product of using the same network for old and new information. One must check however that the same context applies so that the previous answer is still valid. (The only simple case is when the sentence is fully generic, thus independent of context.) Real life cases are rarely simple. Most often, the human mind is expected to formulate an induction hypothesis and deduce the answer from the corresponding rules. Needless to say, human ability to hypothesize is significantly more powerful than what our system can presently hope to achieve. For instance, the first example involves adding 26 and 19. This is shown to begin with the addition of the ones' digits, here 6 and 9. When the later example of adding 23, 46, and 85 is discussed, the question "What do you add first?" is incorrectly answered "6 and 5." The average child would immediately guess (knowing that one can add two or more digits) that all three numbers should be involved, and that the digit to be extracted from the third number is likely to come from the same column. This is a great deficiency in our system. In fact, such questions requiring elementary generalizations are not answered correctly at first. After many examples are processed, the inductive component (discussed in the next two sections) 'learns' the general rule. The same class of questions can now be asked again; they will be answered correctly.

4.4 NODE-LEVEL INDUCTION.

Learning is the acquisition of new knowledge. In its most trivial form, this occurs as a direct by-product of translating all inputs into the standard internal form for memory structures. More interestingly, learning results from generalization over cases. Induction over descriptions of algorithms is a process that deals with complete 'stories' or complex nodes. This section examines the elementary process of performing induction by comparison of elementary sentences.

At the sentence level, the verb nodes and corresponding noun nodes are compared in turn. The attempt is to find the intersection of two spheres of meaning. This is similar to compatibility evaluation as described in the previous chapter (section 3.4). But, instead of looking for rejections, the interest lies here in the determination of common factors. Given two nodes, the basic induction module returns an 'induced node,' together with a similarity coefficient in the interval [0,1]. For entire sentences, the overall value is taken to be the minimum value of this coefficient among verb and noun nodes. The induced nodes are intended to include the candidate similar nodes as particular cases by releasing restrictions at those places where they differ.

4.4.1 Elementary Nodes.

For elementary nodes, comparisons are again made along each 'abstract axis' as in 3.4. Partial similarity coefficients S are computed as detailed below; the final value is the average of the individual coefficients over all axes (note here the difference from C). On each axis where both nodes have identical values (or intervals), this common value is kept as a component of the induced node; S = 1. On an axis where the intervals are different, a tentative induced interval is first formed with both original intervals and whatever lies between them. Thus, (c1-a,c1+a) and

(c2-b,c2+b) yield (min(c1-a,c2-b),max(c1+a,c2+b)). The resulting interval is then compared with those intervals containing it which also correspond to existing nodes. It is enlarged to the smallest such. (In case there are two intervals of similar length which might apply, the system could be made to consult with the 'teacher' concerning the various possibilities.) If the final interval covers either most of or the whole axis, then it is simply dropped. This is most likely to occur in connection with polar axes. For example, "+human" and "-human" result in "human-ness" being left unspecified and "+animate" (which underlies both) to be kept. The similarity coefficient compares the lengths of the original intervals to the length 2L of the induced interval:

$$S = (a + b)/L \qquad (16)$$

Note that the interval enlargement procedure usually produces induced nodes that correspond to an already existing word. This can be checked by comparison with the semantic ancestors of the original nodes. (In general however, the process above does not necessarily lead to a known word or concept. Thus, one should provide for the possibility of creating a new node with some artificial name. This was not needed and not explored.) Some examples follow:

"John," "Bill" yield "boy;" S = 0.87 (17a)
"John," "Sue" yield "human;" S = 0.75 (17b)
"3," "9" yield "digit;" S = 0.96 (17c)
"3," "14" yield "integer;" S = 0.87 (17d)

The most controversial point is the handling of non-identical interval values on some axis. A more conservative approach than the one above might suggest taking the strict union of the original intervals. This corresponds to taking the induced node as the mere disjunction of the nodes under study. There is, however, some psychological evidence to support the contrary (see for example [Manis 1968]): humans seem to generalize more hastily by taking the common factors and sometimes completely ignoring axes of difference. The influence of known concepts in the formation of induction hypotheses is also recognized. There are obvious advantages to such approaches insofar as economics of memory storage and simplicity of structures are concerned. On the other hand, Becker's proposals lie on the other extreme [Becker 1969]: all factors including those in common are dropped. When induction hits upon different nodes, these lose significance in the sentence in which they occur; eventually, they are replaced by dummy variables whose domain is completely unrestricted. In the examples (17a-b-c-d) above, this would mean replacing the induced nodes in all cases by the most general noun node "Noun***." This loss of information would make for

unacceptable over-generalizations.

4.4.2 Intermediate Nodes.

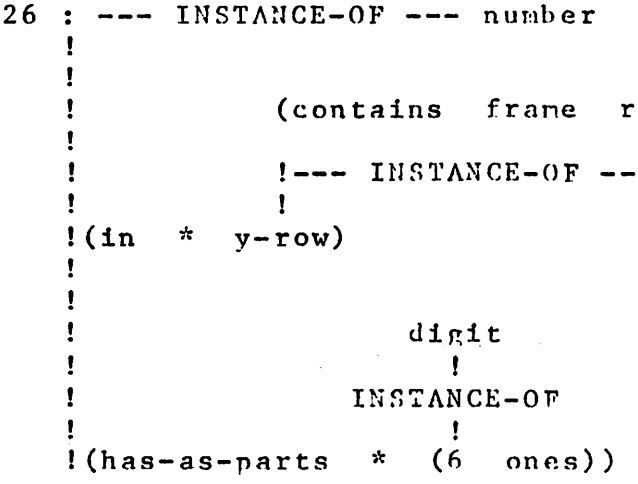
Sometimes a noun node in a sentence is not an elementary node, but a whole relation by itself. The only case which is handled here is that where the verb expects a plural noun which may be an "and" of several nouns. Node level induction must then compare not only nodes from various sentences, but even the coordinated nodes within a simple sentence with each other. One example is the operand for addition, which may include an arbitrary number of arguments playing a symmetric role. As discussed in section 4.1, if the example involves adding 26 and 19, the sentence:

"Add the ones." (18a)

actually means:

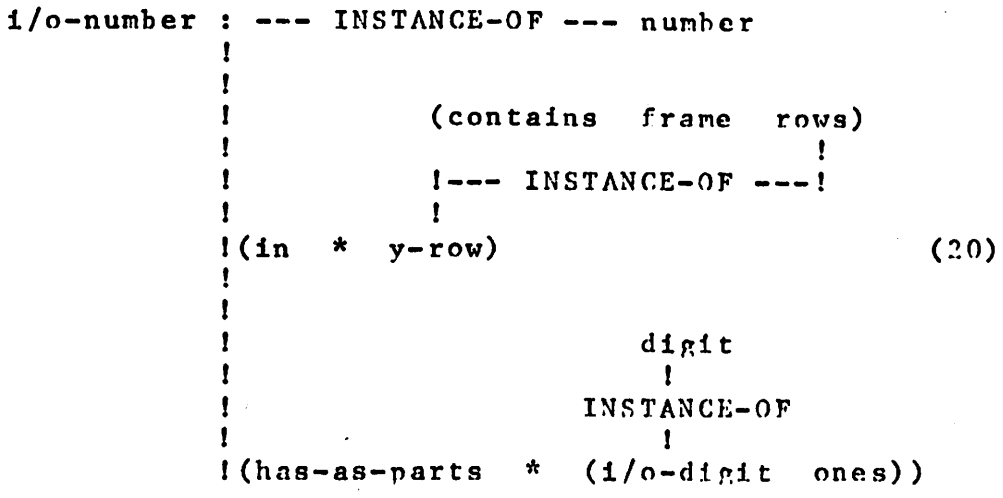
"Add the 6 ones which are in 26 and the 9 ones which are in 19." (18b)

The first operand in (18b), "6 ones," is actually within a whole structure describing the number 26 as it is written in the standard 'frame' for addition. Note that this frame is composed of several rows for operands (named x-row, y-row, etc.) and one bottom row for the answer.



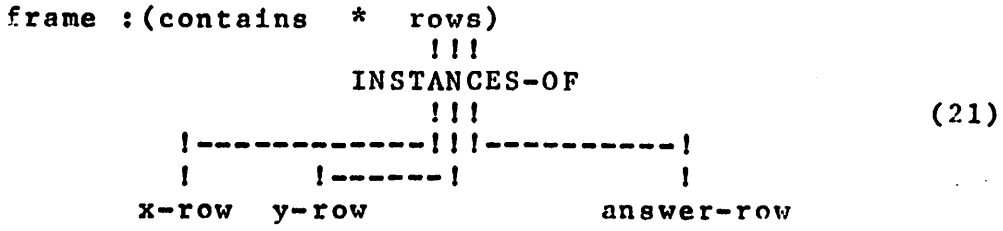
(19)

New various examples will include different numbers and hence different digits in their ones' place. (Note that induction over nodes such as 26 yields an instance of number, denoted "i/o-number," which might be described in English as "some number." The same holds for "i/o-digit.") The induction then yields:



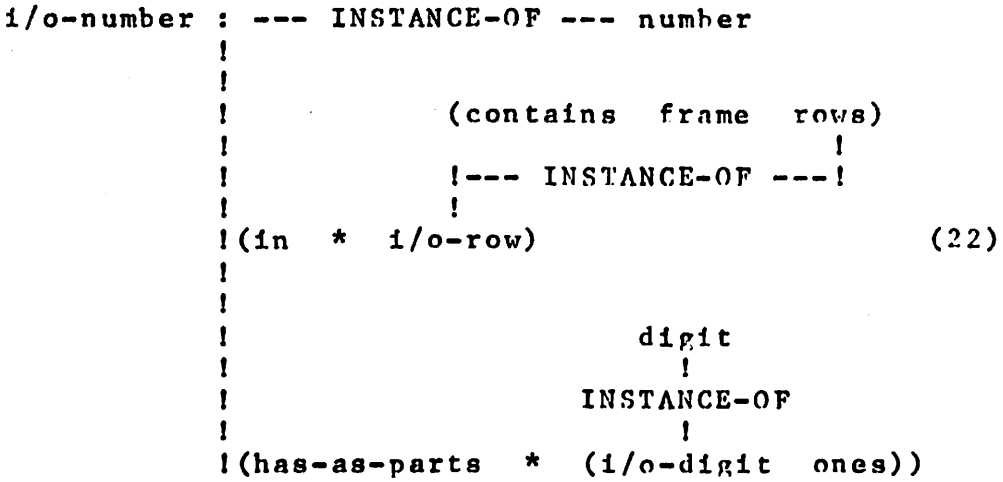
(20)

The structure describing the frame for addition is represented below:



(21)

This determines operand induction to yield finally:



(22)

This is exactly the desired result. One might expect the general rule in the induced algorithm to be more like the original sentence (18a): "Add the ones." rather than (22) above. We claim that this apparent

paradox is mainly due to the fact that surface structures are frequently misleading as to the implicit structures that underlie them. When performing addition, the interpreter will look for all operands satisfying the indicated properties and these are precisely the ones of all numbers (rows) in the picture.

The structure (22) above also shows how difficult it would be to express the induced step in a simple sentence. Even assuming a rather powerful sentence generator, capable of generating relative clauses, prepositions, inflected verbs and nouns, the resulting sentence would certainly be long and complicated. The difficulty is in recognizing those parts of a structure which are inessential to its description so that they can be left understood. In the introduction (section 1.1), in our own attempt at giving an English paraphrase of the induced addition algorithm, the sentences were awkward precisely because they were (intentionally) meant to convey a full idea of the corresponding internal structures.

4.5 ALGORITHMS: CONTROL STRUCTURE.

The general problem of induction over complex nodes representing complete stories is obviously difficult and beyond the scope of our model. Among other problems, one has to decide first which information elements to attempt induction on, by performing the required comparisons. One cannot possibly try to perform generalizations by grouping parts of past information in all possible combinations with the intention of detecting potential similarities. The number of such combinations, even for a relatively small memory, would be enormous. (This is even worse than the corresponding problem of examining the whole memory in search of possible deductions.) Therefore, heuristics are essential. They could include those recognized to be in constant use by humans: temporal and spatial contiguities, and concentration on matters of recognized importance. One would still have to define those subjects to be considered basic by the computer. Here again, the implementation is rather crude. Algorithms are induced only after the program is told which examples it should consider for

generalization. In particular, this prevents the system from trying to merge together addition and subtraction examples.

Our research concentrates on generalization from examples describing the performance of some algorithm. The induction must produce a working algorithm that handles the general execution of the operation under all possible forms of input. Essentially, the process consists of scanning the sequences of sentences which constitute the body of the examples' complex nodes. These sentences are then compared in turn, attempting to generalize on each pair as described above. To be sure, it is unrealistic to expect to find a one-to-one correspondence between sentences from the two nodes. The induction process must bring out the exact flow of control for the computation in the general case. This is mainly connected with what happens in between sentences. It constitutes the main subject of discussion in this section.

4.5.1 Steps and program graphs.

The description of the addition and subtraction algorithm in the text are basically built around a few component steps which are repeated with little variation. In the original text, these can be distinguished by the layout of paragraphs. One could indeed use a special mark to indicate paragraphs. On the other hand, these steps can be simply recognized as they are headed by an imperative sentence. In most cases, a step will consist of this unique sentence which specifies the command to be performed. Sometimes, a declarative sentence follows, stating the result of the command. (The declarative helps in making the appropriate connection between this step and some later step which requires the stated result.) For example,

"Add the ones.
There are 12 ones. (23)
Think of 12 ones as 1 ten 2 ones."

The set of elementary steps for each algorithm is described below. Some of these steps occur only in some examples depending on the values of the operands. If a word may occur with different values in different examples, the various values are indicated between parentheses. Also, as one goes through the addition or subtraction loops, the text repeats itself periodically with minor changes. In connection with this, some steps differ only in one or two words. Word changes that

occur in this manner are indicated between square brackets. Also, this cyclic nature of the algorithm is reflected in an index attached to each step name. The index corresponds to the column of digits currently under consideration (i = 1, 10, 100, ...). Thus, CC10 is the computation of the carry as a result of adding the digits in the tens' column.

Ai (add): "Add the [ones, tens, ...]. There are (1, 2, ...) [ones, tens, ...]."

CCi (compute carry): "(10, 11, ...) [ones, tens, ...] are 1 [ten, hundred, ...] and (0, 1, 2, ...) [ones, tens, ...]."

Wi (write answer digit): "Write (0, 1, 2, ...) in the [ones', tens', ...] place of the answer to show there are (0, 1, 2, ...) [ones, tens, ...]."

WCi (write carry): "Write 1 above the (1, 2, ...) in the [tens', hundreds', ...] place to show there is 1 more [ten, hundred, ...]."

Mi (must borrow): Before you can subtract (1, 2, ...) [ones, tens, ...], there must be more ones, tens, ...]."

Bi (borrow): "Think of (1, 2, ...) [tens, ...] (1, 2, ...) [ones, ...] as (1, 2, ...) [tens, ...] (10, 11, ...) [ones, ...]."

Si (subtract): "Subtract (1, 2, ...) [ones, tens, ...] from (1, 2, ...) [ones, tens, ...].
(1, 2, ...) [ones, tens, ...] - (1, 2, ...) [ones, tens, ...] = (1, 2, ...) [ones, tens, ...]."

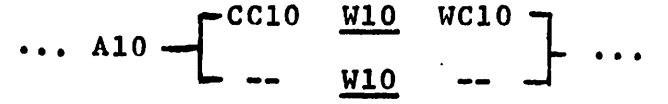
Wi (write answer digit): same as the addition step.

Every individual example can be described by a sequence of steps. Thus, in an example involving the addition of two 3-digit integers, where a carry occurs while adding the tens, the sequence is:

A1 W1 A10 CC10 W10 WC10 A100 W100 (24)

The preliminary idea for induction is then to merge all such paths into a 'program graph.' This diagram is simply intended to indicate at first the different possibilities as a result of empirical collection. Basically, those steps occurring identically in all examples are placed in a central common path. Otherwise, a fork into several parallel paths is formed. These represent all possible variations at that point in the algorithm. (A test will be placed at the root of each such fork but this will be discussed later; see, in particular 4.5.3 below.)

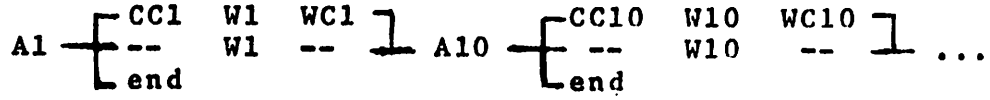
Structural organization is most crucial here. What differentiates two steps is not so much the values of their component nodes, but the relations of these nodes to other nodes in the program graph. Differences in values are indeed 'absorbed' through node induction. This covers digit and operand induction as discussed earlier. On the other hand, two steps may have identical surface structures and still be placed on different branches. Thus, consider the following part of the addition graph.

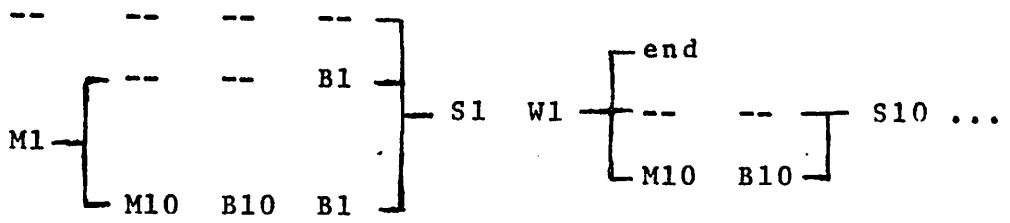


(25)

Where a W10 step occurs twice. The step following the carry involves writing the ones' digit of the sum; this digit is computed and only appears in the preceding CC10 step. The other W10 step involves writing the whole sum; in this case, it is a single digit and it is directly determined in the A10 step. Generally, if a step involves a noun node occurring in a previous step which is itself on a separate path, it is placed on the same path. (e.g., the Wi and WCi steps following a CCi in addition.) Finally, if a step occurs in one example and not in the other, its 'confidence level' is examined. This was discussed earlier. If the sentence is relatively less 'relevant,' its low confidence level will determine deletion from the program path. Otherwise, it is placed on a branch by itself. This is the case of the CCi step in addition, and the Mi and Bi steps in subtraction.

Program graphs for both algorithms are shown below:





4.5.2 Actions: philosophy of performance.

Before going into the inductive inference of step sequence, some general points about actions and performance of actions must be discussed.

Each step consists of the execution of one elementary operation. These operations are supposed to be known beforehand; they are pointed to by the corresponding verb. Each such action, thereafter denoted A, may (and usually does) have some associated conditions which must hold before execution can be performed. In this case, A would be part of some internal relations indicating the nature of these pre-conditions. Generally, the relation:

(not-unless A B) (26)

means that B must hold for A to be true. The interesting case for algorithms is the following: B has some imperatively modified state verb indicating that some situation must be true, and A has some action verb, possibly modified with the modal "can." Also A appears as a subordinate sentence to B with a conjunction expressing antecedence (e.g., "before"). This can be paraphrased in English:

"Before you can do this, things must be that way." (27)

An example can be taken from subtraction:

```
(not-unless
   (subtract somebody x from: y)          (28)
   (not-greater x y))
```

The above relation is in the system's memory as previous knowledge. Note that the actual Mi steps in the subtraction examples are in fact a (less precise) restatement of this condition. They are, however, recognized as such by inference: "there must be more ones" is a particular case of "there must be more ones than ones to be subtracted."

Another fundamental concept is that of corrective action. In the same fashion as each action (in fact a full sentence with an action verb) is associated with the way to carry it out, each state (or static relation) may be linked with some action specifying how it may be realized. (Incidentally, this is the very reason why state verbs may be used with an imperative modal.) Such actions usually depend on the particular situation. Thus, in subtraction, if there are not enough ones, more can be brought in by decomposing one ten into 10 ones. After this, subtraction can proceed as usual. The borrowing operation simply constituted a preparatory action. In practice, the following relation indicates that if B is not true, one should 'prepare' A by a preceding C:

(unless-or-after A B C) (29)

The program does not know any such relation at the beginning. It knows about the concept, and, as discussed later, it is able to induce such a relation.

In other cases, when some necessary condition is not fulfilled, an alternative action is specified. The difference with the above is that the new operation is performed in replacement of rather than prior to the originally planned action. In the following relation:

(unless-or-else A B C) (30)

if B is not true, execution of C replaces that of A. A trivial example follows:

```
(unless-or-else
   (subtract somebody (x tens) from: (y tens))
   (and (exist (x tens))                          (31)
        (exist (y tens)))
   (DONE))
```

Sometimes, no corrective actions of any type are specified and the interpreter in charge of performing the operation must give up. Thus, the decomposition associated with borrowing, e.g. one ten into ten ones, is only possible if there are some tens. Otherwise, the need for a ten is handled similarly by initiating a borrow from the hundreds, etc. This process may fail however, and no other way to bring in more ones is known. The operation fails then completely, and an error message is given.

In all cases mentioned above, conditions associated with some operation A and connected corrective actions can generate different paths in a program graph. Sometimes however, a fork in a program graph may not correspond to an impossibility in

performing some following step. It can be that several branches of the fork demand no prior conditions or else that all these conditions are simultaneously fulfilled. In this case, the choice of the path must result from a test which relies solely on previous information:

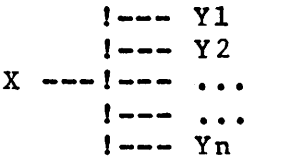
(if-test B A) (32)

Despite the apparent similarity between (32) and (26) above, B is external to A in (32).

One such case arises in connection with addition and is discussed in 4.5.3 below. One can give many other examples of this kind: e.g., a child is shown two numbers and an operator for either addition or multiplication; he is asked to produce the answer. Obviously, given two numbers, he could equally well add them together or multiply them. The test he makes is on the nature of the arithmetic operator, not a condition on the operands.

4.5.3 Analysis of step sequence.

This section shows how a program graph can be analyzed in order to determine which of the aforementioned cases is involved. Obviously, if a step has only one possible successor, there is no problem. In general, depending on the particular example, a step X may be followed by one of the steps Y1, Y2, ... Yn:



(33)

A priori, the branching may be due to any of "unless-or-after," "unless-or-else," or "if-test" situations, or some combination of these.

Steps introduced by "unless-or-after" relations are examined first. For each Yi, examine whether it is in such a relation to a Yj, for some j. Some characteristic cases are actually implemented: Yi can be an explicit statement of a "not-unless" type relation. In this case, it is followed by the corrective action and finally a return to Yj. Yj may occur after quite a few steps only if the corrective action is complex.

Subtraction offers an interesting example where Mi-steps state "not-unless" relations and Bi-steps represent corrective actions which may themselves require conditions and corrective actions. Examples follow:

```
--   --   --   --   S1          (34a)
M1   --   --   B1   S1          (34b)
M1   M10  B10  B1   S1          (34c)
```

In the format of the general fork as in (33) above, this can be redrawn:

```
                !--- Y1 = S1
X = begin ---!--- Y2 = M1                (35)
                !--- Y3 = M1
```

Here, Y2 and Y3 are explicit "not-unless" relations and Y1 does occur in the branch of the program graph following each. All the steps in between are then understood as corrective prior operations. (These are in turn analyzed in the same fashion.)

Another trivial case of "unless-or-after" relation is recognized when the program's prior knowledge includes a relation of the form:

(unless-or-after Yj Condition Yi) (36)

and the "Condition" is actually not true on those paths starting with Yi. On the other hand, one ability which humans do have seems difficult to implement efficiently: when a relation such as (36) is not known but Yj always occurs eventually after a Yi, one might infer that such a relation holds. In the general case, Yj may be separated from Yi by arbitrarily many steps (see (34a-b-c) above). Hence, if nothing is explicit or known beforehand, the task becomes a formidable search and comparison problem.

Somewhat similar problems arise in connection with "unless-or-else" relations. Here again, known or explicit relations of this form directly, or simply "not-unless" relations suffice for recognizing this case. The main difference with the earlier cases is that the step Yj, which is replaced, does not occur again in the same path of the program graph. The example on hand is the trivial one of ending the arithmetic operation when all digits have been exhausted. Thus the replacement occurs as the last step on its path. Otherwise, confusion is possible as Yj may in fact reappear on the same path simply because the algorithm includes repetitions.

Once either of the above two relations is established, the corresponding Yi step can be discarded from the fork under study. What is involved here is a matter of intentionality. One must distinguish what one is eventually trying to achieve rather than what happens in the immediate step. The interpreter is in charge of testing all conditions pertaining to the execution of every operation and takes whatever prior or replacement action (or error message) is necessary. The tests are linked with the operation to be performed, since both appear in some standard relation.

In the pure "if-test" case, some conditional statement must be inferred. In general, this test will examine the range of values of some nodes appearing in the algorithm's semantic structures. Membership in an appropriate set product must be established; this is often, but not necessarily, through some numerical values lying in restricted intervals. Boolean expressions involving several variables may be involved; each of these may appear in the Yi's or in any preceding step. (e.g., in a conditioned reflex type experiment, the presence or absence of electric shock may purely depend on the frequency and the intensity of the preceding whistle.) A useful subset can be defined in which the condition tests the value of a unique node which is itself restricted to be used in some Yi. It must also have been defined by some previous step, perhaps X itself. A good search heuristic further restricts the choice to a true variable, hence a noun involved in node-level induction. Finally, an obvious selection criterion is that the chosen node should have disjoint value ranges characterizing each possible path.

The example here is the test for carry in the addition algorithm:

```
        !--- Wi
   Ai ---!                                    (37a)
        !--- CCi
```

The node to be tested is the sum resulting from Ai and used in both Wi and CCi. The only true variables in these steps are this sum itself and its ones' digit. The former is the only one defined previously by Ai. However, problems arise because node-level induction will have over-generalized the set of values in the case of a carry. Since the semantic memory does not include the concept of a two-digit integer, values such as 12 and 19 will be generalized to the closest embedding concept: "integer." In the no-carry path, the

induction correctly infers that the sum must be a "digit." Hence, the sum seems to have overlapping 'semantic volumes.' As discussed earlier, the program can check with the 'teacher' whether the induction results are correct. Alternate values for the sum are then specified to be in one case: 0, 1, ..., 9 and, in the other case, 10, 11, ..., 29 (note that simultaneous addition of 3 numbers is sometimes performed). The correct test can thus be established:

```
          !--- if sum less than 10: --- Wi
   Ai ---!                                          (37b)
          !--- if sum greater than 10: --- CCi Wi ...
```

4.5.4 Loops.

It is useful (and natural) to perform easier processes before others. Performing node-level induction first is of great help in the study of step sequencing. It is the basis for establishment of program graphs and the nodes involved are used for conditional statements.

Similarly, coalescing repetitions of groups of steps into a loop is much easier once these groups have settled into their induced form. Thus, if one attempts to build a loop from the start, one is likely to be faced with erroneous complications such as:

"When adding the ones or the hundreds, there is a carry, but not when adding the tens."

These confuse the problem unnecessarily. On the other hand, if loop establishment comes as a final process, the problem becomes much easier. At this point, the program graphs for both algorithms have become:

```
                  ┌ CC1  W1  WC1 ┐                  ┌ CC10  ... ┐
A1  T1 ──┤                      ├── A10  T10 ──┤           ├   (38a)
                  └ W1   --  --  ┘                  └ W10   ... ┘

S1  W1  S10  W10  ...                                          (38b)
```

Some steps have been entirely removed, new tests were brought in: Ti tests the sum from Ai for carry purposes. Overall, the structure is much simpler, especially for subtraction (compare with the original program graphs at the end of section 4.5.1).

Determination of loop structure must begin with a decision as to which groups of steps should be merged. This is made essentially by comparing the verbs

involved in the commands. If a verb occurs repetitively within a program graph, the group starting with one occurrence and up to but not including the next, is tentatively set as a candidate for the main body of the loop. In our case, the structure of the various groups are almost exactly identical except for some noun nodes indicated earlier in the description of the component steps. These nouns indicate the column of digits currently participating in the computation. Most crucially, graph and verb structures are the same. (If the verb structure was slightly different, one might still attempt a loop merge by introducing further conditional statements in the same fashion as discussed above.) Thus a loop is decided upon, and an arbitrary loop control index is created: it is assumed to take values 1, 2, 3, ... All noun nodes which differ from one group to another are made dependent on this variable. The simplest (and safest) way to specify this dependence is through a list of successive values; e.g., the following table:

control index: i	dependent variable: V
1	ones
2	tens
3	hundreds

The problem of finding a simpler function which relates these variables to the control variable is very complex. Thus, from the table above, one might like to infer that the following mathematical relation holds:

$$Vi = 10^{**}(i-1) \qquad (39)$$

("**" stands for exponentiation.)
This is beyond the reach of even a fourth grade pupil's mind. The main difficulty, however, is a matter of representation. If there were appropriate structures in the semantic memory to indicate that "one is 10**0," "ten is 10**1," etc., the standard induction process should be able to retrieve the desired relation. A simpler process, which is in fact accessible to the child's mind, is to notice that there is some relation between each two consecutive values of V. In this case, "to-the-left-of" is such a relation. Thus, one could replace the list of values by a different specification of the dependence function: an initial value and a successor relation. Of course, if the loop contains only one variable dependent on the control index, one might drop the index completely, and use the variable itself for loop control.

All of these loop improvements, however, seem unnecessary. In effect, these problems are more relevant to automatic code optimization. It is not our

intention to compete with compiler writers. Rather, it is sufficient for the purpose of this research that the algorithms are induced correctly.

4.6 REVIEW AND FINAL ALGORITHMS.

Before concluding this chapter, it seems desirable to give an overview of the entire processing.

The first phase is the syntactic analysis (SYNPARS) where each sentence is parsed as a grammatical tree structure such as the one presented in section 2.6. These parse trees are kept on file in a prescribed format, ready for input to the next module.

Meanwhile, the semantic network is set up by a group of subroutines called SEMNET. Once all the interconnections are established, the network area can be dumped on a separate file in raw binary form for fast reloading.

Using the semantic network, SEMPARS processes each syntactic tree in the original order of the sentences as they appear in the text. The analysis proceeds as explained in chapter 3 and sections 4.1 to 4.3 in this chapter. Semantic relations within sentences and connections between sentences are established. The new structures (such as the one described in 4.1.4 above) are integrated in the semantic network.

The various examples are then analyzed at a higher, more global level in view of induction. Individual steps are grouped in program graphs representing all the possible step sequences. Elementary level induction merges noun nodes (operands) between similar sentences (see section 4.4). The program graph for an algorithm is then analyzed for general step sequence, and the appropriate tests are inserted wherever necessary (sections 4.5.1 to 4.5.3). Finally, the induced algorithm is examined to detect loops. In both cases of addition and subtraction, a loop is found and the structure is rearranged

accordingly (section 4.5.4).

The final structures obtained for each algorithm are presented below, in the last pages of this chapter. (Again, the reader must be warned that the whole semantic network is connected together. In each case, the figures below only show those parts most relevant to the particular step.) Each algorithm is presented through a global flowchart followed by details of each step.

Note that each step really consists of only one imperative sentence (doubly underlined). The reason why the structures are so complex is that considerable 'static'-type information is needed to specify the action to be carried out: the sub-operands of each step, their relations to each other and to the original operands (these are assumed to be written in the frame or grid used for arithmetic operations).

The job of the interpreter is precisely to examine all the relations pertaining to the nodes under consideration in order to specify its own work. For example, consider Ai, the main, column-wise addition step. The action itself is simply specified as:

(add you *)

where the "*" points to a sizable graph of connected nodes. All the remainder of the structures presented serve to specify what the operands are, where to find them, what to do if there are none left, etc. Similar comments apply to all other steps, and to the restrictive conditions which accompany subtraction.

The notation used requires some additional explanation. In a relation, the verb is always spelled out; in the implementation, there is in fact a pointer to the appropriate verb-node (there is no repetition if the verb occurs in several sentences). Similarly, to improve readability while maintaining a faithful description, noun nodes are written within the relation unless they occur in several relations. In the latter case, the pointers are actually shown with arrows pointing to the appropriate box. Round boxes are used for elementary nodes (simple nouns), and rectangular boxes for embedded relations used as 'nouns' within other relations. On the other hand, some nodes are written in capitals and underlined. These indicate nodes specially recognized by the system (e.g., ERROR, DONE).

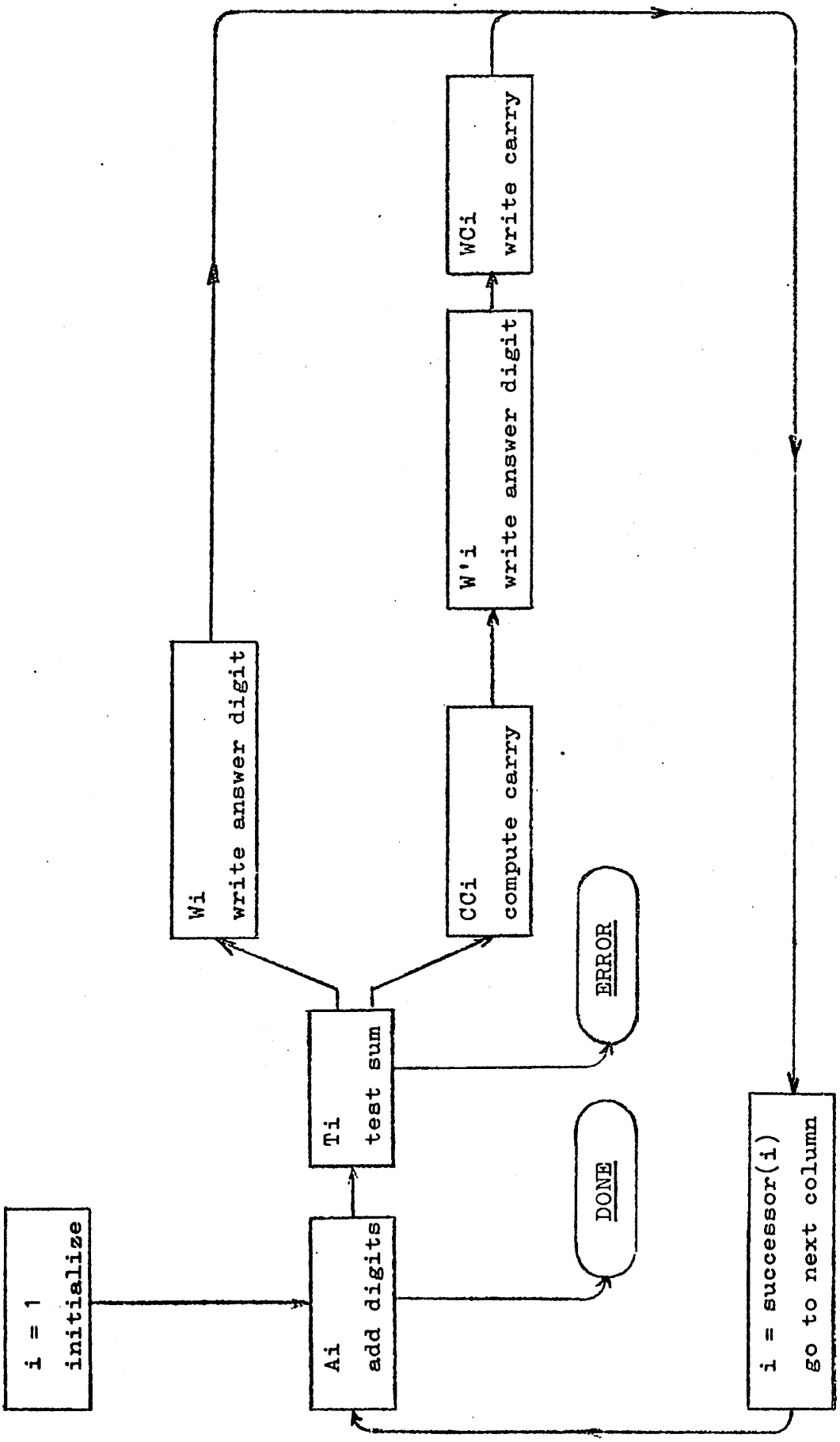
To allow for references across page boundaries, labels, written in capital letters with a ":" sign in front of the original node, can be used. These labels can then be placed inside a box (round or rectangular as appropriate) on another page for reference.

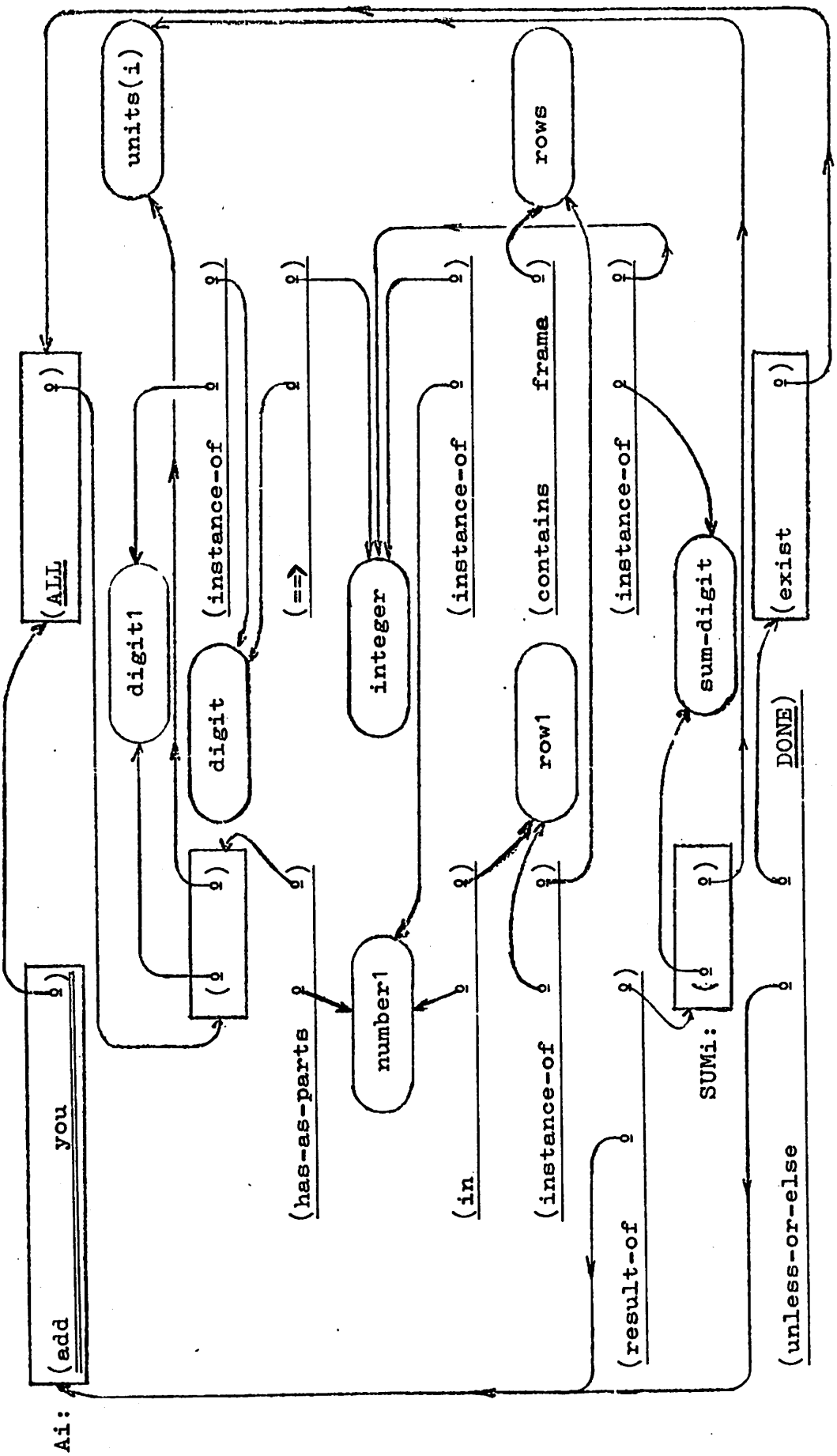
In most cases, descriptive names were chosen for most nouns, even though the name is irrelevant to the actual implementation (e.g., "answer-digit," "upper-number"). In practice, when the system encounters new instances of, say, "digits," it will simply call them "digit1," "digit2," etc.

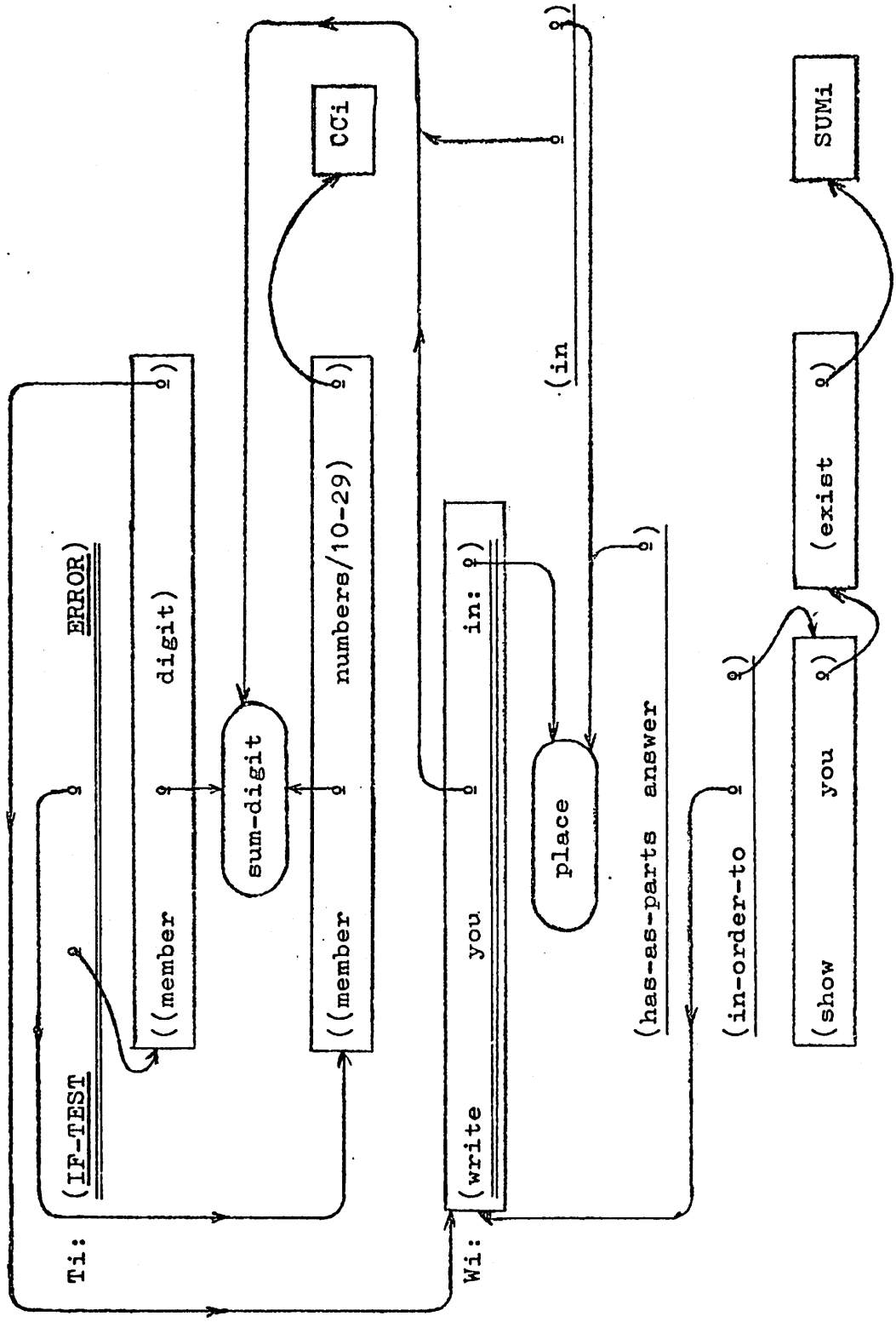
There are two special nodes resulting from the loop merge: "units(i)," "nextunits(i)." These are actually marked as 'dependent' nodes (see section 3.5). Thus, they include a pointer to the dependence specification. This is simply a list where each value of the index is associated with the corresponding value of the dependent variable (see section 4.5.4):

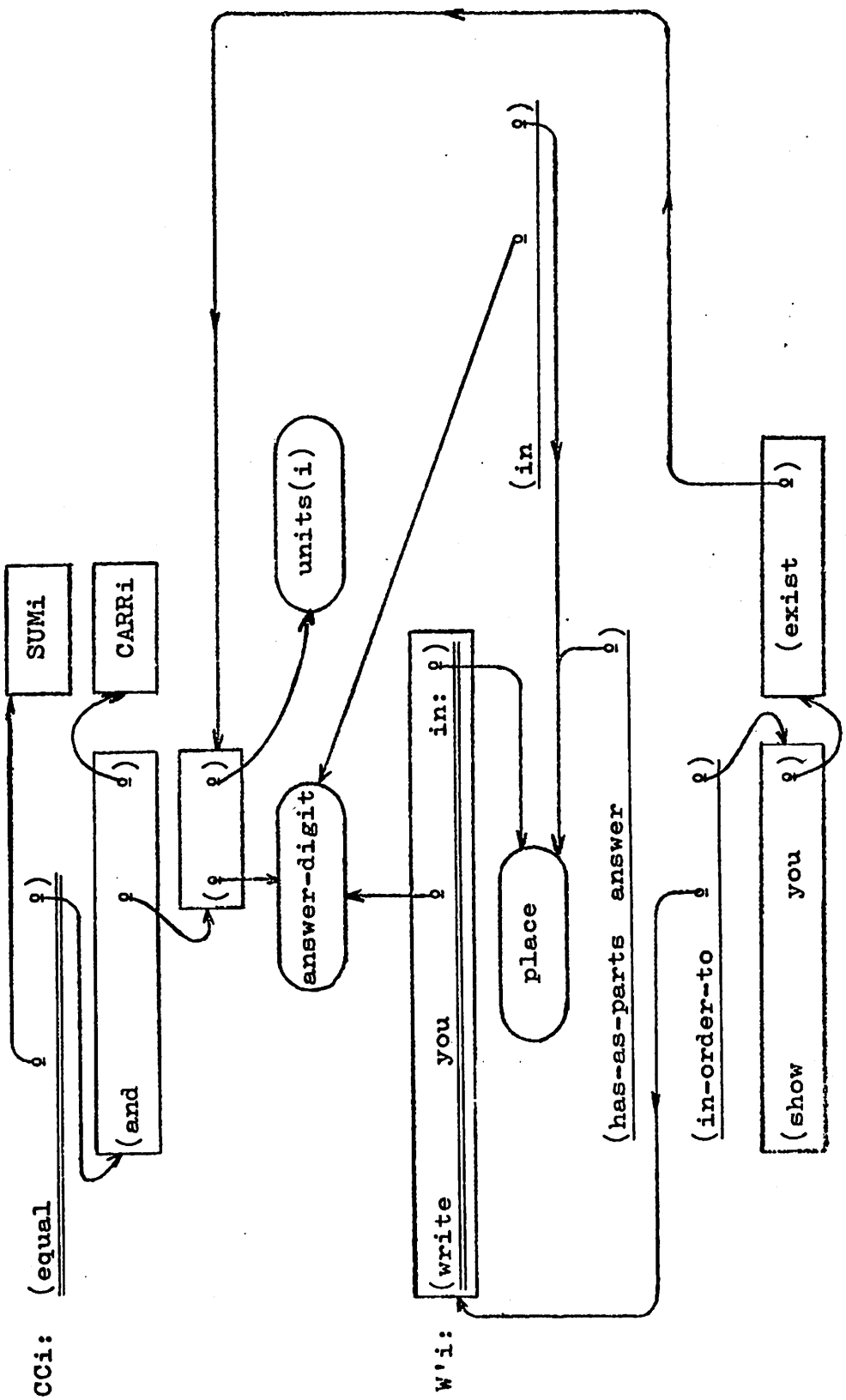
units(i) --► (LIST (1,ones), (2,tens), (3,hundreds))

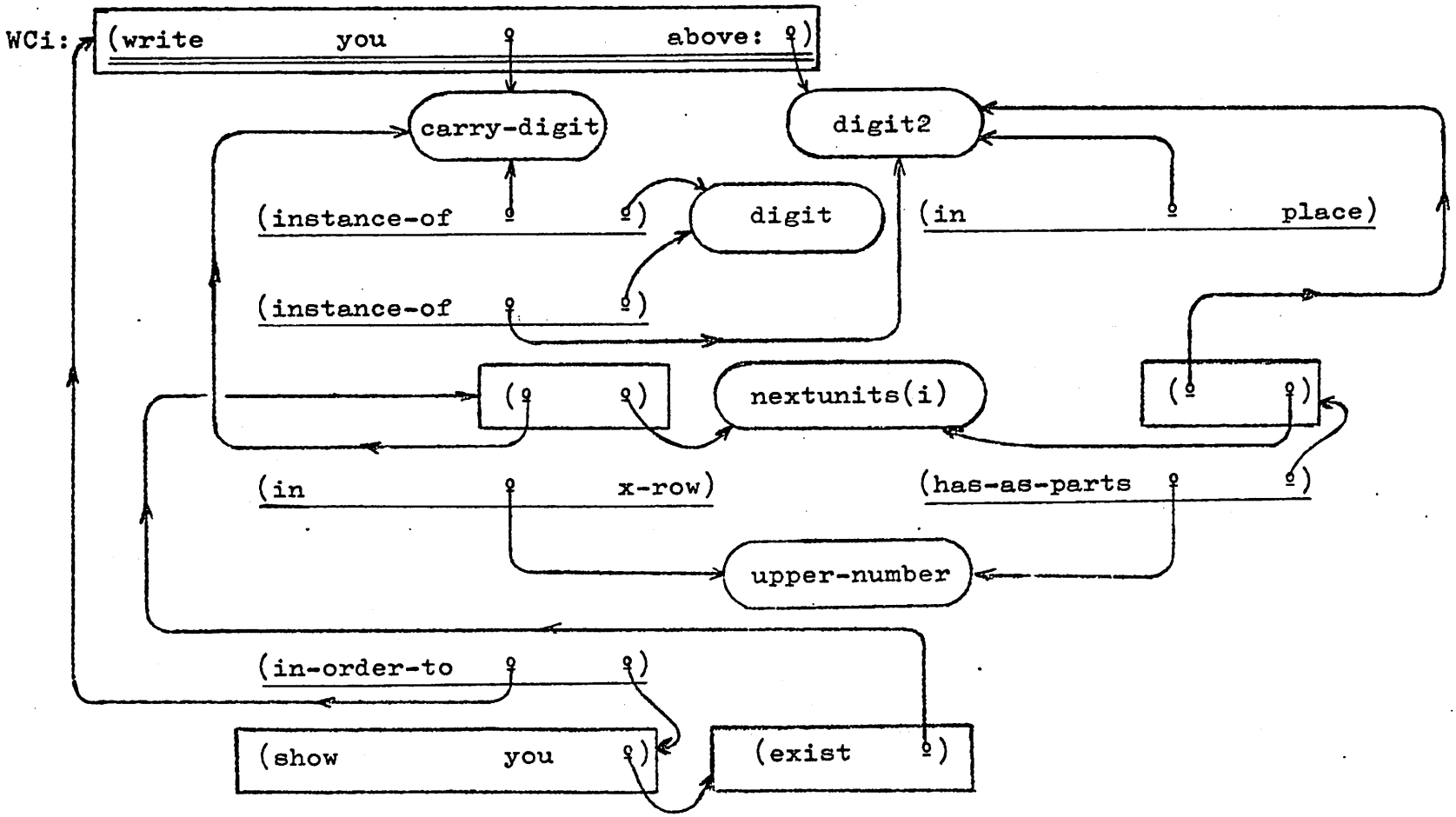
nextunits(i) --► (LIST (1,tens), (2,hundreds), (3,thousands))

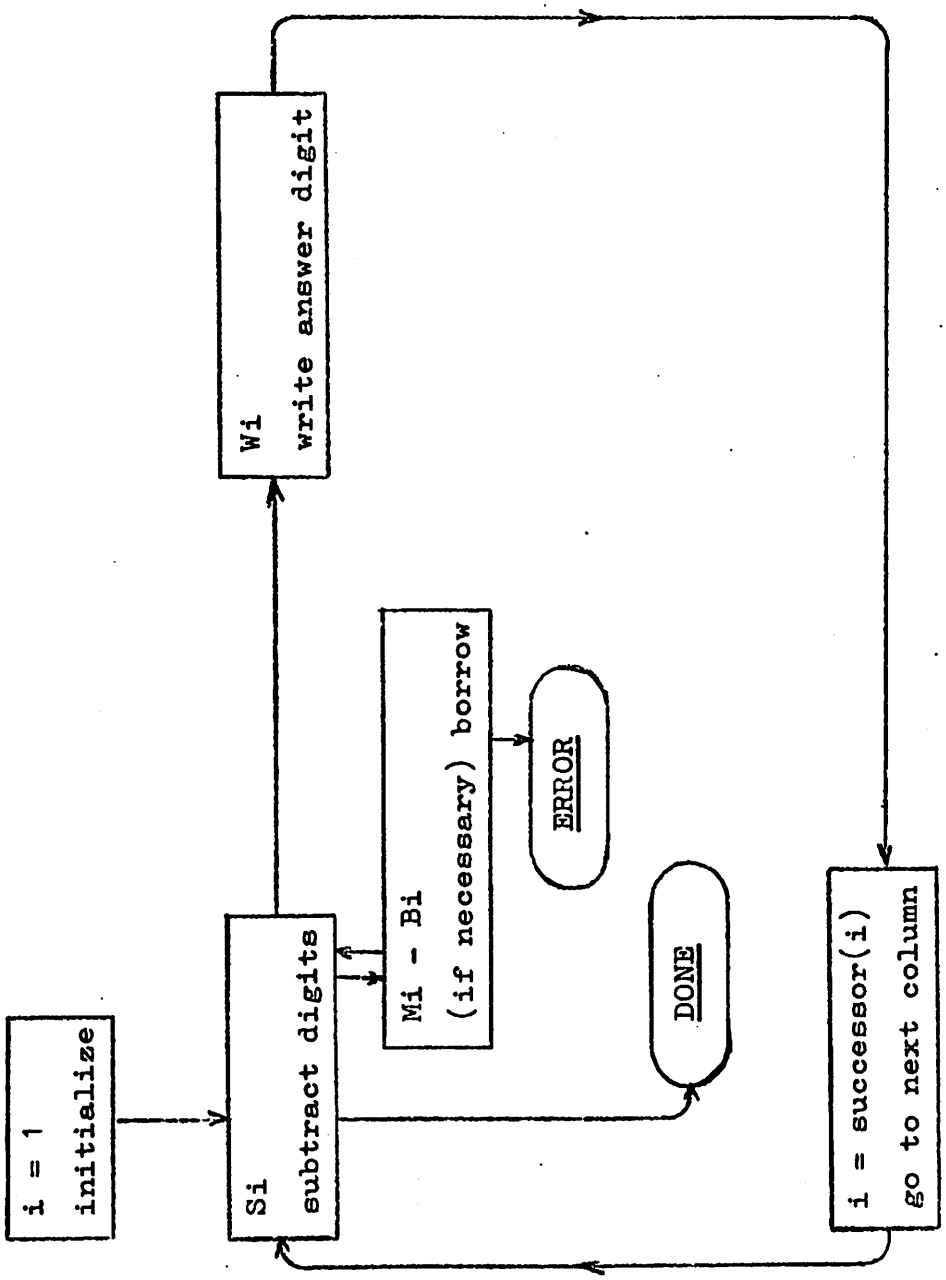

i = 1
initialize
Ai
add digits
Ti
test sum
DONE
ERROR
Wi
write answer digit
CCi
compute carry
W'i
write answer digit
WCi
write carry
i = successor(i)
go to next column

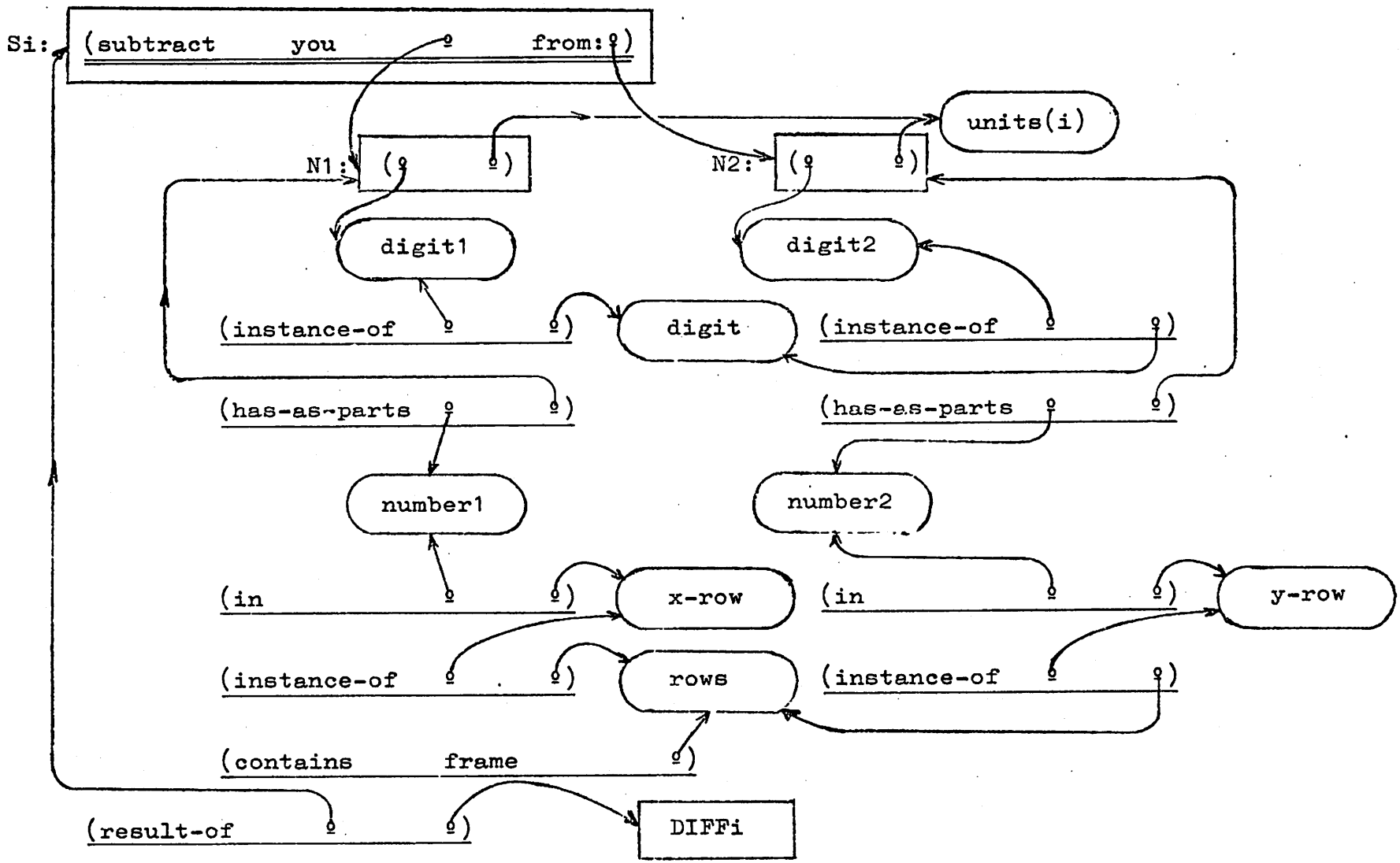

Ai:
(add you)
(ALL)
units(i)
digit1
(instance-of)
digit
(has-as-parts)
(( ))
(==>)
integer
(instance-of)
number1
(in)
rows
(contains frame)
row1
(instance-of)
(instance-of)
sum-digit
SUMi: (( ))
(exist)
DONE)
(result-of)
(unless-or-else)

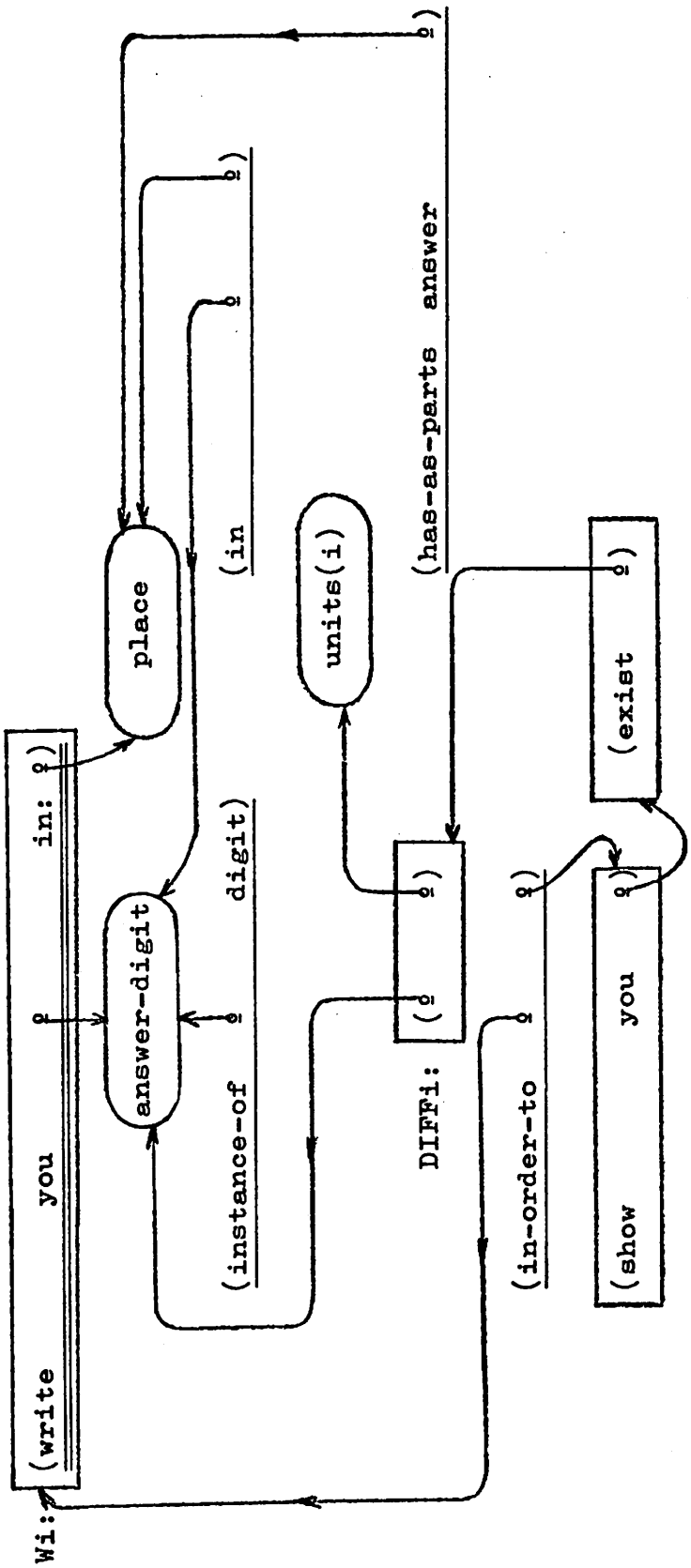

Ti:
(IF-TEST
ERROR)
((member
digit)
sum-digit
((member
numbers/10-29)
CCi
Wi:
(write
you
in:)
place
(has-as-parts answer
(in
(in-order-to
(show
you
(exist
SUMi

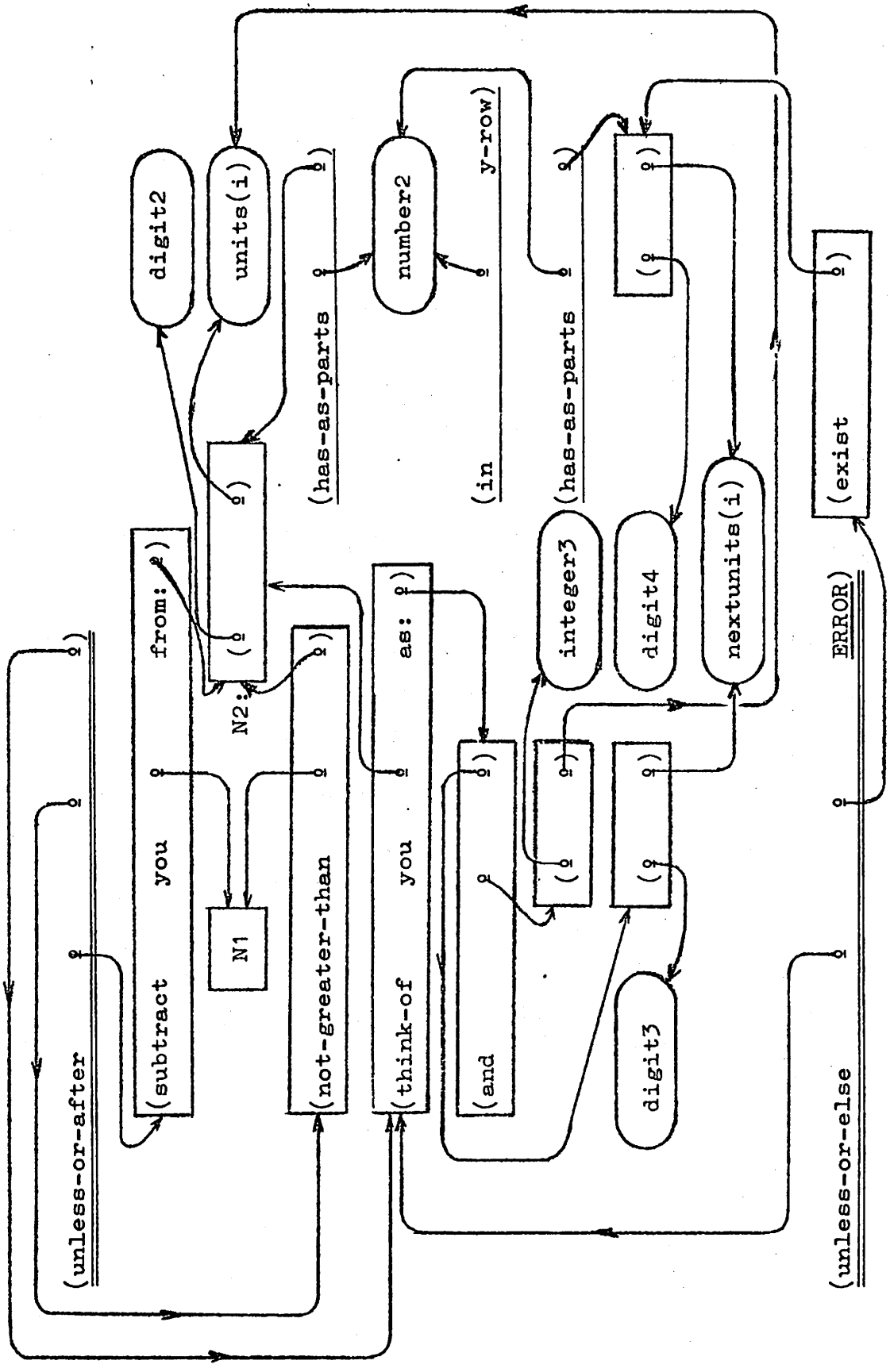

CCi:
(equal
SUMi
CARRi
(and
units(i)
answer-digit
w'i:
(write
you
in:
place
(has-as-parts
answer
(in
(in-order-to
(exist
(show
you

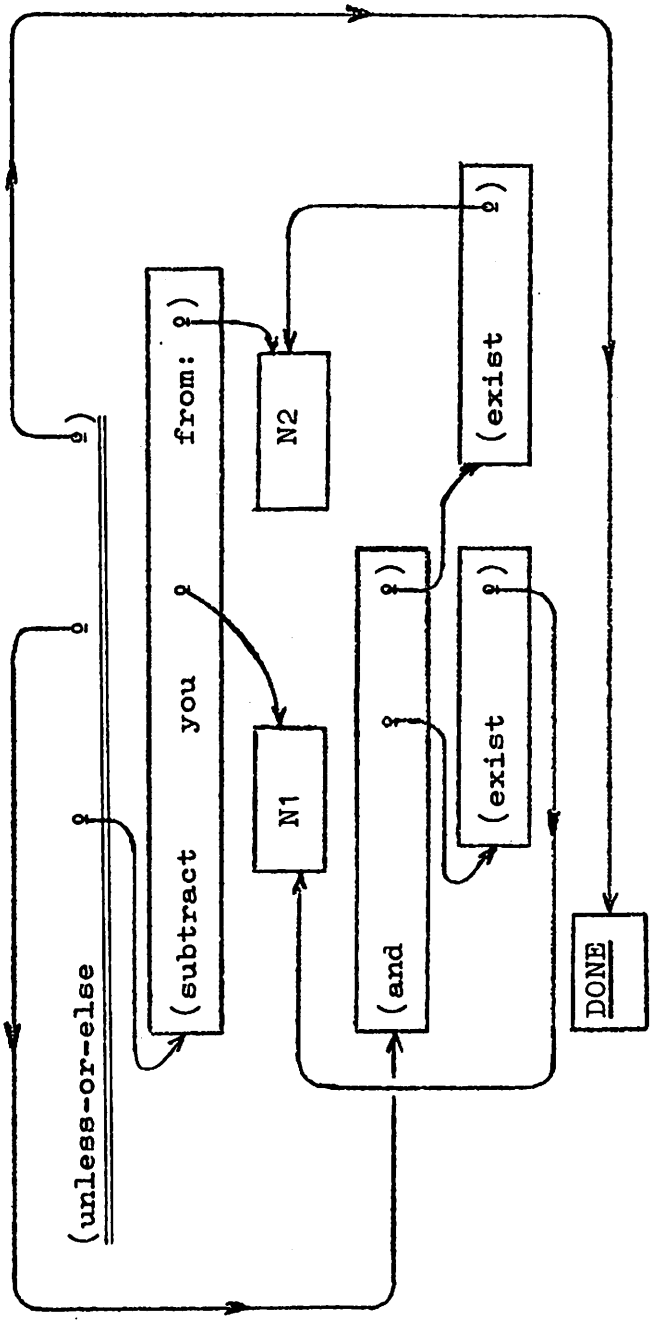

WCi:
(write you ♀ above: ♀)
carry-digit
digit2
(instance-of ♀ ♀)
digit
(in ♀ place)
(instance-of ♀ ♀)
(♀ ♀)
nextunits(i)
(♀ ♀)
(in ♀ x-row)
(has-as-parts ♀ ♀)
upper-number
(in-order-to ♀ ♀)
(show you ♀)
(exist ♀)


i = 1
initialize
Si
subtract digits
Mi - Bi
(if necessary) borrow
ERROR
DONE
Wi
write answer digit
i = successor(i)
go to next column


Si:
(subtract you from:)
N1:
N2:
units(i)
digit1
digit2
(instance-of)
digit
(instance-of)
(has-as-parts)
(has-as-parts)
number1
number2
(in)
x-row
(in)
y-row
(instance-of)
rows
(instance-of)
(contains frame)
(result-of)
DIFFi


Wi:
(write you in:)
place
answer-digit
(instance-of digit)
(in)
units(i)
(has-as-parts answer)
DIFFi:
( )
(in-order-to)
(exist)
(show you)


digit2
units(i)
number2
y-row)
(has-as-parts
(in
(has-as-parts
(exist
(subtract you from:)
N1
N2:
(not-greater-than
(think-of you as:)
(and
integer3
digit4
nextunits(i)
digit3
(unless-or-after
ERROR)
(unless-or-else


(unless-or-else
(subtract you from:)
N1
N2
(and
(exist)
(exist)
DONE

CHAPTER 5
CONCLUSION

This chapter concludes with a review of the results obtained. Various inadequacies of the system and proposals for extending it along several dimensions are discussed. Finally, we analyze the impact of our thesis on some of the deeper problems of artificial intelligence.

5.1 RESULTS.

5.1.1 Programming.

One of the greatest difficulties in our programming effort came as a result of the unavailability of a satisfactory programming language. Snobol was used during the first part of the research. It proved to be a very flexible language, permitting convenient and "natural" programming. The major drawbacks were its great inefficiency both in running time and memory representation, and its inability to compile various subroutines separately. Therefore, in our later work, we reverted to a mixture of Fortran IV and Compass, the CDC6000 series assembler. Extensive use of symbolic parameters and conditional assembly, together with macros, was of great help. It provided sufficient flexibility for complex data structure handling. This was a crucial point, since the internal representation of the semantic network required constant refining even during the latest stages of the research. But many of the assembler facilities are not available for the Fortran IV programmer. On the other hand, writing in low level languages inevitably results in lengthy programs. Hence, some problems were simply due to the difficulty in managing 300 pages of

connected material.

Programming details for the system are summarized in the table below.

	Set	Task	P.L.	Size	Core	Time
1.	SYNPARS	Syntax Analysis	Snobol	60p.	40K	0.2-2.0
	SEMNET	Network Set-up	F/C	25p.	10K	10.0
2.	SEMPARS	Semantic Analysis	F/C	200p.	18K	0.3-3.0
3.	INDUCE	Induction	F/C	20p.	2K	5.0

"Set" is the name of a set of subroutines which are called for the performance of one "task." "P.L." indicates the programming language used: Snobol, or a mixture of FortranIV and Compass denoted "F/C" in the table. "Size" indicates the number of pages of source code.

"Core" indicates central memory requirements. Note that SYNPARS communicates with the later components through its character output file, while SEMPARS and INDUCE communicate through the network area (which can be dumped on a binary file). Thus, total core requirements through the three phases of the entire process are 40K, 28K, and 12K respectively.

"Time" is in seconds of central processing time on the CDC6400. For SYNPARS and SEMPARS, the two numbers indicate the range of processing time per sentence; there are 244 sentences in all. SEMNET takes 10.0 seconds for the entire set-up which need only be done once. INDUCE takes 5.0 seconds for each algorithm, given that 8 examples are used for the induction. Total processing time is therefore approximately 11 minutes.

Although no satisfactory programming language was available, recent efforts in programming language design seem promising, and the importance of flexible data structure facilities has been recognized. Nevertheless, while many new languages have reached the operational stage in a few centers, none is yet widely available. More importantly, most of these languages are still in their infancy; they have not had much use, and concerning reliability, they cannot compete with the more 'mature' languages such as Fortran. In the

future, before large scale research is attempted, it is advisable that some more suitable programming language be designed (or chosen). It should also be fully debugged and documented. Despite its popularity in other artificial intelligence research, LISP 1.5 is not wholly adequate either. The main problem in our work was neither recursion nor binary trees. Rather, emphasis must be placed on flexible and efficient memory structure organization. It may be informative to point out that the semantic network, which is 20K in our system, would have required around 100K using LISP and 200K using SNOBOL.

5.1.2 Review along Evaluation Criteria.

In the introduction, we presented five dimensions along which one can evaluate natural language systems. This section will show to what extent our system fulfills desirable requirements in each of those areas.

a) Syntax. Processing of the text begins with a syntactic analysis of all sentences. The syntax component here is somewhat less powerful than some of the existing analyzers since it uses little semantic information. Yet, its modularization in a Chomsky fashion, and the power of its transformations allow the processing of a large variety of complex sentences.

b) Semantic processing. The next component forms the largest part of the thesis. Syntactic trees are processed in stages at different levels. At the sentence level, the SM-scheme avoids some of the difficulties of feature-based systems. Also, it replaces superficial sentence structures by more meaningful verb-noun relationships. More importantly, the need for discourse analysis beyond the disconnected level shows the necessity of many new developments: recovery of implicit information, serious study of inflections, numerous aspects of reference, and global matters of focus and relevance through activation. These constitute the major contribution towards true understanding of natural language text as a total entity.

c) Semantic Structure. Semantic analysis, deductive and inductive inference, all revolve around the central memory network. Indeed, this network can be thought of as a 'grammar' for semantic processing in the same fashion that context-free productions provide the basis for syntactic analysis. We observe that all types of information (analysis and inference processes as well

as 'static knowledge') should eventually be part of the same memory. The important point, however, is not how much information can be assembled, but how a suitable organization can make such an unwieldy mass of data into a usable structure.

As mentioned in the introduction (see 1.2.2), the two dual concepts of structure and description are in fact gaining increased recognition. Simultaneously, though on a different level, it is interesting to note a gradual shift in computer systems architecture. The focus is not anymore on the central processing unit but on the memory organization. Finally, designers of programming languages have come to realize the primary importance of allowing flexible data structure definitions ([Sammet 1971]). The careful organization of our semantic network was a determining factor in the success of the whole system.

d) Deduction. From its inception, this research was not focused on deduction, and the deductive component is certainly the weakest part of the system. However, we have seen that significant processing can be achieved with very little deductive power (see also 5.3.2-c below). Extensions in this direction will be discussed in the next section.

e) Inductive Inference. The process of generalization over examples of algorithm execution has two distinct parts. One of them is elementary induction over nodes. This process relies heavily on the memory organization. More interesting is the study of flow of control in algorithms. This is a typical application where understanding plays a central role. There is little relation between the problems that arise through our semantic approach to learning, and those faced by perceptron designers. In particular, the idea of <u>possibility</u> is of central importance. Other fundamental notions have yet to appear; more research is needed concerning the philosophy of algorithm description and execution. Meanwhile, within its limitations, CLET has demonstrated that it could actually learn from an ordinary textbook for children.

5.2 SUGGESTIONS FOR FUTURE RESEARCH.

Deficiencies are present at many different levels in the system. They range from the lack of precision in choosing some numeric parameters (e.g., range of activation levels from 0 to 127) to the inadequate handling of the pictorial information which accompanies the text. Arguments concerning the first category are really unimportant. Beyond these, some problems can be solved by extending the system within the same general framework. Finally, a third class of questions remains unanswered; their solution may require major conceptual modifications. These are mostly related to the problem of modeling the human mind which will be dealt with in the following section (5.3).

5.2.1 Extensions.

a) Semantic Implication. There is no real difference between the following two relations:

(V:animal N:human) (1a)

(V:==› N:human N:animal) (1b)

when all verbs and nouns are generic. Both structures represent the sentence:

"Every human is an animal." (1c)

Several reasons justify such a duplication of representation. The first form (1a) is closer to the 'true' structure of the sentence. Standard analysis of (1c) by CLET would yield this form (1a), thus recognizing the surface verb "to be" as merely a copula. On the other hand, (1b) clarifies the hierarchical nature of the relation. It allows easier visualization of noun chains through ==› relations. Following such chains through sentences of the first type would involve switching back and forth between nouns, verbs, and nouns representing the verbs: e.g., here, "a human," "to be an animal," and "an animal."

One may keep this duality. Consistency can still be maintained by establishing a new transformation:

if V is generic, (V N) --• (==› N Nom(V)) (2)

b) Compatibility. The most elementary properties of a node are its ==›-ancestors and the verbs acting on the node as a patient. Compatibility evaluation, as described in section 3.4, relies on these two categories only. In general, one should examine all

properties attached to a node; the entire set is precisely what defines the node. Beyond the elementary properties above, full sentential relations must be handled. For example, the system is at present unable to use relative clauses for determining a referent. (This was not needed since there were no relative clauses in the original text.) Indeed, compatibility should be made to use the full power of a deductive component. Here is one specific place where the efforts of the logic-based approaches could be incorporated. This would greatly enhance the power of the system.

c) Induction and Semantic Grammar. Simple induction can be used to modify the accepted subset of English.

Assume that a sentence of the form (V N1) is originally rejected by the SM-scheme because the verb V is described by (V N2) where N1 is 'incompatible' with N2. Assume further that the analyzer has some reason to believe that the sentence is nevertheless correct (e.g., the speaker is also the teacher). Then, induction could be called upon to construct an induced node N = I(N1,N2) which would replace N2 in the description of V. This could provide for constant refinement of the semantic grammar as represented by the memory structures.

d) Anaphoric Reference. In the present system, a definite noun phrase is represented by the node that it references. The surface structure of the actual reference, i.e., the means by which the referent was accessed, is completely forgotten. This approach is strongly influenced by the transformational school. It loosens further the already weak ties between surface and deep structures. But it is not true that different surface structures may correspond to exactly the same meaning. Generally, there are at least differences in emphasis. Part of this idea bears direct relevance to the problem of anaphoric reference. It is not usually a haphazard process which makes the speaker use one particular expression to refer to some object.

When the teacher says "Add the ones," it is not only because it is more concise than "Add the 3 from 53, the 7 in 17, and the rightmost 2 in 242." An important part of his statement is to point out precisely the relationship between the referents. Thus, the child gets some awareness of what, <u>in general</u>, he is adding. This indeed helps him in performing induction and learning.

A good solution is probably to keep the analysis very close to the surface structure. Each noun node would then be classified according to an attribute of "definiteness" (see 3.5.2-(b)). This introduces no new concept, definiteness becoming an adjective-verb on the same level as "plural" or "red." On the other hand, a relational link between this node and its referent must be established. Incidentally, this provides more flexibility in ambiguous cases. The decisions can be delayed by making reference links between the ambiguous expression and all its candidate referents. Various degrees of certainty can be expressed through confidence levels.

It may prove difficult to implement such a modification of the system. Reference plays an important role in the structure of meaningful stories. Our approach had the advantage of enhancing the similarity between sentences. Having several nodes represent the same object may in fact bring in new difficulties. Much careful research is needed before the change-over can be initiated.

e) Miscellaneous Refinements. Countless minor modifications could be explored with a view toward extending the power of each component. In syntax, both the base grammar and the set of transformations (and their interactions) could be tested on new texts and updated accordingly. Insofar as basic semantic structures are concerned, more research is needed to refine the set of noun-verb relationships. Also, prepositions have different representations depending on whether they are part of larger sentences or occur as predicates themselves. Here again, more consistency can be achieved through the use of a transformation for deletion of the copula.

General induction still requires considerable research, as was pointed out in the previous chapter. However, even within the restricted frame adopted here, several extensions could be explored. Step sequence in algorithms was discussed in terms of a few action primitives: "not-unless," "unless-or-else," etc. More complicated algorithm structures should be examined in order to refine and complete this set of primitives. It is also equally important to devise some general recognition procedure for each primitive. Only a few English constructions can be correctly comprehended at present. An extended deductive component together with more relations in the semantic memory would certainly be helpful. However, it is also essential that more

attention be given to the connectives which link sentences. Very little research has been done at this level so far.

5.2.2 Interaction between Modules.

Many extensions require reorganization of the system in such a way that the various modules can call on each other freely at any time during the processing. Overall, the modules would still be used as a sequence: syntax, semantics, node induction, step sequence, loops. (Reasons for this were pointed out at several places in the text.) But the various modules should be rewritten to work on partly processed pieces of text so that digressions from the normal sequence are allowed. This alone is a major task, and, while it may be consistent with the more fundamental ideas of our research, it would certainly involve a great deal of thought and an enormous amount of programming. The remainder of this section discusses some of the desirable advantages that could be gained from such an effort.

a) Induction and Question-Answering. It was already pointed out that induction hypotheses can (and should) help in answering questions; this itself needs to be investigated further. But the relation is not entirely one-way. When answers made in this manner are successful, they provide great reinforcement to the hypothesis directing the analogy.

For example, "Add the ones" occurring twice at the beginning of the addition algorithm does of course reinforce this sentence as being the algorithm's first step. However, if the second occurrence is replaced by the question "What do you add first?" and it is rightly answered "the ones," the confidence level attached to this step in the whole algorithm would be increased substantially.

This is only useful, however, if the system is often able to generate the right hypothesis in the first place. At present, CLET would fail on the above example (see section 4.3). A different handling of definite noun phrases, as suggested above in 5.2.1-d, would solve this problem.

b) Induction and Node Activation. Making induction simultaneous with semantic analysis also allows the use of activation levels for the process of induction. These are modified as each sentence is processed and lose their significance completely once the whole text has been analyzed. The motivation here is again a psychologically inspired heuristic. Learning is influenced by the focus of one's concentration as symbolized by activation levels.

First, attention helps the learner choose the material on which to attempt induction. When new text is being studied, all relevant information in memory gets 'activated' since activation levels spread out to all connected nodes (see section 4.2). This reduces the search for induction considerably. Thus, the generalization module will compare pairs of nodes in turn; it will simply return a number representing the level of satisfaction derived from similarities between the nodes under study. This number would then be checked against a preset threshold to decide whether the analogy deserves further consideration. If the similarity seems promising, it may attempt to develop a new structure. Inductive inference would not be performed simply on the original pairs, but rather on a whole set of similar nodes.

On a lower level, activation may participate in greater detail in the induction process. As discussed earlier, when an empirical program graph contains a fork, a special test must be established to determine under which conditions each path is taken. This test was seen to depend on the value of some nodes to be selected. Some criteria for the choice of these nodes were discussed. In the general case however, concentrating on the currently active nodes may prove a valuable heuristic.

c) Partial Failure and Recovery. Admitting interaction between the various components of the system paves the way for reorganizing it entirely, putting every part under the control of a top level executive component. This is similar to our approach to syntactic analysis. The executive directs the main sequence of events; essentially, it is responsible for handling the status (success or failure) of returns from each component.

One immediate advantage is in efficiency. If a sentence is ambiguous, but current context imposes a unique parsing, some checking could be performed while investigating each interpretation. With the present

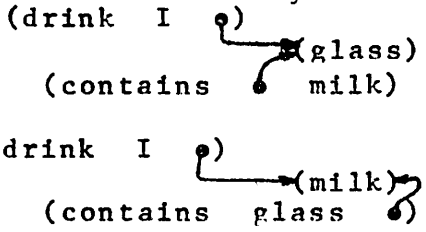
approach, whenever ambiguity resolution cannot be performed purely by syntax, all possible parsings must be developed in full for examination by the semantic module. Essentially, this improvement would allow decisions to be made as soon as they are needed. The gain would be most important in the case of long sentences with an ambiguous construction toward the beginning.

A more interesting by-product is the ease with which one could set-up various processes of recovery from "soft failures." As in syntax, one can handle 'normal' sentences as usual with no significant interference from new additions to the analysis process. Only when failures occur during such primary attempts would the executive try some exceptional procedures. For example, there could be a rule specifying that the real patient may (under appropriate

restrictions) be the contents of the apparent patient:

(*)"I drank a glass of milk." (3)

Failure of the analysis as in (4a) leads to (4b):



(4a)

(4b)

Eventually, one could conceivably comprehend metaphors and humor as particular forms of soft failures. At this stage, however, it is hard to see what makes a metaphor poetic rather than non-sensical, or why a sentence is funny or ridiculous. Present research is still quite far from such considerations.

5.3 COMPUTERS AND CHILDREN.

Modeling the human mind was not the principal goal of this thesis. Nevertheless, in attacking the problems of understanding and learning from English text, one naturally turns to human intelligence for inspiration. This section discusses some of the apparent similarities and differences between computers and children in their approaches to language, and, in general, to thinking. Afterward, special consideration is given to the differences in the learning process. This will show why teaching methods cannot be usefully compared.

5.3.1 Similarities.

a) Structure. We have been emphasizing the importance of well structured information in many places throughout the thesis. It is interesting to note the reasons given to justify the importance of structure to children ([Biehler 1971], p. 264): Structure helps the child in establishing links between the new information and his previous knowledge. Short of this, he finds the material rather dull; he forgets it more easily; finally, generalization becomes exceedingly difficult.

The main reason has a clear equivalent in our system where the unique semantic memory offers a neatly structured framework for old and new information. From the three cited disadvantages of the lack of structure, the last one seems to be the only one relevant to our system. In fact, they all derive from the same main point: without sufficient structure, the child does not understand well enough. Similar deficiencies would also make our system fail to understand the new material. (CLET'S failure would, however, be total; this difference is discussed below in 5.3.2-c.)

b) Previous Knowledge. In our system, previous knowledge participates globally through the use of semantic memory as a 'grammar' for the basic SM-scheme. In a more 'productive' fashion, the initial state of the memory permits recovery of implicit information underlying direct noun-noun relations; it also determines immediate development of new material through IMM-PROP and IMM-CONS relations (see sections 3.3.2 and 4.3.2). Psychologists do not seem to propose any specific reasons for the importance of previous knowledge to human learning. The following are typical statements (from [Scandura 1969]):
- practice on prerequisite material significantly improves the learning of higher order tasks.
- the learner's knowledge affects his future learning only when this knowledge is prerequisite to the material to be learned.
- simple exposure to prerequisite information is often not sufficient to insure later learning.

In general, previous knowledge has great influence on humans' understanding of language, on their literary style, their visual perception, etc. The experiment described below is precisely an attempt by psychologists at demonstrating the effect of one's expectation on one's vision:

> [The experiment] uses a distorted room in which the floor slopes to the right of the observer, the rear wall recedes from right to left and the windows are of different sizes and trapezoidal in shape. When an observer looks at this room with one eye from a certain point, the room appears completely normal, as if the floor were level, the rear wall at right angles to the line of sight and the windows rectangular and of the same size. Presumably the observer chooses this particular appearance instead of some other because of the assumptions he brings to the occasion. ([Ittelson & al. 1951], p. 335)

Influence of previous knowledge in this manner has several advantages: by avoiding details, it simplifies recognition and contributes to economy of representation. However, it is not really clear to what extent it is desirable to allow computers to have illusions. (On this general problem, see [Gregory 1967].)

c) Time Requirements. Earlier, section 4.5 exposed the enormous combinatorial explosion facing the induction process. We also mentioned some of the heuristics that could be used in a fully computerized system. However, these do not solve the problem entirely, they simply reduce the search space. Indeed, it is hard to see how induction could be performed without requiring a substantial time for all the apparently necessary processing. With all its power, the human mind cannot keep up with its learning tasks unless it is given some rest. In fact, sleep seems to significantly improve learning, at least by avoiding interference with daily activities and other concepts which distract the mind ([Manis 1968], p. 27). One might conjecture that part of the brain attends to sensory or other conscious activities, while the other part takes care of sub-conscious activities, such as long-term memory, inductive learning, etc. Furthermore, total brain capacity might be insufficient for the normal waking load, thus requiring sleep. In any case, the main point of this speculation is not so much to formulate conjectures in psychobiology, but rather to suggest that such a background-foreground parallel organization might be desirable for a computer system.

5.3.2 Differences.

When skimming through the artificial intelligence literature, one can easily be misled by false optimism. Computers appear to think, understand, respond, deduce, generalize, and so on. Eventually, one is tempted to exagerate the similarities between present programs and human mental processes. Yet, differences are countless; moreover, they are qualitatively important. The discussion below will be restricted to some points that appeared more relevant to our research.

a) Language Generation. For years now, traditional thinking has suggested little distinction between analysis and synthesis of languages. Insofar as humans are concerned, little is known about either process. Though it is hard to evaluate this objectively,

children seem to learn both more or less simultaneously. In the case of computers, the two problems have been considered equivalent. But there is growing disagreement with this trend. The same syntax cannot be used in both directions, except perhaps at the cost of enormous inefficiency, which makes it very unnatural. The area of semantics is even more obscure. The whole problem of language generation has been very much neglected in artificial intelligence research; very little has been published on the subject. Current results are far more impressive in analysis than in synthesis of language. Extensive research is needed in this area.

b) Integration of Perceptions in Memory. Many diverse sensory inputs enter the human memory. Moreover, they are often recalled in their raw form as if they had undergone little or no post-processing. A child may, for example, have verbal memorization of something he tried to learn, without really understanding much of it. One can remember sceneries, musical themes, food flavors, textures, etc. If memory does have a uniform format, the basic element cannot be the word or any such highly abstract entity.

It is more difficult to make such definite statements concerning that part of memory connected with linguistic input. Does one remember the sound, the written equivalent, some structure similar to those in CLET, or some "deep deep" structure? Consider the following arithmetic problem:

> At noon, John sat at his desk to study. As he was closing his books, two hours later, he realized that 40 minutes ago, his friend had called to remind him of an appointment 15 minutes later. What time was his appointment set for? (Answer: 1:35 P.M.)

When confronted with this problem, adults have a variety of reactions, depending on their tastes and backgrounds. Some concentrate on the verbal aspect, some on the visual. Some mention words, numbers, the clock on the wall of the study room, the events placed on a time axis, etc. Also, if the above story is not announced in advance as an arithmetic problem, and the final question is removed, listeners do not concentrate so much on the timing. Rather, their attention goes to John's anxiety, to his studies, or "globally" to the whole story.

It is not our purpose to provide definite explanations. We note that, in most cases, some 'visualization' of linguistic input does take place. On the other hand, it seems important to distinguish memory and thinking, i.e., the static vs. dynamic

aspects of the human mind, however intricate their relationship may be. When thinking of "this week" and "last week," I may associate them with two adjacent intervals on a line, or with a diary, or many other things. However, these associations can be all recovered even if one chooses a simple representation as in CLET: (this week) or (last week). The main point is that each node is connected to many other concepts in memory, and these connections can be selectively used for further processing.

We must distinguish the objects themselves, our perception of them, and the various forms we can associate with them for the sake of reasoning. Current computer systems are quite far from this stage; they would certainly benefit from any increased knowledge we can have of how this is done in humans. Introspection on this issue is often confusing. In a remarkable paper, Minsky gives deep insight into the heart of this difficulty [Minsky 1965]. But years of research are still needed before computers can truly simulate man's use of models of the world and of himself in his thinking.

c) Thought Processes. Current artificial intelligence systems follow a "black box" approach to intelligence. They generally attempt an overall simulation of some small and well defined aspect of the thought process, with little concern for the way it is performed in humans. Often, they might compare well with human abilities at some particular moment. But the learning process that led gradually to this stage is dramatically different. Computers today compare very poorly with children on many points; yet, they could win a chess game against most adults. Computers are often called upon to perform college level matrix operations; yet, they can hardly learn from elementary grade textbooks. Such differences are a characteristic weakness of the "black box" approach. The author was often asked the following question: "What grade has the computer reached now?" In fact, such a question seems meaningless to someone who is aware of the basic differences.

Children may not have their syntax subdivided into a base component grammar, and a set of various transformations. It is not even clear that they make a distinction between syntax and semantics. Moreover,

> Piaget argues that it is not until the age of eleven or twelve that a child becomes capable of making deductions, of mastering logical conceptions involving cause and effect. ([Biehler 1971], p. 80)

Most crucial is the fact that children can and do achieve surprisingly impressive results in spite of such seeming deficiencies. It is hard to conceive of any current computer system performing something of interest if it were deprived of its deductive abilities. Of course, Piaget could be wrong about deduction in children. But deduction is not the only point. One can compare the child's learning from a complex sentence which is beyond his own syntax and vocabulary, and a typical computer analyzer's complete rejection of sentences that do not fit its syntax exactly.

One could cite many other examples. But are these differences important? We believe they are. We believe that it is more essential for computers to imitate man's abilities to learn than his static abilities to reason. As long as computers are much better on the latter instead of the former, one should pause and consider how humans really do develop their language and problem-solving abilities. This demands long term fundamental research. Practical considerations do not seem to have permitted this, except on a small scale, mostly among psychologists.

5.3.3 Methods of Teaching.

The previous sub-section 5.3.2 demonstrated the enormous difference between children and present computer systems. In view of that, it is no surprise that teaching methods appropriate to each category are hard to compare.

Children do not merely learn from their classroom study.

> It is a great mistake to suppose that a child acquires the notion of number and other mathematical concepts just from teaching. [...] When adults try to impose mathematical concepts on a child prematurely, his learning is merely verbal; true understanding of them comes only with his own mental growth. ([Piaget 1953], p. 76)

On the other hand, children can perform rote learning where they do not really understand the material. Even in this case, they are somehow able to use their acquired knowledge to solve new problems fairly successfully.

Two main schools of thought divide psychologists concerned with learning theory. Led by Skinner ([Skinner 1968]), one school favors programmed instruction. On the other hand, Gestalt psychologists emphasize the importance of insight (e.g., see Piaget's theory in [Ginsburg & al. 1969]; also, [Gagne 1959], and [Bruner 1966]). No contradictions oppose these two schools. Differences in their theories are merely a matter of emphasis. The former focuses on reinforcement, while the latter insists on understanding. It seems reasonable to expect both aspects to be quite important in human learning.

Many experiments were carried out in order to compare the effects, both short and long term, of various methods of teaching. In some cases [Kersh 1963], the experiments revealed the following order, from best to worst: rote learning, guided discovery, programmed instruction! Stephens reports that "about the same amount of learning takes place regardless of the instruction method used" (in [Biehler 1971], p. 201). In fact, so many factors interfere with these experiments that it is hard to draw definite conclusions: "There is no trustworthy evidence that one approach is superior to the other." (ibid., p. 237).

Computer learning does not shed much light on the matter. The controversy over guided discovery vs. rote learning stems from the following idea: if the child infers the rule by himself from examples and questions, one can feel confident that he really understands; if the rule is taught explicitly, his task is easier but he may not have understood. This problem does not exist in today's computer approaches. Similarly, reinforcement is connected with feelings of being rewarded which has no obvious counterpart for computer programs.

On the other hand, one can probably say that the computer, which has no understanding of the pictures accompanying the written text (they are not even part of the input presented to the program), has better understanding of the text itself than the average child. In general, it is much better than the child at handling abstract symbolic inferences and so much worse at dealing with poorly defined material. It is then quite obvious that the computer should require near-perfect understanding and care very little about reinforcement. These results have little relevance for the education of children.

Thus, present research makes little contribution to educational psychology. Nevertheless, as computer approaches get more and more elaborate, we gradually increase our understanding of the phenomenon known as intelligence. One might hope that this will eventually give us greater insight into the human mind.

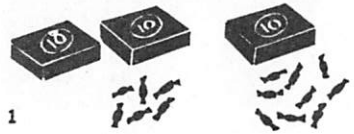
5.4 INTELLIGENCE.

At the present stage of research, computers are still quite remote from achieving a level of intelligence comparable in complexity to human thinking. At this point, it is premature to be concerned with efficiency. Given any amount of memory storage and any speeds of access and of operation, no current approach could solve the entire problem of intelligence. Research is still bound by too many fundamental questions that remain unanswered. (We do not even seem to have reached the stage where these questions can be formulated precisely.)

As computers become more powerful, and thus more influential in human affairs, the philosophical aspects of computer learning become increasingly overshadowed by the practical need to develop an operational understanding of the limitations and feasiblity of machine intelligence. It is hoped that this work has served to clarify at least one aspect of computer learning. Within its narrow field of competence, CLET does serve as a demonstration that computer-learning of elementary, arithmetic procedures based on English text is possible. It is our conviction that such techniques can be extended to larger bodies of text and to other domains. Finally, it is believed that such techniques will be essential for any future system that aspires to increasing levels of human-like intelligence.

APPENDIX 1
T E X T

see



1

A Ed had 26 candies. Sue gave him 19 more. Then Ed had how many candies?

$26+19=t.$

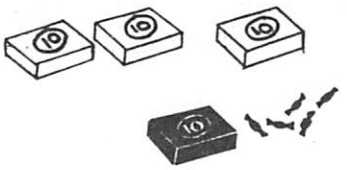
You are to find the sum of 26 and 19.

2

Put the 9 candies with the 6 candies. There are 15 candies.

$$\begin{array}{r} 26 \\ \underline{19} \end{array}$$

Add the ones.
There are 15 ones.



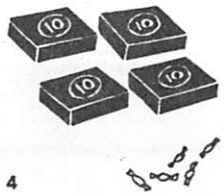
3

Put 10 of the 15 candies into a box. 5 candies are not in boxes of 10.

$$\begin{array}{r} {\scriptstyle 1} \\ 26 \\ \underline{19} \\ 5 \end{array}$$

15 ones are 1 ten 5 ones.
Write 5 in the ones' place of the answer to show there are 5 ones.

Write 1 above the 2 in the tens' place to show there is one more ten.



4

Put the boxes of 10 candies together. There are 4 boxes of 10 candies.

$$\begin{array}{r} {\scriptstyle 1} \\ 26 \\ \underline{19} \\ 45 \end{array}$$

Add the tens. There are 4 tens.

Write 4 in the tens' place of the answer to show there are 4 tens.

5

There are 45 candies in all.

$26 + 19 = 45.$

Then Ed had 45 candies.

think

B David sold 23 tickets, Mark sold 46 tickets, and Jim sold 85 tickets. Altogether the boys sold how many tickets?

$23+46+85=k.$

You are to find the sum of 23, 46, and 85.

$$\begin{array}{r} 23 \\ 46 \\ \underline{85} \end{array}$$

What do you add first?
How many ones are there?

$$\begin{array}{r} {\scriptstyle 1} \\ 23 \\ 46 \\ \underline{85} \\ 4 \end{array}$$

Think of 14 ones as 1 ten ■ ones.

Why is 4 written in the ones' place of the answer?

Why is 1 written above the 2 in the tens' place?

$$\begin{array}{r} {\scriptstyle 1} \\ 23 \\ 46 \\ \underline{85} \\ 4 \end{array}$$

What do you add next?
How many tens are there?

Think of 15 tens as 1 hundred ■ tens.

Why is 5 written in the tens' place of the answer?

Why is 1 written in the hundreds' place of the answer?

$$\begin{array}{r} {\scriptstyle 1} \\ 23 \\ 46 \\ \underline{85} \\ 154 \end{array}$$

$23 + 46 + 85 = k.$

Altogether the boys sold ■ tickets.

try **A** There were 38 chairs in one room, 16 in another, and 26 in a third room. How many chairs were in the three rooms?

$38 + 16 + 26 = m.$
$38 + 16 + 26 = 80.$

80 chairs were in the three rooms.

2 ← Either remember this number or show it.
38
16
26
80

B $28 + 54 = m.$

$$\begin{array}{r} 28 \\ 54 \\ \hline 82 \end{array}$$

$28 + 54 = 82.$

C $61 + 8 + 20 = m.$

$$\begin{array}{r} 61 \\ 8 \\ 20 \\ \hline 89 \end{array}$$

$61 + 8 + 20 = 89.$

D $42 + 59 + 75 = m.$

$$\begin{array}{r} 42 \\ 59 \\ 75 \\ \hline 176 \end{array}$$

$42 + 59 + 75 = 176.$

do **A** 41 candy sticks were in a jar. A clerk put 37 candy sticks with them. Then there were how many candy sticks in the jar?

$41 + 37 = r.$

E Bill delivered 22 newspapers on one street, 56 on another, and 43 on a third street. How many newspapers did Bill deliver?

$22 + 56 + 43 = r.$

block 1

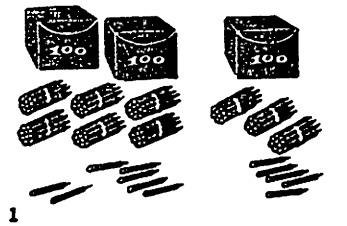
A $67 + 23 = f.$
B $f = 85 + 29.$
C $36 + 21 = f.$
D $90 + 48 = f.$
E $f = 44 + 51.$
F $40 + 86 = f.$
G $f = 62 + 83.$
H $74 + 59 = f.$
I $65 + 27 + 32 = f.$
J $52 + 4 + 59 = f.$
K $25 + 78 = f.$
L $14 + 18 + 47 = f.$
M $f = 19 + 77.$
N $82 + 9 + 68 = f.$
O $f = 12 + 54 + 99.$
P $f = 95 + 35.$
Q $f = 71 + 24 + 69.$
R $f = 8 + 63 + 87.$

block 2

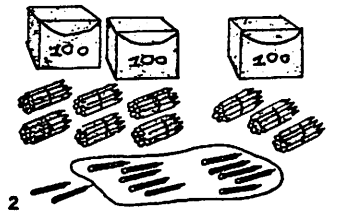
A $a = 49 + 73 + 92.$
B $a = 80 + 57 + 30.$
C $89 + 79 = a.$
D $a = 58 + 10 + 91.$
E $84 + 93 = a.$
F $96 + 6 + 15 = a.$
G $17 + 7 + 5 = a.$
H $60 + 42 + 98 = a.$
I $a = 66 + 76 + 39.$

2 Computation

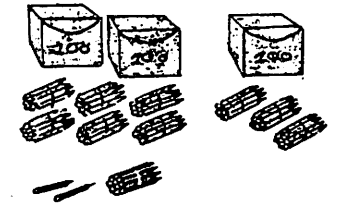
see



1



2



3

A Mr. Clark sold 267 pencils last week and 135 pencils this week. How many pencils did Mr. Clark sell in the two weeks?

$267 + 135 = r.$

You are to find the sum of 267 and 135.

Put the 5 pencils with the 7 pencils. There are 12 pencils.

$$\begin{array}{r} 267 \\ 135 \\ \hline \end{array}$$

Add the ones.
There are 12 ones.

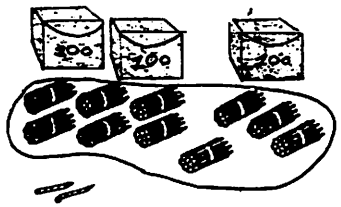
Put 10 of the 12 pencils into a bundle. 2 pencils are not in bundles of 10.

12 ones are 1 ten 2 ones.

$$\begin{array}{r} {\scriptstyle 1} \\ 267 \\ 135 \\ \hline 2 \end{array}$$

Write 2 in the ones' place of the answer to show there are 2 ones.

Write 1 above the 6 in the tens' place to show there is one more ten.

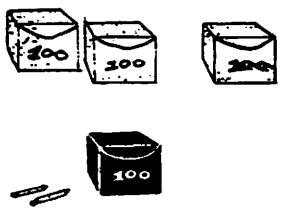


4

Put the bundles of 10 pencils together. There are 10 bundles of 10 pencils.

$$\begin{array}{r} {\scriptstyle 1} \\ 267 \\ 135 \\ \hline 2 \end{array}$$

Add the tens.
There are 10 tens.



5

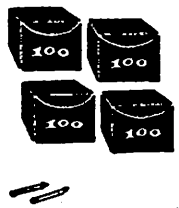
Put the 10 bundles of 10 pencils into a box.

10 tens are 1 hundred.

$$\begin{array}{r} {\scriptstyle 1\,1} \\ 267 \\ 135 \\ \hline 02 \end{array}$$

Write 0 in the tens' place of the answer to show there are no tens.

Write 1 above the 2 in the hundreds' place to show there is one more hundred.



6

Put the boxes of 100 pencils together. There are 4 boxes of 100 pencils.

$$\begin{array}{r} {\scriptstyle 1\,1} \\ 267 \\ 135 \\ \hline 402 \end{array}$$

Add the hundreds.
There are 4 hundreds.

Write 4 in the hundreds' place of the answer.

$267 + 135 = 402.$

Mr. Clark sold 402 pencils in the two weeks.

think ■ Ed collected 698 pennies, Steve collected 475 pennies, and Jim collected 780 pennies. The boys collected how many pennies in all?

$698 + 475 + 780 = r.$

You are to find the sum of 698, 475, and 780.

$$\begin{array}{r} 698 \\ 475 \\ 780 \end{array}$$

What do you add first?
How many ones are there?

Think of 13 ones as 1 ten ■ ones.

$$\begin{array}{r} {\scriptstyle 1} \\ 698 \\ 475 \\ 780 \\ \hline 3 \end{array}$$

Why is 3 written in the ones' place of the answer?

Why is 1 written above the 9 in the tens' place?

$$\begin{array}{r} {\scriptstyle 1} \\ 698 \\ 475 \\ 780 \\ \hline 3 \end{array}$$

What do you add next?
How many tens are there?

Think of 25 tens as ■ hundreds 5 tens.

$$\begin{array}{r} {\scriptstyle 2\,1} \\ 698 \\ 475 \\ 780 \\ \hline 53 \end{array}$$

Why is 5 written in the tens' place of the answer?

Why is 2 written above the 6 in the hundreds' place?

$$\begin{array}{r} {\scriptstyle 2\,1} \\ 698 \\ 475 \\ 780 \\ \hline 53 \end{array}$$

What do you add now?
How many hundreds are there?

Think of 19 hundreds as 1 thousand ■ hundreds.

$$\begin{array}{r} {\scriptstyle 2\,1} \\ 698 \\ 475 \\ 780 \\ \hline 1953 \end{array}$$

Why is 9 written in the hundreds' place of the answer?

Why is 1 written in the thousands' place of the answer?

$698 + 475 + 780 = r.$

The boys collected ■ pennies in all.

buy **A** Today Ann bought 148 sheets of red paper, 195 sheets of blue paper, and 281 sheets of green paper. How many sheets of paper did Ann buy today?

$148 + 195 + 281 = b.$
$148 + 195 + 281 = 624.$

Ann bought 624 sheets of paper today.

2 1 ← Either remember these numbers or show them.

148
195
281
624

B $527 + 84 = b.$

527
84
611

$527 + 84 = 611.$

C $453 + 160 + 294 = b.$

453
160
294
907

$453 + 160 + 294 = 907.$

D $351 + 685 + 29 = b.$

351
685
29
1065

$351 + 685 + 29 = 1065.$

do **A** There were 285 buttons in a box. Carol put 65 more with them. Then how many buttons were in the box?

$285 + 65 = j.$

B 347 chairs are in one circus tent, and 269 chairs are in another circus tent. How many chairs are in the two circus tents?

$347 + 269 = j$ or $269 + 347 = j.$

block 1

A $955 + 18 = x.$
B $346 + 732 = x.$
C $x = 207 + 421.$
D $128 + 189 = x.$
E $560 + 350 = x.$
F $x = 406 + 39.$
G $x = 626 + 796.$
H $x = 597 + 9.$
I $437 + 414 = x.$
J $x = 117 + 290.$
K $693 + 53 = x.$

block 2

Add.

A 594, 131, 254
B 230, 33, 464
C 988, 698, 273
D 791, 48, 3
E 857, 643, 224
F 286, 141, 763, 200
G 672, 349, 83
H 516, 392, 36, 625
I 838, 777, 678
J 176, 262, 395, 578

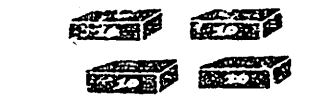
block 3

A $529 + 70 = a.$
B $a = 366 + 112 + 351.$
C $a = 819 + 84.$
D $a = 505 + 27 + 202.$
E $181 + 158 + 479 = a.$
F $987 + 368 = a.$
G $a = 413 + 134 + 201 + 142.$
H $6 + 784 + 15 = a.$
I $444 + 856 = a.$
J $859 + 99 + 878 = a.$
K $a = 380 + 71 + 422 + 45.$

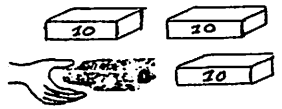
3 Computation

Subtraction (two-digit numerals).

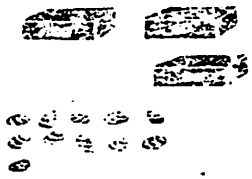
see



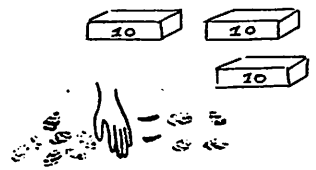
1



2



3



4

A Bill found 41 shells. He gave 27 of them to his brother. Then Bill had how many shells?

$41 - 27 = w.$

You are to find the difference of 41 and 27.

Before you can remove 7 shells, you must open one box of 10 shells.

41
27

Before you can subtract 7 ones, there must be more ones.

3 11
41
27

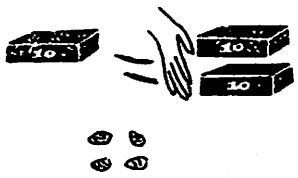
Think of 4 tens 1 one as 3 tens 11 ones.

Remove 7 of the 11 shells. There are 4 shells left.

3 11
41
27
4

Subtract 7 from 11.
$11 - 7 = 4.$

Write 4 in the ones' place of the answer to show there are 4 ones.



5

6

Now remove 2 of the boxes of 10 shells. There is 1 box of 10 shells left.

3 11
4 1
2 7
1 4

Subtract 2 tens from 3 tens.
30 − 20 = 10, or 1 ten.

Write 1 in the tens' place of the answer to show there is 1 ten.

There are 14 shells left.

41 − 27 = 14.

Then Bill had 14 shells.

---

think B There were 50 pine cones in a box. Mary removed 18 of them. How many pine cones were left in the box?

50 − 18 = *s*.

You are to find the difference of 50 and 18.

---

5 0
1 8

What must you do before you can subtract 8 ones?

---

4 10
5 Ø
1 8

Think of 5 tens as 4 tens ■ ones.

4 10
5 Ø
1 8
2

Subtract 8 from ■.
10 − 8 = ■.

Why is 2 written in the ones' place of the answer?

---

4 10
5 Ø
1 8
3 2

Subtract 1 ten from ■ tens.
40 − 10 = ■, or ● tens.

Why is 3 written in the tens' place of the answer?

50 − 18 = *s*.

■ pine cones were left in the box.

try A Mrs. Long had 82 puppets to sell. She sold 39 of them. How many puppets did she have left to sell?

7 12
8 2
3 9
4 3

Either remember these numbers or show them.

82 − 39 = *g*.
82 − 39 = 43.

She had 43 puppets left to sell.

B 75 − 6 = *g*.

7 5
6
6 9

75 − 6 = 69.

C 58 − 42 = *g*.

5 8
4 2
1 6

58 − 42 = 16.

D 97 − 67 = *g*.

9 7
6 7
3 0

97 − 67 = 30.

E 63 − 18 = *g*.

6 3
1 8
4 5

63 − 18 = 45.

---

do A Mary Jane made 71 paper flowers. She gave 25 of them to her grandmother. How many paper flowers did Mary Jane have left?

71 − 25 = *y*.

S There were 34 red candles in a box. Nancy removed 9 of them. How many red candles were left in the box?

34 − 9 = *y*.

block 1

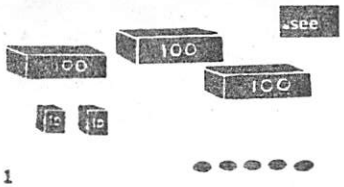
A 52 − 3 = *t*.
B *t* = 23 − 13.
C 90 − 47 = *t*.
D 76 − 59 = *t*.
E *t* = 61 − 12.
F *t* = 20 − 4.
G 96 − 81 = *t*.
H 68 − 29 = *t*.
I *t* = 74 − 57.
J *t* = 80 − 15.
K *t* = 94 − 86.
L 66 − 8 = *t*.
M 79 − 21 = *t*.
N 82 − 54 = *t*.
O *t* = 33 − 26.
P *t* = 92 − 78.
Q 64 − 30 = *t*.
R *t* = 88 − 69.

block 2

A 60 − 56 = *m*.
B 87 − 9 = *m*.
C *m* = 91 − 63.
D 51 − 11 = *m*.
E 43 − 19 = *m*.
F *m* = 35 − 7.
G *m* = 31 − 14.
H *m* = 48 − 45.
I 57 − 28 = *m*.
J 99 − 36 = *m*.
K *m* = 73 − 44.
L 34 − 5 = *m*.
M 85 − 49 = *m*.
N *m* = 76 − 17.
O 90 − 40 = *m*.
P *m* = 22 − 16.
Q *m* = 65 − 38.
R 89 − 32 = *m*.

4 Computation

Subtraction (three-digit numerals)

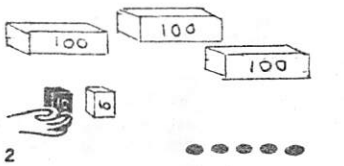


1

A The children at Long School had 325 school buttons to sell. They have sold 168 of the buttons. How many buttons do they have left to sell?

$325 - 168 = v.$

You are to find the difference of 325 and 168.

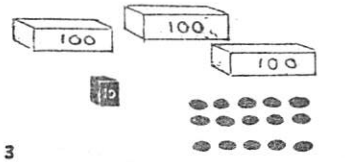


2

Before you can remove 8 buttons, you must open one box of 10 buttons.

$$\begin{array}{r} 325 \\ 168 \\ \hline \end{array}$$

Before you can subtract 8 ones, there must be more ones.

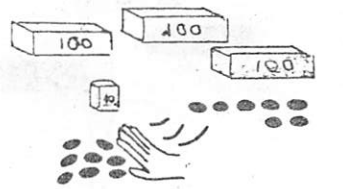


3

Now there are 1 box of 10 buttons and 15 buttons.

$$\begin{array}{r} {\scriptstyle 1\,15} \\ 3\not{2}\not{5} \\ 168 \\ \hline \end{array}$$

Think of 2 tens 5 ones as 1 ten 15 ones.



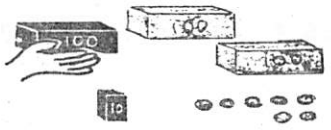
4

Remove 8 of the 15 buttons. There are 7 buttons left.

$$\begin{array}{r} {\scriptstyle 1\,15} \\ 3\not{2}\not{5} \\ 168 \\ \hline 7 \end{array}$$

Subtract 8 from 15.
$15 - 8 = 7$.

Write 7 in the ones' place of the answer to show there are 7 ones.

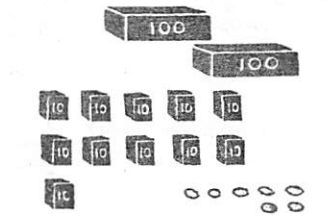


5

Before you can remove 6 boxes of 10 buttons, you must open 1 box of 100 buttons.

$$\begin{array}{r} {\scriptstyle 1\,15} \\ 3\not{2}\not{5} \\ 168 \\ \hline 7 \end{array}$$

Before you can subtract 6 tens, there must be more tens.

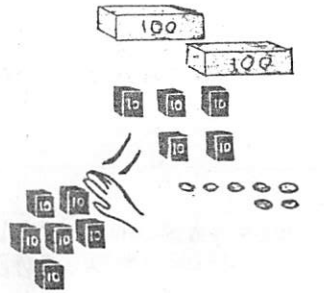


6

Now there are 2 boxes of 100 buttons and 11 boxes of 10 buttons.

$$\begin{array}{r} {\scriptstyle 2\,11} \\ 3\not{2}\not{5} \\ 168 \\ \hline 7 \end{array}$$

Think of 3 hundreds 1 ten as 2 hundreds 11 tens.



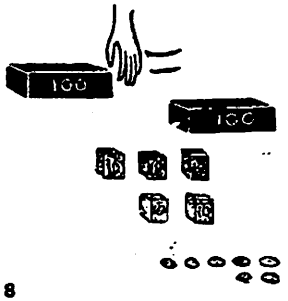
7

Remove 6 of the 11 boxes of 10 buttons. There are 5 boxes of 10 buttons left.

$$\begin{array}{r} {\scriptstyle 2\,11} \\ 3\not{2}\not{5} \\ 168 \\ \hline 57 \end{array}$$

Subtract 6 tens from 11 tens.
$110 - 60 = 50$, or 5 tens.

Write 5 in the tens' place of the answer to show there are 5 tens.

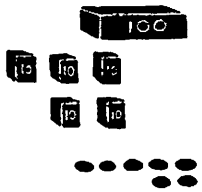


8

Remove 1 box of 100 buttons. There is 1 box of 100 buttons left.

$$\begin{array}{r} {\scriptstyle 2\ 11} \\ 3\not{2}\not{5} \\ 168 \\ \hline 157 \end{array}$$

Subtract 1 hundred from 2 hundreds.
200 − 100 = 100, or 1 hundred.
Write 1 in the hundreds' place of the answer to show there is 1 hundred.



9

There are 157 buttons.

325 − 168 = 157.

They have 157 buttons left to sell.

think B There were 203 sheets of drawing paper in a box. Dan removed 136 of the sheets. Then how many sheets of drawing paper were in the box?

$203 - 136 = r.$

You are to find the difference of 203 and 136.

$$\begin{array}{r} 203 \\ 136 \\ \hline \end{array}$$

What must you do before you can subtract 6 ones?

What must you do before you can think of 1 ten as 10 ones?

$$\begin{array}{r} {\scriptstyle 1\ 10} \\ 2\not{0}3 \\ 136 \\ \hline \end{array}$$

Think of 203 as 1 hundred ■ tens 3 ones.

$$\begin{array}{r} {\scriptstyle 9} \\ {\scriptstyle 1\ 1\not{0}13} \\ 2\not{0}\not{3} \\ 136 \\ \hline \end{array}$$

Think of 10 tens 3 ones as ■ tens 13 ones.

$$\begin{array}{r} {\scriptstyle 9} \\ {\scriptstyle 1\ 1\not{0}13} \\ 2\not{0}\not{3} \\ 136 \\ \hline 7 \end{array}$$

Subtract 6 from ■.
13 − 6 = ■.
Why is 7 written in the ones' place of the answer?

$$\begin{array}{r} {\scriptstyle 9} \\ {\scriptstyle 1\ 1\not{0}13} \\ 2\not{0}\not{3} \\ 136 \\ \hline 67 \end{array}$$

Subtract 3 tens from ■ tens.
90 − 30 = ■, or ● tens.
Why is 6 written in the tens' place of the answer?

$$\begin{array}{r} {\scriptstyle 9} \\ {\scriptstyle 1\ 1\not{0}13} \\ 2\not{0}\not{3} \\ 136 \\ \hline 67 \end{array}$$

What numbers do you subtract now?
100 − 100 = ■.
Tell why you do not write a numeral in the hundreds' place of the answer.

203 − 136 = r.

Then ■ sheets of drawing paper were in the box.

try A There were 480 beads in a jar. The children used 94 of the beads on puppets. How many beads were left in the jar?

480 − 94 = y.
480 − 94 = 386.

386 beads were left in the jar.

$$\begin{array}{r} {\scriptstyle 3\ 17} \\ 4\not{8}\not{0} \\ 94 \\ \hline 386 \end{array}$$

← Either remember these numbers or show them.

B 851 − 549 = y.

$$\begin{array}{r} 851 \\ 549 \\ \hline 302 \end{array}$$

851 − 549 = 302.

C 974 − 480 = y.

$$\begin{array}{r} 974 \\ 480 \\ \hline 494 \end{array}$$

974 − 480 = 494.

D 601 − 373 = y.

$$\begin{array}{r} 601 \\ 373 \\ \hline 228 \end{array}$$

601 − 373 = 228.

A John had 192 football cards. He gave 65 of them to his brother. Then John had how many football cards?

$192 - 65 = c.$

B A farmer had 548 plants to sell. He sold 257 of them. How many plants did the farmer have left to sell?

$548 - 257 = c.$

block 1

A $767 - 682 = z.$
B $536 - 397 = z.$
C $919 - 821 = z.$
D $z = 700 - 246.$
E $z = 839 - 450.$
F $362 - 78 = z.$
G $114 - 51 = z.$
H $z = 607 - 327.$
I $480 - 215 = z.$
J $993 - 704 = z.$
K $285 - 91 = z.$
L $870 - 699 = z.$
M $521 - 84 = z.$
N $z = 756 - 563.$
O $590 - 491 = z.$
P $z = 843 - 387.$
Q $z = 762 - 493.$
R $615 - 139 = z.$
S $421 - 261 = z.$
T $z = 872 - 207.$

block 2

A $k = 948 - 391.$
B $k = 829 - 542.$
C $256 - 78 = k.$
D $k = 979 - 506.$
E $800 - 787 = k.$
F $644 - 396 = k.$
G $750 - 188 = k.$
H $435 - 145 = k.$
I $k = 336 - 252.$
J $k = 711 - 103.$

8 Sets

block 1

For each exercise tabulate the union of the two sets. Then tabulate the intersection.

A set D: {293, 294}
set E: {291, 293, 295}

B set X: {400, 500, 600, 700}
set Y: {200, 300, 800, 900}

C set S: {546, 547, 548, 549}
set T: {547, 549, 548, 546}

block 2

Tabulate each truth set.
Use {0, 1, 2, . . . , 999}.

A $s < 446.$
B $999 < s.$
C $s > 995.$
D $s = 582.$
E $111 > s.$
F $321 < s.$
G $s > 997.$
H $s < 6.$
I s is between 2 and 900.
J s is between 602 and 809.
K s is between 723 and 724.
L s is between 88 and 91.

3 Writing arithmetic

For each problem make an arithmetic sentence, make a true statement, and give the answer to the problem.

A 35 children were ice-skating. 19 other children joined them. Then how many children were ice-skating?

B A clerk sold 4 boxes of puppets. 3 puppets were in each box. In all the clerk sold how many puppets?

C Bob painted 96 pine cones. He sold 27 of them to Mrs. Bell. How many of the 96 pine cones did Bob have left?

D At Field School there were 66 children in one band and 59 in another band. How many children were in the two bands?

E Three girls collect butterflies. Kay has 37 butterflies, Ann has 25, and Ellen has 42. In all the girls have collected how many butterflies?

F Tim put fifteen ounces of candy into bags. He put three ounces into each bag. How many bags of candy did Tim have?

G Mr. Long collected 102 pennies. He gave 40 of the pennies to Bill. How many of the 102 pennies did Mr. Long have left?

H Sally put 678 beads into one box and 428 beads into another box. How many beads did Sally put into the two boxes?

I A farmer had 383 chickens to sell. He sold 192 of them. How many chickens did he have left to sell?

J Ed put 16 postcards into a book. He put 2 postcards on each page. How many pages did he use?

K Ray had 115 dimes in his bank. He removed 17 of the dimes. Then Ray had how many dimes in his bank?

L Last month Mr. Clark delivered 360 quarts of milk. This month he delivered 480 quarts of milk. How many quarts of milk did Mr. Clark deliver in the two months?

M 241 spools of ribbon were on a shelf. A clerk removed 118 of the spools. How many spools of ribbon were left on the shelf?

N Each of three girls wrapped six packages. Altogether the girls wrapped how many packages?

O David has 105 stamps, Steve has 89 stamps, and Dan has 238 stamps. Altogether the three boys have how many stamps?

APPENDIX 2
D I C T I O N A R Y

The dictionary used for syntactic analysis has a somewhat complicated structure. For each word, there may be:
- a syntactic category: ordinary part of speech.
- a set of features (discussed below).
- a root: possibly different from the original word; e.g., "bought" has as its root: "buy." Also, note that homographs are assigned different roots to avoid confusion at the next, semantic level.
- a 'structural category': these are syntactic categories which are used in the semantic tree structures which are output from the analysis; they are not used for the parsing proper. In most cases, the distinction is made simply to conform with standard practice. Thus, "the" and "this" would have the same syntactic category: Nint (Noun introducer). But they would appear in the analyzed structures as Det and Dem (determiner and demonstrative pronoun) respectively.
- a 'special action': this is a Snobol label for a piece of code. This code is executed when the word is encountered in the first phase of initial transformations. Such special routines deal with idiomatic expressions and prepare warning flags (see main text).
- a 'deferred action': this is similar to special actions, except that the corresponding code is executed during base component parsing. This is mostly connected with the use of auxiliaries in interrupted constructions: in declarative sentences, the verb group may include adverbs in virtually any position; similarly, in interrogatives, the verb may be split, and its two parts may be separated by a full noun phrase.

To save on storage, dictionary entries are divided in 10 types (A to J). Furthermore, in the case of homographs, the entry type simply consists of a digit (2 to 5) specifying the number of possibilities. This main word entry would then be followed by the

appropriate number of regular entries.

The following table lists the attributes of each entry type in the same order as they appear in the dictionary entry. (Syntactic and Structural categories are abbreviated to Syncat and Struccat, Special and Deferred action to Specact and Defact, respectively.)

Entry type	Attributes
A	Syncat
B	Syncat Features
C	Syncat Root
D	Syncat Specact
E	Syncat Root Features
F	Syncat Struccat
G	Syncat Defact
H	Syncat Struccat Defact Features
I	Syncat Defact Features
J	Syncat Struccat Root

In the dictionary, the structure of features reflects the possibilities for a given word. Within a particular sentence, features may get further restricted within the limits imposed by the dictionary (e.g., "opened" is derived from "open" and gets specified as having a past tense). A "-" sign as the first character in the feature string specifies that this entry should not be further inflected.

Features are stored as strings composed of feature specifications separated by commas. No particular order is imposed since each specification contains a key. Keys are described in the table below:

Key	Name	Attributes
(Noun features:)		
Abst	Abstract	+, -
Anm	Animate	+, -
Cnt	Count	+, -
Com	Common	+, -
Gen	Gender	M, F, N (masculine, feminine, neutral)
Hum	Human	+, -

(the following three are used for pronouns only)		
Subj	Subject	+, -
Comp	Complement	+, -
Pnum	Person-number	1, 2, 3 - S, P (S, P: singular, plural)
(Verb features:)		
Comp	Complements	0, 1, 2 (number of complements)
Form	Form	0, 1, 2, 3, 4 (infinitive, 3rd singular-person of the present tense, past, past participle, progressive)
Mod	Modal	holds the modal itself, or "Imp" if the sentence is imperative.
Pass	Passive	+, -
Pnum	Person-number	same as above
Prog	Progressive	+, -
S:--	Subject	subject specification. "S:" may be followed by any noun feature specification.
Tens	Tense	Pr, Pa, F (present, past, future)

Note that alternatives may be specified by separating them with a "/," as in "Comp=1/2" for "give." Also, when a feature is left unspecified, the default value usually assumed includes all alternatives of the attribute, except for the following: for a noun, "Pnum=3S/3P;" for a verb, "Comp=0/1," "S:Anm=+," "Tens=Pr," and, if the tense is "present," "Pnum=1S/2S/P."

Finally, to make the features more readable, some abbreviations were used. These are described in the table below:

```
Animal   = Abst=-,Anm=+,Cnt=+,Com=+,Gen=N,Hum=-
C-name   = Abst=-,Anm=-,Cnt=+,Com=+,Gen=N,Hum=-
Female   = Gen=F
Human    = Abst=-,Anm=+,Cnt=+,Com=+,Gen=M/F,Hum=+
Male     = Gen=M
Past     = Form=2/3,Tens=Pa
Present  = Form=0,Tens=Pr
P-name   = Abst=-,Anm=+,Cnt=+,Com=-,Gen=M/F,Hum=+
```

	Word	Entry Type	Syntactic Category	Attributes
1.	a	F	Nint	Art
2.	above	A	Cprep	
3.	add	B	V	
4.	all	A	Q	
5.	altogether	A	Adv	
6.	am	H	Be	Cop Sp.vbe Form=1, Pnum=1S,Tens=Pr
7.	an	J	Nint	Art a
8.	and	D	Conn	Sp.conn
9.	Ann	B	Pn	P-name,female
10.	another	F	Adj	Infadj
11.	answer	B	N	C-name
12.	apply	B	V	
13.	are	H	Be	Cop Sp.vbe Form=1, Pnum=2S/P,Tens=Pr, S:Anm=+/-
14.	arithmetic	B	N	C-name
15.	as	D	Conn	Sp.conn
16.	at	A	Cprep	
17.	bag	B	N	C-name
18.	band	B	N	C-name
19.	bank	B	N	C-name
20.	be	H	Be	Cop Sp.be Form=0, S:Anm=+/-
21.	bead	B	N	C-name
22.	beautiful	A	Adj	
23.	been	H	Be	Cop Sp.been Form=3, S:Anm=+/-
24.	before	A	Sprep	
25.	being	H	Be	Cop Sp.being -, Form=4,Prog=+,S:Anm=+/-
26.	bell	B	N	C-name
27.	Bill	B	Pn	P-name,male
28.	blue	A	Adj	
29.	Bob	B	Pn	P-name,male
30.	book	B	N	C-name
31.	bought	2		
		E	V	buy Comp=1/2,Form=2/3
		E	Vadj	buy
32.	box	B	N	C-name
33.	boy	B	N	Human,male
34.	brother	B	N	Human,male
35.	bundle	B	N	C-name
36.	but	D	Conn	Sp.conn
37.	butterfly	B	N	Animal
38.	button	B	N	C-name
39.	buy	B	V	Comp=1/2,Tens=Pr,

Pass=-,Perf=-

40.	can	I	Mod	Sp.vmod Mod=can
41.	candle	B	N	C-name
42.	candy	B	N	C-name
43.	card	B	N	C-name
44.	Carol	B	Pn	P-name,female
45.	chair	B	N	C-name
46.	chicken	B	N	Animal
47.	children	E	N	child -,Human,plural
48.	circus	B	N	C-name
49.	Clark	B	Pn	P-name,male
50.	clerk	B	N	Human
51.	collect	B	V	
52.	cone	B	N	C-name
53.	Dan	B	Pn	P-name,male
54.	David	B	Pn	P-name,male
55.	deliver	B	V	
56.	did	H	Do	V Sp.vdo Form=2,Tens=Pa
57.	difference	B	N	C-name
58.	dime	B	N	C-name
59.	do	H	Do	V Sp.do Present
60.	does	H	Do	V Sp.vdo Form=1, Pnum=3S,Tens=Pr
61.	doing	E	V	do -,Form=4,Prog=+
62.	done	2		
		E	V	do Form=3
		E	Vadj	do
63.	drawing	3		
		C	Adj	drawing-a
		E	V	draw -,Form=4,Prog=+
		E	N	drawing-n C-name
64.	each	F	Q	Infadj
65.	Ed	B	Pn	P-name,male
66.	either	D	Conn	Sp.either
67.	Ellen	B	Pn	P-name,female
68.	farmer	B	N	Human
69.	field	B	N	C-name
70.	fifteen	2		
		C	Q	15
		E	N	15-n C-name
71.	figure	B	N	C-name
72.	find	B	V	Present,Pass=-,Perf=-
73.	first	2		

		J	Adj	Infadj first-2
		A	Adv	
74.	flower	B	N	C-name
75.	football	B	N	C-name
76.	for	A	Cprep	
77.	found	2		
		E	V	find -,Form=2/3
		E	Vadj	find
78.	from	A	Cprep	
79.	gave	E	V	give -,Comp=1/2, Form=2,Tens=Pa,Pass=-, Perf=-
80.	girl	B	N	Human,female
81.	give	B	V	Comp=1/2,Present, Pass=-,Perf=-
82.	go	B	V	Present,Comp=0,Pass=-, Perf=-
83.	grandmother	B	N	Human,female
84.	green	A	Adj	
85.	had	H	Have	V Sp.had -,Past
86.	has	H	Have	V Sp.vhave -,Form=1, Pnum=3S,Tens=Pr
87.	have	H	Have	V Sp.have Present
88.	having	E	V	have -,Form=4,Prog=+
89.	he	B	Pron	Human,male,Pnum=3S, Comp=-
90.	her	2		
		J	Nint	Poss-f her-p
		E	Pron	she Human,female, Pnum=3S,Subj=-
91.	him	E	Pron	he Human,male,Pnum=3S, Subj=-
92.	his	F	Nint	Poss-m
93.	home	B	N	C-name
94.	how	C	Adv.sprep	Sp.whadv
95.	how+many	A	Q	
96.	hundred	2		
		C	Q	100
		E	N	100-n C-name
97.	ice	B	N	C-name
98.	in	A	Cprep	
99.	in+all	A	Adv	
100.	into	A	Cprep	
101.	is	H	Be	Cop Sp.vbe -,Form=1, Pnum=3S,Tens=Pr
102.	it	B	Pron	C-name,Pnum=3S

103.	Jane	B	Pn	P-name,female
104.	jar	B	N	C-name
105.	Jim	B	Pn	P-name,male
106.	John	B	Pn	P-name,male
107.	join	B	V	
108.	Kay	B	Pn	P-name,female
109.	last	A	Adj	
110.	left	5		
		C	Adj	left-a
		C	Adv	left-adv
		E	N	left-n C-name
		E	V	leave Past
		E	Vadj	leave
111.	long	A	Adj	
112.	made	E	V	make Past
113.	make	B	V	Present,Pass=-,Perf=-
114.	many	A	Q	
115.	Mark	B	Pn	P-name,male
116.	Mary	B	Pn	P-name,female
117.	may	I	Mod	Sp.vmod Mod=may
118.	might	I	Mod	Sp.vmod Mod=might
119.	milk	B	N	C-name
120.	month	B	N	C-name
121.	more	2		
		C	Adjadv	more-c
		J	Adj	Infadj more-n
122.	Mr.	D	Part	Sp.mr.
123.	Mrs.	D	Part	Sp.mrs.
124.	must	I	Mod	Sp.vmod Mod=must
125.	Nancy	B	Pn	P-name,female
126.	newspaper	B	N	C-name
127.	next	2		
		A	Adj	
		A	Adv	
128.	nicely	A	Adv	
129.	no	A	Q	
130.	not	G	Adv	Sp.neg
131.	now	A	Adv	
132.	number	B	N	C-name
133.	numeral	B	N	C-name

134. of	A	Prep	
135. on	A	Cprep	
136. one	2		
	C	Q	1
	E	N	1-n C-name
137. open	2		
	C	Adj	open-a
	B	V	
138. or	D	Conn	Sp.conn
139. other	F	Adj	Infadj
140. ounce	B	N	C-name
141. page	B	N	C-name
142. package	B	N	C-name
143. paint	B	V	
144. paper	B	N	C-name
145. pencil	B	N	C-name
146. penny	B	N	C-name
147. pine	B	N	C-name
148. place	B	N	C-name
149. plant	2		
	E	N	plant-n C-name
	E	V	plant-v
150. postcard	B	N	C-name
151. probably	A	Adv	
152. problem	B	N	C-name
153. puppet	B	N	C-name
154. put	B	V	Form=0/2/3,Tens=Pr/Pa
155. quart	B	N	C-name
156. Ray	B	Pn	P-name,male
157. red	A	Adj	
158. remember	B	V	
159. remove	B	V	
160. ribbon	B	N	C-name
161. room	B	N	C-name
162. safe	A	Adj	
163. Sally	B	Pn	P-name,female
164. school	B	N	C-name
165. see	B	V	Present,Pass=-,Perf=-

166.	sell	B	V	Present,Pass=-,Perf=-
167.	sentence	B	N	C-name
168.	shall	I	Mod	Sp.vmod Pnum=1S/1P, Tens=F
169.	she	B	Pron	Human,female,Pnum=3S, Comp=-
170.	sheet	B	N	C-name
171.	shelf	B	N	C-name
172.	shell	B	N	C-name
173.	should	I	Mod	Sp.vmod Mod=should
174.	show	B	V	
175.	six	2		
		C	Q	6
		E	N	6-n C-name
176.	skate	B	V	
177.	sold	2		
		E	V	sell Past
		E	Vadj	sell
178.	spool	B	N	C-name
179.	stamp	B	N	C-name
180.	statement	B	N	C-name
181.	Steve	B	Pn	P-name,male
182.	stick	B	N	C-name
183.	street	B	N	C-name
184.	subtract	B	V	
185.	Sue	B	Pn	P-name,female
186.	sum	B	N	C-name
187.	tell	B	V	Present,Pass=-,Perf=-
188.	ten	2		
		C	Q	10
		E	N	10-n C-name
189.	tent	B	N	C-name
190.	the	F	Nint	Det
191.	them	E	Pron	they Pnum=3P,Subj=-
192.	then	A	Adv	
193.	there	G	Adv.pron.	Sp.there
194.	these	F	Nint	Dem
195.	they	B	Pron	Pnum=3P,Comp=-
196.	think	B	V	Present,Pass=-,Perf=-
197.	think+of	B	V	Present,Pass=-,Perf=-
198.	third	2		
		J	Adj	Infadj third-2
		A	Adv	
199.	this	F	Nint	Dem
200.	thought+of	B	V	Past
201.	thousand	2		
		C	Q	1000
		E	N	1000-n C-name
202.	three	2		
		C	Q	3
		E	N	3-n C-name
203.	ticket	B	N	C-name

204. Tim	B	Pn	P-name,male
205. to	A	Sprep	
206. today	A	Adv	
207. together	A	Adv	
208. true	A	Adj	
209. try	B	V	
210. two	2		
	C	Q	2
	E	N	2-n C-name
211. use	B	V	
212. was	H	Be	Cop Sp.vbe Form=2, Pnum=1S/3S,Tens=Pa, S:Anm=+/-
213. week	B	N	C-name
214. were	H	Be	Cop Sp.vbe Form=2, Pnum=2S/P,Tens=Pa, S:Anm=+/-
215. what	2		
	J	Adj	Infadj what-2
	B	Pron	C-name
216. why	G	Adv.sprep	Sp.whadv
217. will	I	Mod	Sp.vmod Pnum=2/3, Tens=F
218. with	A	Cprep	
219. would	I	Mod	Sp.vmod Mod=would
220. wrap	B	V	
221. write	B	V	Present,Perf=-,Pass=-
222. written	E	V	write Form=3
223. you	B	Pron	Human,Pnum=2S/2P
224. ***	2		
	A	Q	
	E	N	***-n C-name
225. ****	2		
	A	Q	
	E	N	****-n C-name

APPENDIX 3
INITIAL TRANSFORMATIONS

During the initial phase, dictionary look-up is always performed first. If an entry does exist, no transformation is attempted: the dictionary may override any of the following rules.

1) Numbers:
If the word is an explicit number (string of digits), it is set up as a homograph. Numbers may be used as qauntifiers or nouns.

2) Proper names:
If the word starts with a capital letter, it is considered a proper name unless all of the following holds:
a) it occurs at the beginning of the sentence.
b) the root (same word with first letter non-capitalized) does appear in the dictionary.
c) none of the entries correspond to a proper name.

3) Possessive forms:
If the word ends with "'" or "'s", it is actually shifted within the sentence after the first following noun. A special preposition "xof" is placed in front of it. "xof" is similar to "of" in meaning except that:
a) when there are two or more prepositional phrases, the modification is to the whole preceding group (left-recursion), whereas prepositional phrases introduced by "of" usually modify the last noun (right-recursion).
b) the noun following "xof" may be implicitly preceded by the determiner "the" according to the preceding noun.

4) Composite words:
If the word contains a dash ("-"), it is composite. The individual words are transformed and a group node is set up with the same syntactic category(-ies) as the last word. No further analysis of the inner structure of the group is attempted at the syntactic level.

5) Regular inflections:
These are subdivided into three categories:
a) s-type inflections: plural of nouns and 3rd person of the present tense of verbs, in their various forms ("-s", "-es", "-y" --> "-ies"). Plurals require a count noun; also, modals in the present tense do not take this inflection.
b) ing-type inflections: progressive form of verbs ("ing" ending, last consonant possibly doubled, or final "e" possibly dropped).
c) ed-type inflections: past tense or past participle of verbs ("ed" ending, last consonant possibly doubled or final "e" possibly merged into "ed"). The word is set up as a homograph: it can be a simple verb or a 'verbal adjective,' i.e., a past participle used in post-modification of nouns (see Appendix 4).

APPENDIX 4
BASE-COMPONENT GRAMMAR

The "[]" indicate optional elements, and "!" separates alternatives. Symbols written in upper case are the non-terminals of the grammar. Other symbols represent pseudo-terminal categories assigned by the initial phase and reduced to a unique choice by homograph resolution rules. A list of these categories, with explanation of the mnemonics used, follows:

Adj	: Adjective
Adjadv	: Adjectival adverb (e.g., "more", "very")
Adv	: Adverb
Cop	: Copula (verb "to be")
Cprep	: Common adverbial preposition
N	: Noun (common noun)
Nint	: Noun introducer (determiner, etc.)
Np-b	: Noun-phrase boundary (comma - may be implied)
Pn	: Proper name
Prep	: Preposition - only between NP's ("of")
Pron	: Pronoun
Q	: Quantifier
Rpron	: Relative pronoun
Sprep	: Subordinate preposition or conjunction
V	: Verb
Vadj	: Verbal adjective (past participle)
Xprep	: Special preposition "xof" (see Appendix 3)

Mnemonics used for non-terminal symbols are explained before the production specifying that symbol.

The reader must be warned, before examining the grammar, that the real significance of these productions can be somewhat misunderstood unless one keeps in mind the constant interaction between the base component, the associated "semantic" routines, and especially the transformational component.

```
        Sentence
S           ::= [ADVPS] (DS ! IS) [ADVPS]

        Adverbial phrases string
ADVPS       ::= ADVP [ADVPS]
ADVP        ::= Adv ! (Cprep ! Sprep) NP Np-b ! SSENT

        Subordinate sentence
SSENT       ::= Sprep (PRED ! DS) Np-b

        Declarative sentence
DS          ::= PMNP PRED

        Imperative sentence
IS          := PRED

        Possibly modified noun-phrase
PMNP        ::= NP [ADVPS]

        Predicate
PRED        ::= COPG [PMADJ ! PTAIL] ! VG [PTAIL]

        Verb groups
VG          ::= V
COPG        ::= Cop

        Possibly modified adjective
PMADJ       ::= [Adjadv] Adj

        Predicate tail (Verb complements transformation)
PTAIL       ::= NP

        Noun phrase (full form)
NP          ::= CNP [DEFADJ] ! SNP RCLAUSE
```

```
        (Semi-) Complex noun phrase
CNP         ::= PRONS ! SNP [Prep CNP]


        Deferred adjectival phrase
DEFADJ      ::= Vadj (ADVP ! Np-b)


        Relative clause
RCLAUSE     ::= Rpron (PRED ! DS) Np-b


        Special prepositional phrase (possessives)
SPP         ::= Xprep N


        Simple noun phrase
SNP         ::= [Nint] [ADJCS] NNES ! SNP SPP


        Pronouns
PRONS       ::= Pron ! Rpron
        Nouns non-empty string (proper names  or  common
        nouns string)
NNES        ::= PNS ! NS
PNS         ::= Pn [PNS]
NS          ::= N [NS]


        Adjective-category string
ADJCS       ::= ADJC [ADJCS]
ADJC        ::= PMADJ ! 0


        Special number full-enunciation rule
NP          ::= [SPONNP1] SPONNP
SPQNNP      ::= SPQNNP1 SPONNP2
SPQNNP1     ::= ADJCS N
SPQNNP2     ::= ADJCS NS
```

APPENDIX 5
TRANSFORMATIONS

This appendix describes the transformations used in the syntactic analysis. We will examine in turn interrogative transformations, non-standard (nominal) adverbial phrases, "that" and referent deletions, and finally coordination. Attempting a formal specification along the lines sketched in Chapter 2 would take considerable space. Instead an informal presentation offers advantages of clarity and conciseness. Also, as mentioned in Chapter 2, informality avoids the details of the particular grammar used. Instead, it calls upon some natural grammar which is shared by all native speakers of the language.

Notations: In the examples, a "#" will indicate the position of the scanner at the moment of detecting the 'failure' which leads to the transformation and the scanner position after the transformation has taken place. LE(NP) designates a word whose syntactic category may appear as the leftmost element of a noun-phrase. "!" separates alternatives.

A5.1 Interrogatives.

All interrogative transformations require the interrogative flag to be 'ON'. The sentence must end with a "?" (one could also use WH- type keyword detection, or perhaps better, a trial and error procedure whereby the transformation is accepted whenever it makes sense.)

```
IT1:  (ADVPS ! '') . X1  Cop . X2  NP . X3
        (Adj ! NP) . X4  Remainder . X5  -->
        X1   X3   X2   X4   X5
```

e.g.,
"Are you # happy now?" (1)

--> "You are # happy now."

"Is John # a good student?"

(2)

--➤ "John is # a good student."

IT2: (ADVPS ! '') . X1 NP . X2 Aux . X3 NP . X4
V . X5 Remainder . X6 --➤
X1 X4 VT(X3 X5) X2 LOVT(X3 X5) X6

(VT(x) denotes an analysis of the string x in an attempt to extract a verb, the leftover from the string yielding LOVT(x).)

e.g.,

"How many puppets has she # been selling in the last two quarters?" --➤

(3)

"She has been selling how many puppets # in the last two quarters."

"What could you # have eaten instead?"

(4)

--➤ "You could have eaten what # instead."

"How many candies did he # have left in his

pocket?" --➤ "He had how many candies # left (5)

in his pocket."

(Note: there are some complicated problems associated with IT2. Details appear in the main text.)

IT3: (ADVPS ! '') . X1 Aux . X3 NP . X4 V . X5
Remainder . X6 --➤
X1 X4 VT(X3 X5) X6

(Here LOVT(X3 X5) must be empty.)

This is a trivial variant of IT2 in the case where the first NP, X2, is missing. In fact, the main difference is in the implementation: this case will first appear as a pseudo-imperative with the wrong verb inflections (IT3 must take place once this is noticed):

e.g.,

"Did you # go to the movie?"

(6)

--➤ "You went # to the movie."

"Why is 2 # written there?"

(7)

--➤ "Why 2 is written # there?"

See also the note at the end of TD ("that" deletion transformation) below.

A5.2 Non-standard, nominal adverbial phrases.

Rephrasing is not meaningful here. Rather, the recognized adverbial phrases will be enclosed between "*" signs. It is understood in both AP1 and AP2 below that the NP's have a particular form that makes them eligible for transformation into an ADVP. Checking for this form is now restricted to a crude table look-up for a few special time or location references: TLNP(NP) = 'True'.

```
AP1:  (ADVPS ! '') . X1  NP . X2  LE(NP) . X3
        Remainder . X4  -->
        ADVPS(X1 *X2*)  X3  X4
```

e.g.,

"*Last week* # you were very happy." (8)

but, of course, no transformation occurs in:

"Last week was a beautiful week." (9)

AP2: This transformation assumes that the parser first collects all NP's appearing in the input sentence following the main verb, then only checking that the actual number does not exceed the allowed number of complements: n. If there is an excess of m NP's (m = 1 or 2), and these are eligible, i.e., TLNP(NP) = 'True', then AP2 applies.

```
AP2:  (ADVPS ! '') . X1  NP . X2  V . X3  n*NP's . X4
        m*NP's . X5  Remainder . X6  -->
        X1  X2  X3  X4  *X5*  X6
```

e.g.,

"I went *home* *last week* # ." (10)

"Mr. Clark sold 207 pencils *last week* # and
135 pencils *this week*." (11)

(In the second example, note that the reconstitution of the second NP into an *ADVP* is a by-product of the coordination transformation, by comparison and matching of the coordinated predicates.)

A5.3 "That" deletion.

TD: (ADVPS ! '') . X1 NP . X2 V . X3 n*NP's . X4
NP . X5 V . X6 Remainder . X7 -->
X1 X2 X3 X4 [that] X5 X6 X7

Note: the restored "that" is assumed to introduce a sentential NP which becomes another complement to the verb X3. Thus the number of allowable complements must be at least (m+1). Also X5 and X6 must fulfill subject-verb feature agreement, even though X5 is first mistaken for a complement.
e.g.,
"Write 4 in the tens' place of the answer to show there # are 4 tens." --> "Write 4 in the tens' place of the answer to show [that] there are 4 tens." (12)

This transformation could a priori occur simultaneously with AP2 above. No such case was encountered and the combination does not seem very grammatical. On the other hand, there is a more serious recognition problem between TD and the earlier IT2-IT3: in both cases, the failure occurs because of the unexpected presence of a 'second verb'. The rule used is to apply TD unless the interrogative flag is 'ON' and some inversion may be expected because the 'first verb' was in fact a simple auxiliary. (see IT2-IT3 for examples.)

A5.4 Referent deletion.

RD: ... . X1 Adj . X2 (V ! Cop ! P-mark) . X3
Remainder . X4 -->
X1 X2 [X-Dummy] X3 X4

where P-mark is any punctuation mark, and X-Dummy represents a dummy noun which is introduced with no inherent features. Thus, RD may occur when, while expecting a full NP, an adjective occurs without any following noun; further conditions on X1, or others, are not examined.

Semantic routines will, in a later phase, attempt to recover some features of the artificial NP by studying the adjectives and quantifiers which modify it. These features will in turn provide the basis for referent recovery.

e.g.,
"Ed had 3 candies. Sue gave him 5 more # ." -->
"... Sue gave him 5 more # [X-Dummy]."

(13)

(Semantic study later restores: "5 more [candies].")

A5.5 Coordination.

When the sentence includes coordinated structures, the transformation required may be very complex, especially when word deletions have occurred. In short, when the coordination flag is set, the executive will direct the parser to return a sequence of subtrees: the input sentence is piece-wise analyzed by structuring it as much as possible with respect to the base component grammar. The sequence of subtrees serves then as input to the coordination module: deleted words are restored (expectedly, this is the most difficult part), and the following transformation can then be applied:

$$SC\ [,\ SC,\ \ldots\ ,]\ Conn\ SC \longrightarrow \begin{array}{c} SC\ \ Conn\ \ SC\ \ \ldots\ \ Conn\ \ SC \\ \text{'} \qquad\qquad\qquad\qquad\qquad\quad \text{'} \\ \text{----------------------------} \\ \text{'} \\ SC \end{array} \qquad (14)$$

where SC is any syntactic category, Conn is any connective, and the brackets indicate optional elements.

In the following example, the "[]" indicate the words restored by the analysis, and the "()" surround the complete predicates:

"Bill delivered 22 newspapers on one street, 56 on another, and 43 on a third street."
--> "Bill ((delivered 22 newspapers on one street) [and] ([delivered] 19 [newspapers] on another [street]) and ([delivered] 35 [newspapers] on a third street))."

(15)

For more details on the coordination transformation in general, and the analysis of this

particular example (15), see sections 2.4 and 2.5 in the main text.

B I B L I O G R A P H Y

Throughout the bibliography, the following abbreviations will be used:

CACM : Communications of the Association for Computing Machinery.

FJCC : Fall Joint Computer Conference.

IJCAI : First International Joint Conference on Artificial Intelligence.

[Abelson 1969]
Abelson, R. P., and Reich, C. M.
"Implicational Molecules: A Method for Extracting Meaning from Input Sentences."
Proc. IJCAI, Washington, D.C., 1969, pp. 641-647.

[Baranofsky 1970]
Baranofsky, S.
"Some Heuristics for Automatic Detection and Resolution of Anaphora in Discourse."
Unpublished M.A. Thesis, Univ. of Texas at Austin, Jan. 1970.

[Becker 1969]
Becker, J. D.
"The Modeling of Simple Analogic and Inductive Processes in a Semantic Memory System."
Proc. IJCAI, Washington, D.C., 1969, pp. 655-668.

[Biehler 1971]
Biehler, R. F.
"Psychology Applied to Teaching."
Houghton Mifflin Co., Boston, Mass., 1971.

[Bobrow 1964]
Bobrow, D. G.
"A Question-Answering System for High School Algebra Word Problems."
Proc. AFIPS 1964 FJCC, vol. 26, Pt. 1, Spartan Books, New York, pp. 591-614; also in [Minsky 1969], pp. 135-215.

[Bruner 1966]
Bruner, J. S.
"Toward a Theory of Instruction."
Belknap Press of Harvard University Press, Cambridge, Mass., 1966.

[Chafe 1970]
Chafe, W.
"Meaning and the Structure of Language."
Univ. of Chicago Press, 1970.

[Chomsky 1957]
Chomsky, N.
"Syntactic Structures."
Mouton and Co., The Hague, 1957.

[Chomsky 1959]
----.
"On Certain Formal Properties of Grammars."
from Information & Control, 1959, vol. 1, pp. 91-112; also in [Luce & al. 1965]

[Chomsky 1965]
----.
"Aspects of the Theory of Syntax."
MIT Press, Cambridge, Mass., 1965.

[Colby & al. 1968]
Colby, K. M., and Enea, H.
"Inductive Inference by Intelligent Machines."
Scientia, vol. 103, no. 19-20, (Jan.-Feb. 1968).

[Colby & al. 1969a]
----, and Smith, D. C.
"Dialogues between humans and an artificial belief system."
Proc. IJCAI, Washington, D.C., 1969, pp. 319-324.

[Colby & al. 1969b]
----, Tesler, L., and Enea, H.
"Experiments with a Search Algorithm for the Data Base of a Human Belief Structure."
Proc. IJCAI, Washington, D.C., 1969, pp. 649-654.

[Coles 1967]
Coles, L. S.
"Syntax Directed Interpretation of Natural Language."
Ph.D. Thesis, Carnegie-Mellon Univ., Pittsburgh, Pa., June 1967; also in "Representation and Meaning," H. A. Simon and L. Siklossy (Eds.), Prentice Hall, Inc., 1972, Chapter 5, pp. 211-287.

[Coles 1968]
----.
"An On-Line Question-Answering System with Natural Language and Pictorial Input."
Proc. 23rd ACM Nat. Conf., Brandon Systems Press, Princeton, N.J., 1968, pp. 157-167.

[Coles 1969]
----.
"Talking with a Robot in English."
Proc. IJCAI, Washington, D.C., 1969, pp. 587-596.

[Coles 1972a]
----.
"The Application of Theorem Proving to Information Retrieval."
Proc. Fifth Hawai International Conf. on Systems Sciences, Honolulu, Hawai, 1972, pp.410-412.

[Coles 1972b]
----.
"Techniques for Information Retrieval Using a Deductive Question-Answering System with Natural Language Input."
Final Report on Project 8696 to the National Science Foundation, Stanford Research Institute, Menlo Park, California, September 1972.

[Earley 1969]
Earley, J. C.
"A Practical Incremental LR(1) Parsing Algorithm."
Dept. of Computer Sc., Univ. of Calif. at Berkeley, Calif., 1969.

[Earley 1970]
----.
"An Efficient Context-Free Parsing Algorithm."
CACM, vol. 13, no. 2, Feb. 1970, pp. 94-102.

[Evans 1963]
Evans, T. G.
"A Program for the Solution of a Class of Geometric Analogy Intelligence Test Questions."
in [Minsky 1969], pp. 271-353.

[Feigenbaum & al. 1963]
Feigenbaum, E. A., and Feldman, J., Eds.
"Computers and Thought."
McGraw Hill, Inc., 1963.

[Fikes & al. 1971]
Fikes, R. E., and Nilsson, N. J.
"Strips: A New Approach to the Application of Theorem Proving to Problem Solving."
Artificial Intelligence Journal, vol. 2, pp.189-208, 1971.

[Fikes & al. 1972]
Fikes, R. E., Hart, P. E., and Nilsson, N. J.
"Learning and Executing Generalized Robot Plans."
Accepted for publication in the Artificial Intelligence Journal, 1972.

[Gagne 1959]
Gagne, R. M.
"Problem Solving and Thinking."
Annual Review of Psychology, vol. 10, 1959.

[Ginsburg & al. 1969]
Ginsburg, H., and Opper, S.
"Piaget's Theory of Intellectual Development: An Introduction."
Prentice Hall, Englewood Cliffs, N. J., 1969.

[Green & al. 1961]
Green, P. F., Wolf, A. K., Chomsky, C., and Laugherty, R.
"BASEBALL: An Automatic Question Answerer."
in [Feigenbaum & al. 1963], pp. 207-216.

[Green & al. 1968]
----, and Raphael, b.
"The Use of Theorem-Proving Techniques in Question Answering Systems."
Proc. ACM National Conf., 1968, pp. 169-181.

[Green 1969]
----.
"Application of Theorem Proving to Problem Solving."

Proc. IJCAI, Washington, D.C., 1969, pp. 219-240.

[Gregory 1967]
Gregory, R. L.
"Will Seeing Machines Have Illusions?"
in "Machine Intelligence 1," N. L. Collins and D. Michie (Eds.), pp. 169-177, Oliver & Boyd, Edinburgh, 1967.

[Hartung & al. 1967]
Hartung, M. L., VanEngen, H., Gibb, E. G., Stochl, J. E., Knowles, L., and Walch, R.
"Seeing Through Arithmetic - 4"
Published by Scott, Foresman & Co., Glenview, Illinois, 1967.

[Ittelson & al. 1951]
Ittelson, W. H., and Kilpatrick, F. P.
"Experiments in Perception."
Original paper written in August 1951; in "Psychobiology - The Biological Bases of Behavior," J. L. McGaugh, N. M. Weinberger, and R. E. Whalen (Eds.), W. H. Freeman & Co., San Francisco, Calif., 1967.

[Kellog 1968]
Kellog, C.
"A Natural Language Compiler for On-Line Data Management."
Proc. FJCC, 1968, pp. 473-492.

[Kersh 1963]
Kersh, B. Y.
"The Motivating Effect of Learning by Directed Discovery."
in "Human Learning in the School," J. P. DeCecco (Ed.), pp. 277-287, Holt, Rinehart & Winston, New-York, N. Y., 1963.

[Lindsay 1964]
Lindsay, R.
"Inferential Memory as the Basis of Machines which Understand Natural Language."

in [Feigenbaum & al. 1963], pp. 217-236.

[Luce & al. 1965]
Luce, R. D., Bush, R. R., and Galanter, E., Eds.
"Readings in Mathematical Psychology."
vol. II, J. Wiley and Sons Publishers, 1965.

[Manis 1968]
Manis, M.
"Cognitive Processes."
Brook/Cole Publishing Co., Belmont, Calif., 1968.

[McCalla & al. 1972]
McCalla, G. I., and Sampson, J. R.
"MUSE: A Model to Understand Simple English."
CACM, vol. 15, no. 1, (Jan. 1972), pp. 29-40.

[McCarthy 1959]
McCarthy, J.
"The Advice Taker."
from "Programs with Common Sense", Proc. of Symp. on Mechanization of Thought Processes, I, London, England, 1959; also in [Minsky 1969], pp. 403-410.

[McCarthy 1963]
----.
"Situations, Actions, and Causal Laws."
Stanford A. I. Project, Memo 2, Stanford Univ., Stanford, Calif., 1963; also in [Minsky 1969], pp. 410-418.

[Minsky 1965]
Minsky, M. L.
"Matter, Mind, and Models."
Proc. IFIP Congress, vol."1, pp. 45-49, 1965. Also in [Minsky 1969], pp. 425-432.

[Minsky 1969]

----, Ed.
"Semantic Information Processing."
MIT Press, Cambridge, Mass., 1969.

[Minsky & al. 1969]
----, and Papert, S.
"Perceptrons."
MIT Press, Cambridge, Mass. 1969.

[Olney & al. 1966]
Olney, J. C., and Londe, D. L.
"A Research Plan for Investigating English Discourse Structure with Particular Attention to Anaphoric Relationships."
TM(L)-3256, System Development Corp., Santa Monica, Calif., Nov. 1966.

[Petrick 1965]
Petrick, S.
"A Recognition Procedure for Transformational Grammars."
Ph.D. Thesis, MIT, Cambridge, Mass. 1965.

[Petrick & al. 1969]
----, Postal, P. M., and Rosenbaum, P. S.
"On Coordination Reduction and Sentence Analysis."
CACM, vol. 12, no. 4 (April 1969), pp. 223-233.

[Piaget 1953]
Piaget, J.
"How Children Form Mathematical Concepts."
Scientific American, vol. 189, pp. 74-79, 1953.

[Quillian 1966]
Quillian, M. R.
"Semantic Memory."
Ph.D. Thesis, Carnegie-Mellon Univ., Pittsburgh, Pa., Feb. 1966; also in [Minsky 1969], pp. 216-270.

[Quillian 1969]
----.
"The Teachable Language Comprehender."
CACM, vol. 12, no. 8, (Aug. 1969), pp. 459-476.

[Raphael 1964]
Raphael, B.
"SIR: A Computer Program for Semantic Information Retrieval."
Ph.D. Thesis, Math. Dept., MIT, Cambridge, Mass., June 1964. In Proc. AFIPS 1964 FJCC, vol. 26, Pt. 1, Spartan Books, New York, pp. 577-589; also in [Minsky 1969], pp. 33-134.

[Reibel & al. 1969]
Reibel, D. A., and Schane, S. A., Eds.
"Modern Studies in English."
Prentice-Hall, Inc., Englewood Cliffs, N. J., 1969.

[Robinson 1965]
Robinson, J. A.
"A Machine-Oriented Logic Based on the Resolution Principle."
J. of the ACM, vol. 12, no. 4, (Oct. 1965), pp. 536-541.

[Sammet 1971]
Sammet, J. E. (Chairman)
"Symposium on Languages for Systems Implementation."
Proceedings constitute vol. 6, no. 9, October 1971 issue of SIGPLAN notices (available from ACM headquarters).

[Scandura 1969]
Scandura, J. M.
"Expository Teaching of Hierarchical Subject Matters."
Psychology in the Schools, vol. 6, no. 3, pp. 307-310, 1969.

[Schank & al. 1970]
R. C. Schank, L. Tesler, and S. Weber
"Spinoza II: Conceptual Case-Based Natural Language Analysis."
Stanford A. I. Project, Memo AIM-109, Stanford University, Jan. 1970.

[Schank 1971]
R. C. Schank
"Intention, Memory, and Computer Understanding."
Stanford A. I. Project, Memo AIM-140, Stanford University, Jan. 1971.

[Schank 1972]
R. C. Schank
Private communication.

[Simmons 1965]
Simmons, R. F.
"Answering English Questions by Computer: A Survey."
CACM, vol. 8, no. 1, (Jan. 1965), pp. 53-70.

[Simmons 1966]
----, Burger, J. F., and Long, R. E.
"An Approach Toward Answering English Questions from Text."
Proc. FJCC 1966, pp. 357-363.

[Simmons & al. 1968]
----, Burger, J. F., and Schwarcz, R.
"A Computational Model of Verbal Understanding."
Proc. AFIPS 1968 FJCC, vol. 33, Thompson Book Co., Washington, D.C., pp. 441-456.

[Simmons 1970]
----.
"Natural Language Question Answering Systems: 1969."
CACM, vol. 13, no. 1 (Jan 1970), pp. 15-30.

[Skinner 1968]
Skinner, B. F.
"Science and Human Behavior."
Macmillan, New York, N. Y., 1953.

[Thompson 1966]
Thompson, F. B.
"English for the Computer."
Proc. FJCC, 1966, pp. 349-356.

[Weizenbaum 1966]
Weizenbaum, J.
"ELIZA: A Computer Program for the Study of Natural Language Communication between Man and Machine."
CACM, vol. 9, no. 1, (Jan. 1966), pp. 36-45.

[Winograd 1971]
Winograd, T.
"Procedures as a Representation for Data in a Computer Program for Understanding Natural Language."
Ph.D. Thesis, MIT, Cambridge, Mass., Jan. 1971.

[Winston 1970]
Winston, P. H.
"Learning Structural Descriptions from Examples."
Ph.D. Thesis, MAC TR-76, Sept. 1970.

[Woods 1968]
Woods, W. A.
"Procedural Semantics for a Question-Answering Machine."
Proc. AFIPS 1968 FJCC, vol. 33, Pt. 1, MDI Publications, Wayne, Pa., pp. 457-471.

[Woods 1970]
----.
"Transition Network Grammars for Natural Language Analysis."

CACM, vol. 13, no. 10, Oct. 1970, pp. 591-606.

[Zadeh 1970]
Zadeh, L. A.
"Quantitative Fuzzy Semantics."
Memo. no. ERL-M281, Electronics Research Lab., Univ. of Calif. at Berkeley, Calif., Aug. 1970.