American Journal of Computational Linguistics *Microfiche 23*

AUTONOTE2: NETWORK MEDIATED NATURAL LANGUAGE COMMUNICATION IN A PERSONAL INFORMATION RETRIEVAL SYSTEM

William E. Linn, Jr.[2]
and
Walter Reitman

University of Michigan
Ann Arbor

[1]*This paper is based on a doctoral dissertation by the first author. Support from the National Science Foundation under Grant No. DCR71-02038 is gratefully acknowledged. Those wishing more complete details about system commands and implementation should write the second author for a User's Manual.*

[2]*Now at Southern Railway System, 125 Spring Street, S.W., Atlanta, Georgia 30303*

ABSTRACT

Natural language combines nouns and adjectives into noun phrases, and links phrases by means of prepositions to form complex descriptions of objects and topics. AUTONOTE2, a file-oriented retrieval system, allows the user to employ such descriptions to characterize the items of information he wishes to store and retrieve. In addition, the system also constructs a network representation of the user's subject matter, using syntactic analysis to derive dependency structures from his descriptions. The dependency information, expressed as subordinate and coordinate linkages among the phrases, is represented by a tree of nodes, with simple phrases at the terminal branches. The PARSER uses the network to disambiguate descriptions, querying the user only about residual ambiguities.

Associated with the PARSER is a network LOCATOR, which determines whether a current user description refers to an existing topic at some level in the network. The LOCATOR also builds a table specifying the changes, if any, to be made in a network in order to represent the topic inferred from the current input description. For example, if the user's description contains one or more simple phrases (thereafter referred to as active) directly describing at least one existing node in the network, the description as a whole quite likely references an existing network topic. To locate it, the PARSER first determines the focus phrase, the active phrase at the highest dependency level. The nodes directly described by the focus phrase are used to generate candidate topics. These then are matched against the

remaining active phrases obtained from the description to determine the most likely referent.

Many of the procedures employed in description and representation also are used in network-mediated retrieval. The user may initiate retrieval with a FIND command, supplying a description as argument. The resultant phrase table is passed along to the network LOCATOR, which returns a node number to the FIND processor. The FIND processor constructs a set of item numbers by extracting the textual references from the node. The system then checks for upward pointers from the node. If there are structurally related topics, the FIND processor so informs the user. Note that by virtue of network mediation of retrieval, if a user description is imprecise or incorrect, the system may be able to direct the user to relevant related topics.

When the system queries the user about a topic, for example to determine the intent of a description, the topic node number is passed to a SPEAKER component. A phrasal description of the node is returned. To minimize redundant communication, a level indicator may be set according to the level of detail in the user's description. For example, if the user describes an item as RESULTS OF THE EXPERIMENT and the system must ask if he is referring to SMITH'S EXPERIMENT ON THE SHORT TERM MEMORY OF WHITE RATS, the resulting query would be: ARE YOU REEERRING TO SMITH'S EXPERIMENT ON MEMORY?

Construction of a description from the network takes place in two stages. The first stage steps thorugh the network recursively, collecting the simple phrases that directly or indirectly describe the specified node.

The level indicator blocks collection of simple phrases below the specified level. The second stage is carried out by a recursive algorithm that operates on the tabled simple phrases and their interrelations to construct the phrasal description.

The last major component of the system handles network modification and reorganization. This enables the user to add or remove references and phrases, and to modify, delete, or reorganize his topic structure.

A detailed case study comparing AUTONOTE2 with a good keyword-based retrieval system showed that for a coherent body of material, the communicative efficiency of AUTONOTE2, as measured by the ratio of the number of words conveyed to the number of words entered, was more than double that of the keyword-based system. Retrieval capability was enhanced considerably, and the representation network effectively distinguished among the many topics partially indexed by the same words. Furthermore, SPEAKER output of topics from the representational network proved a useful retrieval intermediary, greatly reducing the need for perusal of item texts.

TABLE OF CONTENTS

Page

I. INTRODUCTION

When two humans communicate, each party builds up a conceptual representation of the topics of discussion. Such representations are fundamental to human communicative efficiency. The listener's representation of the topics already discussed facilitates communication in that the speaker is spared the trouble of describing in complete detail those things to which he refers. Furthermore, the speaker can proceed to related topics without having to describe them in full. For example, a speaker who has been talking about the design of a particular experiment can safely move on to discuss the results of the experiment without specifying anew the experiment he has in mind.

We use the term referential communication to indicate the process by which a speaker communicates a reference to some subject or topic to a listener. If within the environment of a personal information system, we view the information universe as a collection of textual materials each pertinent to one or more "topics," then one can readily construct an analogy. The user and system take on the roles of speaker and listener, respectively. The domain of discourse is a set of topic descriptions characterizing the user's textual materials. The user enters his materials and describes to the system the topic or topics to which they pertain. During this process, the system constructs its own representation for the subjects the user has described, and associates each piece of text with its corresponding topic representations.

This paper describes the design and implementation of a personal information storage and retrieval system based on the foregoing analogy with

human referential communication. It presents a hierarchical network data structure for representing topic descriptions formulated within a phrasal description language. Called the representational network, this structure enables the system to move easily from one subject to other related ones. It provides a means for representing the user's working context, thereby enabling the user to describe his materials much more tersely than is possible in keyword-based systems. The system makes use of the syntactic dependencies among the words and phrases of descriptions in order to represent structural relationships among the user's topics. Consequently, the user imparts structure to the data base in a particularly natural way, eliminating much of the organization activity normally associated with keyword-based systems. Our central thesis is that the network mediated techniques provide for more effective man-machine communication during the processes of description, organization, and retrieval within a personally generated information universe

The procedures used here differ substantially from the typical keyword indexing and retrieval mechanisms of other personal retrieval systems. The central objective is to provide the user with a framework for defining the important topics or informational objects he deals with, and to enable him to easily associate items in his data base with these entities. Rather than viewing the data base as a collection of items and associated index terms, the user deals with "objects" that are in some sense meaningful to him. Whether retrieving information or indexing new material the user conveys references to the appropriate topics. This shift in the user's view of his information universe, coupled with the mechanisms we have developed for

building up and referring to the topic framework, constitute the substance of our approach to personal information storage and retrieval.

II. THE AUTONOTE SYSTEM

The system described here uses the AUTONOTE information storage and retrieval system (Reitman et al., 1969) as a base. AUTONOTE is an on-line retrieval system that runs as a user program under the Michigan Terminal System (MTS), a time-sharing system implemented on the IBM 370/168. The basic units of information stored in AUTONOTE are called items. The user may enter arbitrary textual materials into an item and may assign descriptors by which these materials can be retrieved. Retrieval requests take the form of single descriptors or combinations of descriptors connected by AND, OR, or NOT logical operators. Facilities are provided for deleting, replacing, linking, and hierarchically organizing text items.

AUTONOTE makes extensive use of the MTS disk file system. MTS disk files (line files) may be read or written either sequentially or in an indexed fashion by specifying a line file number. AUTONOTE maintains two line files for each user's data base; one for storing textual materials and bookkeeping information, the other for storing a descriptor index. Each text item occupies a specific region of the line number range of the text file. The descriptor index, on the other hand, is accessed through an efficient hash coding algorithm that maps each descriptor into an index file line number. The descriptor index is organized as an inverted file, that is, each line in

the index contains pointers to each of the text items assigned the descriptor for that line.

Basic AUTONOTE Commands

Text entry. To enter a new text item, the user first types the command ENTER and the system responds with a numerical tag for the new item. The system then enters a "text insertion mode" and indicates its readiness to accept successive text lines with a question mark. After entering text, the user may return to "command mode" by entering a null line or an end-of-file indication. Should the user at any time wish to continue inserting text into the current item, he may re-enter text insertion mode via the INSERT command. Subsequent lines are placed below the most recent line for the current item in the text file.

In command mode, the system prompts the user for input with a minus sign. The user may give each command in full or he may abbreviate by giving any initial substring of the command name.

Descriptor entry. To associate one or more descriptors with the current text item, the user enters a list of words, beginning the input line with an at sign (@). Any character string up to 16 characters in length may be used as a descriptor. In addition to updating the descriptor index, the system also places the actual "@-line" in the text file in a subregion beneath the text of the current item.

Retrieval. To display a partciular text item the user may enter the command PRINT followed by the appropriate item number. Sequential blocks of items can also be specified in the PRINT command, e.g., PRINT 77...85.

In most cases, however, the specific item number(s) will not be known. The LIST command accepts a descriptor or logical combination of descriptors as its argument and responds with a list of the item numbers that satisfy the query. The functions of the PRINT and LIST commands are combined in the RETRIEVE command. It also takes a descriptor specification as argument and causes each item in the resulting list to be PRINTed.

Definitional facility. AUTONOTE also provides a definitional facility that allows the user to create sets of items referenced by arbitrary combinations of descriptors. For example, the command CREATE $IRS= INFORMATION AND RETRIEVAL AND SYSTEMS adds a new descriptor, $IRS, to the index that references each item having the words INFORMATION, RETRIEVAL, and SYSTEMS as descriptors. Any defined term may be used just as any other descriptor in retrieval requests; they may also be used to define other new terms (e.g., CREATE $OTHERSYSTEMS= $IRS NOT AUTONOTE).

The definitional facility is also invoked implicitly each time the user issues a retrieval query. The set of items referenced by the most recent LIST or RETRIEVE command, called the active set, is assigned the name $. Should the user wish to refine the results of the previous query, he has access to the active set. To facilitate this process, each time a missing descriptor is noted in a retrieval request the descriptor $ is inserted automatically by the system. For example, the command LIST NOT FORTRAN is interpreted as LIST $ NOT FROTRAN, i.e., the old active set of items is restricted to include only those not referenced by the descriptor FORTRAN. This operation, of course, redefines the active set.

Item-item linkages. The ability to define associative links between any two text items is provided by the APPEND command. When an item is displayed, its associative links to other items may optionally be printed along with a user-specified comment indicating the nature of the association.

Tutorial feature. Throughout the course of its development, AUTONOTE has been employed to collect, organize, and maintain up-to-date documentation of its capabilities, usage strategies, and so on. This information is stored in a publically available data base. It includes brief descriptions of each of the commands, announcements of recent developments and system changes, and other instructive information. The AUTONOTE user may call upon this store of material by entering a HELP command. The user s data base is temporarily set aside and the public data base is attached to the system. The user may then retrieve instructive information in the same way that he operates with his own data base. To assist novice users, the system will optionally print instructions for accessing the HELP data base.

Grouping. AUTONOTE provides a grouping facility which permits the user to organize text items in several useful ways. It enables the user to define a "grouping item" which references an arbitrarily ordered list of other items This is done by entering into an item an @-line of the form:

$$@GROUP \quad N_1 \quad N_2 \quad N_3 \ , \ \ldots$$

Since any item can represent a group, it is possible to form a complex hierarchical structure in this way.

A grouping item can be viewed as a node of an inverted tree structure with downward branches to those items listed in its "@GROUP" line. A request to display a grouping item initiates recursive processing of the tree

structure to identify the terminal and nonterminal items of the hierarchy. The user may request that only terminal or nonterminal items be displayed, or that the entire list of materials be printed.

The organization of the HELP data base described above provides an excellent example of the power and flexibility of the grouping facility. The HELP text file contains at this writing approximately 150 items of documentation. Using the grouping convention, these are organized into five subgroups: (1) general information; (2) input and editing facilities; (3) output (retrieval) facilities; (4) organizational facilities; and (5) utility commands. There is one major item which groups all of these subgroups into a single tree structure. The top node of the structure is indexed by the descriptor USERS-MANUAL. As new facilities are incorporated into the system, their descriptions are entered into the manual structure, thus assuring that complete and up-to-date documentation is always available. At any time, the single command: RETRIEVE USERS-MANUAL causes the entire updated data base to be displayed in organized form.

Command modifiers. AUTONOTE includes a set of modifiers or option settings that control the execution of many commands. These include options that affect the format of displayed items, the expansion of grouping structures, the nature and extent of system feedback, etc. Each of the modifiers has a default value that is chosen to simplify use of the system by a novice. The more experienced user may alter the modifiers via the SET command to tailor the system to his own needs, usage patterns and level of competence.

AUTONOTE also provides a large number of auxillary commands and facilities. A list of the major AUTONOTE commands, each accompanied by a brief description, is included in Linn (1972, Appendix A).

AUTONOTE System Organization

AUTONOTE has been designed as a modular system so that as new facilities become available they may be tested and later added with little or no reprogramming of the existing system. The majority of AUTONOTE commands are implemented as subroutines, each of which resides permanently in an MTS disk file. The basic system is organized around a central monitor that accepts user input and calls upon appropriate modules to service the user's requests. In addition to the monitor, the core resident system includes a dynamic loader, a disk file interface and a set of frequently used utility routines. A number of primitive commands, text entry, and descriptor assignment are also handled by the resident system. As the user requests more complex services (LIST, RETRIEVE, or PRINT, for example), the monitor calls upon the dynamic loader to bring the appropriate modules into core storage. These routines then become a part of the resident system, remaining in core storage until the user explicitly requests their removal. An organizational diagram of the AUTONOTE system appears in Fig. 1.

The modular design of AUTONOTE coupled with the dynamic loading facility offers two important benefits. From the user s viewpoint, he has access to the complete repertory of AUTONOTE services, yet he pays core storage charges only for those routines he actually uses during a given session. To the developers of the system, the modular framework facilitates the

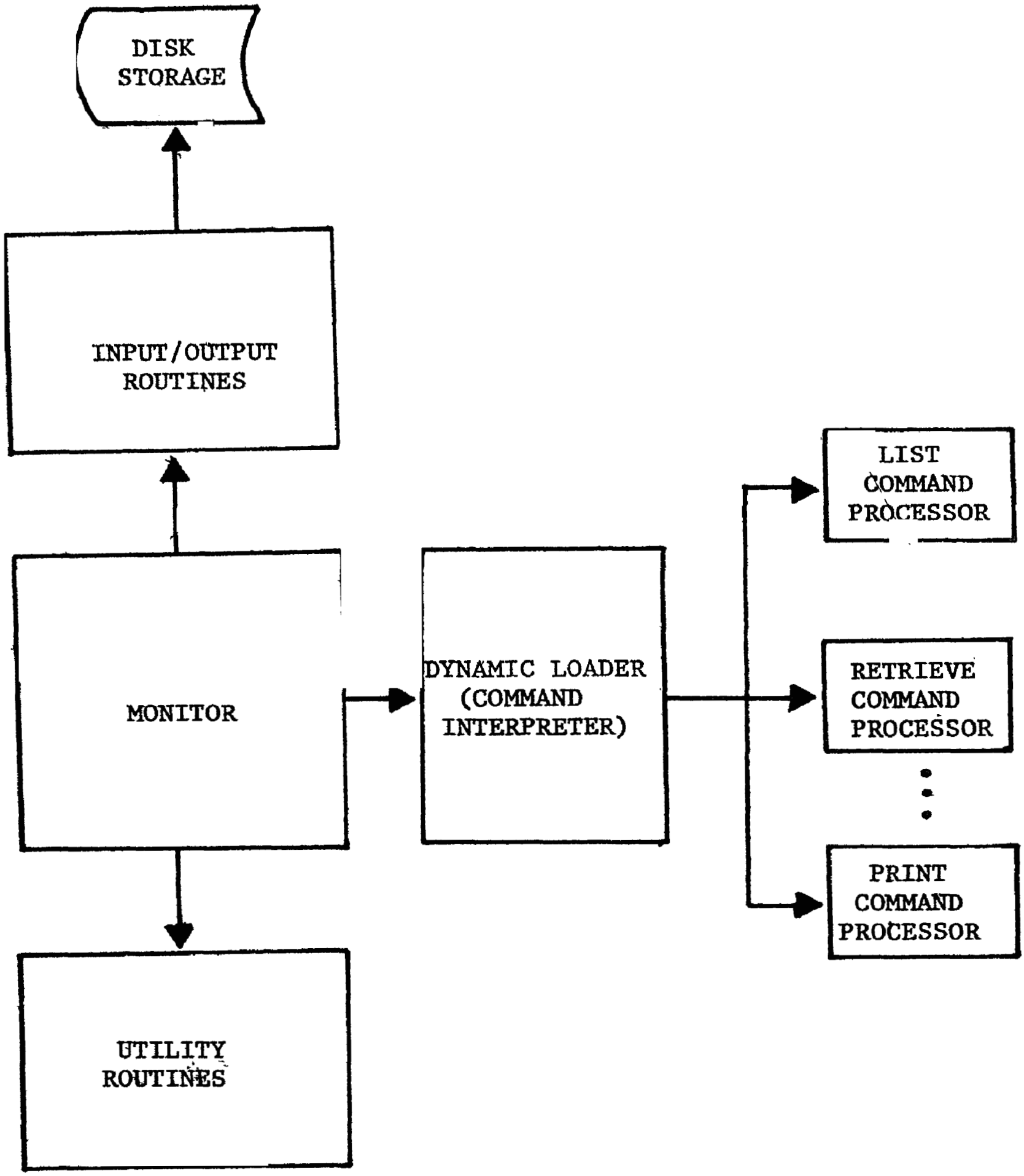


Fig. 1 - AUTONOTE System Organization

addition of new system components. The latter has been an important factor in the implementation of the AUTONOTE2 system.

III. OVERVIEW OF AUTONOTE2

The AUTONOTE2 system uses ideas (Reitman, 1965; Reitman et al., 1969) concerning the use of our "knowledge of the world" to disambiguate and fill in implied facts when conversing with one another. In particular, the system design is based upon the assumption that efficient human communication... "depends upon the listener's ability to make inferences from prior information, from context, and from a knowledge of the speaker and the world. Communicating in this way, we risk occasional misunderstanding as the price for avoiding verbose, redundant messages largely consisting of material the listener already knows" [Reitman et al., 1969].

In our more restricted domain of discourse, we view the process of human referential communication as one guided by some form of internal representation of the various topics or referents discussed earlier. When a listener can be assumed to have such a representation, the speaker is spared the difficulty of describing in complete detail the things to which he refers. He need only give enough information to allow the referent to be discerned in full. Our goal then is to develop a representational scheme for our retrieval system that allows the user analogous communicative efficiencies.

The Description Language

The first step in devising a representational framework was the formulation of a language for expressing topic descriptions to the system. Although an underlying factor in the design of AUTONOTE2 was to make communication with the system more "natural," it should be noted that the emphasis of this research is not upon parsing or "understanding" natural language. Rather, our goal is to investigate the notions of topic representation and referential communication as a means for improving the user's ability to describe, organize, and retrieve his materials. Consequently, a minimal subset of noun phrases was chosen--minimal in the sense that it excludes most of the complexity of natural English, yet still retains a degree of descriptive richness sufficient to explore the underlying ideas of this study.

Natural language enables us to combine nouns and adjectives into noun phrases and to interlink noun phrases via prepositions to form complex descriptions of objects in the real world. The AUTONOTE2 description language provides such a framework for composing topic references. A formal grammar for the language is given in Fig. 2 along with a few sample descriptions that illustrate the flexibility of expression achievable with the language. These grammatical rules are not in fact used explicitly by the system in actually parsing topic descriptions. The grammar is presented here only to specify precisely the set of descriptions acceptable to the system. The actual AUTONOTE2 parser is heuristic-based, making use of previously analyzed phrases, noun-preposition co-occurrences, and a set of heuristics to guide

<description>	::=	<noun-group> | <noun-group> <preposition> <description>
<noun-group>	::=	(<article>) (<modifier-group>) <noun>[a]
<modifier-group>	::=	<modifier>[a] | <modifier> <modifier-group>
<preposition>	::=	about | to | from | in | on etc.
<article>	::=	a | an | the

(a) Grammar for the description language.

The paper

The paper about microprogramming in the proceedings of the fall joint computer conference

Notes on the organization of AUTONOTE2 for use in the presentation of the ACM paper

The use of recall precision measures in the evaluation of the SMART information retrieval system

Quotes from Feldman's 1969 paper for use in the introduction of the second chapter[b]

(b) Sample descriptions.

Fig. 2 - The AUTONOTE2 Description Language

[a]Modifiers and nouns are arbitrary character strings not recognized as articles or prepositions. When a number of consecutive "words" are encountered, the last is parsed as a noun and the preceding words as modifiers.

[b]Possessive adjectives are treated as a special case of adjectival modification.

the parsing process. In some instances, the user may even be asked for parsing assistance.

Representational Framework

Central to the design of AUTONOTE2 is the idea of viewing the user's information universe as a collection of "informational objects" or topics, each having associated with it a number of text items. When the user wishes to describe a text item, we assume he has such a topic in mind. Using the phrasal language specified above, he composes a description of that topic and presents it to the system. AUTONOTE2 then constructs an internal representation of that topic. When a text item is described, the system must consult the representation to determine if the description (1) references an existing topic, (2) is related to an existing topic, or (3) defines a new topic. In any case, the ultimate goal is to associate the text item with a topic representation, possibly augmenting the representation in the process.

Criteria for the Representation

Efficiency of communication. Efficient man-machine communication implies that the user should not in general have to formulate a complete description of a particular topic in order to convey a reference to it. The system should be capable of accepting and correctly interpreting incomplete references by filling in missing information. As an example, a topic fully described as THE PAPER BY SALTON ABOUT THE SMART SYSTEM might be referred to as THE PAPER, THE PAPER BY SALTON, THE PAPER ABOUT THE SMART SYSTEM and so on.

A description in the AUTONOTE2 language consists of a noun modified by adjectives and prepositional phrases. The words that modify any given term

may themselves be modified in exactly the same way. In effect, each adjective and prepositional phrase functions as a phrase component that imparts greater detail to the overall description. In the example above, BY SALTON and ABOUT THE SYSTEM provide information about the paper; SMART specifies which system is meant.

To facilitate efficient communication we require a representational framework that makes explicit the component phrases of each topic description. Given such a framework, we have a basis for comparing incomplete descriptions with the representation to determine possible topic referents.

Descriptive power. A system that makes use of syntax in the user's descriptor entries increases descriptive power in that it permits distinctions that, in general, will not be made in keyword-based retrieval systems. A description such as THE ORGANIZATION OF THE PAPER ABOUT MTS is semantically quite different from THE PAPER ABOUT THE ORGANIZATION OF MTS, despite the fact that both contain the same words. A system that takes into consideration the syntactic relationships that hold among the words ORGANIZATION, PAPER, and MTS can discriminate between the two.

The considerations outlined thus far lead quite naturally to some form of dependency representation for the user's topics. Essentially, a dependency representation for the AUTONOTE2 language would reflect the syntactic dependence of each adjective and prepositional phrase upon an appropriate noun. Such a framework provides the essential information for enhancing descriptive power and communicative efficiency as defined above.

Hierarchical representations. We view a topic as a group of interconnected subtopics, each bearing on a central theme yet with varying levels of generality. To make this notion more concrete, consider a user of AUTONOTE2 putting down his thoughts and ideas for a book he is writing. He begins by entering some general material which he describes simply as "THE BOOK ABOUT...." At some later time he may enter an outline for the book, a list of reference materials he will use, publishing arrangements, etc. Still later, he will enter materials for the chapters of his book and perhaps outlines for each chapter. In time he will have defined a host of related descriptions. Fig. 3 gives a pictorial representation of the resultant complex "topic." The representational scheme of AUTONOTE2 was designed with complex hierarchies such as this one in mind. In other words, we want to represent related topic descriptions via interconnections in a network.

The essential idea is that such a network corresponds to a map of the organization of the associated textual materials--a map that should reflect important structural relationships among the materials from the user's viewpoint. A hierarchical representation of this kind is especially effective during retrieval. If the user requests materials dealing with his book, for example, the system can also inform him that he has more specific items dealing with the publishing arrangements, the component chapters, and so on.

The notion of a representational network fits well with the dependency framework we require. The syntactic dependencies among the words and phrases of a description may be used to represent structural relationships among the user's topics. In the example above, the network connection between the

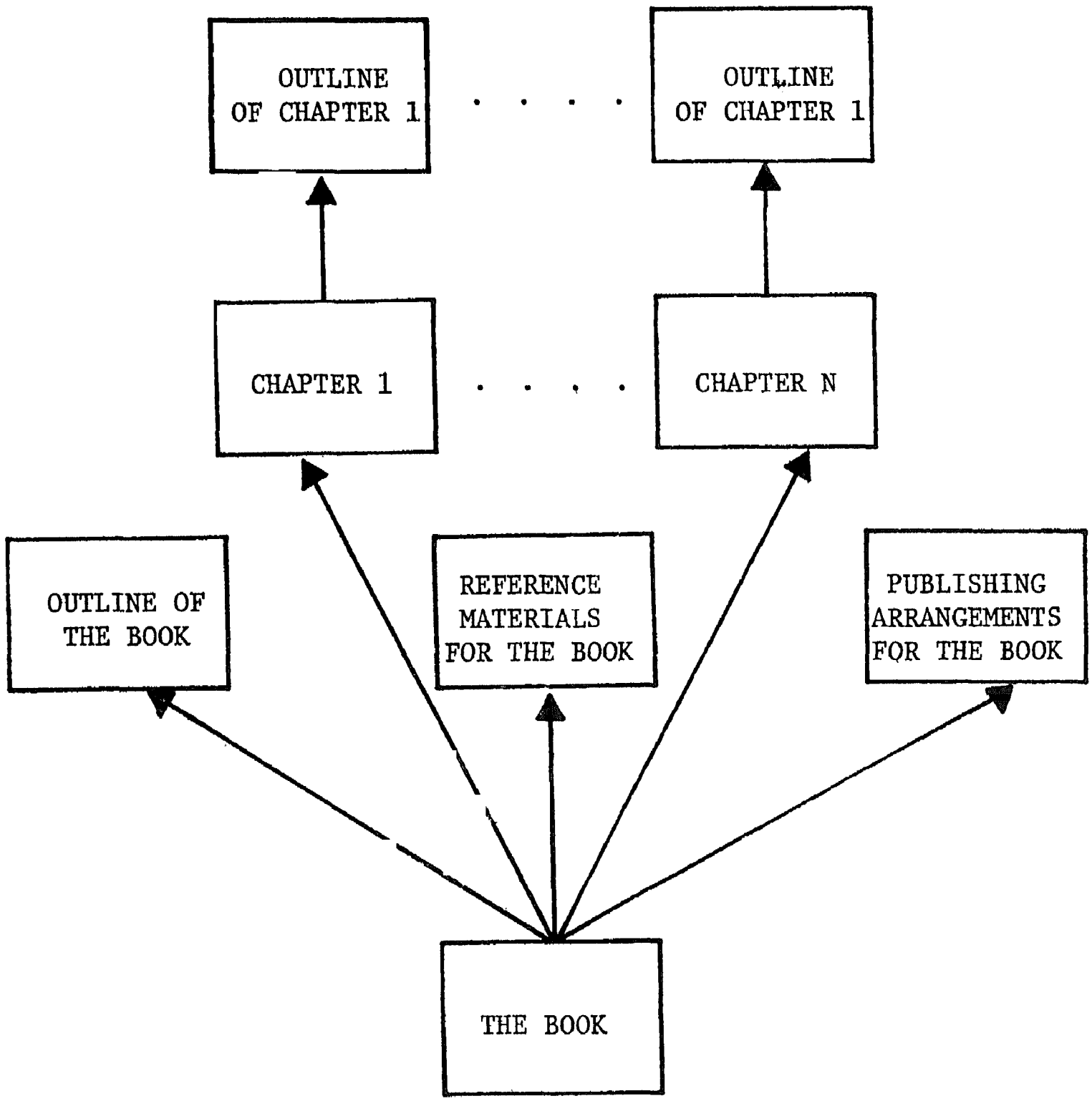


Fig. 3 - A Topic Hierarchy

"outline" and the "book" corresponds to the syntactic dependency of "book" upon "outline" in the description THE OUTLINE OF THE BOOK ABOUT....

Augmentation of the representation. In the previous discussion of communicative efficiency we were concerned with associating an incomplete description with its corresponding topic. In designing the representational framework we also had to consider the case in which a reference provides a more detailed description of an existing topic. In such instances we want to enrich the topic representation to include the additional information. Whether additional descriptive information is encountered in a subsequent item description or in a retrieval request, we want the system to incorporate it into its existing knowledge of the user's topics. This requires that the representation be structured in such a way that dynamic augmentation is easily accomplished.

The representation of context. In providing a framework for interpreting terse, incomplete references we naturally are confronted with the problem of ambiguity. A description such as THE PAPER, or THE PAPER ABOUT MICROPROGRAMMING may in fact satisfy a large number of distinctly described topics. To deal with this problem we require some kind of contextual framework that enables the system to infer, where possible, the intent of a vague or ambiguous reference. A user who has been entering material for a paper he is writing should be able to describe a subsequent item as, say, THE OUTLINE OF THE PAPER, and have the system infer which paper he means. In general then, we want the representational framework to include information that identifies the "working-context," i.e., those topics the user has referred to recently.

System interrogation of the user. Presented with an ambiguous description "out of context," the system is faced with much the same dilemma a human listener would face. In such instances, we want the system to be capable of asking pertinent questions to resolve the ambiguous reference. This implies, of course, that the representation preserve sufficient information to enable it to reconstruct descriptions of the user's topics.

Overview of the AUTONOTE2 Implementation

Data structures. We have now presented the major design requirements for the representational framework. These preliminary criteria suggest a representation organized as a network of (possibly interconnected) dependency structures obtained from syntactic analysis of topic descriptions. The network data structures are discussed in section IV in terms of the representational criteria and also the computational requirements--how they are to be accessed, modified, and so on.

The parser. The parsing of descriptions is guided by the state of the representation at the point they are entered. For this reason, the parsing algorithm is treated in section IV in conjunction with the representational data structures. The presentation includes detailed discussion of the parsing problems encountered and the heuristics employed in dealing with them.

Network location. The function of the network locator is to analyze the parse tree to decide whether the description references an existing topic or defines a new one. Once this decision is made, it constructs a list of any network modifications required to represent the topic and its associated item reference. The network location algorithm is described in section V.

Retrieval. The AUTONOTE2 retrieval component is invoked via a FIND command. The command takes a topic description as its argument. The FIND processor in turn calls upon both the parser and network locator, regaining control after the appropriate network topic has been identified. Text items directly associated with the topic then may be retrieved from the data base. Alternately, the retrieval component will move to structurally related topics in the representational network to collect additional item references for subsequent display.

To reconstruct topic descriptions from the network, AUTONOTE2 includes a SPEAKER module. If the user's description is ambiguous, for example, the network locator may call for a display of the alternative topics. The FIND processor employs the SPEAKER to present descriptions of topics structurally related to the user's original query. The user also may invoke the SPEAKER explicitly, via a DESCRIBE command, to obtain descriptions of some subset of the topics in the representational network. The retrieval component, the SPEAKER, and the DESCRIBE command are treated in section VI.

Network modification. The last major component of AUTONOTE2, the network modification processor, is described in section VII. It allows the user to delete topic representations, create new ones, and merge multiple topics into a single representation. It also enables the user to move through clusters of related topics in order to explore associations in the network.

Auxillary commands. Various auxillary commands and facilities are given in Linn (1972, Appendix B). This appendix also includes some discussion of usage strategies for achieving the most effective use of AUTONOTE2.

Fig. 4 depicts the organization of the AUTONOTE2 components within the AUTONOTE system framework.

IV. THE PARSER AND THE REPRESENTATIONAL NETWORK

Overview

When the user wises to describe a text item, we assume he has in mind some subject, topic, or informational object that can be characterized by a phrasal description. A description may convey a reference to a topic the user has dealt with earlier; or it may define a new one. The description is analyzed to determine a dependency tree--a structure that preserves the original words and phrases of the description and the syntactic dependencies among them.

In constructing this tree, the parser incorporates primary units called simple phrases. A simple phrase may consist of a modifier and a noun (e.g., ACM CONFERENCE), or of a noun followed by a preposition and modifier (e.g., OUTLINE OF PAPER). The parser extracts these basic phrases from the original description and records the syntactic dependencies among them. A description such as THE OUTLINE OF THE PAPER ABOUT AUTONOTE2 FOR THE ACM CONFERENCE will be analyzed into four simple phrases: (1) THE OUTLINE OF THE PAPER, (2) THE PAPER ABOUT AUTONOTE2, (3) THE PAPER FOR THE CONFERENCE, and (4) THE ACM CONFERENCE. Each simple phrase consists of a subject noun and a modifier word. When two simple phrases have a common subject noun, we say they are coordinate simple phrases. When a modifier word of one simple phrase subsequently becomes the subject noun of another, we say the latter phrase is subordinate

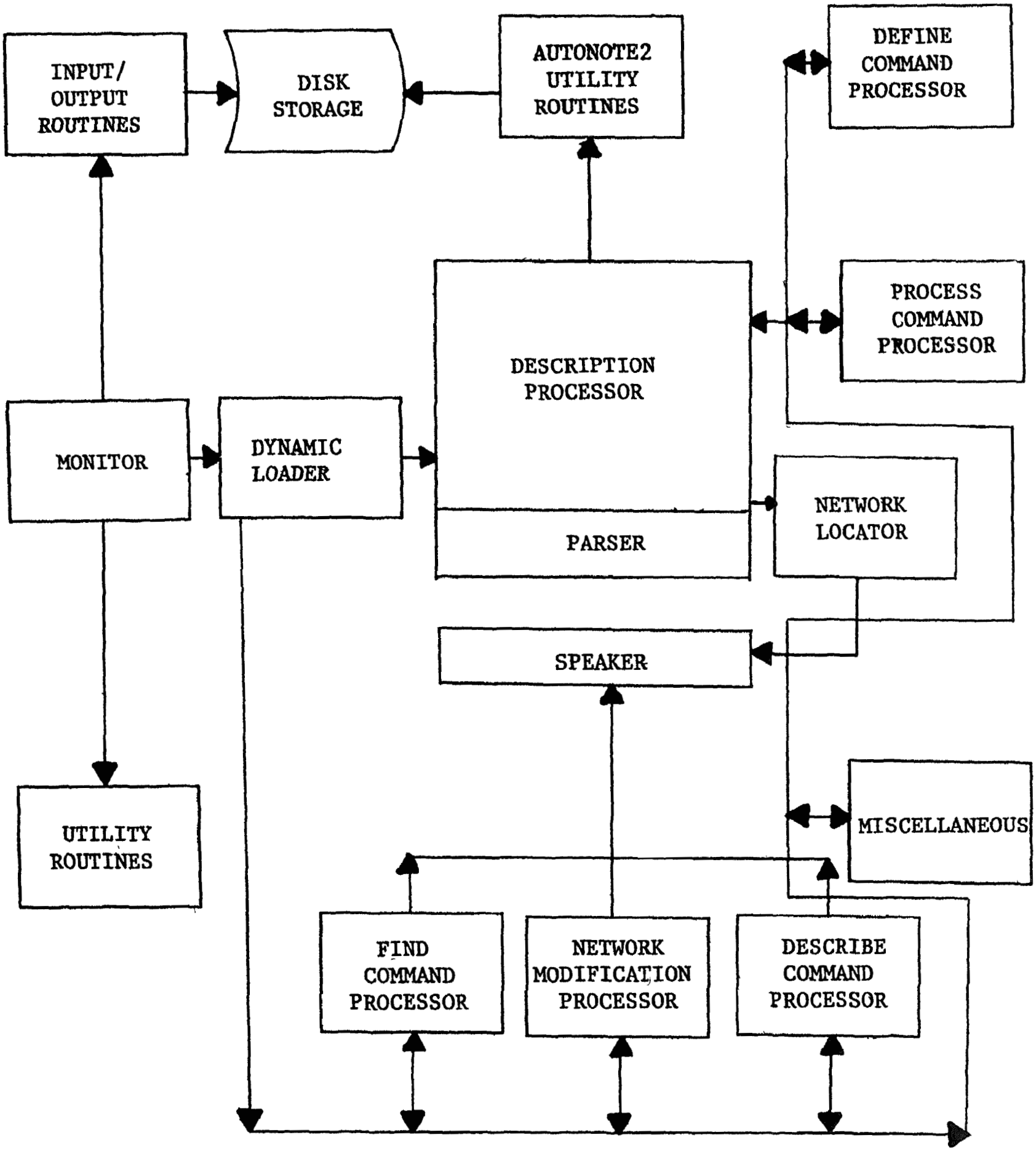


Fig. 4 - The Organization of AUTONOTE2 within the AUTONOTE System

to the former. In the above example, THE PAPER ABOUT AUTONOTE2 and THE PAPER FOR THE CONFERENCE are coordinate simple phrases--both have PAPER as their subject noun. Both of these are subordinate to the phrase OUTLINE OF THE PAPER, in which PAPER appears as a modifier word. Additionally, the simple phrase THE ACM CONFERENCE is subordinate to THE PAPER FOR THE CONFERENCE. Subordinate phrases simply qualify the use of their subject words. For example, phrases subordinate to THE OUTLINE OF THE PAPER provide a more detailed description of the paper ("ABOUT AUTONOTE2," and "FOR THE CONFERENCE"); the phrase subordinate to THE PAPER FOR THE CONFERENCE further qualifies the conference.

In effect, two kinds of dependency information are extracted by the parser. The first is the dependency of adjectives and prepositional phrases upon a noun. This information is reflected in the selection of the simple phrases themselves. Second, there are the dependency relationships among the simple phrases of the description. This information, expressed in terms of subordinate and coordinate linkages, may be represented by a tree structure consisting of nodes with simple phrases at the terminal branches. Fig. 5 gives the tree structure for the example. Simple phrases with an immediate linkage to a node are said to directly describe that node. Note that the two coordinate phrases from the example directly describe a common node, node B. The subordinate relationship of the node B phrases to the node A phrase, and in turn, that of the node C phrase to the node B phrases is reflected by downward branches connecting those nodes.

The resultant tree structure defines the representation of its corresponding topic. Representations of each of the user's topics are organized into a

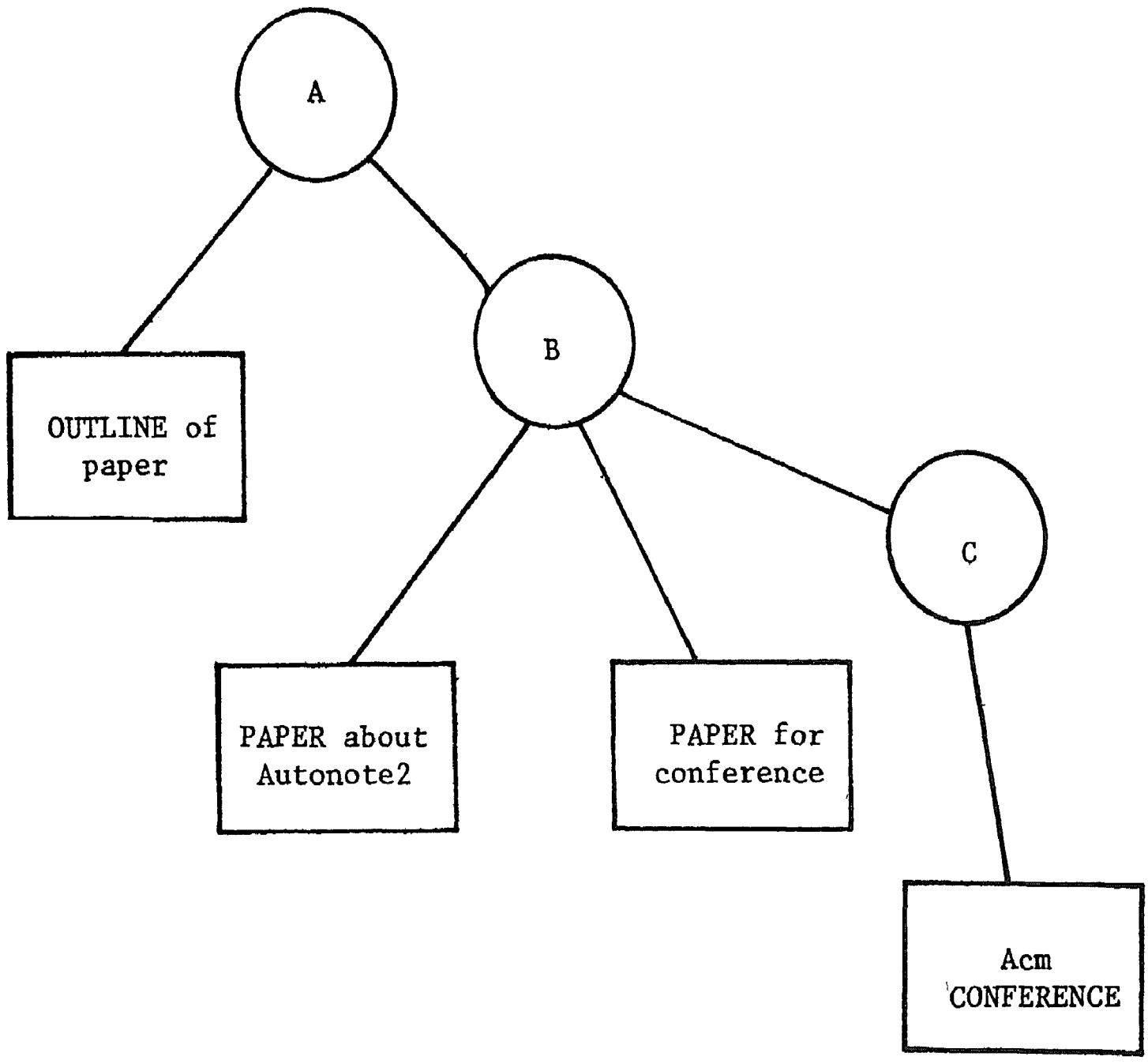


Fig. 5 - Simple Phrase Dependency Structure

hierarchical data structure called the representational network. The representational network is composed of interconnected nodes, simple phrases, and words. When description is mapped onto the network, the number of the associated text item is stored with the highest order node in the corresponding topic representation.

Each node in the network may have up to four types of linkages: (1) pointers down to simple phrases that directly describe the node; (2) pointers down to subordinate nodes; (3) pointers up to superior nodes; and (4) pointers to textual materials associated with the node. Each simple phrase or single word is directly accessible as a unit in the network through hash coding procedures similar to those used in maintaining the AUTONOTE keyword index. Associated with each simple phrase are the linkages to the node(s) the phrase directly describes. In turn, each single word has associated pointers that lead the system to the simple phrases containing the word. Fig. 6 illustrates the network representation of the example.

Once a topic is defined in the network, the user can refer to it using a word, a simple phrase or composition of simple phrases. For example, should the user later describe a new text item as say, OUTLINE OF THE PAPER or OUTLINE OF THE PAPER ABOUT AUTONOTE2, the system will note that it already has a representation for the topic. The only change to the network in such cases is the addition of new item reference linkage to the identified node (node 1). In general, the system attempts to relate each new item description to those it already "knows" about. For new topics, new nodes are allocated in the network. Should some subset of the simple phrases of a new description refer to an existing topic, the additional simple phrases are

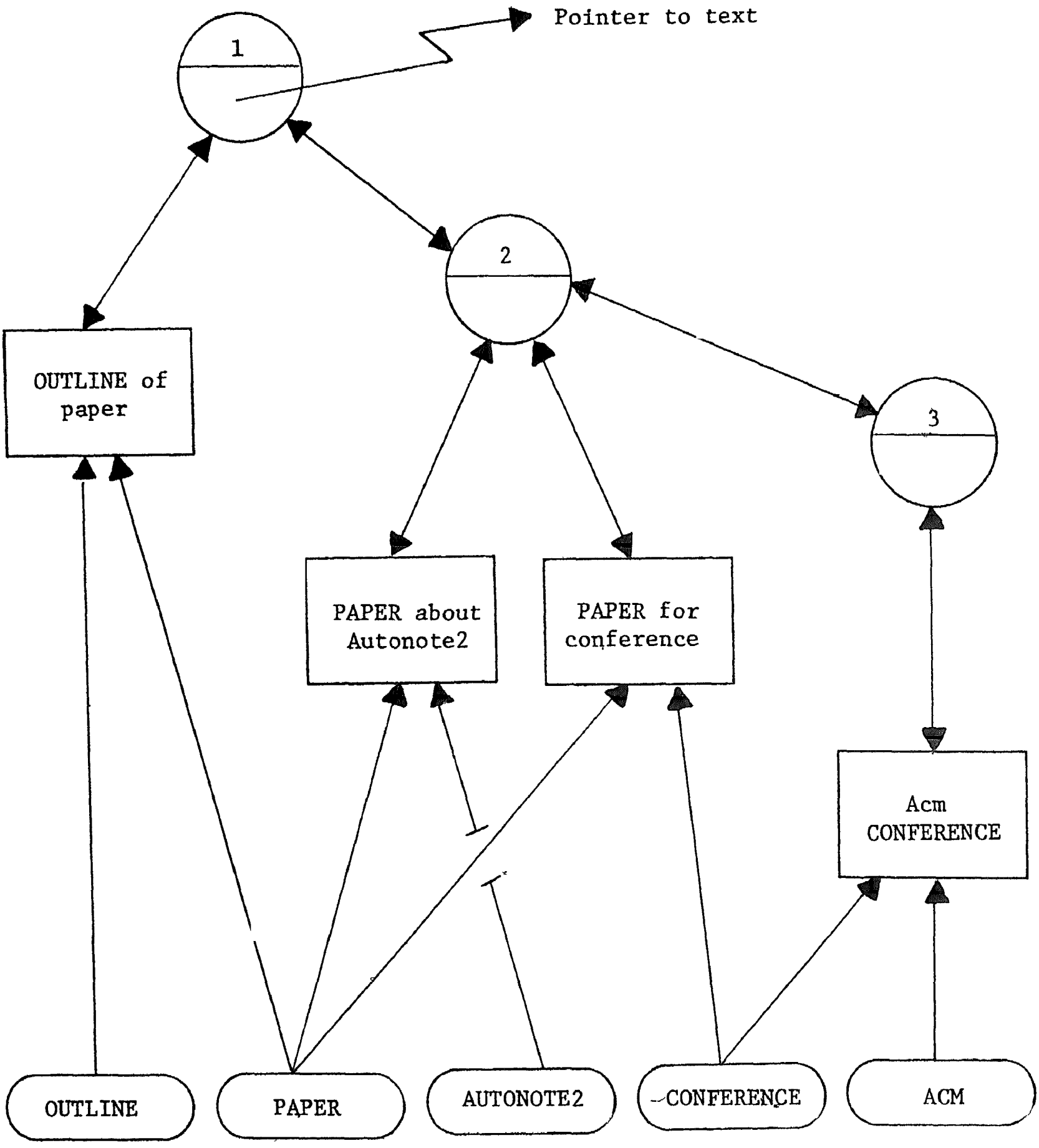


Fig 6 - Corresponding Representational Network Structure

linked to that existing representation. For example, if in reference to the same paper the user describes another item as THE ABSTRACT OF THE PAPER ABOUT AUTONOTE2, the system would modify the network to that shown in Fig. 7.

Design of the Network Data Structures

List structures. As noted above, the representational criteria dictate a hierarchical network-type organization, based upon dependency analyses of topic descriptions. List structures are particularly well suited for this kind of application. They provide a convenient representation for dependency trees and are especially appropriate for dealing with complex, evolving structures.

In designing special purpose list structures for the representational network, we first specified the logical components of the structure and defined the interconnections among these primitives. Three logical components were forumlated--simple phrases, nodes, and words. The following subsections present the major design considerations for each structural component.

Simple phrases. Given our goal of communicative efficiency, we chose the simple phrase as a primary unit for the network. By analyzing a topic description into simple phrases we are in effect isolating possible "shorthand" references to the given topic. The representational data structures have been designed to allow a topic to be referenced through any of its component simple phrases.

Simple phrases are formed from either adjectival or prepositional modification of a noun. Very often, an adjectival modification can be equivalently expressed by a prepositional phrase dependent upon the same noun

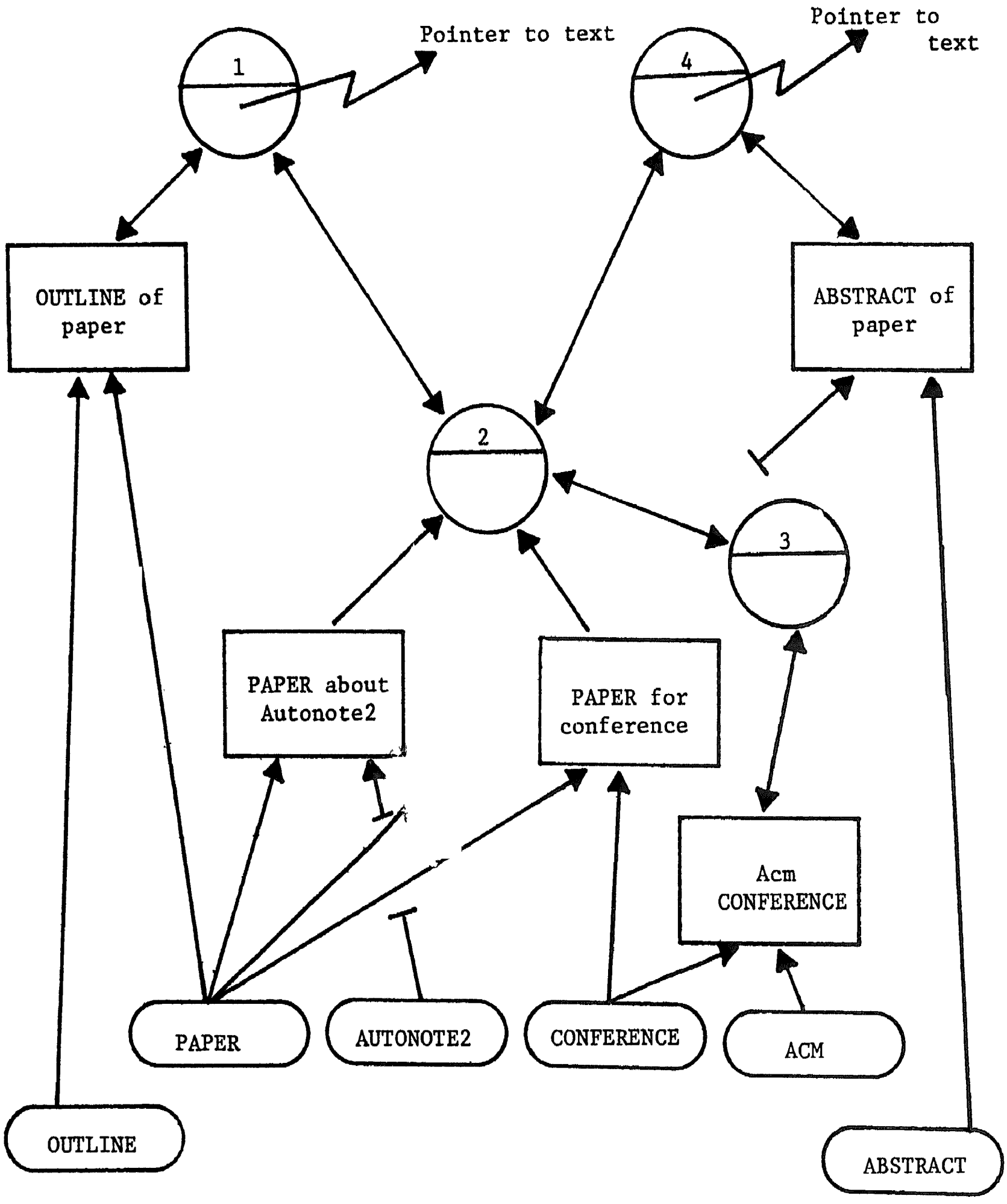


Fig. 7 - Network Representation of a Topic Hierarchy

(example: THE PAPER ABOUT AUTONOTE and THE AUTONOTE PAPER). In other instances the adjectival form may have multiple interpretations; THE SMITH ARTICLE could refer to an article by Smith or possibly an article about Smith. Some prepositions may be used synonymously in a particular context (THE PAPER ABOUT (ON) SHORT TERM MEMORY); others convey distinctly different meanings (THE MEMO TO THE COMMITTEE versus THE MEMO FROM THE COMMITTEE). We do not deal with these problems to the extent of providing a semantics for "understanding" natural language. However, the representational structure makes explicit the various possibilities, so that the system is able to generate plausible alternatives.

We treat adjectival modification as a special case, as if the modifier and subject noun were related by an unspecified preposition. In terms of the data structure design, all simple phrases composed of the same two words are mapped into a larger unit, each subunit of which represents a particular instance of a simple phrase in a topic description. This arrangement assures that all information on simple phrases involving any two words is accessible collectively. This information will then be at hand to provide a basis for interpreting the alternative referents of each incoming simple phrase. For example, should the user make reference to THE SMITH PAPER and the system finds only PAPER ABOUT SMITH in the network, then that single alternative is chosen. On the other hand, if PAPER BY SMITH also is present, the system considers both possibilities.

Network nodes. The next structural component of the representational network is the node. A node groups together a set of simple phrases that comprise the description of the node. The node also functions as a collector

of item references. Each node in the representational network corresponds to a topic or concept pertinent to the items of textual material associated with it. The simple phrases that directly describe a node define the corresponding concept. Any given node may be linked to more general (lower) nodes, or to more specific (higher) nodes. For example, a node that represents a particular paper may be linked downward to another that describes a conference at which the paper was presented; it may also be linked to several higher order nodes corresponding to, say, a summary, an outline, and a review of the paper. As more and more items are described, additional topics may be tied into the same conference node. The ultimate result will be a highly interconnected set of concept nodes, each with its own set of associated textual materials.

To achieve this kind of structural organization for the network, we make use of the dependency relationships in the user's descriptions: each node level corresponds to a syntactic dependency level. In terms of the example above, the adjectives and prepositional phrases modifying the noun "paper" are formed into simple phrases that will directly describe a common node. Simple phrases identifying the conference will describe a subordinate node due to the syntactic dependency of "conference" upon "paper" in a phrase of the form, PAPER AT THE...CONFERENCE. Superior nodes are assigned to the outline, the summary, and the review, reflecting the dependence of "paper" upon those nouns in appropriate descriptions.

A node may be viewed as a collection of pointers to simple phrases, other nodes, and text items. All node linkages are two-way. Pointers down from a node to its simple phrases are required in order to reconstruct a

description of the node. Pointers down to subordinate nodes are necessary for the same reason. Both upward and downward pointers to other nodes provide a means for moving from any topic to structurally related ones. Associated with each instance of a simple phrase is a pointer to the node where item references are stored. Finally, bookkeeping information stored with each text item includes pointers to each topic node with which the item is associated. Item-node linkages enable the system to provide the user with topic descriptions of any text item.

Words. The representational structures considered thus far provide simple phrases as the sole means for accessing the nodes in the network. A less restrictive access mechanism also is required, for several important reasons. First, it would be unrealistic to assume that the user will always phrase references to a particular topic in exactly the same way. Second, single word descriptions play an important role in achieving our goal of communicative efficiency. Since we anticipate that users will make frequent use of single word references when working in the context of a particular topic, we want to provide a natural and convenient treatment of such descriptions. Finally, a phrasal description can convey a higher order categorization of an existing topic without containing a simple phrase for that topic. For example, THE REVIEWER'S COMMENTS ON THE PAPER may reference a paper mentioned earlier; yet it contains no simple phrases describing that paper.

These considerations lead us to the third logical component of the network data structures, the single word. Essentially, each component word provides access to a series of pointers to simple phrases in which the word occurs. Word-to-phrase pointers are of two types: those indicating usage

as subject noun; and those indicating modifier usage in a particular simple phrase. As we shall see later, this distinction is required in order to relate new simple phrases to existing topics at an appropriate node level.

Having specified the three logical components and the linkages in the representational network, we now turn our attention to the storage implementation of these structures.

Storage Implementation of the Network Structure

There are three directories needed to maintain the representational network, one for each of the components of the structure. All directory information must, of course, be saved in permanent storage between AUTONOTE2 sessions. Two design alternatives were considered for maintaining the network during execution of the program. The directories could be accessed and updated on disk, or they could be brought into core storage for the duration of the session. We adopted the former strategy for a number of reasons. First, AUTONOTE is highly oriented toward the use of disk file storage. Several file interface routines were available at the outset for conveniently storing and accessing information through the MTS file system. Second, as the network grows in complexity, it becomes increasingly unlikely that the user will reference the major portion of the network during any given session. By maintaining the network in disk files, the amount of core storage required is substantially reduced. Finally, the file approach greatly simplified the programming effort, especially in those system components that operate recursively on the list structured network. We will elaborate on this point further in section VI, which illustrates the simplification of recursive processes in AUTONOTE2.

Rather than store all the directories in a single disk file, we chose to maintain each directory separately. This strategy preserves the logical distinction among the three types of directory information, and has also simplified the programming of the system. We now describe the organization of each of the directory files.

The node directory. Each node in the representational network has a corresponding integral node number which is also the line number in the node directory file. As new node numbers are needed to represent new topics, the next sequentially numbered line in the node directory is assigned as the node number. Each node directory line contains four fields--one for bookkeeping information and three fields for the upward, downward, and item reference pointers for the node. The item reference region contains a list of integer item numbers. The upward pointer region also contains a list of integers that represent immediate linkages to superior nodes. The two types of downward pointers (to nodes and to phrases) are stored in a common region. Each node, simple phrase, and single word has a corresponding file line number in its respective directory file. In the case of nodes, the line number is simply the node number. In the case of words and simple phrases, the line number is the result of a hash coding process on a compact character representation of the word or phrase. Thus a "pointer" is actually a file line number. Downward pointers to nodes and phrases are distinguishable in the node directory on the basis of the magnitude of the line number.

Since each of the three pointer fields is of fixed length, there is a maximum number of each type of pointer for a given node directory line.

Each field consequently has an associated continuation pointer to a line where additional pointers are stored if necessary.

The phrase directory. To locate the phrase directory line for a particular simple phrase, a hash coding function is applied to the character string formed by concatenating the modifier word, a slash, and the subject word. For example, the directory line for the simple phrase PAPER ABOUT AUTONOTE is the hashcode for the string "AUTONOTE/PAPER." Since the hashing function operates only on the modifier and subject word, simple phrases formed from the same two words, but with differing (or no) prepositions, are mapped into the same directory line number.

To distinguish among the various instances of the same two-word combination, the directory line for simple phrases consists of a series of pointer blocks. Each pointer block contains a code for the particular preposition used, some additional bookkeeping information, and a pointer to the node directly described by that occurrence of the simple phrase.

The word directory. The word directory incorporates the same pointer block principle as the phrase directory. The pointer field of the block in this case is a pointer into the phrase directory. The preposition code field contains a binary flag indicating whether the particular word occurs as the subject noun or modifier word in the simple phrase specified by the pointer. Like phrases, each word directory line is accessed through an efficient hash coding algorithm.

The word directory also maintains preposition usage information for each word. For example, the entry for MEMO may indicate that the word has

occurred with the prepositions ON, ABOUT, TO, FROM, etc. This information is used to guide the parsing of descriptions.

The organization of the three network directories is depicted in Fig. 8.

The Representational Network: An Example

To help fix ideas, we now present a more detailed example that illustrates the structure of the representational network. Suppose the user describes Items 157, 158, and 159 as THE PAPER ABOUT AUTONOTE FOR THE ACM CONFERENCE: he enters materials on the organization of that paper into Items 201 and 202. A summary of the paper is placed in Item 230. The user also describes Item 270 as SMITH'S PAPER, and enters a summary of that paper into Item 312. A pictorial representation of the resultant portion of the network is given in Fig. 9, while the corresponding directory contents appear in Fig. 10. For simplicity, the simple phrase hash codes are represented by the alphabetic characters U through Z. (In subsequent diagrams, we also omit word-to-phrase linkages for simplicity.)

Parsing of Descriptions

This section outlines our general approach to parsing topic descriptions. The parsing of prepositional phrases, consecutive modifiers, and possessive modifiers is considered.

Prepositional phrases. Despite the apparent simplicity of the description language there are several nontrivial parsing problems. One of these is the difficulty in determining the noun referent of prepositional phrases. The determination of noun referents is partially a semantic problem rather than a purely syntactic one. Consider the following two descriptions:

Internal Form of the Word or Phrase	Pointer to Continuation Lines	Collision Pointer[a]	Preposition Usage	Number of Upward Links	Series of Fixed Length Blocks[b] (Format Given in (b) Below)

(a) Format for the word and phrase directories.

Upward Link	Access Recency	Preposition Code[c]	Article Codes	For Future Expansion

(b) Pointer block format in the word and phrase directories.

Number of Upward Links	Number of Downward Links	Number of Items	Access Recency	Continuation Line Pointers	List of Downward Pointers	Preposition Code for Each Downward Pointer	List of Upward Pointers	List of Item References

(c) Format for the node directory.

Fig. 8 - Representational Network Directory Formats

---

[a]Used in conjunction with the hash coding mechanism.

[b]For single words, there is one block for each phrase containing the word. For phrases, there is one block for each node that the phrase directly describes.

[c]For single words, the preposition code is used to distinguish between words used as subjects or modifiers.

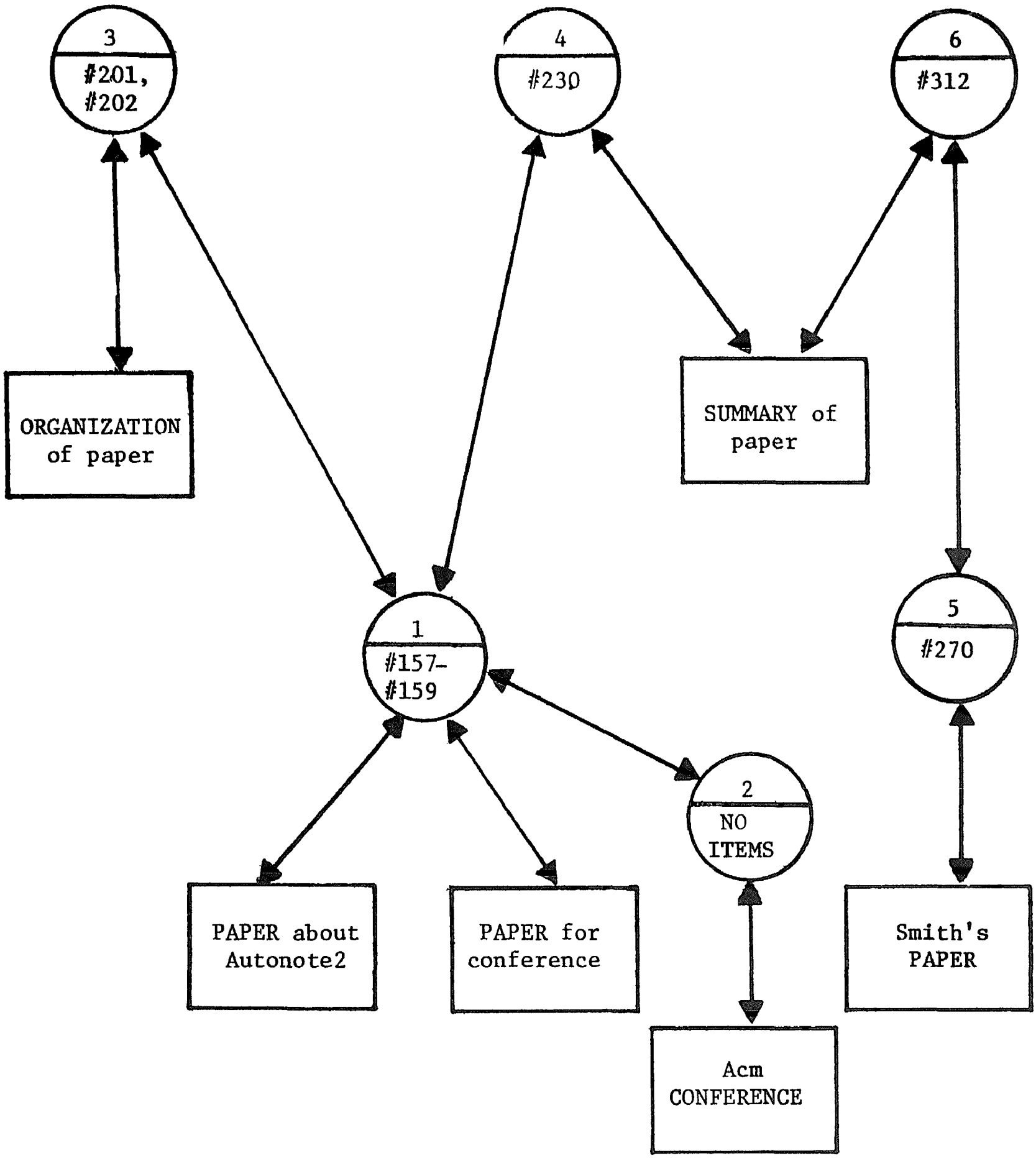


Fig. 9 - A Complex Representation

Word	No. Blocks	Blocks				
Organization	1	U (sub)				
Paper	5	W (sub)	X (sub)	U (mod)	V (sub)	Z (mod)
Autonote	1	X (mod)				
Conference	2	W (mod)	Y (sub)			
Summary	1	Z (sub)				
ACM	1	Y (mod)				
Smith (poss)	1	V (mod)				

(a) Word directory.

Line No.	Phrase	No. Blocks	Blocks	
U	paper/organization	1	Node 3 (of)	
V	smith/paper (poss)	1	Node 5 (adj)	
W	conference/paper	1	Node 1 (for)	
X	autonote/paper	1	Node 1 (about)	
Y	acm/conference	1	Node 2 (adj)	
Z	paper/summary	2	Node 4 (of)	Node 6 (of)

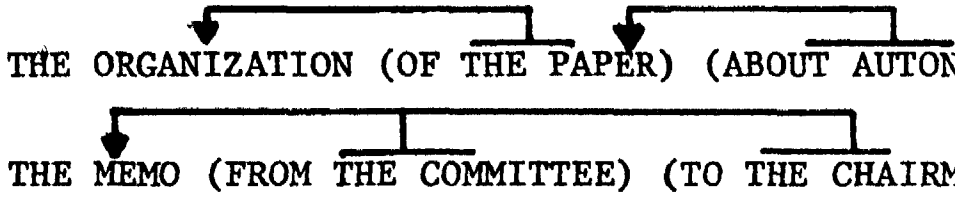
(b) Phrase directory.

Line No.	Pointers Up	Pointers Down	Item References
1	3,4	2,W,X	#157, #158, #159
2	1	Y	...
3	...	1,U	#201, #202
4	...	Z,1	#203
5	6	V	#270
6	...	Z,5	#312

(c) Node directory.

Fig. 10 - Corresponding Directory Contents[a]

[a]See Fig. 9.



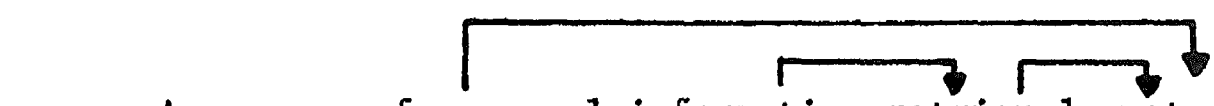
In the first example, both prepositional phrases refer to the immediately preceding noun. In the second case, both refer back to the noun MEMO at the beginning of the string. Although neither of these examples is intuitively ambiguous, the parsing algorithm must consider each preceding noun as a possible referent of any given prepositional phrase.

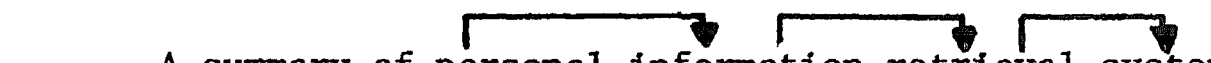
The AUTONOTE2 parser deals with this problem to a limited extent, by utilizing prepositional clues. For example, if the system finds that the noun MEMO can form a simple phrase with the prepositions ON, ABOUT, TO, and FROM, then phrases introduced by these prepositions will be associated with that noun. Such clues will not always yield a unique parsing, of course, as in the case of inherently ambiguous descriptions. THE PAPER FOR THE CONFERENCE ON GENETICS, for example, could refer to a paper on genetics to be delivered at a conference, or to a paper which is to be delivered at a conference on genetics.

In such instances we rely upon the user to supply the referent noun upon request. In the example above, the system may prompt: DOES "ABOUT GENETICS" REFER TO PAPER OR CONFERENCE? Should the user reply CONFERENCE, the simple phrase CONFERENCE ON GENETICS will be added to the network. If at some later time, the parser is attempting to find a referent for the prepositional phrase ON GENETICS where CONFERENCE is one of the alternatives, it forms that simple phrase directly.

Consecutive modifiers. A parallel problem arises in determining the noun referents for a string of consecutive modifiers. Descriptions

containing at most a single adjective for any particular noun are parsed in the obvious manner. A simple phrase is formed from each modifier and the noun following it. In the event a noun is preceded by two or more modifiers, the parser is confronted with a task similar to that of determining the referent of a prepositional phrase. The modifier occurring immediately before the noun is first processed as above. Each of the remaining modifiers, however, can modify any one of several words depending upon their "distance" from the head noun. Specifically, any such modifier can refer to either the head noun or any of the other modifiers following it. Consider the descriptions:





In both of the cases above, INFORMATION modifies the modifier RETRIEVAL which in turn modifies the head noun SYSTEMS. Depending upon the user's intent, PERSONAL can modify either INFORMATION or SYSTEMS. The choice of modifier referents is an especially important problem when there are multiple parsings, each resulting in a different semantic interpretation. For example, LARGE COMPUTER CONFERENCE could refer to a conference on large computers, or a large conference on computers. Another important reason for our emphasis upon correctly identifying modifier referents concerns the use of paraphrasing. In the example, PERSONAL INFORMATION RETRIEVAL SYSTEMS, if we determine that INFORMATION modifies RETRIEVAL and PERSONAL modifies SYSTEMS, then the resultant topic can be paraphrased as (1) PERSONAL SYSTEMS FOR

INFORMATION RETRIEVAL, or (2) PERSONAL SYSTEMS FOR THE RETRIEVAL OF INFORMATION. Depending upon context and the nature of other topics in the network, the following incomplete descriptions will in most cases identify the topic:

1. SYSTEMS
2. SYSTEMS FOR RETRIEVAL (or RETRIEVAL SYSTEMS)
3. PERSONAL SYSTEMS
4. PERSONAL SYSTEMS FOR RETRIEVAL (or PERSONAL RETRIEVAL SYSTEMS)
5. SYSTEMS FOR INFORMATION RETRIEVAL
6. SYSTEMS FOR RETRIEVAL OF INFORMATION

A different choice of modifier referents determines a correspondingly different set of paraphrases. If PERSONAL was intended to modify INFORMATION, we would have the paraphrase SYSTEMS FOR THE RETRIEVAL OF PERSONAL INFORMATION, with a corresponding list of incomplete references to the topic.

As in the prepositional case, the choice of modifier referents is guided by the current state of the representational network. After processing the last modifier in the string, the parser positions itself at the preceding modifier and moves left in the input string until the first word in the modifier string is processed. In the above example, after associating RETRIEVAL with SYSTEMS, the parser next examines the modifier INFORMATION. A list of simple phrase candidates is formed. In this case, the list contains INFORMATION RETRIEVAL and INFORMATION SYSTEMS. If neither of the candidate phrases has been previously used, the system queries: WHAT DOES INFORMATION MODIFY? The user's reply is matched against the candidate referents and the appropriate simple phrase is formed.

Possessive adjectives. Possessives are processed in much the same way as normal modifiers. The system recognizes the 's word stem and marks the root word as a possessive. The root word is later stored in the network directories along with a possessive flag. Thus the phrase SMITH'S PAPER is stored internally as SMITH/PAPER (possessive). The removal of the stem insures that a subsequent simple phrase incorporating a preposition (PAPER BY SMITH) will hash to the same directory line thus allowing the use of either prepositional or possessive forms in referencing topics.

A particularly interesting case arises when a possessive occurs in a string of consecutive modifiers as in SMITH'S LATEST MEMORY EXPERIMENT. The string is first processed as described above; that is, a check is made to see if SMITH has been used in a simple phrase with LATEST, MEMORY, or EXPERIMENT. In the event that this yields no clues, the system then checks to see if SMITH was rendered as a possessive. Upon noting that it was, the parser carries out a heuristic that assumes that the possessive modifies the head noun, EXPERIMENT.

The possessive heuristic can be fully stated as follows. A possessive occurring in a string of modifiers will be assumed to modify the head noun unless another possessive occurs between it and the head noun. In the latter case, the first possessive will be assumed to modify the second. This is similar to the possessive feature employed by the REL parser (Dosert & Thompson, 1971). Thus in SMITH'S RESEARCH GROUP'S MEMORY EXPERIMENT, SMITH'S is assumed to modify GROUP, and GROUP'S is assumed to modify the head noun EXPERIMENT.

The question now arises, why check the phrase directory first instead of applying the possessive heuristic immediately? To answer this, suppose a topic was originally described as THE RESULTS OF THE MEMORY EXPERIMENT BY SMITH and

the user now attempts to refer to it as SMITH'S MEMORY EXPERIMENT RESULTS. If the possessive heuristic were applied immediately, the system would incorrectly form the simple phrase SMITH'S RESULTS, not SMITH'S EXPERIMENT. By checking the network first, the simple phrase EXPERIMENT BY SMITH will be detected and the system will parse the description appropriately.

Implementation of the Parser

The ultimate goal of the parser is to determine the simple phrases of a topic description. The parsing algorithm is implemented as a two stage process. The first stage is a preliminary scan to ascertain that the string is in a form acceptable for analysis. The description is segmented into an ordered list of words, each of which is marked as either WORD, POSSESSIVE, ARTICLE, or PREPOSITION. The parser makes no distinction between nouns and modifiers until completing the scan. At this point, the last in a series of consecutive WORDs is marked as a NOUN; the preceding words are marked as MODIFIERs. Possessive modifiers are an exception as they can be recognized explicitly during the scan. A record of article usage is also kept, but the articles themselves are not placed on the word list.

The preliminary scan of the description can be viewed as a simple finite state process. Of course, to be completely formal, the recognizer would have to examine each input character. For convenience we will assume a five state automaton with inputs: WORD, POSSESSIVE, PREPOSITION, and ARTICLE. The state transition graph for the machine is given in Fig. 11. The machine starts in state S_0, examines the next input and moves to a new state. If at the end of the input string, the machine is in state S_1, called the final state, the input is accepted; otherwise, the user is asked to rephrase. Note that in state

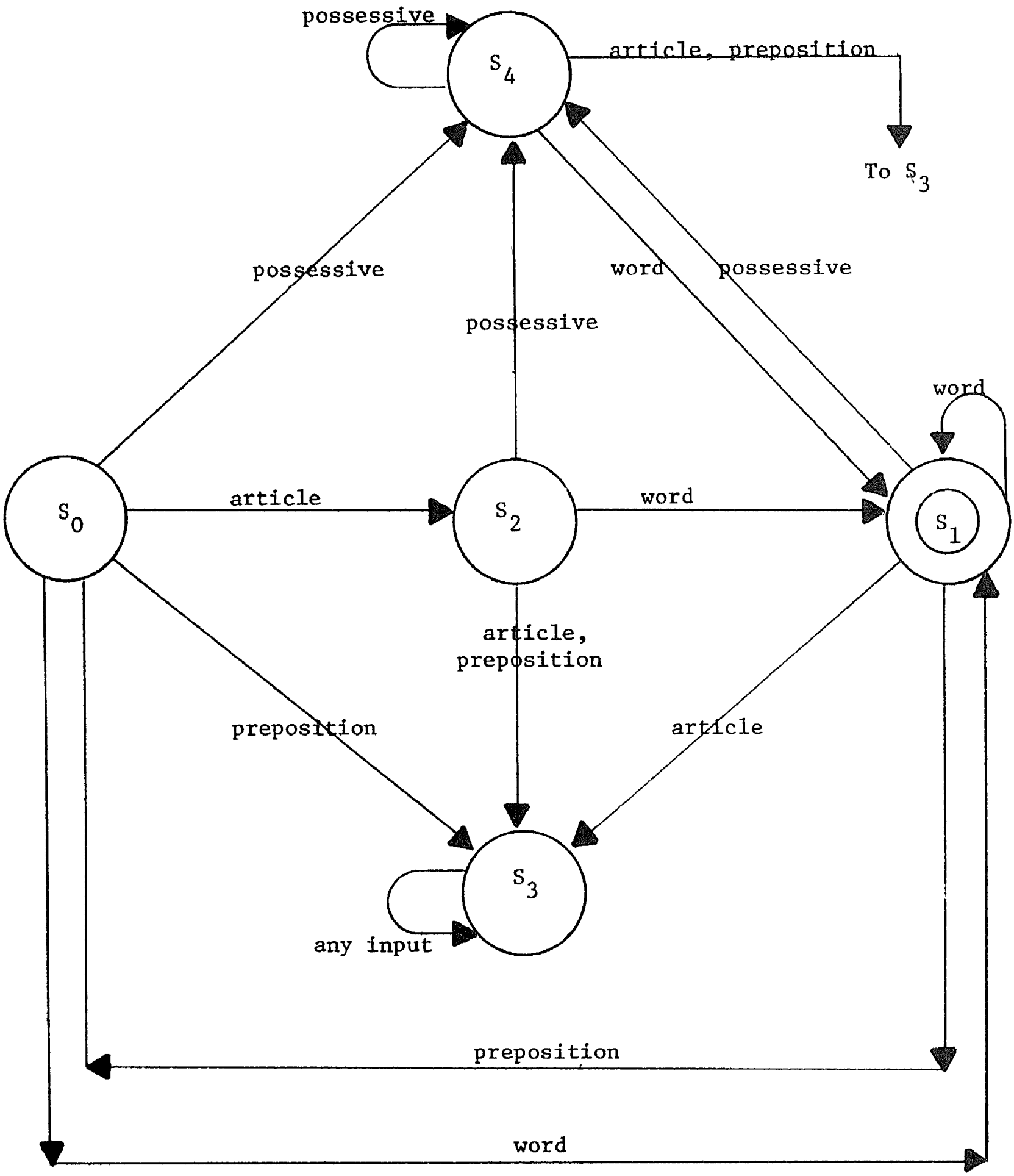


Fig. 11 - A Finite State Acceptor for the Autonote2 Description Language

S_2, the machine has just encountered an article and is anticipating a "word." If the machine is in state S_0 upon completion, it has just recognized a preposition and is expecting an object; thus, the string is rejected. The state S_4 is reached whenever a possessive is encountered. Since a possessive must have an object noun, a "word" input is required to reach state S_1. State S_3 is a trapping state; once entered, the machine remains in that state regardless of the remaining input and the description is consequently rejected. State S_3 corresponds to various error conditions--two consecutive prepositions or articles, an article between two words, a phrase beginning with a preposition, etc.

The state transitions for the description BRUNER'S FIRST EXPERIMENT ON THE CONSERVATION OF LIQUIDS are given below along with the resultant word list.

INPUT	TYPE	RESULTANT STATE
Bruner's	Possessive	S_4
First	Word	S_1
Experiment	Word	S_1
On	Preposition	S_0
The	Article	S_2
Conservation	Word	S_1
Of	Preposition	S_0
Liquids	Word	S_1 Accept

WORD	TYPE
Bruner's	Modifier
First	Modifier
Experiment	Noun
On	Preposition
Conservation	Noun (the)
Of	Preposition
Liquids	Noun

Descriptions found acceptable by the scanner next undergo analysis by the second stage procedure. This algorithm steps through the word list and builds a table of simple phrases called the phrase table. Each entry in the phrase table includes (1) the internal character representation of the phrase for use in hash coding, (2) a numerical code for the preposition used, (3) the hash code (directory line number) for the phrase, (4) a list of nodes directly described by the phrase, and (5) a coordinate or subordinate link to another phrase table entry.

We now illustrate the construction of the phrase table by following through several examples.

The parser in operation. Let us assume that a user is running the system for the first time; consequently, the three network directories are initially empty. Item No. 1 is opened, some text is inserted, and the user describes it as THE PLANNED PAPER ABOUT AUTONOTE FOR THE CONFERENCE. The description successfully passes the preliminary scan and the word list is constructed. The parser then moves on to determine the simple phrases.

The modifier PLANNED is first noted. Since it is followed immediately by a noun, the simple phrase PLANNED PAPER becomes the first entry in the phrase table. Next the prepositional phrase ABOUT AUTONOTE is encountered. Again there is only one possible noun referent. The phrase PAPER ABOUT AUTONOTE is entered into the table and marked as coordinate with the first entry. To determine the referent of FOR CONFERENCE, the system must consider two alternatives: AUTONOTE FOR CONFERENCE and PAPER FOR CONFERENCE. The network is interrogated to determine if either of the candidate phrases has been previously used. This test fails since the network is empty at this point. A check is

then made in the word directory to determine if either AUTONOTE or PAPER has headed a simple phrase with the preposition FOR. This also fails so the system asks the user: DOES "FOR CONFERENCE" REFER TO PAPER? A yes response results in the addition of PAPER FOR CONFERENCE to the phrase table. Since the noun referent, PAPER, is the same as the previous phrase, the new entry is marked as coordinate with PAPER ABOUT AUTONOTE. The completed phrase table is given in Fig. 12.

Phrase	Preposition	Article	Dependency	Links
(1) autonote/paper	about	...	root	...
(2) planned/paper	adj	...	co-ord 1	...
(3) conference/paper	for	the	co-ord 1	...

Fig. 12 - Sample Phrase Table

The phrase table is next passed to the network locator. We will assume it determines that the user is defining a new topic. Using the syntactic dependencies in the phrase table, the network locator assigns new node numbers to the phrases in the description. In this case, all three phrases are coordinate; each will directly describe node No. 1 in the network. In addition, a reference to Item No. 1 is stored with the topic node (see Fig. 13).

The user next enters Text Item No. 2 describing it as THE ORGANIZATION OF THE PAPER FOR THE ACM CONFERENCE. The system proceeds as before until it encounters the prepositional phrase FOR THE CONFERENCE. It forms the two alternatives PAPER FOR CONFERENCE and ORGANIZATION FOR CONFERENCE. Upon interrogating the network, it finds that PAPER FOR CONFERENCE has been defined

previously and accepts that candidate. ACM CONFERENCE is added to the phrase table and the parsing is complete (Fig. 14).

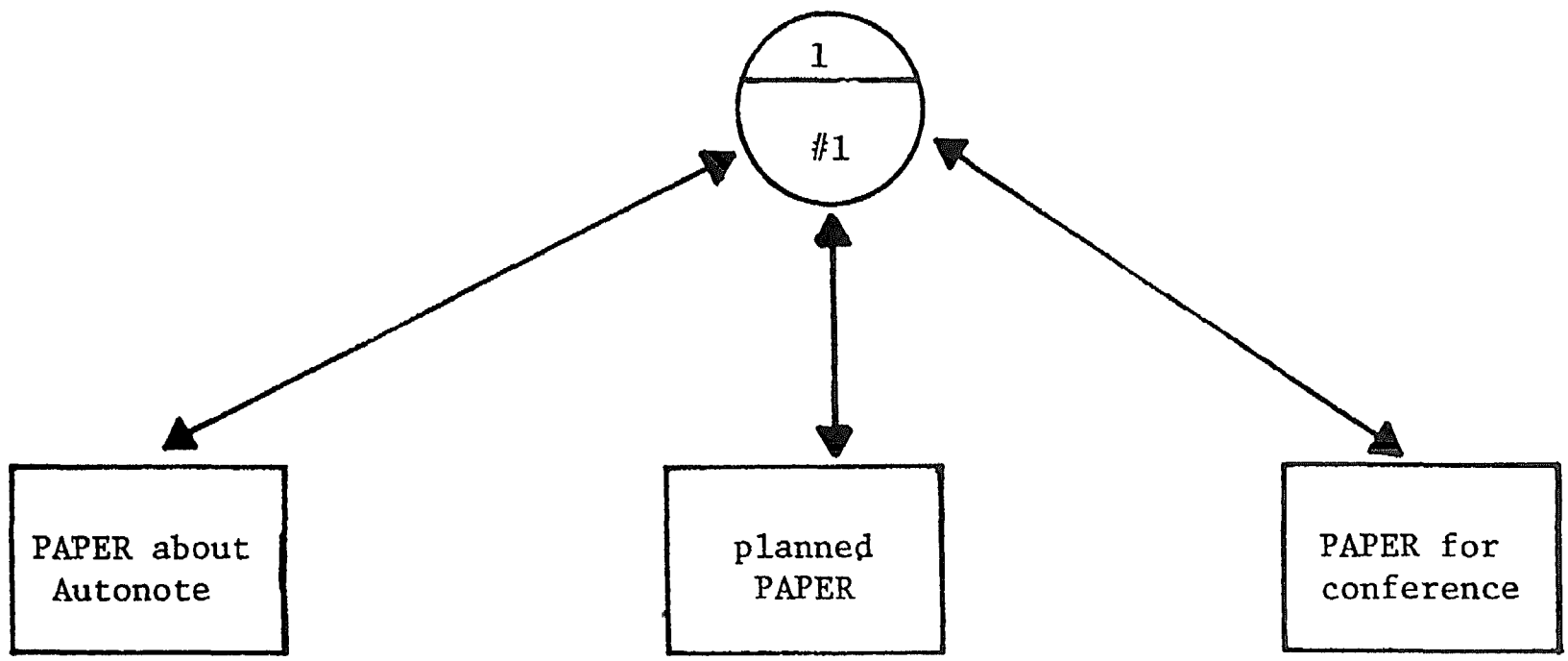


Fig. 13 - Corresponding Network Representation

Phrase	Preposition	Article	Dependency	Links
(1) paper/organization	of	the	root	...
(2) conference/paper	for	the	subord 1	Node 1
(3) acm/conference	adj	...	subord 2	...

Fig. 14 - Sample Phrase Table

The network locator must then determine if the user is referring to the same paper or a new one. The operation of the network locator will be discussed in detail in the next chapter. Let us assume for now that the current description is indeed a reference to the same paper. The simple phrase PAPER FOR CONFERENCE is already in the network. The system must then decide what to

do with THE ORGANIZATION OF PAPER, and with ACM CONFERENCE. Since the former is superior to the node 1 phrase, it is assigned to node 2 and a downward pointer from node 2 to node 1 is added. The phrase ACM CONFERENCE, on the other hand, is subordinate to a node 1 phrase. Thus it is assigned a new node number (node 3) and a pointer up from node 3 to node 1 is added. Phrase-to-node and node-to-node pointers are two way, thus corresponding pointers down from node 1 to node 3, and up from node 1 to node 2 are also added. The resul tant network is illustrated in Fig. 15. This example points out an interesting feature of the AUTONOTE2 system. Although Item No. 1 was originally described as a paper for some unspecified conference, a subsequent reference to that paper has enriched its description.

V. THE NETWORK LOCATOR

The purpose of the network locator is to determine whether the user's description makes reference to an existing topic in the representational network. Its decision is based on the information in the phrase table and the current state of the network. Once the decision is made, the locator builds a table, called the links table, that specifies the changes to be made in the network to represent the description.

In cases where the input description matches exactly some structure in the network, the links table will specify only the addition of an item reference. When the description defines a new topic, every phrase in the phrase table will be assigned a new node and links entries will be made for the proper node-node linkages.

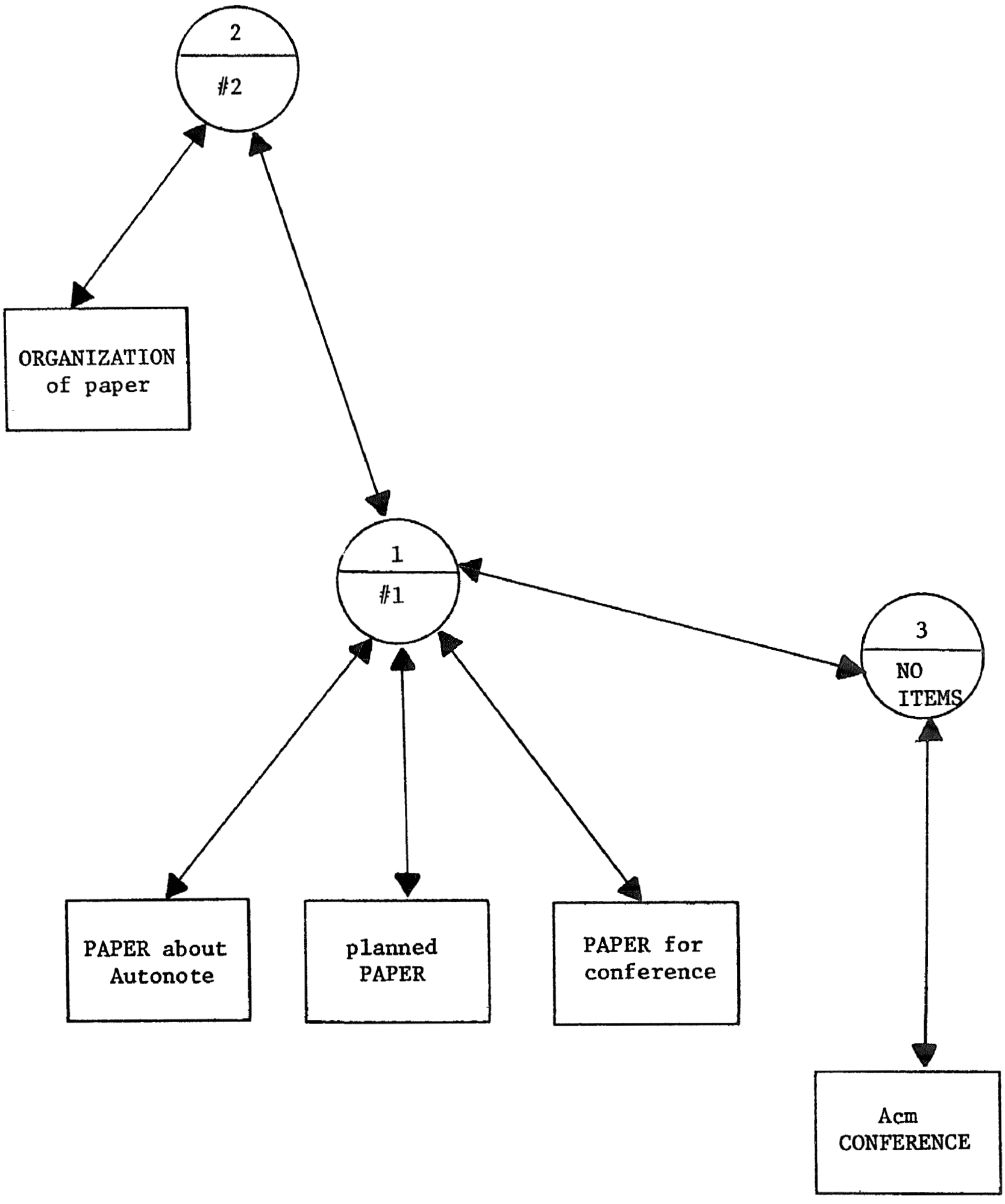


Fig. 15 - Network after Augmentation

In order to describe more precisely the operation of the network locator, let us assume the network has evolved to the state depicted in Fig. 16. Note that by starting at any node and tracing downward through the network, it is possible to reconstruct the description of the topic the node represents. The nodes in the network represent the following topics.

Node 1. THE PLANNED PAPER ABOUT AUTONOTE FOR THE ACM CONFERENCE.

Node 2. ORGANIZATION OF THE PLANNED PAPER ABOUT AUTONOTE FOR THE ACM CONFERENCE.

Node 3. THE ACM CONFERENCE.

Node 4. AN ABSTRACT OF THE FIRST PAPER ABOUT AUTONOTE.

Node 5. THE FIRST PAPER ABOUT AUTONOTE.

Node 6. THE REVIEWER'S COMMENTS ON THE PLANNED PAPER....

Node 7. THE PROCEEDINGS OF THE ACM CONFERENCE.

Node 8. TRAVEL ARRANGEMENTS FOR THE ACM CONFERENCE.

To illustrate the network location procedures, we will now go through several subsequent references to topics already defined in the representation.

Before passing the phrase table to the locator, the parser first checks to see if the description contains any active phrases, simple phrases that directly describe one or more nodes in the network. When the locator gets control, it checks an internal flag that indicates one of three conditions: the description contains one or more active phrases; the description contains no active phrases; or the description contains only a single word.

As our first example, consider the subsequent item description: THE PAPER ABOUT AUTONOTE FOR THE CONFERENCE. The phrase table is given in Fig. 17. The locator notes that there are two active phrases and focuses its attention on

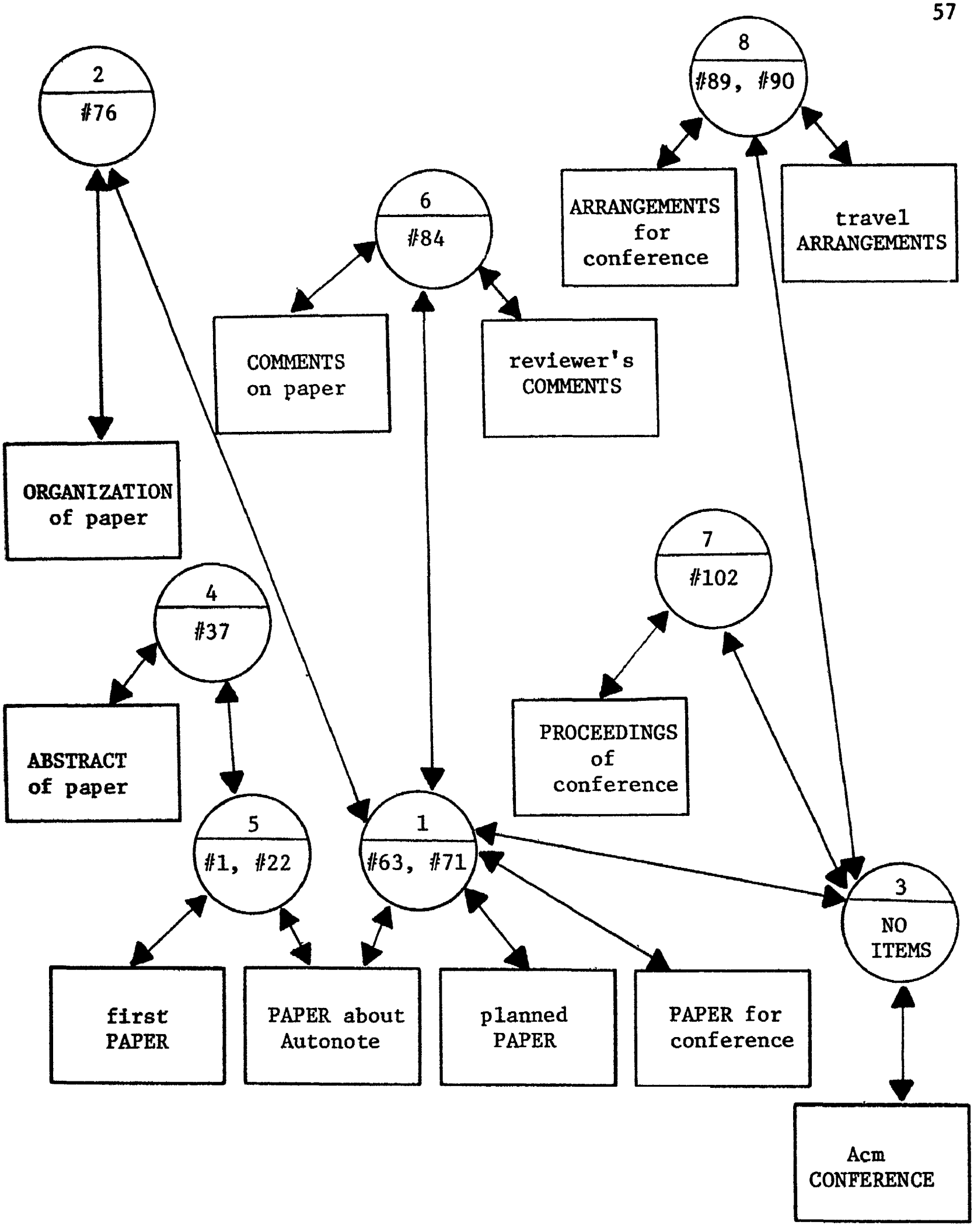


Fig. 16 - A Cluster of Related Topics

Phrase	Preposition	Article	Dependency	Links
(1) autonote/paper	about	the	root	Node 1
(2) Conference/paper	for	the	co-ord 1	Node 1

Fig. 17 - Sample Phrase Table

the first of these, PAPER ABOUT AUTONOTE. From information in the phrase table, it sees that PAPER ABOUT AUTONOTE plays a role in two distinct topics represented by nodes 1 and 5. It then considers both of these alternatives, checking to see if the remaining phrases in the phrase table either directly or indirectly describe either of the two nodes. Since both phrases directly describe node 1, the locator assumes that it is the topic node of user reference. As an option, the user may request the locator to display its assumptions, in which case the system replies: I ASSUME YOU MEAN THE PLANNED PAPER ABOUT AUTONOTE FOR THE ACM CONFERENCE. Other than the addition of an item reference to node 1, no network changes are made in this example. Note that the user has efficiently made reference to the desired topic, relying on the system to fill in the gaps in his description. The system would proceed in much the same way in processing shorthand descriptions such as SUMMARY OF THE PAPER or REVIEWER'S COMMENTS ON THE PAPER, in each case assuming that the user is referring to the same paper about AUTONOTE

In previous discussion we have alluded to the use of contextual clues in deciding among the alternative referents of a vague or ambiguous description. Context in the AUTONOTE2 system takes the form of an access recency number (context number). Each time the user refers to some topic in the network,

each of the component nodes is assigned the current context number. The current context number is incremented at the beginning of each AUTONOTE2 session and each time the user defines a new topic. Thus when deciding among alternative topic nodes, the system can readily determine which was referred to most recently.

Another class of interesting cases are those in which the description consists of a single noun. If the current description is THE PAPER, for example, the system would use the word directory to locate those simple phrases where PAPER is the subject noun. Using the resultant list of simple phrases, a list of nodes directly described by these phrases is generated. In this case, this process generates two alternatives (node 5 and node 1). The system then functions as before, either choosing a node in context, or interrogating the user.

The foregoing discussion has described our approach to network location. We now give a more detailed presentation of the algorithm.

Case I: Active phrases in the description. Should the user's description contain one or more active phrases, there is a good possibility that it references an existing network topic. The first step in processing such a description is to determine the focus phrase, the active phrase at the highest dependency level. Note that the focus phrase may be subordinate to other (non-active) phrases in the description. The basic idea is to use the nodes directly described by the focus phrase to get a set of candidate topics. Once these candidates are determined, they are matched against the remaining active phrases in the phrase table to determine the most likely referent.

Before describing the matching process, let us first consider a few special cases. Suppose, for instance, that the focus phrase directly describes only one topic node and that any additional active phrases are also present in that topic representation. The presence (or absence) of non-active phrases in the description is, in this case, an important parameter. Any non-active phrases may serve to distinguish the description from the existing topic. On the other hand, they could very well represent additional description of the topic at hand. If the topic under consideration is recent, we first assume the latter case. In addition, when processing descriptions rendered for retrieval, the network locator naturally rules out the possibility that a new topic is being described and accepts the one at hand.

The matching process. When the focus phrase directly describes two or more nodes, a network matching procedure is used to determine which of the associated topics the description references. The matching routine uses a list of the candidate nodes, a list of the active phrases in the description, and the current contents of the representational network. For each candidate node, the routine determines how many of the description's active phrases directly or indirectly describe that node. The matching routine returns a table of this information along with the number of the node, if any, that best matches the input description. The "best" node is the one that has the highest number of matching phrases. If two or more nodes have an equal number of matching phrases, an attempt is made to choose one of them on the basis of context.

The simplified flow diagram appearing in Fig. 18 summarizes the decision procedure for the case in which the description contains one or more active

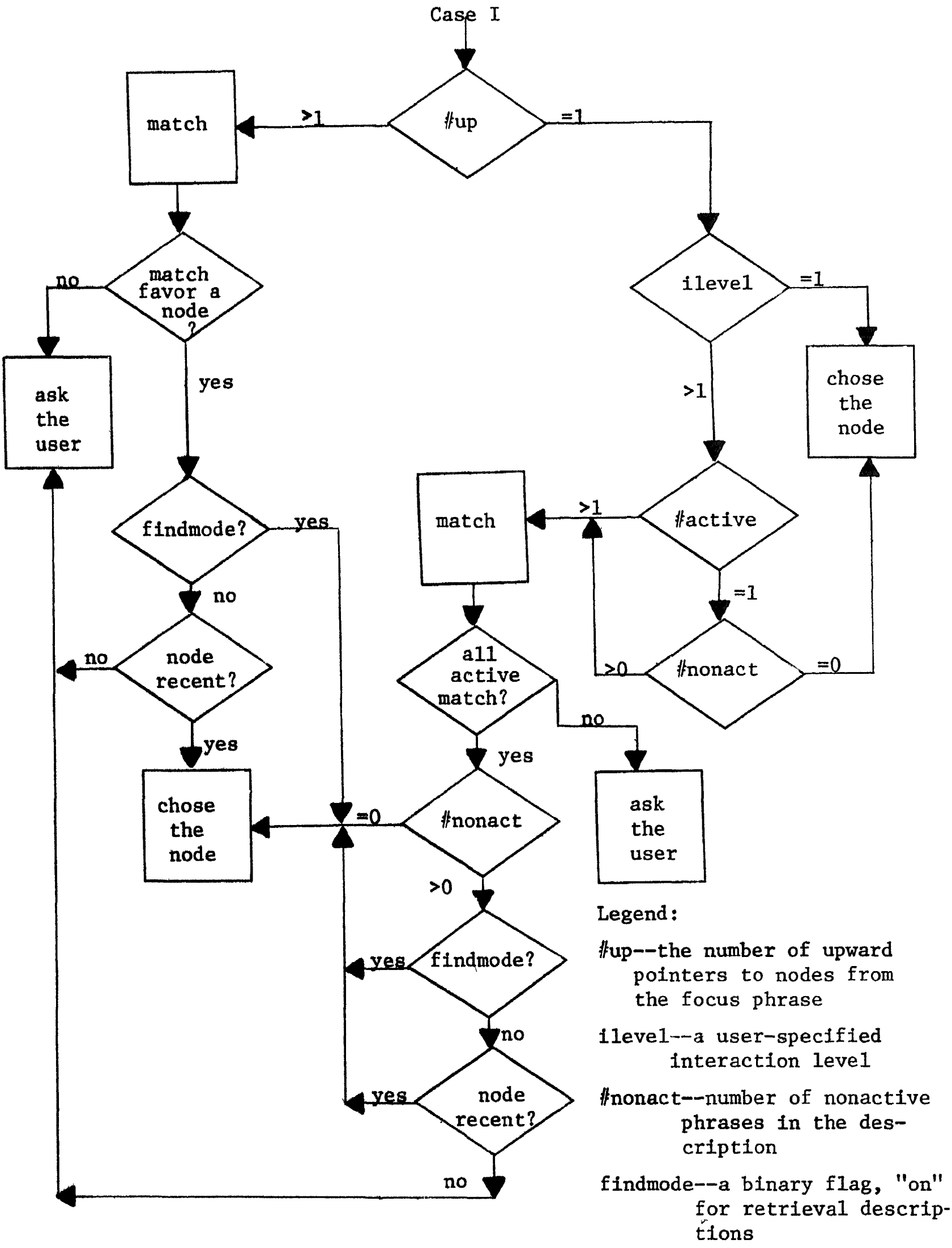


Fig. 18 - Network Location Procedure for Descriptions Containing Active Phrases

phrases. Note in particular that in case the description consists of only a single active phrase that directly describes a single node, the network locator assumes the node immediately. The rationale is that we anticipate the user will frequently make use of such terse descriptions in reference to previously define topics. Recalling our earlier discussion of human referential communication, a speaker makes incomplete references with the assumption that the topic can be inferred by the listener; otherwise, he describes his subject more precisely to avoid being misunderstood. The network locator was designed with this in mind. That is, whenever a terse description references a single node directly, or in context, that node is taken as the referent.

Case II: No active phrases. The first step in processing a description with no active phrases is to examine its component words, attempting to identify possible referents by utilizing the word and phrase directories. If no candidate nodes are generated by the procedure, the network locator assumes a new topic is being defined and allocates new nodes in the network.

Non-active descriptions that reference existing topics fall into two categories. First, there are those that paraphrase some existing topic description. For example, a topic originally described as THE USE OF THESAURI IN THE SMART SYSTEM may subsequently be referred to as THESAURI TECHNIQUES IN SALTON'S SYSTEM. Second, the description may constitute a more specific classification of some topic. While working in the context of a particular paper, for example, the user may describe a new item as THE ORGANIZATION OF THE PAPER, where ORGANIZATION OF PAPER is a non-active phrase.

The word search procedure involves the use of selected words from the description and the word directory to obtain a set of active phrases containing

those words. From there, a set of nodes is obtained by collecting the upward pointers from those phrases in the phrase directory. The resultant set of nodes then is processed as a list of candidate topics just as if they had been obtained immediately from a description containing active phrases.

An important consideration here is which of the words in the description to use in the search for candidate topics. In an early version of the system, we tried using each noun and modifier in turn. Although this approach was successful in many cases, it often resulted in an extremely lengthy list of alternatives. We also noted that words at the highest dependency level more often led to identification of the topic node than those words occurring at subordinate levels. For this reason, it was decided to curtail the word search, using only the words in the "root phrase" of the description. For the non-active description PAPER ON CLUSTERING IN THE SMART SYSTEM, the words PAPER and CLUSTERING would be used in the search for candidate nodes. As a user option, the system will expand the search to include the remaining words in the description.

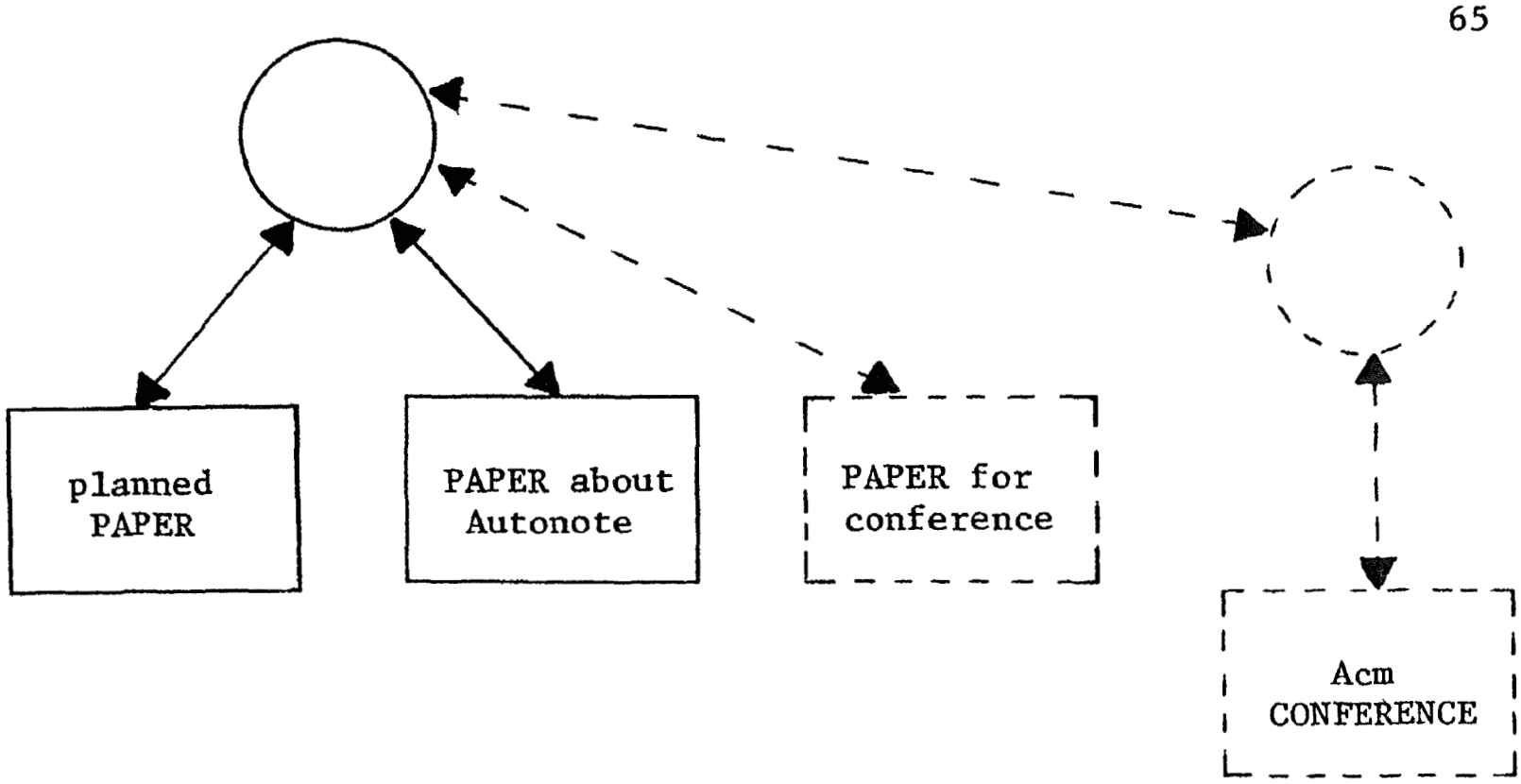
There are four stages in the search for candidate topics (see below). Stages 1 and 2 deal with the subject word of the root phrase; Stages 3 and 4 with the modifier word. In Stages 1 and 3, only those nodes directly described by phrases having the particular word in subject position are considered. In Stages 2 and 4, topic nodes with the word in modifier position are considered.

	Role of word in the description	Role of word in the network
Stage 1	Subject	Subject
Stage 2	Subject	Modifier
Stage 3	Modifier	Subject
Stage 4	Modifier	Modifier

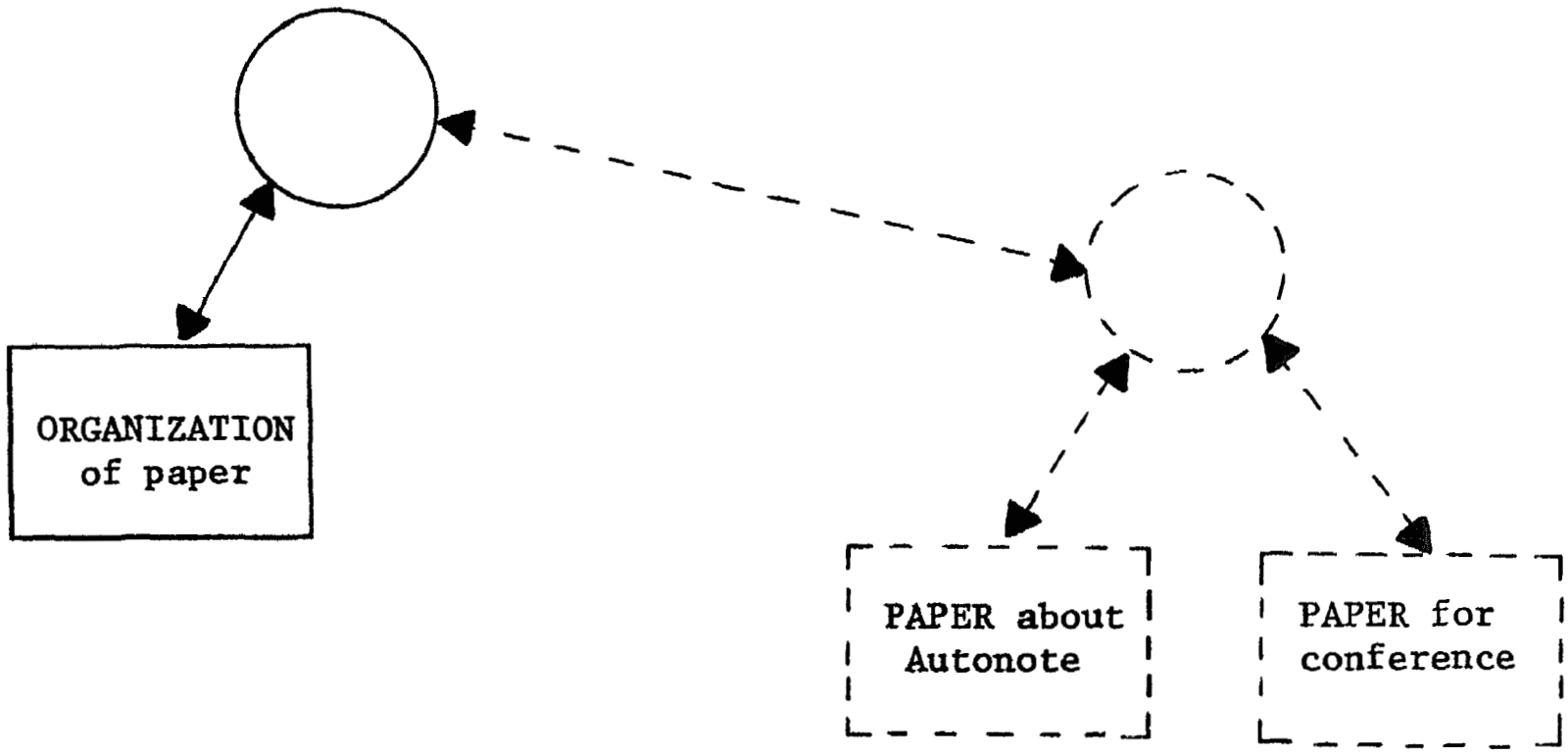
After completion of each stage, if there were any nodes generated they are passed through the recency check. If no node is distinguished, the user is presented with a list of the alternatives. The user may then choose one of the topics or reply that none is the intended referent, in which case the next stage is tried.

If a node eventually is identified by this process, the locator must note the stage it is in, since each case implies a distinct links table. Fig. 19 gives an example of each case. The state of the network before processing the description is illustrated by solid lines; the network additions by dashed lines. Note that in the Stage 2 example, the user originally described a new topic simply as ORGANIZATION OF THE PAPER and then gave a more complete description of the paper. The description of the paper was also left pending in the Stage 4 example. In fact, Case 2 and 4 can only occur in this situation since the presence of a simple phrase with paper as the subject noun would have been picked up earlier in either Stage 1 or 3.

The stage in which topic identification is made also is important when processing retrieval descriptions. Recall that a principal advantage of the referential system is that it enriches its representation of the user's topics during retrieval. Whenever a retrieval description contains simple phrases not already present in the representation of the identified topic, they are added to the representation. However, if topic identification occurs in Stage 2, note that a simple phrase will be added at one level higher than the decided topic. If processing a retrieval description, the addition would be meaningless as it will not enrich the description of the identified node. For example, if the user has previously described a paper and later calls for the retrieval

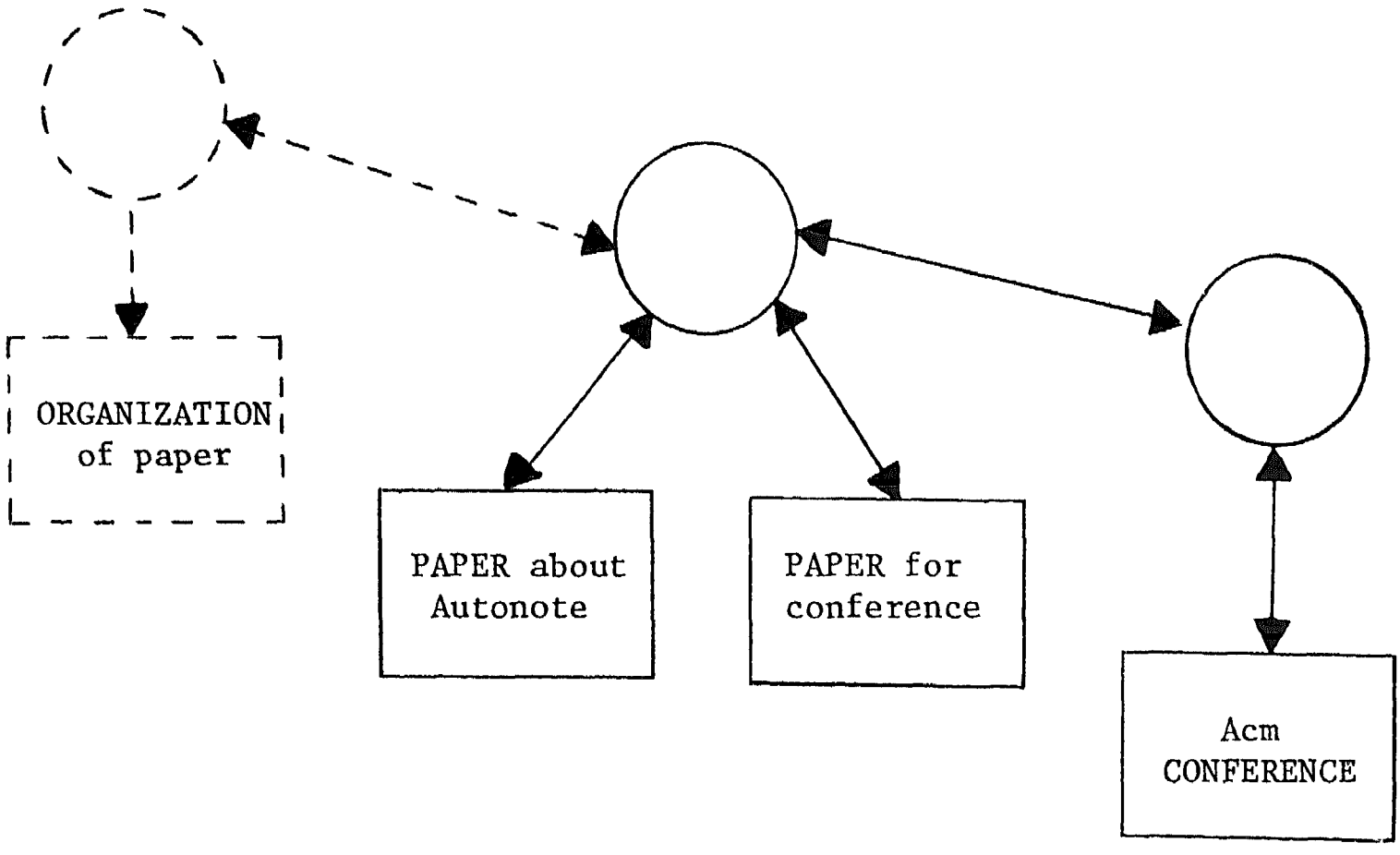


(a) Stage 1. User input: The paper for the Acm conference.

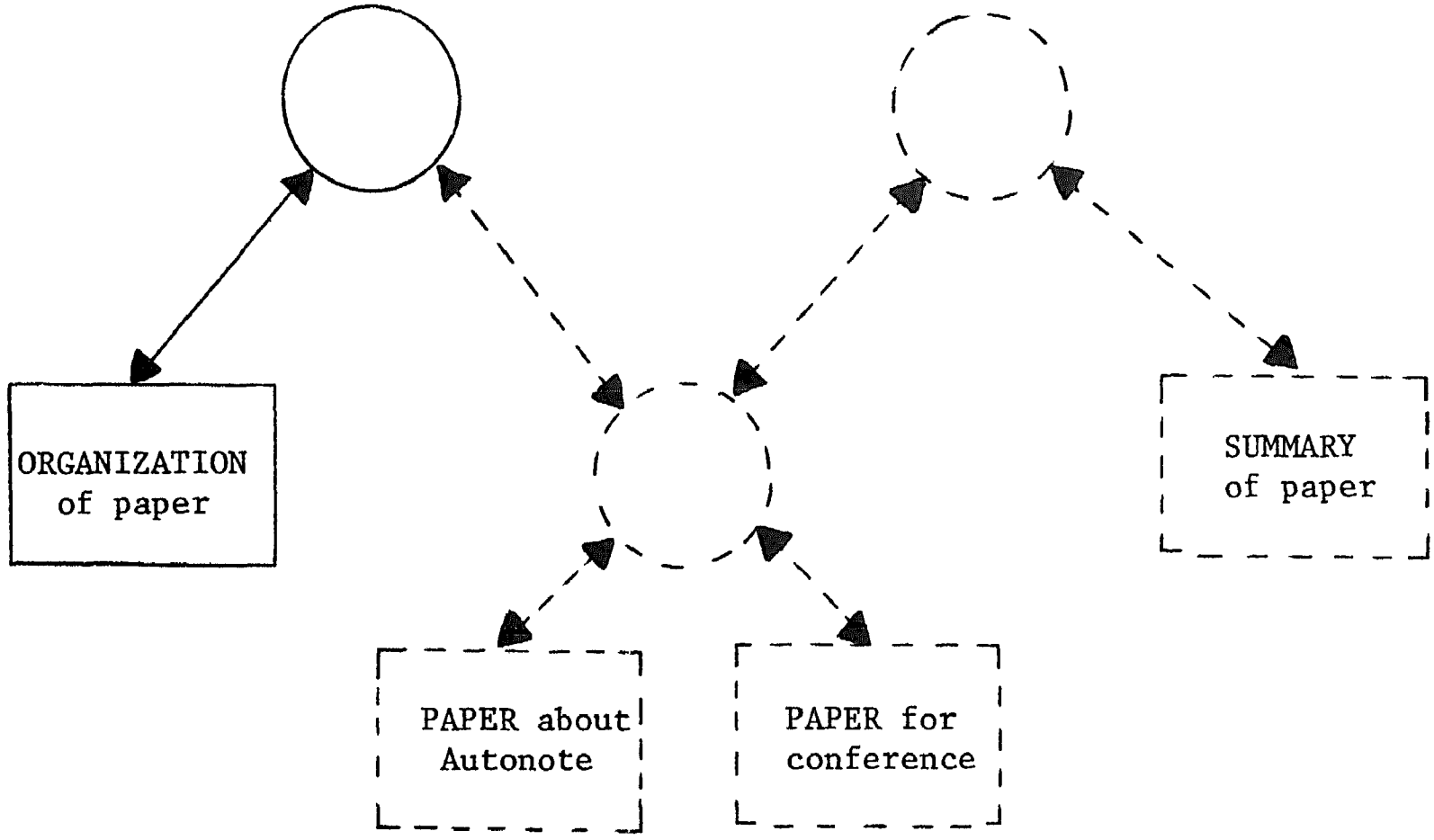


(b) Stage 2. User input: The paper about Autonote for the conference.

Fig. 19 - Network Alterations Arising from a Successful Word Search in Each of the Four Stages



(c) Stage 3. User input: The organization of the paper.



(d) Stage 4. User input: Summary of the paper about Autonote for the conference.

Fig. 19 - Continued

of MATERIAL FOR THE PAPER, the addition of a higher order node pointing down to the "paper" node is of no value in later referencing the topic. In such cases, the network locator returns the located node to the retrieval processor and suppresses the network addition. To state this more generally, retrieval descriptions are employed to augment the representation only when the additional phrases constitute co-ordinate or subordinate description of the located topic.

Although we have found the word searching procedure quite effective, its success ultimately depends upon a co-occurrance of some word in both the description and the representational network. A proposed extension to AUTONOTE2, as described in Linn (1972), would augment this procedure to include word stem and synonym processing.

The major objection to this procedure is that as the network grows larger it generates too many candidate nodes and consequently more queries to the user. To alleviate this problem, we allow the user to cancel processing of the description any time he decides the system is having difficulty relating his description to the current representation. In addition, if the user is unsure how he previously described a particular topic, a facility is provided that allows him to obtain topic descriptions from a specified region of the network. Upon noting the topic he originally intended, he may then give a more precise description.

Case III: One word descriptions. Descriptions consisting of a single noun are processed in much the same manner as non-active descriptions. The noun is treated as if it resulted from a deletion on a simple phrase. The word directory is first searched for simple phrases in which the word appears as the

subject and, if necessary, the modifier. The nodes obtained from the phrase directory then are processed as described earlier.

VI. NETWORK MEDIATED RETRIEVAL

The previous sections dealt primarily with the process of item description, that is, the process of constructing a representation from descriptions of the user's textual materials. This section discusses the AUTONOTE2 procedures that retrieve information through the representational network.

Retrieval via Descriptions

Many of the procedures described earlier for item description and representation are used in retrieval. The user initiates retrieval by giving a FIND command, supplying a description as argument. Retrieval descriptions are first passed to the parser, and are therefore subject to exactly the same constraints as item descriptions. If the description is acceptable, the resultant phrase table is passed along to the network locator which ultimately returns a node number to the FIND processor.

The FIND processor constructs a set of item numbers by extracting the textual references from the node returned by the network locator. The system then checks for upward pointers from the node, to more specifically described materials. If there are structurally related topics, the FIND processor so informs the user and asks if he would like to explore further. If so, the user is presented with descriptions of the higher order alternatives. Using the network depicted in Fig. 16, for example, consider the retrieval request FIND THE PLANNED PAPER ABOUT AUTONOTE. The network locator would determine

that node 1 is the desired referent and return that fact to the FIND processor. After storing away the item references of node 1 the system would ask:

DO YOU WANT:

A. THE ORGANIZATION OF THE PAPER

B. THE REVIEWER'S COMMENTS ON THE PAPER

The user may respond with an appropriate letter indicating which topic he desires. If the topic selected also has higher order nodes, the process is repeated until the user terminates the search.

If the node returned by the network locator has no associated item references, the system searches upward in the network for a node with text item pointers. If a node is reached with multiple upward paths, the system stops and queries the user. For example, if a user has entered only an outline and some bibliographic references for a paper he is writing, then a retrieval description that maps onto the "paper" node would elicit a query such as:

DO YOU WANT:

A. THE OUTLINE OF THE PAPER

B. BIBLIOGRAPHIC REFERENCES FOR THE PAPER

This example illustrates a distinct advantage of the referential system over simple keyword indexing. When the user's description is imprecise, AUTONOTE2 directs the user to related topic nodes with associated textual materials.

Upon termination of the search, the resultant set of textual references is stored internally. Depending upon the user's option settings, a reference count and the set of item numbers then may be displayed on the user's console. The user may PRINT those particular items he wishes to see, or he may simply RETRIEVE the entire set.

In dealing with groups of related items, network mediated retrieval has three major advantages over simple keyword-based techniques. First, the user need only make his descriptions more specific in order to "zero in" upon correspondingly specific textual materials. Second, the representational network enables the system to use the user's original description as a starting point in guiding him to structurally related topic nodes. Finally, the possibility of network exploration can help the user recall the structure of the materials represented in some portion of the network. This can be quite valuable after the user has spent an extended period working with other topics, or as the number of topics and their interconnections become large.

After processing a retrieval request, the system determines if the user's description contained any prepositional phrases or adjectives not already present in the identified topic's representation. If so, the topic description is enriched accordingly. For example, if the representation of the located node is THE PAPER FOR THE ACM CONFERENCE, and the user referred to it by the retrieval description THE PAPER FOR THE FALL CONFERENCE, the system will augment its representation to include the simple phrase FALL CONFERENCE. This is an important aspect of AUTONOTE2. Whether descriptions are employed for the purpose of characterizing text items or retrieving them, the system continually updates its representation of the user's topics. In addition, this example illustrates how the system is able to establish a limited form of phrase synonomy. There will subsequently be a node in the network directly described by both ACM CONFERENCE and FALL CONFERENCE, and any topic associated with that node may later be referenced using either or both of the two simple phrases.

Interrogating the Network

As the network grows complex, the user must be able to question the system about the current representation. This capability may help him recall the structure of some set of related topics. Or, prior to formulating a new topic description, the user may wish to examine the representational network for possible related topics. Finally, periodic perusal of the network may strengthen the user's own conceptual representation of the various topics and their interrelationships.

The DESCRIBE command retrieves topic descriptions from the representational network. It accepts a variety of arguments and first generates a set of topic nodes. Then, using the SPEAKER routine, it outputs a description of each node in the set. The various input forms include the following.

DESCRIBE ITEM <list>. Each time the description processor adds a textual reference to a node, the node number is placed in a predetermined location in the text file region of the item. The DESCRIBE processor consequently has access to the desired set of associated node numbers. For any particular text item, the user may wish to know which topics it currently is associated with. Initially, when an item is first described by the user, the actual description line is placed in the data base beneath the text. To recall how he described an item originally, the user need only request that the item be printed (omitting the text if he chooses). But the original description may have been only a terse reference, in context, to a more fully described node. Furthermore, the description of that node may have been enriched or altered subsequent to the entry of the item. To obtain a full description of each topic presently

associated with the item, regardless of how the item originally was described, the user employs DESCRIBE ITEM.

DESCRIBE CURRENT [TOPIC]. A pointer to the node most recently referenced in the representation is maintained in the node directory. In response to this command, the DESCRIBE routine simply determines the node number and displays its description. The current node number is saved between AUTONOTE2 sessions; this command is often employed at the beginning of a session to remind the user of the previous working context.

DESCRIBE TOPICS. This command causes every node in the network having associated item references to be described. Because of the voluminous output, it is most frequently employed in batch mode.

DESCRIBE <description>. When the DESCRIBE routine encounters an argument that is not in one of the special forms discussed above, it treats the input as a phrasal description. Using the parser and network locator, an attempt is made to map the input into a unique topic node. If successful, a complete description of the node is presented to the user. Thus, if the user cannot recall precisely how he described some topic, he may supply an incomplete reference to obtain the topic description in full.

The network locator functions somewhat differently when processing a description for the DESCRIBE command. If it is unable to discern a unique node using the matching procedure and context, a list of the alternatives is returned for subsequent display.

The FULLY modifier. The user may request the display of a host of related topics by employing the FULLY modifier. Specifically, the user types DESCRIBE FULLY, followed by any of the argument forms discussed above. As before, this

generates a node or set of nodes. When describing FULLY, each node is in turn expanded into a set of structurally related nodes also having associated textual references.

As an example, consider again the network in Fig. 16. The user types DESCRIBE FULLY, THE PAPER ABOUT AUTONOTE. Assuming no choice is possible in context, the description is ambiguous, and the network locator returns nodes 1 and 5.

The two nodes then are passed to a routine that displays an indented outline representing the structurally related topics reached by moving upward in the network. Each level of indentation represents a node level traversed in the network. In this example the following outline would be printed:

A. THE PLANNED PAPER FOR THE ACM CONFERENCE
THE ORGANIZATION OF THE PAPER
THE REVIEWER'S COMMENTS ON THE PAPER

B. THE FIRST PAPER ABOUT AUTONOTE
THE ABSTRACT OF THE PAPER

DESCRIBE STRUCTURES. This command functions as if FULLY was specified, displaying outlines of each topic cluster in the representational network. To accomplish this, the network is searched for nodes having no downward pointers to other nodes. Each such node corresponds to the lowest order node level in a particular cluster of related topics. When described FULLY, the effect is to reveal the structural outline of its associated cluster.

The SPEAKER Component

As we have seen, SPEAKER is invoked during many phases of AUTONOTE2's operation. The calling routine passes the SPEAKER a node number. A buffer containing a phrasal description of the node is returned. A second, optional

input parameter specifies the level of detail desired in the resultant description. The level indicator corresponds to the number of node levels in the representation to be employed in formulating the description.

The level indicator is particularly useful when the system must question the intent of a description. When querying the user during the network location process, for example, the system requests topic descriptions from the SPEAKER with the level indicator set according to the user's current preferred level of detail, as inferred from his most recent description. For example, if the user describes an item as RESULTS OF THE EXPERIMENT and the system must ask if he is referring to SMITH'S EXPERIMENT ON THE SHORT TERM MEMORY OF WHITE RATS, the resulting query would be ARE YOU REFERRING TO SMITH'S EXPERIMENT ON MEMORY?

The process of constructing a description from the network takes place in two stages. The first stage steps through the network recursively, collecting the simple phrases that directly or indirectly describe the specified node. The level indicator, if applicable, blocks the collection of simple phrases below the specified level. During this stage, the SPEAKER constructs two tables of words, one for subject nouns and another for modifiers. Each entry in the subjects table is linked to a list of adjectives for that subject, and a list (called the modification chain) of prepositional modifications of the subject noun. For example, the subjects tables entry for PAPER may have an adjective list containing PLANNED, and a modification chain consisting of (ABOUT) AUTONOTE and (FOR) CONFERENCE. Both of the lists are chained through the table of modifiers. Note that some words will appear in both the subject and modifier tables. For example, PAPER may be in the modifier table as part

of the modification chain of the word ORGANIZATION, and also in the subjects table with a modification chain of its own.

The subjects table also maintains article usage information for each of its entries. Fig. 20 illustrates the subject and modifier tables constructed from a typical topic node.

Subject Noun	Article	Level	Adjective Chain	Modification Chain
organization	the	1	...	(2)
paper	the	3	(1)	(3)
conference	the	3	(4)	...

(a) Subjects table

Modifier Word	Preposition	Chain Link
(1) planned	adj	...
(2) paper	of	...
(3) autonote	about	(5)
(4) acm	adj	...
(5) conference	for	...

(b) Modifier table

Fig. 20 - SPEAKER tables generated from the network representation of THE ORGANIZATION OF THE PLANNED PAPER ABOUT AUTONOTE FOR THE ACM CONFERENCE.

The second stage is carried out by a recursive algorithm that operates on the two tables to construct the phrasal description. The process begins with the first word in the subjects table, in this case ORGANIZATION. If an article applies, it is added to the description buffer. Next, the adjective chain is

traversed adding each adjective in turn to the buffer. In this case there are no adjectives so the current subject word (ORGANIZATION) is added to the buffer and the system continues with the modification chain. This leads to the second entry in the modifier table, (OF) PAPER. The preposition is then added to the buffer yielding THE ORGANIZATION OF. Next, a check is made in the subjects table to determine if the current modifier word (PAPER) is further described. Since there is an entry for PAPER, the current position in the modification chain for ORGANIZATION is placed on a push down stack (the goal stack) and the algorithm recurses on the word paper. After adding the article, the adjective (PLANNED), and the subject word (PAPER), the description buffer contains THE ORGANIZATION OF THE PLANNED PAPER. The system now begins processing the modification chain of PAPER. The first piece of the chain adds ABOUT AUTONOTE to the buffer. Note that there was no recursion on AUTONOTE because that word does not have a subjects table entry. The pointer to the next piece of the modification chain, (FOR) CONFERENCE, is then picked up from the link field of the AUTONOTE entry. After adding the preposition (FOR), the algorithm recurses on CONFERENCE, adding THE, ACM, and CONFERENCE in turn to the buffer. The goal stack is then popped in search of remaining modification chain pointers. The first "pop" restores the PAPER modification chain. Since there is no additional modification of the paper, the goal stack is popped again to restore the ORGANIZATION chain. We are at the end of this chain also, and thus the process terminates with the description buffer reading: THE ORGANIATION OF THE PLANNED PAPER ABOUT AUTONOTE FOR THE ACM CONFERNECE.

SPEAKER heuristics. The addition of phrases to a topic in many cases could reduce the readability of its SPEAKER-generated description. For

example, suppose a topic is first defined as SAMPLE DESCRIPTIONS FOR USE IN THE ACM PRESENTATION, and later is referred to as SAMPLE DESCRIPTIONS FOR USE IN THE NSF PROPOSAL. Given only the algorithm just presented, the SPEAKER generated description would be SAMPLE DESCRIPTIONS FOR USE IN THE ACM PRESENTATION IN THE NSF PROPOSAL. To avoid such unreadable descriptions, whenever the modification chain for a subject noun contains two or more prepositional phrases headed by the same preposition, the SPEAKER sets off each phrase after the first with parentheses. The above example then becomes SAMPLE DESCRIPTIONS FOR USE IN THE ACM PRESENTATION (AND THE NSF PROPOSAL). Note that a description such as COMMENTS ON SMITH'S ARTICLE ON CLUSTERING is not processed in this manner since (ON) ARTICLE is in the modification chain of COMMENTS, while (ON) CLUSTERING is in the modification chain of the work ARTICLE. Note also that although parenthetical phrases are excluded from topic descriptions generated for the purpose of interrogating the user, when the user requests a description of a topic via the DESCRIBE command, the complete description is provided.

Simplification of list processing. It may be added here that our decision to maintain the representational network in disk file storage has greatly simplified the list processing in recursive algorithms such as the SPEAKER. The network can be envisioned as a complex list structure where the links are simply line file numbers. To illustrate this point, consider the recursive collection of simple phrases carried out in the first step of the SPEAKER. The main body of the routine collects the simple phrases that directly describe a node. If the node processed has downward links to subordinate nodes, they are placed on a push down stack. Next the stack is popped and the routine is

called recursively to operate on a new node number. Thus all the concomitant problems of storage management that are normally present in list processing systems are avoided. Recursive deletion, discussed in the next section, similarly is simplified. To delete a portion of the list structure requires only the removal of a line from a directory file. Thus the process of "garbage collection" is both automatic and transparent to AUTONOTE2.

VII. NETWORK MODIFICATION

Procedures for modifying the representational network are required for several reasons. Should the system incorrectly parse a description, the user's ability to reference the associated topic will be impaired. The user may wish to alter the description of a topic to (a) make it more precise, (b) insure that it is not confused with similarly described topics, or (c) enable a topic to be referenced in more than one way. After initially describing a text item, the user may discover that the item should also be associated with other topics in the representation. Alternatively, he may decide that a text item should be dissociated from some topic. The user may wish to delete an obsolete topic from the representation altogether, or replace a description in its entirety by a more suitable one while maintaining the same list of associated textual references. Finally, when dealing with a group of structurally related topics, the user may wish to delete an entire structure, or certain components of a structure, from the network.

We cannot expect a typical user to think in terms of list structures, nodes, linkages, etc. Thus we sought to provide a command language and

feedback more or less independent of the internal data structures that implement the representation. In addition, care was taken to avoid the possibility of accidental damage to the representation stemming from misunderstanding or misapplication of the modification procedures.

The resultant processor includes procedures for removing or adding item references to a topic, deleting topics, adding or removing simple phrases from the description of a topic, etc. Rather than require the user to identify the particular topic to be altered each time a modification is to be performed, primitives are implemented as local commands to a generalized modification processor.

The modification processor is invoked by issuing a CHANGE command which accepts a phrasal description as its argument. A node in the network is established as the current identified topic. The processor then prompts the user for modification instructions. After all modifications are completed, the user types DONE and control is returned to the regular command monitor. The CHANGE command also may be issued while in modification mode, thereby changing the current topic. Each of the local commands is discussed separately below, using the hypothetical representation depicted in Fig. 21 for illustration.

Adding References and Phrases to the Network

The ADD command associates additional text references with the current topic, and adds simple phrases to the topic's description. To add item references, the user types ADD ITEM[S] followed by a list of item numbers. This procedure is quite useful if the user has a large set of items that pertain to a particular topic. He simply identifies the topic and adds the list of references.

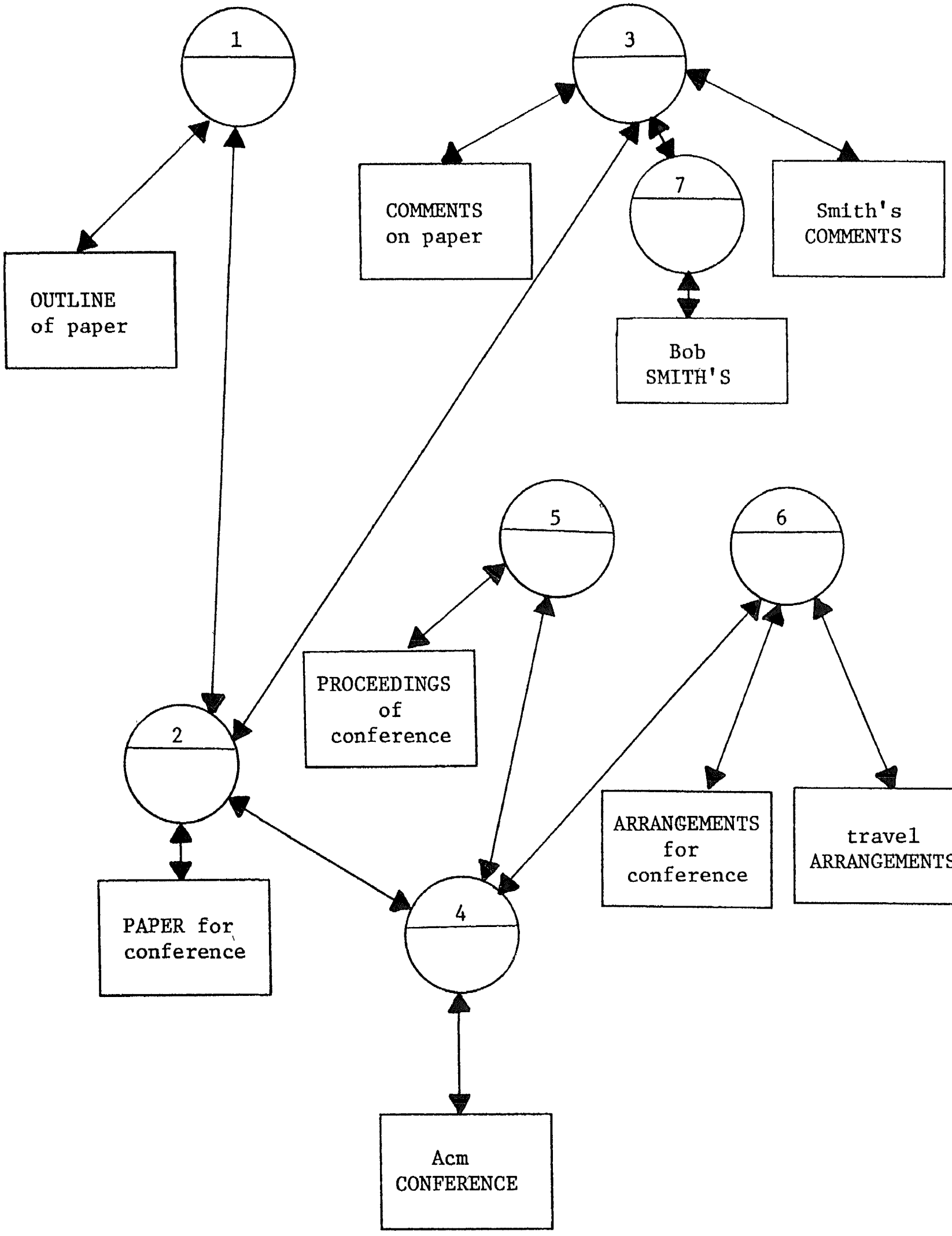


Fig. 21 - Sample Representation for Discussion of Network Modification

If the supplied argument is a phrase, it is added to the current node. For example, if the current topic is THE PAPER FOR THE ACM CONFERENCE, the command ADD PAPER ABOUT AUTONOTE causes the prepositional phrase ABOUT AUTONOTE to become a part of the topic description. Adjectives may also be added to a description (example: ADD SMITH'S PAPER). If only a single word occurs as the argument, it is assumed to be an adjective which is to modify the current subject noun.

Moving through the Network

The MOVE command allows the user to change the current node pointer from its present position to structurally proximate topics without having to enter a description. For example, if currently located at the "paper" node, the command MOVE DOWN causes the ACM CONFERENCE to become the current topic. If the current topic is the ACM CONFERENCE, MOVE UP will produce three higher order topics. Each is saved, and the leftmost node becomes the current topic. Subsequently, the user may MOVE LEFT or RIGHT to the other topics. After a successful move, a brief description of the new topic is displayed.

The Caching Facility

The CACHE command stores item references for subsequent use. If the command is given with no argument, the set of text references associated with the current topic is added to an internal cache. The caching facility may be used to manipulate large sets of item references, for example, in transfering all item references from one topic to another. This may be accomplished by identifying the first topic and issuing a CACHE command. After identifying the

second topic, the command ADD CACHE causes the set of cached items to be merged with those of the new topic.

Retrieval Commands

The retrieval commands of the network modification processor are analogous to their counterparts in AUTONOTE. LIST outputs a list of the item numbers associated with the current topic. LIST CACHE displays the numbers of the items in the cache. PRINT outputs selected text items. All items associated with the current topic, or those in the cache, will be printed in response to RETRIEVE and RETRIEVE CACHE, respectively.

By employing the IDENTIFY and MOVE commands, the user may explore the representation, LISTing the associated references for each topic. During the exploration, the CACHE command can be used to store selected references for later retrieval, or the user may choose to PRINT or RETRIEVE pertinent references as he goes. These procedures allow the retrieval set to be shaped interactively, and more selectively than is possible with the FIND command discussed earlier.

Removing References and Phrases from the Network

The REMOVE command accepts the same argument forms as the ADD command and simply performs the inverse operations. The argument ALL also is recognized, causing all item references to be removed from the current topic.

Topic Deletion

DELETE may be employed to remove obsolete topics from the representation, or as the first step in replacing a topic description with a more appropriate one. CREATE then may be used to enter the replacement topic into the network.

The major problem in the design of the topic deletion algorithm can be stated as follows. When deleting a topic, under what circumstances are structurally related nodes to be deleted as well? Consider the following cases. In the hypothetical network, a request to delete the "paper" node involves a decision about deleting (a) the outline of the paper, and (b) Smith's comments on the paper. Note that if the paper node were deleted and the two higher order nodes were not, the higher order nodes would no longer be structurally related. In addition, their descriptions will still contain the word "paper," but which paper no longer is specified. For these reasons, we concluded that a topic deletion should also entail the deletion of more specifically described topics. In many cases, this convention is an advantage, since the user can delete an entire structure by identifying and deleting a single lower order node.

The considerations involved in dealing with lower order nodes are a bit more complex. Some lower order nodes serve only to augment the description of superior nodes. In THE EXPERIMENT ON BLIND RATS, there will be a subordinate node directly described as BLIND RATS. In this case, deletion of the EXPERIMENT node should include deletion of the subordinate node. On the other hand, deletion of the "paper" node in the previous example does not imply deletion of the ACM CONFERENCE topic. The ACM CONFERENCE node also plays a role in other topics.

In the instances we have examined, the distinction between the two cases seems to be that "unimportant" nodes have neither textual references nor upward pointers to other topics. The deletion process employs a heuristic based

upon this observation. When a subordinate node is deemed unimportant, it is deleted; otherwise, the user is asked to confirm its deletion.

The deletion algorithm. First, a list of all nodes to be deleted is constructed. After the list is complete, the user is presented with a brief summary of the deletions to be made and is prompted for confirmation.

The algorithm for constructing the deletions list is recursive. Two pushdown stacks are employed: one for storing upward node paths yet to be explored, and one for saving downward paths. The procedure is most easily explained with an example. Suppose the user identifies the "paper" node and requests its deletion. The algorithm begins with node 2, first pushing down any upward pointers along with the node number (2) that was being processed when the pointers were added to the "up stack." In this case the pairs (1,2) and (3,2) are pushed down. Next, the downward pointers are placed on the "down stack" along with the current node number. The current node number is then added to the deletions list. The current state of the pushdown stacks and the deletions list is now:

UP STACK	DOWN STACK	DELETIONS LIST
(1,2) (3,2)	(4,2)	2

The down stack is then popped and node 4 is established as the next node to be examined. After setting a flag indicating that we have just moved down a level in the network, the algorithm recurses on node 4. The system detects three upward pointers (to nodes 2, 5, and 6). It should now be apparent why we save the fact that node 4 was reached by moving down from node 2. When

placing a node's upward pointers on the up stack, the node that led down to the current node must be ignored.

Upon noting that node 4 has upward pointers in addition to node 2, the system checks to see if it has just moved down. In this case it has; consequently, node 4 is deemed "important" and the system asks DO YOU WANT THE ACM CONFERENCE DELETED? Assume the reply is NO. Since the ACM CONFERENCE node will remain, the system records that the linkage between nodes 2 and 4 must be severed. The algorithm then recurses without adding node 4 to the deletions list. Note also that the upward pointers from node 4 to nodes 5 and 6 are not placed on the up stack. The current state of the process is now:

UP STACK	DOWN STACK	DELETIONS LIST
(1,2) (3,2)	empty	2

The attempt to pop the down stack fails, so the up stack is popped and a flag set to indicate upward movement (to node 3). Node 3 has no upward pointers but it has two downward pointers, to nodes 2 and 7. Node 2 is ignored because it led up to node 3. Node 7 is placed on the down stack and node 3 is added to the deletions list yielding:

UP STACK	DOWN STACK	DELETIONS LIST
(1,2)	(7,3)	2 3

The down stack is popped and the algorithm recurses on node 7. Node 7 has neither text references nor upward pointers (besides node 3). Consequently, it is deemed unimportant and is added to the deletions list. We now have:

UP STACK	DOWN STACK	DELETIONS LIST
(1,2)	empty	2 3 7

The down stack is empty so the up stack is popped and the system recurses on node 1. The node has no new upward or downward pointers so it is added to the deletions list. Both stacks are now empty and the algorithm terminates having collected nodes 2, 3, 7, and 1 for deletion.

Carrying out the deletion involves several steps. First, any linkages between those nodes that are to be deleted and those that will remain are severed. These changes will have been detected and recorded during the recursive collection process. Next the system executes a REMOVE ALL for each node on the deletions list so that no associated text reference points to a non-existent topic. Then the system removes all pointers from simple phrases to the obsolete nodes. Finally, each deleted node is removed from the node directory file.

Creating New Topic Representations

CREATE enables the user to define a new topic for the representation. The command takes as argument a description which is processed in the normal way, except that no item references are associated with the topic. The new topic becomes the current node. ADD is used to associate any appropriate text references. During topic deletion, the system adds to the cache all text references previously associated with deleted topics. Consequently, ADD CACHE will now associate those items with the new description.

When processing a CREATE description, the network locator attempts to associate the description with an existing topic, for two reasons. If the description is to be a new topic but the network locator confuses it with another one, the user may want to alter the description. Second, this permits use of the CREATE command in adding to an existing description.

VIII. A CASE STUDY OF SYSTEM PERFORMANCE

The Inapplicability of Recall and Precision

The most widely accepted methods for retrieval system evaluation are based upon recall and precision measures. As applied to the results of retrieval queries, precision is defined as the proportion of retrieved material that is deemed relevant to a query; recall is the ratio of relevant documents retrieved to the total relevant in the data base. But recall and precision cannot meaningfully be applied to the evaluation of AUTONOTE2. The AUTONOTE2 user describes each piece of textual material himself. Even within a large personal data base, the user will certainly recollect some of his topics and the key words and phrases that define them. Furthermore, subject to the user's own limitations in describing his materials, the topic framework of AUTONOTE2 implies "perfect" precision and recall once a particular topic is identified during retrieval.

A principal motivation for the AUTONOTE2 system was the desire to overcome the disadvantages of keyword indexing techniques, which force the user to translate ideas and concepts pertinent to a given document into discrete content indicators. In developing AUTONOTE2 we have sought to provide mechanisms for

defining and efficiently referencing these concepts directly. An evaluation of AUTONOTE2 should therefore provide some comparisons of keyword indexing vs. indexing by topic. To achieve a direct comparison, protocols of both types of indexing activity with a common data base are required.

The Sauvain Data Base

The original AUTONOTE system was employed in a study (Sauvain, 1970) aimed at uncovering structural communication problems within a keyword-based system. The resulting data base is related primarily to Sauvain's dissertation research. It includes reading notes, bibliographic references, research ideas, expository material, and so on. The collection brings together a broad range of topics and ideas touching upon various aspects of computer science, information retrieval, man-machine interaction, and psychology.

Copies of the item texts, the originally assigned keywords, and protocols of Sauvain's activities during data base indexing, organization, and retrieval were acquired. We then proceeded to re-index the collection with AUTONOTE2 topic descriptions. Each of the roughly 400 items in the data base was viewed and described in a sequential fashion; that is, there was no look-ahead or pre-planning of topic phrasings to facilitate network structuring. Protocols were collected of all interaction with the system and the state of the network was recorded at periodic intervals. (For details, see Linn, 1972).

Results

For brevity, AUTONOTE2 reports of parsing assumptions are excluded.

However, system responses that elicit a user reply are shown to provide a feeling for user interaction under AUTONOTE2.

Indexing activity. The AUTONOTE2 protocols show a high degree of terse, efficient referencing of previously defined topics. The communicative efficiency was especially great in instances where several consecutive items were entered on a common topic. This situation frequently occurred when entering a set of reading notes on a particular paper or collection of papers. Typically, the first item in such a set of entries was assigned to one or more new topics. In describing the subsequent items, references to these topics often were conveyed by a single word or phrase, or by a null description (a description line consisting of only a slash is treated as a reference to the topic just mentioned).

To illustrate, consider the materials dealing with various aspects of artificial intelligence. A total of 17 of these items contained notes taken at a 1968 conference at Case Western Reserve University. Of the AUTONOTE2 descriptions supplied for these items, three mention only the word CONFERENCE; five include the subphrase 1968 CONFERENCE; two include CWRU CONFERENCE; and seven make no explicit reference to the conference at all. Each of the items, however, was associated with a topic node linked in some way to the "conference" node. Furthermore, though none of the descriptions contain the words ARTIFICIAL or INTELLIGENCE, each of the associated items can be accessed in the network through the "artificial intelligence" node.

In the AUTONOTE protocol for these materials, there was frequent use of descriptor abbreviations and other idiosyncratic tags (CWRUAICONF, AI, COGPSY,

etc.). These suggest a strong desire to eliminate repeated entry of lengthy descriptors and phrases. The major drawback of this strategy, however, is that abbreviations (especially the more uncommon ones) are not as easily remembered as the words they represent. In addition, once an abbreviation has been used, the user must remember that he has done so in order to maintain consistent indexing. In contrast, there was little motivation for descriptor abbreviations under the AUTONOTE2 system. Once a lengthy phrase had been defined in the network there was generally no need to reference it again with a full description.

The Sauvain study identified a clear need for mechanisms to assist the user in maintaining consistent indexing.

> The second type of need (how a descriptor has been used) frequently occurs when a text item is being entered. The user has some ideas for candidate descriptors, suspects that there has been prior usage of these words, and needs a way to check the prior usage to keep his indexing consistent. He also may want to look at prior usage contexts to get ideas about other descriptors to use, or to weed out candidates that look too general.

The topical view of the data base under AUTONOTE2 eliminates part of this problem. When describing a new topic the AUTONOTE2 user need not be as concerned about prior word usage in other contexts. The representational network provides a means for discriminating among the various topics in which a particular word occurs.

If, on the other hand, the user suspects that the item at hand is somehow related to a previously existing topic, there is an analogous need to interrogate the representational network for candidate topics. This capability is provided by the DESCRIBE command. There were, in fact, numerous instances in

the AUTONOTE2 protocols of network interrogation prior to entering descriptions. An example is given in Fig. 22. In response to the user's description

```
USER:  OPEN

       #100

USER:  /SALTON'S CCS COLLOQUIUM ON EVALUATION

       NEW TOPIC ASSUMED

USER:  RELOCATE

       WHICH DO YOU MEAN:

       A.  THE COMPUTER EVALUATION OF INDEXING (AND TEXT PROCESSING)

       B.  EVALUATION OF CRT DISPLAY USAGE

USER:  A
```

Fig. 22 - Network interrogation during description entry

the system indicates that no association will be made with a prior topic. The user recalls talking about the evaluation of automatic indexing techniques earlier so he requests the system to search further by entering a RELOCATE command. Two candidates are generated, one of which is the desired referent.

Under the keyword system, searching for candidate descriptors and usage contexts was much more tedious. Typically, a RETRIEVE command was issued calling for the display of all keyword lines of items indexed by a particular term or logical combination of terms. In some cases a large number of items were accessed necessitating time consuming perusal of the data base.

The discussion thus far should convey some feeling for the degree of communicative efficiency achievable with AUTONOTE2. To provide a more precise

indication of this aspect of system performance we calculated the ratio of content words conveyed to content words entered for three samples of data base items (articles and prepositions were excluded). The average number of AUTONOTE2 words entered and conveyed were compared with the average number of AUTONOTE keywords assigned within each sample (under the keyword system this ratio will always equal one). The three samples taken were (1) a random sample of 50 items, (2) 41 sequential items, and (3) all items dealing with some aspect of artificial intelligence.

The results of this tabulation are summarized in Fig. 23. In all three samples more than three content words were conveyed for every two entered, on the average. The conveyed-entered ratio was lowest for the random sample and highest for the artificial intelligence items. This is because most of the items dealing with artificial intelligence were entered with a rich global context of topic nodes defined. The sequential items, on the other hand, were related to several smaller, more localized topic structures. Consequently, many more of these items were described in full. The random sample lacked a consistent contextual framework, and consequently had the least communicative efficiency on the average.

Retrieval activity. We have seen that retrieval activity is an essential part of the indexing and organizational processes. The protocols show a frequent need to search for related material and item numbers in the keyword system and a corresponding need for network interrogation prior to description entry under AUTONOTE2. However, the AUTONOTE2 topic framework eliminates much of the text file perusal so common in the AUTONOTE protocols. Each topic description typically provides a clear indication of the content of its associated

	Random Sample	Sequential Sample	Artificial Intelligence Sample
No. of items in sample	50	41	30
No. of keywords originally assigned	270	251	155
Avg. No. of keywords per item	5.4	6.1	5.1
No. of content words entered	256	233	109
Avg. No. of content words entered per item	5.1	5.7	3.6
No. of content words conveyed	388	396	235
Avg. No. of content words conveyed per item	7.9	9.7	7.8
No. of words conveyed per word entered	1.5	1.7	2.2

Fig. 23 - A comparison of entered and conveyed content words for three samples of data base items

items. Consequently, there was very little need to examine the text file prior to describing new items or relating them to others in the data base. A perusal of candidate topics generated by the DESCRIBE command was sufficient in most cases.

The most outstanding improvement during retrieval activity occurred in instances of very general queries. Under the keyword system a request for all items pertinent to, say, ARTIFICIAL INTELLIGENCE or PROBLEM SOLVING will access a large set of items. Queries of this kind were employed to peruse large segments of the data base relevant to a general topic area--for example, in search of item candidates for a particular grouping. The important difference between the two systems in this situation is that AUTONOTE gives the user no indication of the subtopics within the general topic area. The AUTONOTE user essentially has two alternatives: he may display each of the accessed items (optionally suppressing text); or he may further restrict the set of items with an addi-

additional set of descriptors. Both options have notable drawbacks. The first entails time-consuming perusal of the data base. The second raises a more significant problem. Which descriptors should be used to restrict the size of the accessed set of items? Some descriptors may restrict the set too greatly, eliminating relevant material; others may discriminate very little or not at all. In the absence of system feedback, this discrimination process places a major burden on the user's memory.

The AUTONOTE2 system, on the other hand, provides the user with very meaningful feedback in response to general queries. Consider, for example, the retrieval protocol presented in Fig. 24. At each level in the representational network, the user is given an opportunity to choose among several subtopics. This example very effectively demonstrates a marked improvement over keyword indexing--the ability to discriminate among subsets of material indexed under a common set of general descriptors.

Conclusion

An analysis of man-machine dialogs collected during the description of a realistically diverse collection of textual materials has shown the communicative ease and efficiency and the descriptive power attainable under the referential system. The results indicate that the referential mechanisms developed in this study constitute a viable alternative to keyword indexing techniques as applied to personal information systems. The referential approach offers four primary contributions toward the improvement of man-machine communication; each corresponds to a particular kind of facilitation during storage and retrieval activity.

```
USER:  FIND ARTIFICIAL INTELLIGENCE
       DO YOU WANT:
       A.  THE 1968 ARTIFICIAL INTELLIGENCE CONFERENCE AT CWRU
       B.  THE INFLUENCE OF ARTIFICIAL INTELLIGENCE ON COGNITIVE
           PSYCHOLOGY
       C.  THE GENERAL APPROACH TO ARTIFICIAL INTELLIGENCE
       D.  ARTIFICIAL INTELLIGENCE RESEARCH AT MIT
       E.  RELATION OF ARTIFICIAL INTELLIGENCE TO PSYCHOLOGY
USER:  A
       1 ITEM SAVED.  WANT TO EXPLORE?
USER:  YES
       DO YOU WANT:
       A.  AMAREL'S TALK AT THE CONFERENCE ON THE REPRESENTATION OF A
           PROBLEM SOLVING SYSTEM
       B.  ROBINSON'S TALK AT THE CONFERENCE ON THEOREM PROVING SYSTEMS
       C.  LIN'S PAPER AT THE CONFERENCE ON THE HEURISTIC SOLUTION OF
           LARGE COMBINATORIAL PROBLEMS
       D.  BANERJI'S OVERVIEW OF GAME PLAYING PROGRAMS AT THE CONFERENCE
       E.  SIMMONS REVIEW OF QUESTION ANSWERING SYSTEMS AT THE CONFERENCE
       F.  OTHER TALKS AT THE CONFERENCE
       G.  SLAGLE'S DISCUSSION AT THE CONFERENCE ON HEURISTIC SEARCH
           PROGRAMS
       H.  FIKES PRESENTATION AT THE CONFERENCE OF AN ALGOL-LIKE
           LANGUAGE FOR PROBLEM SOLVING PROCEDURES
       I.  FEIGENBAUM'S DISCUSSION AT THE CONFERENCE OF THE DENDRAL
           PROJECT
       J.  BANERJI'S REVIEW OF THE CONFERENCE IN THE SIGART NEWSLETTER
```

Fig. 24 - Topic descrimination during retrieval activity

First, the concept of a representational network provides the user with a particularly natural view of both the content and organization of his data base. In essence, the user explicitly defines the important concepts and topics within his own area of interest; he specified structural relationships among concepts and, in general, manipulates these informational objects during all phases of problem solving activity. This topical view of the data base is, in a very real sense, more "meaningful" to the user than the artificial view inherent in keyword indexing systems.

Second, the developed techniques provide a unified treatment for both indexing and organizational activity. The communication of structural

associations is achieved through exactly the same descriptive mechanisms used in categorizing material. In effect, the topic serves as the focal point in all aspects of communication with the system.

Third, retrieval capability is considerably enhanced by the discriminatory power of the referential system. The representational network provides an effective means for distinguishing among the many topics that may be partially indexed under a common set of words. As noted earlier, this discriminatory power is especially useful in providing meaningful user feedback in response to general retrieval queries. Further, since each topic description serves to identify the content of its associated items, the representational network may be used as a retrieval intermediary. That is, the user can essentially engage in retrieval activity by utilizing mechanisms for exploring the topic structures in the network. This aspect of the system greatly reduces the need for lengthy perusal of document texts in search of desired materials.

Finally, the utilization of the structural context provided by the network approach makes it possible for the user to describe, organize, and retrieve materials with considerable communicative efficiency. This is a fundamental aspect of the system design--to provide a framework for interpreting terse, efficient, sometimes ambiguous references to the topics in the information universe.

In light of the increasing availability of on-line computing facilities today, it seems reasonable to expect that personalized retrieval systems will play an expanding role in the computer support of individual research activity. It is hoped that this study will suggest new directions for the design of such systems.

REFERENCES

Dosert, B. H., & Thompson, F. B. How features resolve syntactic ambiguity. In J. Minker & S. Rosenfield (Eds.), Proceedings of Symposium of Information Storage and Retrieval, University of Maryland, April, 1971.

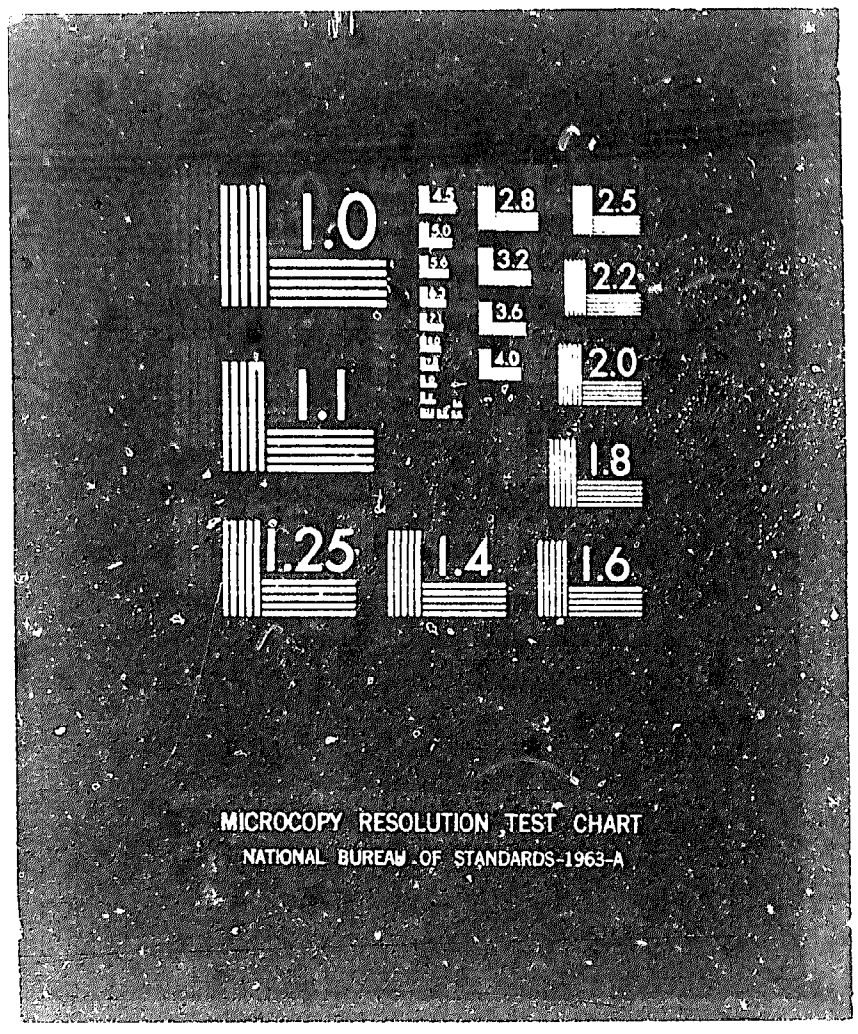
Linn, W. E. Jr. Man-machine referential communication in a personal information retrieval system. (Doctoral dissertation, The University of Michigan) Ann Arbor, Michigan: University Microfilms, 1972. No. 73-6867.

Reitman, W. Cognition and thought. New York: Wiley, 1965.

Reitman, W., Roberts, R. B., Sauvain, R. W., Wheeler, D. D., & Linn, W. AUTONOTE: A personal information storage and retrieval system. Proceedings of the 24th National Conference of the Association for Computing Machinery. New York: Association for Computing Machinery, 1969. Pp. 67-76.

Sauvain, R. W. Structural communication in a personal information storage and retrieval system. (Doctoral dissertation, The University of Michigan) Ann Arbor, Michigan: University Microfilms, 1970. No. 70-21782.

END


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A