MIT/LCS/TR-155

MECHANIZATION OF TEMPORAL KNOWLEDGE

Kenneth M. Kahn

September 1975

This blank page was inserted to preserve pagination.

MECHANIZATION OF TEMPORAL KNOWLEDGE

Kenneth M. Kahn

September 1975

This research was supported by the Department of Health, Education, and Welfare (Public Health Service) under Grant Number 1 R01 MB 00107-01.

Massachusetts Institute of Technology

Project MAC

Cambridge **Massachusetts 02139**

ABSTRACT

The design and implementation of a collection of computer programs knowledgable about time "in general", called the time specialist, is described. The thesis that this time specialist can be placed in the service of larger more general problem solvers is demonstrated for two examples, medical diagnosis and the understanding of a time-travel story.

The time specialist accepts a wide variety of facts and questions relating to the time of events. These include dates, vague terms such as "a few weeks ago", and two kinds of intervals. The "fuzziness" or inexactness of the time of events is handled differently for each of the representation types. The time specialist contains routines that compare, combine and translate between these various representation types.

The time specialist attempts to maintain a consistent data base. As facts are entered into the system, they are checked for their consistency with previously accepted facts. The time specialist corrects, to the extent possible, the data base after previously believed facts are doubted.

Incoming facts are organized by the time specialist to facilitate inference. Events are organized by their dates, by their position in a sequence of events, and by their relation to other very common events such as "now" and "birth". Routines that create, maintain, correct and use these organizational structures are described. The importance of organizing principles for facts is indicated.

Thesis Supervisor: G. Anthony Gorry

ACKNOWLEDGEMENTS

This research was carried out in the environment provided by the Clinical Decision Making Group at Project MAC, the people of whom I am very grateful to for their help, encouragement and criticism. I would also like to express my graditude to others in Project MAC and the Artificial Intelligence Laboratory who have helped me in various ways. I would like to give special thanks to Tony Gorry without whose advice and guidance much less would have been accomplished. Others who I am very grateful to are Michael Genesereth, Rand Krumland, Byron Davies, Peter Miller, Dave Taenzer, Paul Goldenberg, Peter Szolovits, Bill Martin, Johan de Kleer, Andee Rubin, Juylie Minsky, Laurie Wadsworth, Carl Hewitt, Dr. William Schwartz, and Margaret Stern.

Table of Contents

Abstract 2

Acknowledgements 3

Table of Contents 4

Table of Illustrations 5

Chapter I. Introduction and Overview 6

Chapter II. Representation of Temporal References 45

Chapter III. The Structure of the Time Specialist 64

Chapter IV. The Application of the Time Specialist to Medicine 100

Chapter V. Understanding Time-Travel Stories 115

Chapter VI. Extensions 124

Appendix A. Fuzzy Arithmetic 130

Appendix B. Human Time Specialists 138

Appendix C. The Interface Mechanism 143

Appendix D. Generation of English 148

Bibliography 150

TABLE OF ILLUSTRATIONS

1. The Users of the Time Specialist 10

2. Organizational Structures for the Facts in the Scenario ..... 15 and 16

3. A Schematic Representation of the Time Specialist 18

4. Organizations of Facts and Functions 68

5. The Time Course of AGN 102

6. The History of the Patient in the Scenario 109

7. The "Time Line" of "All you Zombies" 119

TABLE OF CONTENTS CHAPTER I

I. Introduction 8

II. A Session with the Time Specialist 12

III. The Representation of Temporal References 19

IV. Organizing Temporal Specifications 21

A. Organizing by Dates 21

B. Organizing by "Special Reference Events" 22

C. Organizing by "Before-after Chains" 22

D. Analyzing an Incoming Statement 23

V. Answering Questions 24

A. Equivalent Temporal Specifications 24

B. Deduction in Question Answering 25

1. Using Dates 25

2. Using Relative Expressions 26

3. Using Before-after Chains 27

4. The Last Resort 28

5. Selecting a Method for Application 29

VI. Maintaining the Consistency of the Data Base 30

VII. Correcting Errors 32

VIII. The Relation between the Time Specialist's Language and English 34

IX. Comments About Philosophical Questions 37

X. The Time Specialist Applied to Medicine 38

A. The Time Specialist's Role in Hypothesis Matching 38

B. How the Time Specialist was Applied to Diagnosis 39

1. Some Problems 39

2. The Hypothesis Matcher 40

XI. Understanding a Time-travel Story 41

XII. Other Applications of the Time Specialist 43

A. Reminder System 43

B. Date-Finding System 43

C. Trip Planner 44

Section 1: Introduction

In a wide variety of situations, problem solving requires a rather extensive and detailed knowledge of time.[1] To deal effectively with everyday life, a person must understand how events are related to one another in time and how plausible deductions can be drawn from the temporal characteristics of these occurences. The paramount importance that such an understanding plays in human problem solving argues forcefully that no computer program can be "intelligent," if it lacks a sophisticated temporal sense. In this report, I will consider the problems of equipping a computer with such an understanding.

Despite the importance of an understanding of time in many problem-solving situations, research on mechanized intelligence conducted within the artificial intelligence community largely has ignored the temporal characteristics of problems. [2] The application areas have been chosen deliberately to illuminate only particular aspects of a current theory of intelligence. Thus for example, the so called world of "toy blocks" has received considerable attention, not so much because artificial intelligence researchers are enamoured of blocks, but because they thought that the rather clear-cut "physics" of this world is a more suitable vehicle for testing rudimentary theories of knowledge acquisition, organization, and application. Because the consideration of time within such settings would lead to rather messy problems, researchers in artificial intelligence

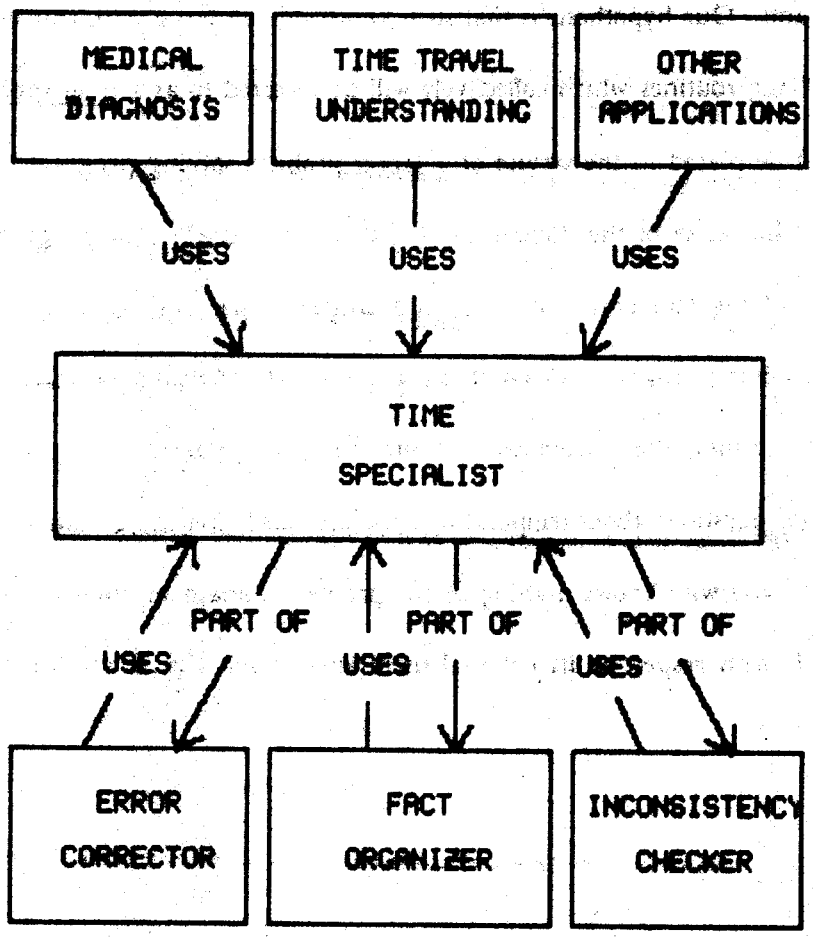
---

1 Much of this chapter is identical to the paper, "Mechanizing Temporal Knowledge" by Kenneth Kahn and G. Anthony Gorry.

2 The two exceptions known to the author are the work of Bruce [Bruce 1972] and the work of Findler and Chen [Findler 1973]. The former primarily is concerned with the computational linguistic aspects of temporal reference, and neither work deals with the full range of problems considered here. For example, neither work considers in any detail intervals of time, the fuzziness of temporal references, or the use of various organizing structures.

generally have factored temporal considerations out of the problem domain in question or deliberately have treated them in rather cursory fashion.

This paper considers one way in which knowledge about time can be incorporated into problem-solving programs. Our hypothesis is that such knowledge, in large part, can be embodied in a set of problem-solving routines which collectively will be referred to as a time specialist. The time specialist then can be placed in the service of a larger problem-solving program to deal with the temporal questions that arise in the domain dealt with by the latter. Having given the time specialist statements involving temporal references (in a language that is determined by the time specialist), the problem-solving program can ask it to make a variety of deductions and to answer a variety of questions concerning these statements. Thus the time specialist is an *idiot savant*, a program that can answer questions about temporal matters (provided that the questions are framed appropriately), but that otherwise knows nothing of the problem domain in question. The role of the time specialist, both with respect to its external users and its self-use, is visually depicted in Figure 1.



THE USERS OF THE TIME SPECIALIST

Figure 1

To test the idea, I undertook the construction of a rudimentary time specialist program to understand temporal specifications. A temporal specification is a statement that partially specifies, in some manner, the time of one or more events. Examples are:

(1) Three weeks ago John had a cold.
(2) John was born June 6, 1966.
(3) Two or three years after graduating, John went back to school.
(4) A few months from now, John will finish his thesis.
(5) After his cold, John died.
(6) John saw Mary a while ago.

In any temporal specification, some relation is given between at least two events, each of which can be considered a point in time. We dealt exclusively with the case of two events, because temporal specifications with more events can be represented as a series of statements concerning only two events. The more primary event is called the "event;" and the other, the "reference event." Sometimes, as in sentence (1), the reference event "now" is implicit (through the use of the word "ago"). In date expressions such as sentence (2), the reference event is the "zero" point for the calendar in use.

No attempt was made to deal with what is commonly called subjective time, and I devoted relatively little attention to the linguistic problems associated with understanding and generating temporal specifications in natural language. Within these constraints, however, we did implement a rather extensive program for understanding and answering questions about temporal references. To test the notion that such a program would be useful in a variety of problem-solving applications, I undertook an initial assessment of its use in understanding the temporal aspects of some everyday occurences, disease scenarios used in medical diagnosis, and time-travel stories.

Section II: A Session with the Time Specialist

The following is a session with the time specialist. For ease of exposition, I have translated the stilted language [3] required by the program into English. Although this has substantially improved the form of the conversation, it has not altered the content of the interactions.

(U1) User: I was born January 25, 1952. When I was a few weeks old I had an operation. I began grade school when I was 5. I graduated grade school in June 1965. I went to England two years later. I stayed in England for 8 or 9 weeks.

(U2) User: How old was I when I left England?

(S1) System: 15.5 years give or take 1.6 months

(U3) User: Explain your answer.

(S2) System: The time of the end of the trip to England is August 17, 1967 give or take 1.6 months and the time of birth is Friday, January 25, 1952.

(U4) User: How long ago did I go to England?

(S3) System: Between 8 and 8.3 years ago

(U5) User: What happened before I was 10?

(S4) System: The beginning of grade school, the operation, and birth

(U6) User: I was born May 5, 1955.

(S5) System: I thought the time of birth is January 25, 1952 not May 5, 1955, are you referring to the same birth?

(U7) User: Yes

(S6) System: Is May 5, 1955 correct?

(U8) User: Yes

---

3 For example, the first two sentences are given to the time specialist as "(time-of (beginning-of life) (date (1952 1 25) (fuzz nil)))" and "(time-of (all-of operation)) (after (beginning-of life) (fuzzy-amount (nil a-few weeks)))." The meaning of "fuzz" and "fuzzy-amount" will be discussed below.

(S7) System: Earlier I was asked how old the person was at end of the trip to England, and I replied 15.5 years give or take 1.6 months. However, my reasoning was based on the fact that the birth was January 25, 1952, which you now say is incorrect. We could either forget it or I could answer the question again. Shall I re-run it?

(U9) User: Yes

(S8) System: The answer is now "12.3 years give or take 1.6 months"

As can be seen by the user's story (U1), the time specialist can accept facts in many different forms. Some of the statements specify the dates of events. The others give the time of one event relative to some other event. Portions of the date can be missing as in "June 1965." In addition to accepting time intervals in terms such as "2 years" or "between 8 and 9 weeks" the specialist accepts time intervals stated in vague terms such as "several days," "nearly a month," or "a few weeks" as in U1. The time specialist also can answer questions about events whose time was not explicitly given. Questions that require the time specialist to search for events that occurred at a specified time expression such as "before I was 10" (U5 and S4) also are accepted.

The time specialist checks incoming facts for contradictions with previously entered facts. The statement "I was born May 5, 1955" (U6) is inconsistent with a previous statement. The time specialist begins to resolve the contradiction, first by making sure the same event is being referred to (S5). Then it asks whether the new fact is correct (S6). (If more than one old fact had been involved in the contradiction, then the time specialist would have asked about each one in turn.)

After finding that the old fact, "I was born January 25, 1952" is invalid, the time specialist attempts to correct the situation. It marks the old fact as "not to be believed" and searches for facts that were deduced from that fact and doubts those in turn. For example, it had deduced that the end of the trip to England was August 17, 1967 and it found that this deduction was partially based

(S7) System: Earlier I was asked how old the person was at end of the trip to England and I

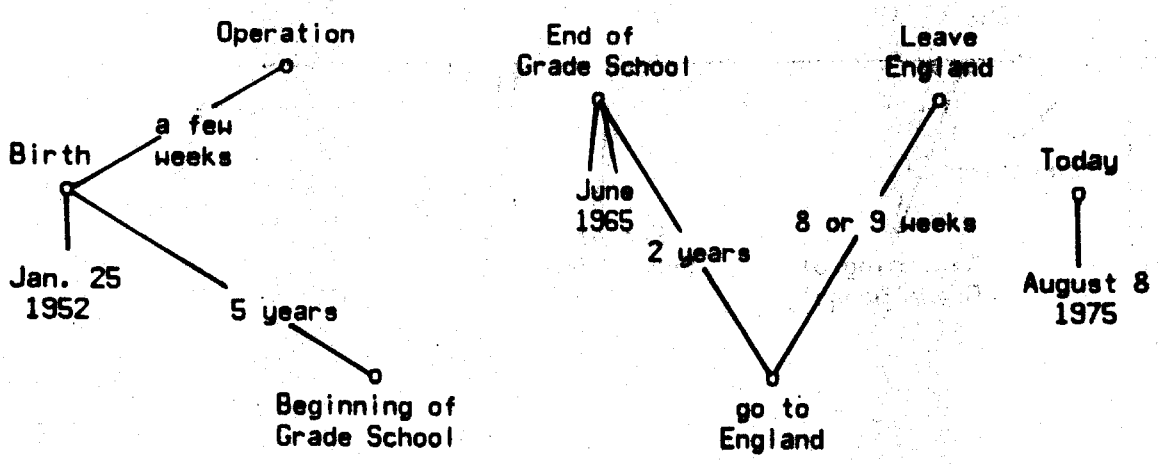
birth was January 25, 1952, which you now say is incorrect. We could either forget it or I could

As can be seen by the user's story (U1), the time specialist can accept facts in many different

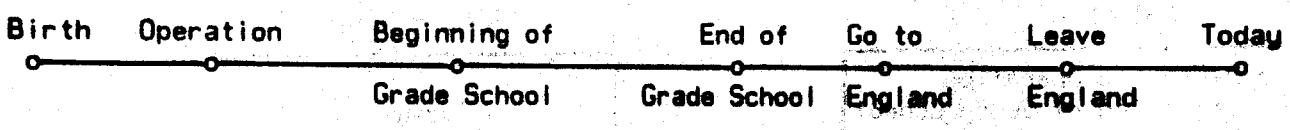
The statement "I was born May 5, 1955" (U6) is inconsistent with a previous statement. The time specialist begins to resolve the contradiction, first by making sure the same event is being referred to (S5). Then it asks whether the new fact is correct (S6). (If more than one old fact had been involved in the contradiction, then the time specialist would have asked about each one in turn.)

After finding that the old fact, "I was born January 25, 1952" is invalid, the time specialist attempts to correct the situation. It marks the old fact as "not to be believed" and searches for facts that were deduced from that fact and doubts those in turn. For example, it had deduced that the end of the trip to England was August 17, 1957 and it found that this deduction was partially based

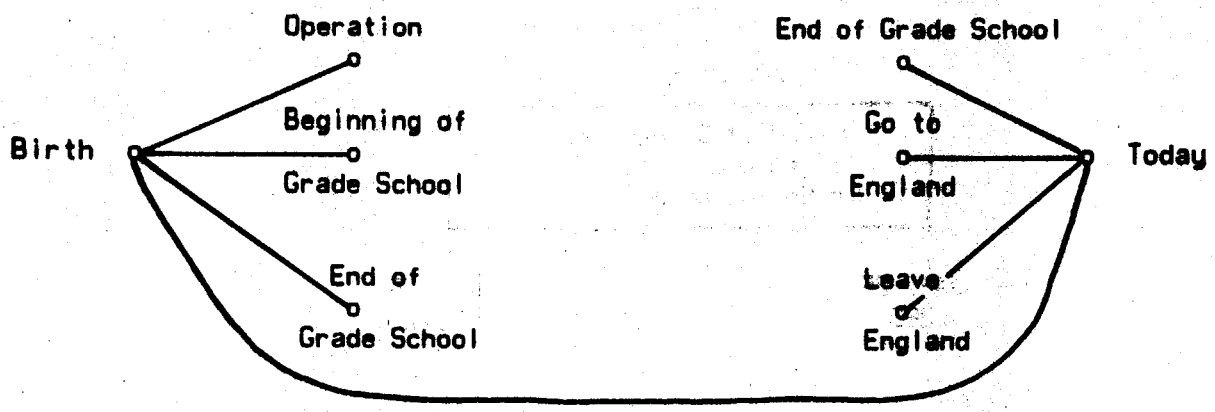
Original Story in the Data Base:



Before-After Chains:



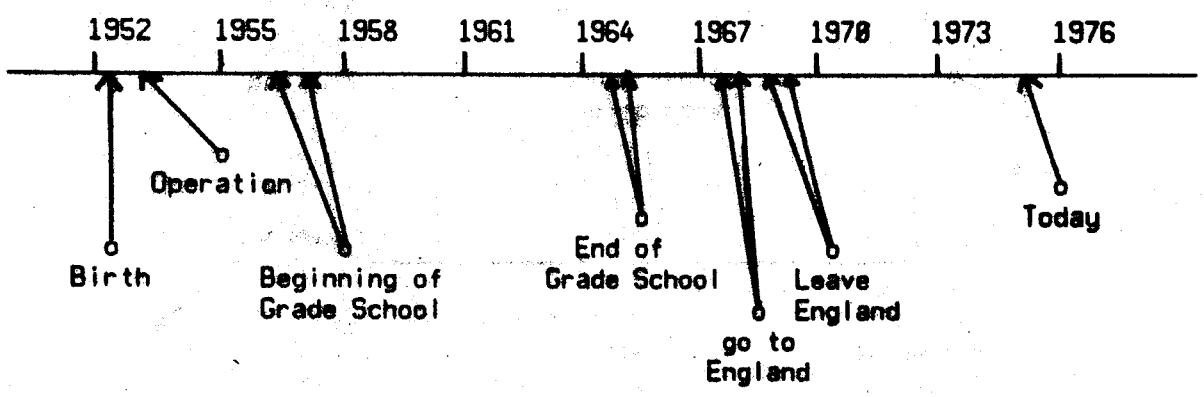
Special Reference Events:



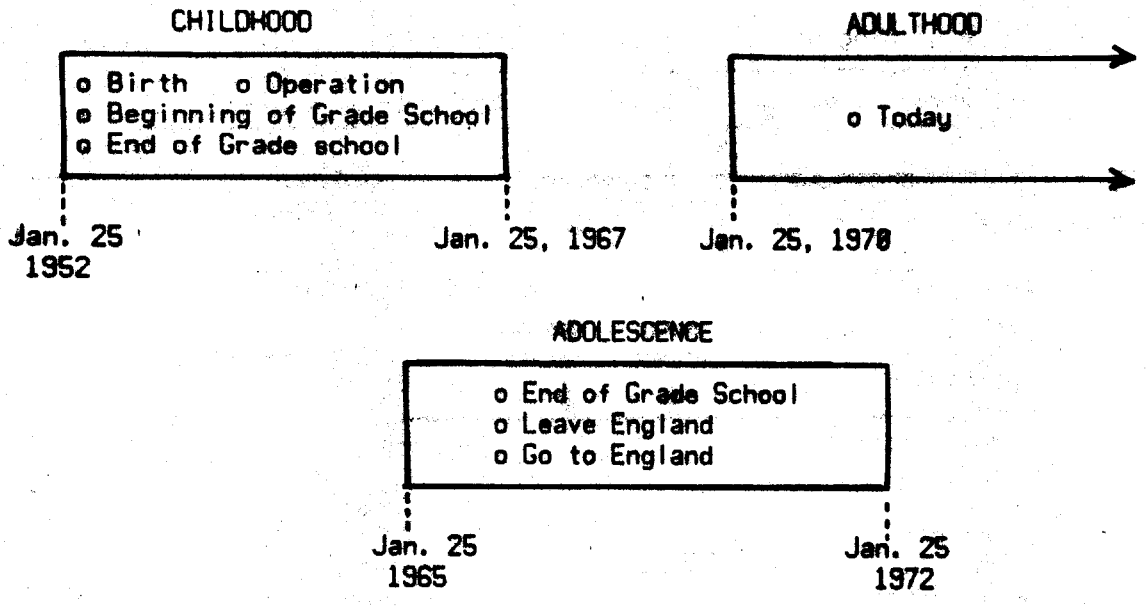
Organizational Structures for the Facts in the Scenario

Figure 2 Part I

The Date-Line:



Proposed Periods Structures:



Organizational Structures for the Facts in the Scenario

Figure 2 Part II

In what follows, I will discuss the operation of the program in terms of its three major functions. These functions of the time specialist are depicted schematically in Figure 3. First, I will consider how statements of knowledge which embody some reference to time can be represented in a consistent way. Second, I will discuss how plausible deductions concerning these statements can be made by a program that knows only about the temporal characteristics of the knowledge involved. Finally, I will discuss the problems of maintaining a consistent set of temporal references by detecting inconsistent or potentially contradictory pieces of knowledge.

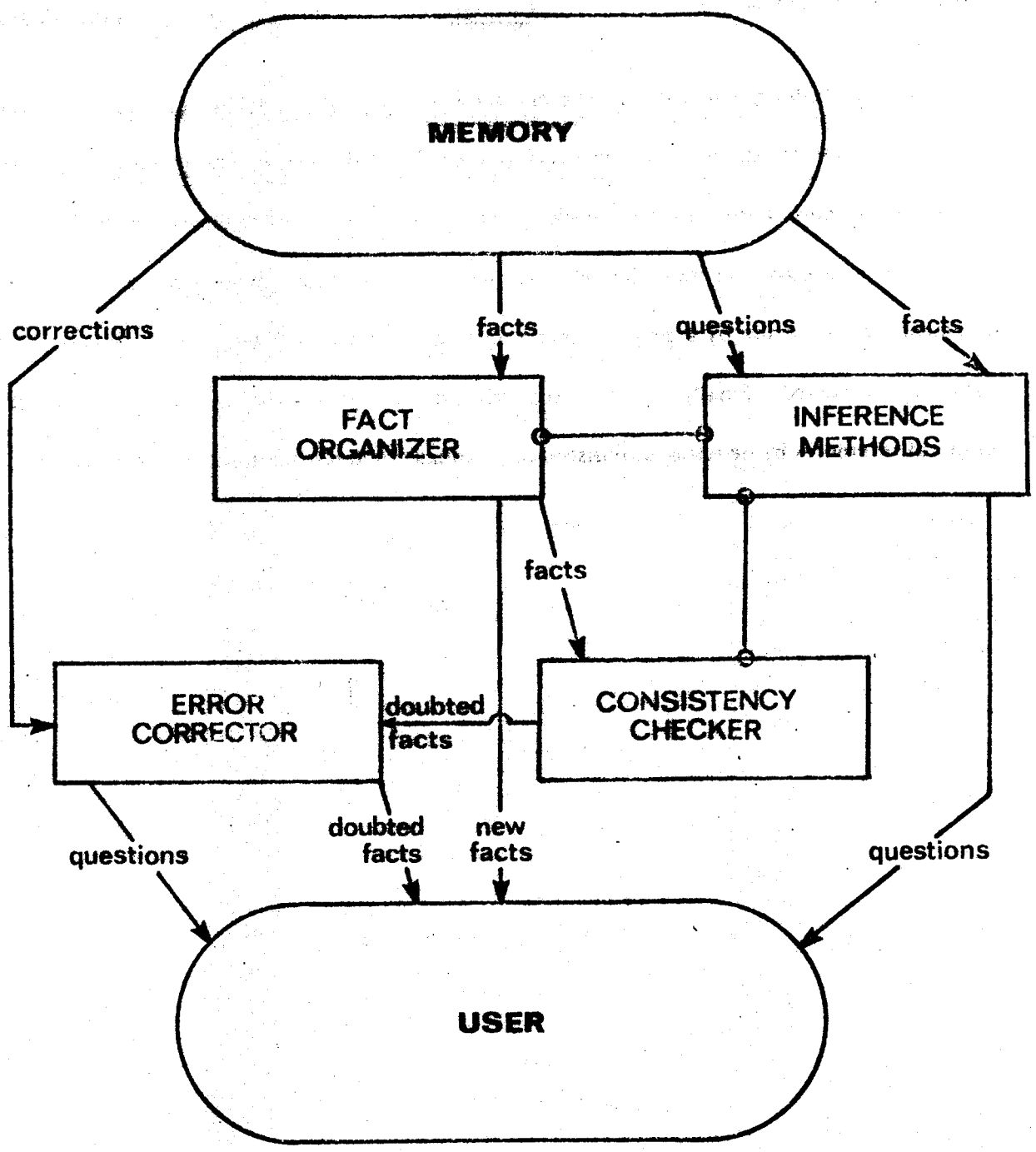


Figure 3

A Schematic Representation of the Time Specialist

Section III: The Representation of Temporal References

It is natural first to approach the problem of representing statements involving temporal references through the use of a single representation into which all such statements are converted. For example, each occurrence can be associated with the date on which the it began and the date of its conclusion. All events that happened during a particular time interval, then, can be retrieved simply from a date-line, a list of events sorted into chronological order.

In many situations, however, the date of an event is unknown or irrelevant. Suppose for example the following two statements were given to the time specialist: "Event A was after event B by four months" and "Event B was before event C by six weeks." Surely one would like the time specialist to be able to answer the question of how much later A was than C through the obvious means of subtracting six weeks from four months. To force the conversion of these expressions to ones involving dates seems inappropriate, even if the dates of A, B, and C are known or computable.

Another problem with using dates (or any point estimates) as the only means for specifying temporal items is that it is difficult to preserve the inexactness or fuzziness of statements such as "A few weeks ago I had a cold." If the time specialist converts such a statement to one involving a date and an associated interval (the latter reflecting the uncertainty about the precise time), it may lose much of the original information. If asked, for example, "When did the patient have a cold?" it may not be able to produce the reasonable answer "a few weeks ago."

For reasons such as these, the time specialist was constructed to deal with different kinds of temporal specifications. The time specialist has routines that can compare time expressions of

different kinds, translate between them, and combine time expressions in making inferences. The complete specification of statements accepted by the time specialist is given in the next chapter.

Section IV: Organizing Temporal Specifications

The manner in which the time specialist organizes the incoming statements is important, because it may markedly influence the efficiency with which questions can be answered. A temporally sorted date-line will make the retrieval of those events which occurred within a particular interval both simple and fast. Other organizations are preferred, however, if many dates are unknown or different types of questions are anticipated. Therefore, the time specialist has several different ways of organizing the events in its memory. The choice of which organizing schemes are employed is under the control of the user.

A) Organizing by Dates

One way in which the time specialist can organize events is by their dates. If it has been requested to do so, the time specialist checks new facts to see if they are "date" types, and if they are, it inserts them into the date-line in an appropriate chronological position. The date-line is represented by a list of event specifications, each including a date and a pointer to the time specification from which the date was obtained or inferred. Thus the date-line can be revised when the fact in question is no longer to be believed.

If the incoming fact contains a date and an implicit fuzzy expression (e.g., "June, 1965"), then the upper and lower bounds for the date are computed, and each is inserted into the date-line, marking an interval for the event. When no date is obvious for a time specification, the time specialist can invoke a program that will attempt to discover the date of the event so the event can be placed properly in the date-line. This program is one of the "methods" which will be discussed below.

B) Organizing by "Special Reference Events"

Another way the time specialist has of organizing facts is in terms of special reference events, events (e.g., "birth" and "now") that often are referred to (perhaps implicitly) in giving the time of particular occurrences.foot <The choice of which events are special reference events is made by the user of the time specialist, but the time specialist itself could make such decisions after inspecting statistics on which events are referred to most often by other events.> Typically, the time of a special reference event is known quite precisely. Therefore, it can be used to assess the time of events related to it. For example, the time of the event "A" can be determined with considerable confidence from the statement "Event A happened a few days after my twelfth birthday." If a new event entering the system does not refer to some special reference event, then the time specialist attempts to find one for it. If it succeeds, a link between the incoming event and the special reference event found is added to the data base. Thus, in terms of the previous example, if the incoming statement is "Event B began when I was 15" then the system can use the "the day of birth" as a special reference event to find the relation between A and B.

C) Organizing by "Before-after Chains"

Another form of organization employed by the time specialist uses the notion of "before-after" chains. Such a chain occurs often in a story where the principal events form a sequence. For example, suppose the system had been told

(1) Event A was a week before event B.
(2) Event B was three weeks before event C.
(3) Event C was a month before event D.
(4) Event A was two months before event D.

The time specialist would create a chain "(Event A, Event B, Event C, Event D)" and a chain

"(Event A, Event D)" to reflect the two "temporal" paths through this series of events. Such paths often are useful in answering questions about the time of events in the chain, particularly when additional temporal characteristics are known for some members of the chain.

D) Analyzing an Incoming Statement

When a fact is added to the system, then there may be a number of analyses brought to bear on it by the time specialist depending on the commands from the higher-level program. We have indicated the three major ones in the above discussion, namely the deduction of a date for the facts, the association of the event to the nearest special reference event, and the insertion of the event into an appropriate before-after chain. Any or all of these functions may be applied to a given event. It is not necessary, however, that the methods be applied at the time that the event is entered. The time specialist leaves a record whenever a given method is applied to a fact. If, at a later time, the higher-level system wants another method applied to the fact in question, it is easy to do so. For example, if the higher-level program decides that a date-line would be useful, it can activate the function which constructs the date-line to make the date line current. Thus, the time specialist can accept statements without deducing all those aspects which may have some potential use.

Section V: Answering Questions

The time specialist can answer three types of questions about the facts it has in its data base. These types are sufficient for the representation of the essence of a wide variety of superficially different requests. The three basic question types are:

(1) Did event X happen at time expression T?
(2) When did event X happen?
(3) What happened at time expression T?

The question-answering ability of the time specialist is embodied in a set of programs collectively called the fetcher. The tasks of the fetcher are to accept a pattern which specifies a particular question, to interpret the pattern to determine the type of question involved, and to select one or more appropriate methods for answering the question. The fetcher has a rather large number of methods at its disposal. Each method is an independent program which is designed to answer a particular kind of question by making use of a particular organization of facts in the data base. For example, there is a method which is designed to answer questions about dates by making use of a date-line. In this section I will review briefly the various methods employed by the fetcher and indicate the way in which they are used to answer various kinds of questions.

A) Equivalent Temporal Specifications

The simplest way a question can be answered by the time specialist is that the fetcher finds that the answer is already known explicitly or that a fact equivalent to the answer is known. For example, suppose the system had been told that Event A was three or four days after Event B. If later it was asked if Event B was about a half a week before Event A, the time specialist should know the answer.

The time specialist employs a method to see if an equivalent means of expressing the requested item is already known. In assessing the equivalence of two temporal specifications, this method uses knowledge of the meanings of such relationships as "before," "after," and "same as," as well as the meanings of the various time units. It may be possible to make the required assessment with only rather straightforward conversions. If the items are different representation types, however, (for example, one in terms of the fuzzy notation and the other in terms of intervals) translation of one item into the type of the other is required before the application of the equivalence tests. [4]

B) Deduction in Question Answering

When the fetcher fails to find an equivalent expression in the data base, it must invoke other methods to try to deduce the answer to the question. As noted above, these methods are specialized to particular kinds of questions and to particular kinds of organizations of the facts.

1. Using Dates

Many questions can be answered using dates. In the case where the dates are stored in the system, then the question can be answered simply. For example, if the time specialist is asked what events occurred within a particular time interval, then it will retrieve the section of the date-line

---

4 There is one problem in the assessment of equivalence, if the pattern being used to retrieve items from the data base has a "fuzziness" within it. Currently, the test of whether one fuzzy event occurs within the interval specified by another fuzzy event depends on which type of representation is involved. Each representation has associated with it a specialized program which computes the percentage of overlap between a retrieved item and the pattern which represents the question involved. These routines leave behind comments on how well the items in the question matched the item retrieved from the data base.

that falls between the dates of the end points of the interval. [5] Alternatively, if the time specialist is asked whether one event occurred before another, a comparison of the dates of the event will be sufficient. The subtraction of two dates is sufficient for answering questions about the relationship of two events.

Dates are so useful that it is often worthwhile determining the date of an event when one is not given. The time specialist has an indirect method for inferring the date of an event using reference events. The idea is simple; to find the date of an event, find a relative time expression for the event in question and look for the date of that time expression's reference event. If that date is found, then the date of the original event can be determined. For example, suppose the system had been previously told:

(1) A week after the sore throat I had headaches.
(2) The sore throat was March 21, 1975.

To answer the question "When were the headaches?" a method would find the first fact. Once the date of the reference event, the sore throat, is retrieved, it is a simple matter to produce a date for the headaches. If the date of the reference event is not known, the date-finding method can be recursively applied to try to find it.

2. Using Relative Expressions

Suppose the time specialist has been told "Three weeks ago I had a cold and two weeks ago it was over," and later it is asked "How long did my cold last?" It seems natural to simply subtract two weeks from three weeks and respond "one week," rather than to calculate the dates of the

---

5 This retrieval can also incorporate the so-called group specialists to filter out those events which are not of the desired type. At present the time specialist does not segregate events from different groups into different date-lines.

beginning and the end of the cold and then to subtract the dates. Further there are many such cases where one does not have the choice of computing dates.

The fetcher possesses a method which tries to deal with such a problem. When a question concerning the relationship between two events is posed, this method attempts to find facts in a data base which relate the events to a common special reference event. If it succeeds, it can easily determine the temporal relationship among the events in question. If it fails, it attempts to find two special reference events linked to one another which in turn link the two events in question. For example, consider "Three years ago I graduated from college, and I started college at the age of 18." In order to determine how much time was spent at college, the time specialist can use the special reference event link from the beginning of college to birth and another such link from the end of college to now. Because it knows the "distance" between the two special reference events, in this case age, it can answer the question. Typically such links between special reference events do exist, because of the importance of these events in answering a variety of questions. [6]

3. Using Before-after Chains

The fetcher also employs the before-after chains in trying to find relationships between events. Basically it seeks a chain that contains both the events in question, and then attempts to find the shortest path between the two events in the chain. When this path has been determined, the time specialist uses the intervals of each link in the chain to compute the overall time interval between the two events. For example, if the before-after chain "(A, B, C, D)" is in the data base,

---

6 There is another version of this method that uses the full power of the fetcher to try to establish such links if they do not exist. This method will not be invoked on the second call to the fetcher, because it does not apply to questions that explicitly mention a special reference event.

and the time specialist is later told that A is before C by some amount, then in determining the interval between A and D then it only needs to combine the "distance" between A and C and the "distance" between C and D.

The fetcher cannot be confident of the intervals determined from a long before-after chain, because in combination of fuzzy information, considerable inaccuracy is introduced. Links between special reference events, however, often are known with great accuracy. (For example, the "distance" between birth and now). Therefore, whenever possible the fetcher seeks to "jump over" segments of a before-after chain through connections between two special reference events.

The before-after chains are also useful in several other contexts. For example, if the question pattern requires only the relative order of the events, not the temporal distance between them, then finding a before-after chain may solve the problem. Similarly, if the question requests all events of a particular group which occurred between two specified events, then finding the specified events in before-after chains will yield at least some of the required answer. Of course in this latter case it may be necessary to filter the events by a group specialist.

4. The Last Resort

When all other methods fail, a breadth-first search can be invoked as a last resort. The search follows all chains away from the event until the reference event in question is found or until the chain terminates. This method could be used for all fetches, but of course it would be exceedingly slow. Because of its computational inefficiency, it is used only sparingly. Another problem with the use of this method is that although a chain between two events may be found, the chain may include so many events with fuzzy specifications that the overall time interval may be so uncertain as to make the answer virtually meaningless.

5. Selecting a Method for Application

When the fetcher is called with a question pattern, it must decide which methods it will employ in trying to provide an answer. Currently its choice depends on two factors. First, it considers the type of time specification which occurs in the pattern. Second, it determines whether events are specified or whether events are required to fit within a certain time expression. Once these determination has been made, it is a simple matter to select a subset of the available methods for application. For example, if the question asked what events occurred between two dates, then the method that uses the date-line and the method that checks if each known event occurred within the interval would be selected. The selection of appropriate methods is easily accomplished, but the order in which the methods should be applied remains a problem. Presently the order of the methods is determined by a static estimate of their likelihoods of success. In a more advanced system, of course, other factors such as the characteristics of the items in the data base and statistics on the past success of methods would influence the default order.

The effort the caller wants to put into the task is reflected in the use of the selected methods. Clearly the time specialist cannot know how important any particular fetch is to some higher level problem-solving program. For this reason an effort measure can be passed to the fetcher along with the question pattern. Presently this effort measure is approximated by the amount of CPU time necessary to complete a fetch.

Section VI: Maintaining the Consistency of the Data Base

In addition to accepting temporal specifications, the time specialist attempts to assess the plausibility of a particular statement and its consistency with statements already in the data base. Consider the following facts:

(1) The cold ended last week.
(2) Three weeks ago there was a party.
(3) A couple of days after the party the cold ended.

If the same cold and party are being referred to, the "fuzziness" of these expressions is not sufficiently great to account for the discrepancy among these statements. We would like the time specialist to recognize this and to correct the situation if possible.

The module that detects this kind of inconsistency was easy to implement by making use of other portions of the time specialist. The time specialist invokes its question-answering function to check the consistency of the latest fact with facts previously accepted. It asks the fetcher whether the latest fact is true. Three outcomes are possible for this request. The fact already may be known. There may be insufficient information to answer the question, in which case the fact is accepted. Of course it may be contradicted later. The third possibility is that enough is known to determine that the new fact is inconsistent with facts previously learned. Notice that an apparent inconsistency need not be a real one. It may be that the events being referred to are in fact different events than the ones known in the data base. Thus in the above example, the party that is referenced may be a different party in the two statements.

When an inconsistency is detected, the normal course for the time specialist is to engage in a conversation with the user to ascertain whether the events in question are the events currently

known to the time specialist. If they are indeed different events, then the new item can be added to the data base, but the time specialist must store with it a reminder that the new event is different from the events it previously knew about. When the events are the same, then the apparent inconsistency is explained to the user, and the user is asked if the new fact is wrong. If it is, then the fact is rejected and the user is free to add a revised version of it. If the new fact is correct, then the user is asked about each of the other facts which are involved in the apparent contradiction. If the user asserts that each fact is correct, then the inconsistency is allowed to stand and a certain amount of indeterminism has been introduced into the data base. If, however, one of the offending items is no longer to be believed, then its status is changed, and the entire mechanism for undoing the consequences of having believed that fact is invoked.[7]

---

7 Of course, this rather simple approach will not solve all problems. One inconsistency that it will not discover is that the beginning of an event may be said to be after its end, but this inconsistency can be detected easily by asking the question-answering program somewhat different questions. Notice, however, that in time-travel stories, this normally obvious constraint is not always valid. It should not be applied to time trips.

Section VII: Correcting Errors

It is one thing to reject an item that appears to be implausible or inconsistent with knowledge in the data base, but it is an entirely different matter to undo the effects of having believed something incorrectly. Thus when it is told that a previously accepted fact is now to be doubted, the time specialist faces a rather complicated problem. Other facts may have been deduced from the suspect fact, and these, too, perhaps should be under suspicion. On the other hand, the questioned fact may not have come directly from the higher-level problem-solving program, but itself may have been deduced from other facts. At times, these other facts should be doubted, but at other times the original facts should remain. In addition, answers to previous questions now become suspect when the reasoning employed the doubted fact.

For example, when a previously accepted fact is no longer to be believed, the before-after chains which reference this fact should be corrected. Assume the time specialist has formed the chains "(Event A, Event B, Event C, Event D)" and "(Event A, Event D)," and it subsequently is told to doubt that "Event B is before Event C." The program generally will destroy those chains containing both Events B and C. If a chain can be easily saved, however, (as when both of the events in question are at the beginning or both are at the end of a chain), then the chain is shortened in the appropriate manner. Notice that were the chain "(Event A, Event D)" based on a fact that in turn was based on the doubted fact it too would be destroyed.

When it makes deductions, the time specialist retains information about which facts were used in the process. Then when it is told to doubt a particular fact, it can determine whether each of the deduced facts remains consistent with the facts known in the data base. Some of these

deduced facts will come under suspicion, and the entire process will be applied recursively to the newly-doubted facts. Quite understandably this process requires a considerable amount of computation; it will, however, maintain the data base in a consistent fashion.

As noted above, the fact under suspicion may have caused the time specialist to answer questions incorrectly. Unfortunately the time specialist cannot know what has been done with the answers it provided, but it can interact with the user, if it no longer believes an answer formerly given. To discover which answers should be doubted, the time specialist uses comments left behind by the fetcher that indicate what facts were used in answering a particular question. The higher level program is given an opportunity to ask each such question again.

Although these solutions are not complete, they do allow the time specialist to undo most of the effects of incorrectly believed facts. Clearly much more work is required to solve this problem in a general way. The complexity of the problem is such that in some circumstances, the simplest and most efficient solution may be to change the fact in question and to re-learn the whole set of facts that are affected.

Section VIII: The Relation between the Time Specialist's Language and English

An earlier version of the time specialist was connected to a crude parser of English sentences and questions about the times of events. Using LINGOL, ([Pratt 1973] and [Pratt 1975]) a syntactic parse tree was obtained. At all nodes on the tree small programs tried to identify components of a temporal specification such as the reference event, the units, the fuzz, etc.. These suggestions were then combined to produce a time specification understandable by the time specialist. The present version, however, does not include a parser.

Despite the lack of a parser, the relationship between the aspects of time that this research covers and that available in everyday English needs to be clarified. Ideally, all English temporal specifications should be mapable into the language of the time specialist. There are some sentences, however, that the time specialist cannot, at present, handle. Some sentences are not intelligible to the time specialist because of minor deficiencies in the representation of dates and amounts. For example, the time specialist does not "understand" the units of time less than one day. These problems can be easily remedied by slight modifications of the time specialist and are of little theoretical interest. Sentences about recurring events are often unintelligible to the time specialist, however, an extension of the system to cover these cases is described in Section I-E of Chapter II.

One might think that since the representation of time presented herein is concerned with events of no significant duration that occurences with duration would not be understood by the time specialist. This is clearly not the case. For example, the sentence, "The cold lasted a few days," is represented internally as, "the beginning of the cold was a few days before the end of it." This transformation can always be done with no loss (or gain) of information. It is one of the tasks of the parser and as such is transparent to the human interacting with the system.

Other kinds of temporal specifications that cannot be handled are a more serious problem. Sentences that depend upon "real world" knowledge cannot properly be handled by the time specialist. Examples are sentences containing phrases such as "recently" or "after the ball game." Were certain facts such as the duration of ball games or the frequency of the event that occurred "recently" known, then they could be used to transform the sentence into one that can be understood by the time specialist. The time of the event after the ball game might be assumed to be within a few hours of the end of the ball game and thus representable. The problems of determining which such assumptions should be made and when were not investigated.

Interesting problems relating to meanings of context-dependent expressions like "a while ago" or "intermittent" are difficult to resolve. The meaning of "a while ago" varies with the context and frequency of the event being referred to. For example, if someone were to say, "Mary saw John a while ago," then the length of the interval involved depends on how often they see each other, whether the speaker and hearer of the sentence saw John and Mary together a few hours ago, etc.. Expressions like "last Friday" are interesting because they also require interaction between the time specialist and real world knowledge. The expression "last Friday" is ambiguous if "today" were Saturday for it can mean "one day ago" or "eight days ago." However, "eight days ago" is what should be "parsed" since the word "yesterday" would have probably been used if it did happen one day ago.

The remaining type of sentence that causes problems for the time specialist are those that are ambiguous. For example, the sentence, "John works on Friday," by itself, does not contain enough information to decide whether a particular event or a recurring class of events is being referred to. These kinds of sentences are not very useful for answering questions, however, combined with

other sentences they may constrain the meaning sufficiently to be useful. Suppose that the following sentence was "so we better not schedule the classes on Friday," then we can safely assume that a cyclic event occurring on Fridays was being referred to.

Notice that both the ambiguous and the context-dependent sentences become intelligible to the time specialist when enough other information is brought to bear to produce a precise enough meaning. While it is clear that the time specialist should be able to help combine the information to constrain the meanings of such sentences, these problems are of general interest to AI natural language researchers and are removed from the central problem addressed in this research -- understanding time.

Clearly communication between the parser and time specialist should be in both directions. When inconsistent or unlikely times for events are entered into the system, the time specialist should ask the parser to double check its answer. Consider the case where a time specification was parsed indicating the beginning of an occurrence was after its end. Here the time specialist would ask the parser if it is sure of its conclusion; did it mix up the main event and reference event? These questions, and those concerning the role of tense, while interesting, were not investigated.

The inverse problem of mapping from the internal representation of temporal specifications to English was investigated to a limited extent. Very simple heuristics were used to generate English statements that are adequate for the purpose of communicating with the user. This is described in more detail in Appendix D.

Section IX: Comments About Philosophical Questions

In philosophy or AI, people occasionally wonder whether time is linear, circular, cyclic, hierarchical, or spacial. Does time have a direction? Is time primary or is it secondary to space? Does time play a special role in the structure of memory or is it an another aspect of events along with location, motive, participants, and desirability? How do subjective and objective time differ? What is the relationship between time and causality?

These are all interesting questions which should be considered in designing a representation of time. One must make a commitment to answers to some of these questions. As presently concieved the time specialist "thinks" of time in the following ways:

1) as linear, as evidenced by its "date line"

2) as hierarchical, in that it will accept statements about periods, unfortunately it does not maintain special structures for dealing with periods

3) as explicit, in that the basic data base explicitly represents the time of events

4) as implicit, in that in "before-after" chains the time of events is indicated by their relative position

5) objective, except that the time specialist also seems adequate for dealing with many aspects of subjective time (that is the time of events experienced as opposed to told about)

6) uni-dimensional, in that its reasoning applies equally well to distances

7) as a lattice, where the nodes are events and the partial ordering is based on the temporal "distance" between them

Since both people and the time specialist can think of time as a multi-faceted concept, it seems reasonable to answer the philosopher's questions with "yes." "Yes" time is linear, "yes" time is hierarchical, "yes" time is a primitive.... Time is what people think it is.

Section B: The Time Specialist Applied to Medicine

As mentioned earlier an important thesis of this research is that once a specialist knowledgable about time is developed, it can be used as a component of diverse problem solvers. The problem of doing medical diagnosis was chosen as an example to demonstrate the general applicability of the time specialist. The complicated "time course" of a disease is often important in diagnosis. An "hypothesis matcher" that matches a patient's history against the time course of a suspected disease was implemented and tested on a description of Acute Poststreptococcal Glomerulonephritis (AGN).

A) The Time Specialist's Role in Hypothesis Matching

Many of the activities of the time specialist can be viewed as that of an intelligent pattern matcher. In medical diagnosis, for example, one can view the typical time course of a disease as a complex pattern against which one wishes to match the patient's history. For example, in a chapter on AGN by Schwartz and Kassirer [Schwartz 1971] there are these sentences:

> Between the onset of a streptococcal infection and the development of symptoms or signs of acute glomerulonephritis, there is a latent period that usually ranges between one and two weeks and that averages 10 days.
>
> Latent periods shorter than a week are not uncommon, however, and occur in as many as one-fifth of the cases.
>
> Only in an occasional patient does acute glomerulonephritis develop as long as three to four weeks after the inciting infection.
>
> Latent periods of more than one month have not been noted, and it is doubtful whether an infection that precedes the onset of renal abnormalities by such a period has a relation to subsequent renal disease.

While the patient may have said:

"I had a bad sore throat a couple weeks ago. And I first felt lousy and had red urine the day before my last visit."

Assuming the doctor knew that the patient's previous visit was last Wednesday, he or she could safely assume that the patient's history matched the norm presented in the AGN chapter. Yet for a computer to do the same many problems of representation, meanings of words (such as "couple weeks," "last Wednesday," etc.) and inference must be solved.

B) How the Time Specialist was Applied to Diagnosis

1. Some Problems

The purpose of the hypothesis matcher is to demonstrate the usefulness of the time specialist, not to do medical diagnosis. It was decided that the hypothesis matcher should be written as if it were a module of a larger present illness system. Problems of determining if, for example, the patient's red urine indicates hematuria, were assumed already resolved. The times of the major symptoms and phases are the inputs to the program. The main problem with designing the hypothesis matcher is to have it do enough to demonstrate the usefulness of time specialist in medical diagnosis without trying to solve the entire present illness problem.

Another major problem with the hypothesis matcher is deciding how good a match one has, or "scoring." The time specialist does remark as to how well an individual temporal expectation matches the facts in the data base, however, it does not provide a means of combining these measures of goodness of fit into an aggregate score. The hypothesis matcher does this aggregating in a rather *ad hoc* manner; it was considered a problem the present illness system must resolve and that there are no features of the problem special to time.

2. The Hypothesis Matcher

The hypothesis matcher is given an hypothesis and an effort measure; it will then either accept, accept with reservations, reject the hypothesis, or redirect the matching. In addition, it remembers the reasons for its recommendation in detail and a summary of its activities in a data base context. It also saves the state of its exploration so that the exploration of the hypothesis can be resumed.

The basic steps of the hypothesis matcher are:

1) initialize or resume old exploration

2) accept the patient's history

3) go through the facts of the story seeing what expectations they meet or fail to meet

4) ask about those expectations that are important and for which not enough information is available to confirm or deny

5) note those facts that are "left over"

6) summarize findings

The time specialist is essential for the operation of the hypothesis matcher. A few modifications to the time specialist were necessary for this application and were implemented. The hypothesis matcher can handle a complex story, ask the appropriate questions and give a reasonable evaluation. In doing all these tasks the time specialist is often called upon. A more detailed description of the application of the time specialist to medical diagnosis is given in Chapter IV.

Section XI: Understanding a Time-travel Story

In order to test the various functions of the time specialist and to assess whether they were sufficient to deal with a variety of temporal references, I applied the time specialist to the problem of understanding a time-travel story. Although this application may seem somewhat frivolous and esoteric, it is, nonetheless, one which provides a good exercise of the time specialist's capabilities. The very fact that travel through time permits a number of ostensible temporal paradoxes to exist exposes virtually the full range of assumptions implicit in the program. Of course, an important aspect of time-travel stories is that some of these assumptions need to be disabled if the story is to be understood properly. For example, it is clearly the case that a person can be alive in a time-travel story at a point in time which is before his birth.

Often the plots of time-travel stories hinge on the reader's ability to make temporal inferences from different frames of reference such as the time traveler's and the "real world's." One such story, "All You Zombies" by Robert Heinlein, [heinlein 1959] was chosen as a test of the time specialist. The time specialist demonstrates its understanding only by answering questions posed to it. A more complete time-travel understander would be capable of deciding what the interesting questions are, but from the time specialist's point of view there is no essential difference.

The story is about a man who had grown up in an orphanage as a girl, had a baby and became a male. His baby was stolen and he meets a time traveller who offers to bring him to the thief. He finds a girl instead who he gets pregnant. The time traveller manages to recruit him into the time service. Through various clues, some of them temporal, the reader can infer that the baby, the girl, the guy, and time traveller are all the same person with different ages. To provide

support for these inferences one needs to confirm that the certain events occurred simultaneously. This role is played by the time specialist.

To deal with time-travel stories, I had to extend the capabilities of the time specialist somewhat. The most important extension was to equip the time specialist with the ability to look at facts from two different points of view. The first point of view is the normal "universal" point of view in which time follows its normal, somewhat pedestrian, course. From this point of view, the experiences of the time-traveler are seen as creating paradoxes. But if the time specialist is to truly understand time-travel stories, it must be able to appreciate the point of view of the time-traveler himself. From this point of view, paradoxes vanish and the traveler enters and leaves the linear course of time at will. Therefore, in order to answer questions about time-travel stories, the time specialist must assume an appropriate point of view. For this reason, the understanding of such stories presents an interesting test case in which the capabilities of the time specialist can be investigated. The time specialist is capable of answering many interesting questions concerning the temporal aspect of the story as is shown in Chapter V.

Section XII: Other Applications of the Time Specialist

Other tests of the time specialist were considered but not implemented. One could go on trying out the time specialist *ad infinitum*. Two applications plus a few ideas for other applications are hopefully adequate to demonstrate the feasibility of a time specialist applicable to many diverse domains.

A) Reminder System

One problem domain for testing the time specialist that was considered was a reminder system. It would accept statements as to what was going to happen, and what regularly occurs and would answer questions and more importantly remind the user of appointments at the appropriate time. The time specialist would have to be extended to handle recurring events as described in Section I-E of Chapter II. It would also need methods that combine many descriptions of the time of an event to form a more refined description. The representation would have to include the importance and flexibility of the scheduled events to answer useful questions like, "Can I squeeze in an hour on Friday?" This system would fit in very well with the "personal assistant" project at the MIT AI Laboratory [Winston 1973].

B) Date-Finding System

Another application of the time specialist that was considered was a date-finding system. The user would be trying to remember the date of some event in his or her life, perhaps the first opera the user ever attended, or when the user last saw a particular friend. The system would contain many heuristics for asking the right questions of the user to obtain information to deduce

the answer. For example, it would know that one often knows the weather, or the day of the week of an event, without remembering the year. An analysis of the knowledge of such a system would perhaps provide interesting insight into human memory.

The time specialist would be called upon often to help combine various partial specifications to narrow down the possible times of the event in question.[8] It would need to be extended to include declaratives that include logical operators. This extension is described in Section IV-B in Chapter II. This application would be different from the others in that the time specialist always has the same task: determine the date of the event in question.

C) Trip Planner

A system that would help one plan a trip was also considered. It should be thought of as a module in a larger hypothetical "travel agent" system. Clearly an understanding of time is necessary to plan a trip. The more interesting reason for considering this application is that for a limited class of trips the time specialist with few modifications could be turned into a "distance" specialist. For the trip, to be "time-like" there must be no situations which depend upon the second or third spacial dimensions. For example, the trip from A to B to C to A without going back thru B would require a more sophisticated distance specialist than a slightly modified time specialist. This problem is interesting in that two different specialists; time and distance, would be interacting.

---

8 Unfortunately this problem was overlooked, though clearly the time specialist should be able to "intersect" many partial specifications of the time of a single event.

TABLE OF CONTENTS CHAPTER II

I. Temporal Specifications 47

A. Dates 49

B. The "Relative" Representation Types 49

C. The Representation of Fuzz 51

D. Currently Unrepresentable Temporal Specifications 52

E. The Representation of Recurring Events 52

II. The Equivalence of Temporal Specifications 55

III. The Translation of Temporal Specifications 58

IV. Question Patterns 60

A. The Normal Question Patterns 60

B. A Fuller Question Pattern 61

One could represent facts in the computer as English sentences, just as they were originally stated. Obvious problems follow from using English surface structure as the sole means of remembering facts in a machine. Paraphrasings that mean essentially the same thing would not be equal and retrieval based on partial information would be difficult. One needs a canonical representation (or a few canonical representations) that entering natural language input is mapped into. The problems designing and using a representation are discussed in this chapter. Included is a description of the four major representation types, the representation of recurring events, and the representation of questions. Also, the problems of mapping between the different representation types and the equivalence of temporal specifications is discussed. The problems of translating to or from English and the language of the time specialist are not considered.

Section I: Temporal Specifications

A temporal specification is a statement that partially specifies, in some manner, the time of one or more events. Examples are:

(1) Three weeks ago John had a cold.
(2) John was born June 6, 1966.
(3) Two or three years after graduating, John went back to school.
(4) A few months from now, John will finish his thesis.
(5) After his cold, John died.
(6) John saw Mary a while ago.

Despite the variety in the form of temporal specifications one can make generalizations about them. In any temporal specification, there are at least two events, both of which can be considered as points in time, and some relation between those events is given. I will deal exclusively with the case of two events, because temporal specifications with more events can easily be represented as a series of statements concerning only two events. I call the more primary event the "event" and the other, the "reference event." Sometimes, as in sentence (1), the reference event "now" is implicit (through the use of the word "ago"). In date expressions, such as sentence (2), the reference event in not obvious. Dates basically mean some amount of time after a particular, arbitrary, agreed upon, "zero point." Julian dates, which represent the date as the number of days since "day 0," are the clearest example of this view of dates.

The representation of the time of events as points in time is adequate for all temporal specifications. The duration of an occurrence is represented as the interval its beginning to the occurrence's end. Questions as to whether an event occurred during another are answered by paraphrasing the question as, "did an event occur after the beginning of some occurrence and before its end?" One of the advantages of breaking any happening of significant duration into two

events corresponding to its beginning and end is that the fuzziness of a temporal representation is easier to represent. Each "point" in time can become a fuzzy interval which is interpreted as the event occurred during this interval. The amount that the end follows the beginning might also be fuzzy, and by a different amount. This situation is more difficult to represent using events with durations. The more important reason for chosing points over intervals is that it simplifies the methods and thoughts of the time specialist.

We could define a time specification as follows:

time-specification ==> (<event> <relation> <reference-event> {<amount>})

Which means that a time-specification is defined as, (denoted by "==>"), an event, followed by a relation, followed by a reference-event, and finally an optional (denoted by the "{}") amount. For the sake of uniformity and convenience of processing, a different form was chosen. A marker called "TIME-OF" is inserted in the beginning, so this object's type is easily ascertained. More structure was introduced, and for the time specialist the formal definition of a time specification is:

time-specification ==> (TIME-OF <event> <time-expression>)

where time-expression is defined as,

time-expression ==> (<relation> <reference-event> {<amount>}).

As stated earlier, the time specialist considers events to be points in time. Any occurrence with a duration greater than the minimum unit of interest[1] is broken down into two events, one corresponding to the beginning of the occurrence; the other its end. This is expressed as:

event ==> (<event-type> <event-description>)

---

1 In most of the applications considered, this minimum is a unit of time of one day.

event-type ==> *BEGINNING-OF | END-OF | ALL-OF*.[2]

The representation of the "event-description," while a problem of interest to AI researchers, is not directly related to this research, and therefore it is usually handled in an *ad hoc* fashion.

A) Dates

Although dates can be expressed in the basic representation, for convenience and naturalness a special representation for date-expressions was designed. It is defined as follows:

time-expression ==> <*date-expression*>
date-expression ==> (*DATE* <*date*> <*fuzz-expression*>)
date ==> (<*year*> <*month*> <*day*>).

"Fuzz-expression" will be explained later. The "parsing" of the sentence discussed earlier, "John was born June 6, 1966," is:

```
(TIME-OF (BEGINNING-OF (JOHN'S LIFE))
         (DATE (1966 6 6) (FUZZ NIL))).
```

B) The "Relative" Representation Types

The "by-amount" representation type, along with the "interval" type and the "fuzzy-amount" type, correspond to time-expressions of the form:

time-expression ==> (<*relation*> <*reference-event*> <*amount*>).

The three representation types differ only in the form of the "amount" portion. The by-amount type is an interval of time, plus or minus some fuzz factor. The amount in the interval type is represented by two intervals of time, which is interpreted as the event occurred sometime between those intervals. The amount in the fuzzy-amount type is an interval of time expressed in vague terms such as "a few," "several," "nearly one," etc..

---

2 The "all-of" marker is simply used to easily determine the type of an event, and could easily be deleted.

The by-amount representation type is defined as,

amount ==> by-amount
by-amount ==> (BY-AMOUNT <interval> <fuzz-expression>)
interval ==> (<time-unit> <number>)
time-unit ==> DAYS | WEEKS | MONTHS | YEARS [3]

Example (l), "Three weeks ago John had a cold," is represented as,

```
(TIME-OF (ALL-OF (JOHN'S COLD))
         (BEFORE (ALL-OF TODAY)
                 (BY-AMOUNT (WEEKS 3) (FUZZ (DAYS 4)))))
```

The size of the interval in the fuzz-expression is difficult to determine and is discussed later.

The interval representation type is defined as,

amount ==> interval-amount
interval-amount ==> (INTERVAL <interval> <interval>)

This representation was designed to represent sentences such as, "Two or three years after graduating, John went back to school." This sentence is represented as,

```
(TIME-OF (ALL-OF (JOHN'S GOING-BACK-TO SCHOOL))
         (AFTER (ALL-OF (JOHN'S GRADUATING))
                (INTERVAL (YEARS 2) (YEARS 3))))
```

The interval representation is very similar to the by-amount representation, so much so that there need not be two separate types. There are two types, however, because the fuzz-expression in the by-amounts type is considered only a crude approximation to what is meant by "fuzz" or "lack of exactness."

The fuzzy-amount representation is defined as follows:

amount ==> fuzzy-amount
fuzzy-amount ==> (FUZZY-AMOUNT <fuzzy-expression>)
fuzzy-expression ==> (<qualifier> <fuzzy-number> <time-unit>)
qualifier ==> ABOUT | NEARLY | SOMEWHAT-LESS-THAN |

---

3 Extending the time specialist to "understand" hours, minutes and seconds would not be very difficult.

SOME-WHAT-MORE-THAN | A-BIT-MORE-THAN | NIL
fuzzy-number ==> A-HALF | ONE | A-COUPLE | A-FEW | SEVERAL | MANY | PLURAL

The fuzzy-amount representation type is intended to suffice for temporal specifications such as example (4), "A few months from now, John will finish his thesis," which is represented as,

```
(TIME-OF (END-OF (JOHN'S WORKING-ON THESIS))
         (AFTER (ALL-OF TODAY)
                (FUZZY-AMOUNT (NIL A-FEW MONTHS))))
```

The qualifiers and the fuzzy-numbers are explained in detail in Appendix A.

C) The Representation of Fuzz

"Fuzz" is a term that describes the uncertainty about exactly when an event occurred, not the uncertainty of the event having occurred at all. The fuzz-expression is part of the "date" and "by-amount" representation types. The fuzz-expression is defined as,

fuzz-expression ==> (FUZZ <interval>)

This is interpreted by the time specialist as the extremes, plus or minus that the amount in the "by-amount" type or the date in the "date" representation can possibly deviate. This is a simple and useful scheme.[4]

Regardless of the representation chosen, however, there needs to be a way to combine fuzz from different temporal references. In following chains of events, for example, one needs a way of combining the fuzz of each expression. In the current implementation, this is done in the simplest manner, the fuzz-expressions are simply added together. This tends to overestimate the fuzz, however, this is seldom important enough to produce answers that differ from humans' responses. Probability curve representations solve this problem in that procedures exist for combining such curves in a "rational" manner.

---

4 More complex schemes involving probability distributions are reasonable alternatives.

D) Currently Unrepresentable Temporal Specifications

Because of the limited scope of this research, certain temporal specifications are unintelligible to the time specialist. Example (6), "John saw Mary a while ago" is one such temporal specification. The problem is that the phrase "a while ago,"[5] is extremely context-dependent. For example, if John and Mary live together, the meaning is quite different than if they live in different countries. The frequency of their being together is only one consideration; if the speaker and listener both had seen John and Mary a few hours before the sentence was spoken, a different meaning should be inferred than if that was not the case. Also, it seems best to put off until as late as possible the evaluation of the interval involved, since more may be learned about the context after accepting the sentence but before needing it for any inferences. In some sense, the time specialist can handle example (6) by paraphrasing the sentence as, "John saw Mary before now" and making no assumptions about the interval concerned. This is, however, only a partial solution since there are cases where a person will make certain assumptions about the interval (usually with a large fuzz factor) and use those assumptions in answering questions.

E) The Representation of Recurring Events

One very common and useful kind of temporal specification is the description of the time of recurring events. This type of temporal specification was not, however, incorporated into the time specialist. This is considered an improvement of the system that should be done, although none of the applications of the time specialist which were implemented would benefit much from it. This ability would be crucial to other applications, such as the reminder system discussed at the end of Chapter I.

---

5 There are many such phrases, a few of which are "recently," "a long time ago," "in a while," "often," "near" (some event), and "just after" (some event).

A representation was developed for recurring events and the changes necessary to the time specialist were considered. It is my opinion that this would only be a few weeks work to incorporate into the implementation. The representation designed is,

time-specification ==> occurs-specification
occurs-specification ==> (OCCURS <event-class> <occurs-expression>)
event-class ==> <event>
occurs-expression ==> <time-expression>

This is not very different than for the other representations, however, the differences become more apparent inside the date-expressions. Date-expressions are extended as follows,

date-expression ==> (DATE <date> <fuzz-expression>)
date ==> (<year-specification> <month-specification> <day-specification>)
year-specification ==> <number> | ANY | <predicate>
month-specification ==> <number> | ANY | <predicate>
day-specification ==> <number> | ANY | <predicate>

The major modifications are the additions of the word "ANY" and a predicate. "ANY" is to be interpreted as matching anything, an example would be the sentence, "Christmas is on December 25th," whose representation is,

```
(OCCURS (ALL-OF CHRISTMAS) (DATE (ANY 12 25) (FUZZ NIL)))
```

The "predicate" can be any LISP predicate that can be applied to that position in an item being matched. Examples are:

Elections are the first Tuesday in November.
```
(OCCURS (ALL-OF ELECTIONS) (DATE (ANY 11 (FIRST TUESDAY)) (FUZZ NIL)))
```

John was at camp every summer from 1960 to 1967.
```
(OCCURS (BEGINNING-OF (JOHN'S BEING-AT CAMP))
        (DATE ((BETWEEN 1960 1967) 6 21) (FUZZ (MONTHS 1))))
(OCCURS (END-OF (JOHN'S BEING-AT CAMP))
        (AFTER (BEGINNING-OF (JOHNS BEING-AT CAMP))
               (BY-AMOUNT (MONTHS 3) (FUZZ (MONTHS 1)))))
```

Every Monday John goes to class.
```
(OCCURS (ALL-OF (JOHN'S GOING-TO CLASS))
```

```
(DATE (ANY ANY (MONDAY)) (FUZZ NIL)))
```

In addition, a rather normal looking temporal specification can be a recurring event type, for example,

John's birthday is exactly one month after Christmas.

```
(OCCURS (ALL-OF (JOHN'S BIRTHDAY))
        (AFTER (ALL-OF CHRISTMAS) (BY-AMOUNT (MONTHS 1) (FUZZ NIL))))
```

The representation is pointless unless it can be used by the system. Suppose all the above examples had been added to the data base. Then if the time specialist were asked, say, "When is Christmas this year?," it would need to merge information that this year is, say, "1975" and the date in the occurrence specification for Christmas, "(ANY 12 25)," resulting in "(1975 12 25)." If asked, "Could John have gone to camp on June 12th, 1964?," the fetcher would return yes. The fetcher would need to be extended to, in addition to doing a simple fetch, to fetch for any time expression about John being at camp and then call the equivalence and containment modules, discussed in the next section. These modules would need to be changed to check to see if a predicate is in the date and if so apply that predicate to the value in the corresponding position. In this example, it would result in the predicate "(between 1960 1967)" being applied to "1964" and responding "true."

This scheme would treat recurring events as virtual statements of all the times the event can occur. Of course, in general, one can not add these explicitly, but by the use of the predicates discussed above, this can be simulated. None of the fetcher's methods should need any changes, due to the addition of recurring events. The analyzer's methods, however, are not so fortunate. They could be left alone, the price for that being much slower and unguided fetches. How each such method should be changed and whether new structures should be built for just this type of statements is not clear. Also translation between this type and the others would be a useful extension of the system.

Section II: The Equivalence of Temporal Specifications

Clearly, the time specialist needs to know when two temporal specifications are equivalent. Equality testing is provided by the LISP language, however the following non-equal expressions are considered roughly equivalent by the time specialist:

```
(1) (TIME-OF (ALL-OF COLD)
             (BEFORE (ALL-OF TODAY)
                     (BY-AMOUNT (WEEKS 3) (FUZZ (DAYS 7)))))

(2) (TIME-OF (ALL-OF TODAY)
             (AFTER (ALL-OF COLD)
                    (BY-AMOUNT (DAYS 21) (FUZZ (WEEKS 1)))))

(3) (TIME-OF (ALL-OF COLD)
             (BEFORE (ALL-OF TODAY)
                     (INTERVAL (WEEKS 2) (WEEKS 4))))

(4) (TIME-OF (ALL-OF COLD)
             (BEFORE (ALL-OF TODAY)
                     (INTERVAL (WEEKS 4) (WEEKS 2))))

(5) (TIME-OF (ALL-OF COLD)
             (BEFORE (ALL-OF TODAY)
                     (FUZZY-AMOUNT (ABOUT A-FEW WEEKS))))

(6) (TIME-OF (ALL-OF COLD)
             (BEFORE (ALL-OF TODAY)
                     (FUZZY-AMOUNT (A-BIT-MORE-THAN A-HALF MONTHS))))
```

The number of equivalent expressions of the same fact, "The cold was three weeks ago," is very large. One may wonder if the problem of determining the equivalence of paraphrasings is as bad here as with English. It is not the case, however, and the time specialist can rather easily determine the equivalence of these expressions.

The first step to solving this equivalence problem is to break the problem down into parts.

One part is based on the meaning of the relations, "before," "after" and "same-as." For example, the cold can either be before today or, equivalently, today can be after the cold. An additional part of the problem is that there may be synonyms for the same event in the items. Another part of the equivalence problem is concerned with the meanings of the time units, e.g., one week is equivalent to seven days. Another part is that two expressions can be in different representation types. This is resolved by the translation of one item into the type of the other and then testing for equivalence.

There is another aspect to equivalence that is connected with the notion of fuzz. If the pattern being used to retrieve items in the data base has a fuzz factor then one would like to consider those events that occurred sometime during that fuzzy interval as matching. This test of containment is intimately connected with the test of equivalence. For example, if searching for events whose time of occurrence matched,

```
"(BEFORE (ALL-OF TODAY) (INTERVAL (WEEKS 2) (WEEKS 4)))"
```

the time specialist would accept the items,

```
"(BEFORE (ALL-OF TODAY) (INTERVAL (DAYS 20) (DAYS 22)))" and
"(BEFORE (ALL-OF TODAY) (BY-AMOUNT (DAYS 15) (FUZZ NIL)))"
```

In addition, a near miss is treated differently than being far apart in time.

This containment testing is always done in the same representation type. If different types are involved, one is translated to the other type. The majority of the containment testing code is dependent on which type is involved. By-amount, interval, and date representation types are handled by computing the percentage of overlap that the retrieved item has with the pattern of the retrieval. For example, the pattern

```
(TIME-OF ? (BEFORE (ALL-OF TODAY)
                   (BY-AMOUNT (WEEKS 3) (FUZZ (WEEKS 1)))))
```

will match

```
(TIME-OF (ALL-OF COLD) (BEFORE (ALL-OF TODAY)
                               (INTERVAL (WEEKS 1) (WEEKS 3))))
```

with an overlap of fifty percent. The fuzzy-amount representation type is handled differently as described in Appendix A. These routines also leave behind comments about how well the items in question matched.

Section III: The Translation of Temporal Specifications

When many different representation types exist within a system, the need to translate or map one type into another exists.[6] This could be done by separate modules that can translate from one representation type to a different one. This would require the number of different representation types times one less than that number of different translating modules, or twelve in this case. This was cut in half by implementing only those modules that translate to or from the "by-amount" representation type. The other translations are done by simply translating the item to by-amount type and then translating the result into the desired representation type.

The translation between the by-amount type and the interval type is trivial. The translation between the fuzzy-amount and by-amount types is based on the meanings of the fuzzy-numbers and qualifiers.[7] The translation between dates and by-amount type representations provided the most trouble. Should the translation of, say,

```
(TIME-OF (ALL-OF COLD) (DATE (1975 6 15) (FUZZ NIL)))
```

into the by-amount type be,

```
(TIME-OF (ALL-OF COLD)
         (AFTER (ALL-OF (DAY-ZERO))
                (BY-AMOUNT (YEARS 1975.53) (FUZZ NIL))))?
```

I think not. The notion of "day-zero" is rather strange, clearly not something that the usual person uses. Explanations based on this concept, without translation to the more normal representation, would be strange to most people. On the other hand it would provide a uniformity to the

---

6 One would like translation to be performed as seldom as possible. In that way, the computation could be performed in the representation type of the question and facts involved.

7 More detail can be found in Appendix A.

representation by being able to treat "dates" as normal relative expressions when desirable. A reasonable extension to the system is provide both kinds of translation.

It was decided that the translation modules, that map to and from dates, would ask the fetcher for help. The translation from dates to a relative by-amount temporal specification is supplied a reference event. The above translation to by-amount, with respect to, say, "today" would be,

```
(TIME-OF (ALL-OF COLD)
         (BEFORE (ALL-OF TODAY)
                 (BY-AMOUNT (WEEKS 1) (FUZZ NIL))))
```

assuming that,

```
(TIME-OF (ALL-OF TODAY) (DATE (1975 6 22) (FUZZ NIL)))
```

is in the data base. If no reference event is provided to the translator, it will pick the special reference event closest to the event of the item being translated. The translater can operate in two modes, one where it only will do simple fetches to determine the amount, and the other where the full power of the fetcher is utilized.

The translation in the other direction, from by-amount type to date type, passes the "buck" to the fetcher. The fetcher is asked for the date of the event of the item to be translated and what the fetcher returns is what the translator returns.

Section IV: Question Patterns

Questions also need to be represented so that the fetcher knows what is being asked. Ideally, the syntax of the questions would be identical to facts, except for markers indicating the unknown or partially restricted parts. This was accomplished in the implementation, with a few exceptions noted below.[8]

A) The Normal Question Patterns

Question patterns are interpreted by the fetcher. If the pattern contains no missing parts, indicated by a "?," then the question is interpreted as "Is this temporal specification correct?" For example, the question, "Did John have a cold three weeks ago?" would be represented the same as the sentence, "John had a cold three weeks ago." The difference between them is only in what the time specialist is told to do with them, "believe" it or "verify" it. The answer in turn is simply the same item back again, or "nil," which has two meanings, one is that the answer to the question is "no" and the other is that the answer is "I don't know." In the cases, where one needs to distinguish between these cases, one can easily inspect the answer contexts.

Questions of the form "When did some event happen?" have the "time-expression" position filled by a "?". Questions, that are slightly more specific, like "What date did some event occur at?," or "How long ago did some event happen?," are represented by partially completed "time-expressions." For example, the question, "How long ago did John have a cold?" is represented by,

```
(TIME-OF (ALL-OF (JOHN'S COLD))
         (BEFORE (ALL-OF TODAY) ?))
```

if the, say, "fuzzy-amount" type is indicated, then it is,

---

8 There is no claim being made that this can always be done, only that for those types of questions which the time specialist needs to answer for the applications considered it was possible.

```
(TIME-OF (ALL-OF (JOHN'S COLD))
         (BEFORE (ALL-OF TODAY) (FUZZY-AMOUNT ?)))
```

A variant of this type of question, is of the form, "Did event-a occur after event-b?," where the amount of time between the events is of no concern to the caller. This is represented as,

```
(TIME-OF (EVENT-A) (AFTER (EVENT-B)))
```

Another common form of question is, "What happened during some interval of time?" This is the "all" events type fetch described later in Section II of Chapter III. This is sometimes representable following this scheme, for example, the question, "What occurred between 2 and 3 weeks ago?" is represented as,

```
(TIME-OF ? (BEFORE (ALL-OF TODAY) (INTERVAL (WEEKS 2) (WEEKS 3))))
```

There are cases, however, which this scheme cannot handle, and an extension to the question representation to cope with those cases is discussed in the next section.

B) A Fuller Question Pattern

There are three types of questions that require extensions of the representation of questions to be answerable. One of them is for questions with a restricted set of events (discussed later in Section II in Chapter III). The second type requires the use of the comparatives "Earlier-than" and "Later-than." And the last type requires the logical operators "And," "Or," and "Not."

The comparatives are used to represent questions like "Was event A after turning 13?" This question would be represented as,

```
(TIME-OF (ALL-OF EVENT-A)
         (LATER-THAN
          (AFTER (BEGINNING-OF LIFE)
                 (BY-AMOUNT (YEARS 13.) (FUZZ NIL)))))
```

The way in which it is answered is to create an temporary anonymous event corresponding to the point in time when the person in question was 13 and then doing a standard fetch of the form:

```
(TIME-OF (ALL-OF EVENT-A) (AFTER (ALL-OF EVENT-137))).
```

This construct becomes more useful when used in conjunction with the logical operators described below.

Logical operators are needed to answer questions of the type, among many others, "What happened between two events?"[9] That would be represented as:

```
(AND (TIME-OF ? (AFTER <EVENT1>))
     (TIME-OF ? (BEFORE <EVENT2>)))
```

If, instead of events, the question's form was, "What happened between two time expressions?," then the comparatives described above would be used. As one might suspect the meaning of "And," "Or" and "Not" differ greatly depending on whether the question has the event specified or not. With the event specified they have their usual logical meaning (which corresponds very closely to LISP's "AND," "OR," and "NOT"). However when the event is not specified (as in the example above) their meaning becomes the set operators of "intersection," "union," and "complement" respectively. These set operators apply to the events, not the time expressions, of course. "Complement" is defined in terms of the group given or the total list of events known to the system.

These extensions to the question pattern could profitably be extended to the representation of declaratives. The comparatives "Earlier-than" and "Later-than" would be convenient, however, by creating intermediate events one can always operate without them. The logical operator "And" is

---

9 A common special case of this type of question is, "Did event1 occur during event2?" The two events in this case are the beginning and end of event2.

implicitly what connects all the facts in the data base. The operator "Or" would be useful, but would require significant changes to the system, to be able to maintain these disjunctions. The issue of how complicated the time specialist reasoning should be allowed to be, appears here--would an explanation in terms of many disjunctions combined in complex ways be understandable? The negation operator "Not" would be very useful, the medical hypothesis matcher (described in Chapter IV) does have to occasionally re-compute things, since a negation of an item cannot be added to the data base.[10] This extension would require changes to many parts of the system, however, it would not create such complex explanations as "Or" would.

---

10 The addition of an item with the status "not to be believed" will not work. It does not mean that the system should believe the negation is true only that it should not believe the statement to be true.

TABLE OF CONTENTS CHAPTER III

I. Overview 66

II. The "All" Events Type Fetch 69

III. The Simplest Method 71

IV. Organizing by Dates 72

A. The "Know-date" Method 72

B. The "Compare-dates" Method 73

C. The "Find-event-and-reference-event-times-for-date" Method 73

D. The "Use-date-line" Method 74

E. The "Add-date" Method 75

F. The "Maintain-date-line" Method 75

G. The "Fix-date-line" Method 76

V. Organizing by Special Reference Events 77

A. The "Try-using-special-reference-events" Method 77

B. The "Add-special-reference-event" Method 78

VI. Organizing by "Before-after" Chains 80

A. The "Try-following-chains" Method 80

B. The "Look-at-chains" Method 81

C. The "Use-chains" Method 81

D. The "Maintain-chains" Method 82

E. The "Fix-chains" Method 83

F. The "Chain-too-vague" Method 84

VII. Last Resorts 85

A. The "All-out-search" Method 85

B. The "Go-thru-all-known-events" Method 86

VIII. Organizing by Periods 88

IX. More about Methods 90

A. The Order of the Method's Application 90

B. Possible Interactions between Methods 91

X. More about the Methods of the Fact Organizer 92

XI. Maintaining the Consistency of the Data Base 94

A. The "Look-for-inconsistencies" Method 94

B. The "Look-for-beginning-end-problems" Method 95

C. The Handling of Inconsistencies 96

D. Plausibility Checking 96

E. The "Re-think" Methods 97

Section 1: Overview

The time specialist performs three major functions. First, it accepts new temporal references, checking that the new facts are consistent and updating special data structures. The second function of the time specialist is to answer questions about the facts in the data base by fetching items from the data base and making inferences. The time specialist can answer three types of questions. These types are sufficient for the representation of the essence of a wide variety of superficially different requests. The three basic types are:

1) Did event X happen at time expression t? [1]
2) When did event X happen?
3) What happened at time expression t?

As we shall see later, the first two types of fetches are dealt with in a similar manner, while the last one requires special methods and data structures. The third function of the time specialist is to undo the effects of having believed those items whose status is put in doubt.

The performance of these functions is greatly improved by organizing the facts. The time specialist organizes facts primarily in the following three ways:

1) by dates and a temporally-sorted "date-line"
2) by special reference events such as "now" and "birth"
3) by before-after chains or sequences of events

Various methods use these special data structures to answer questions, other methods maintain these structures and others correct them when a fact is no longer believed. These organizations for the times of events are also a means of conceptually organizing the methods or functions of the time specialist.

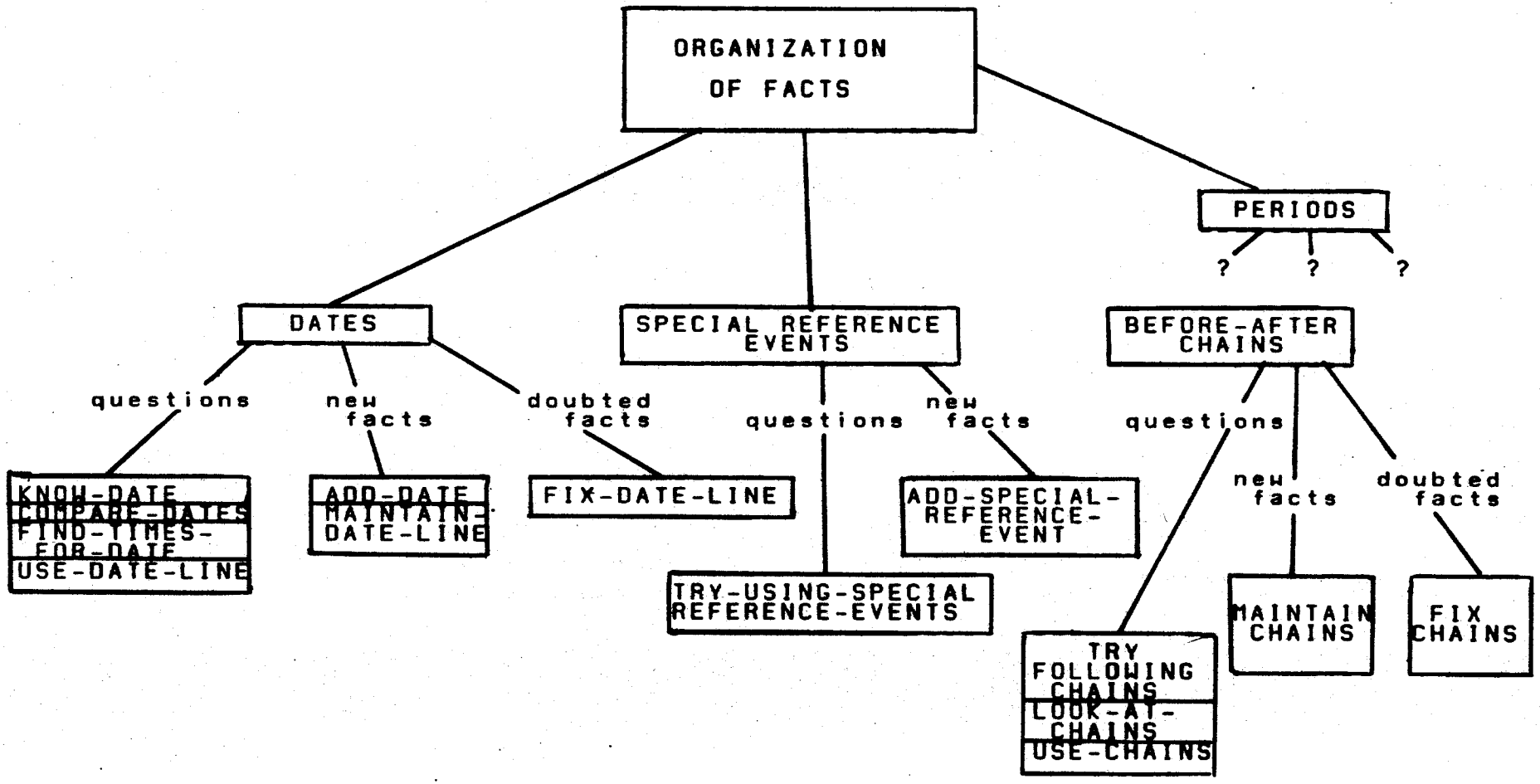
---

1 The time expression is varied enough to cover diverse questions such as, "How long did X last?" or "What happened between X and Y?"

The question-answering ability of the time specialist is embodied in a set of programs collectively called the fetcher. The tasks of the fetcher are to accept a pattern which specifies a particular question, to interpret the pattern to determine the type of question involved, and to select one or more appropriate methods for answering the question. The fetcher has a rather large number of methods at its disposal. Each method is an independent program which is designed to answer a particular kind of question by making use of a particular organization of facts in the data base. For example, there is a method which is designed to answer questions about dates by making use of the date line. If more than one method is available then they are applied one at a time until one of them can answer the question.

When the user wants to add a temporal specification to the data base the inconsistency checking methods are applied to that item, and if they accept it, the other methods of the fact organizer are applied one at a time. Associated with each fact are the names of those methods that were applied, so that were the same temporal specification added later only those methods that were not applied previously are applied. The methods that are to be applied are ordered in a list so it is very easy to enter an item into the data base and have only some of the permissible methods applied and in whatever order desired. If the item's status is later changed to "not to be believed" then those methods of the error corrector that undo the effects of the methods applied to that item are invoked together with more general error correcting methods. In the following sections the various methods of the time specialist are discussed. Many of these functions can be organized by the special data structure they are associated with is visually depicted in Figure 4. Before these methods are described a slight digression discussing the ways in which open questions are handled is presented.

put figure 4 on organization of facts and functions here



Organizations of Facts and Functions

Figure 4

This empty page was substituted for a blank page in the original document.

Section II: The "All" Events Type Fetch

I call open questions of the type, "What events occurred at time expression t?" "all" events type fetches. They are answered by the methods "Use-date-line," "Use-chains" and "Go-thru-all-events-known." An "all" events type fetch is slightly ambiguous, in that the caller may want all events that fit that pattern or only a few to think about and maybe more later. Even more of a problem is that there doesn't seem to be a human equivalent to the question, "What were all the events that happened between, say, two dates?" One does not literally mean "all" events; one generally means all or some events in some sort of semantic group, for example, medical symptoms. One normally does not expect an answer including events from many different semantic domains.

Two different mechanisms were developed to cope with these problems. One is to consider the fetcher for these types of questions as a process that returns what each method finds as it finds it and can be resumed to invoke other methods. For example, the method "Use-chains" may be invoked by the fetcher to find events that match the time expression of the pattern. Its answers will be returned to the caller, who can ask the same question again, in which case another method, if any are left, will try to find events and return those found. The other mechanism, special to "all" type questions, makes use of what group the events are to come from. Depending on the method, the time specialist would call a (hypothetical) group specialist asking if an event was in the group in question or what events are in this group. The group specialist may in turn call on the time specialist to help it group events on a temporal basis but this would be exceptional. As an example of event grouping, the question may have been, "What signs and symptoms occurred in the last two weeks?" "Going to a friend's birthday party," even if it occurred within the last two weeks would

not be an appropriate answer to this question. As described in the following sections some methods will be greatly aided by the group filtering of events while others will use it only to throw away inappropriate events already found to fit the question pattern.

Section III: The Simplest Method

The simplest way to answer a fetch is to know the answer explicitly already. The method "Known-already" handles this case. It does checking for equivalent means of expressing the same thing in one statement. For example, if the system knows that Event A is before Event B by 3 or 4 days, then "Known-already" would be able to answer the question, "Was Event B after Event A by about a half a week?" In addition, this method, like all the other methods, checks for synonyms of the event names. For example, the beginning of AGN's latent phase is by definition the same as the end of the strep infection. This is different from expressing this as a time expression where the temporal relation is "same-as." If someone says, "I had an exam the same day my strep infection began" this does not mean that the having of the exam and the beginning of the strep infection are the same, only that they occurred at roughly similar points in time. The operational differences within the system are significant, the most important being that the use of the temporal relation "same-as" can increase the fuzz or uncertainty of an inference, while synonymity does not. This method like the others will translate between different representation types when necessary.

Section IV: Organizing by Dates

One way that people and the time specialist organize events is by their dates. When a time of an event whose date is not explicitly known is entered into the system and the time specialist is so directed, then it attempts to discover the date. If a date is found it is added to the data base and to a temporally sorted date-line. Dates are very useful for easily determining the time of event relative to some other event. The date-line is used to answer questions about what happened during a particular interval (the "all" events type fetch). The date-line, were it complete (i.e. the dates of all events are either known or computed), would provide a fast access to all events that occurred within a specified interval. Unfortunately the date-line usually is not complete and so provides access to many, but not all, events and other methods and data structures are necessary for questions of this type.

The following sections describe the methods for answering questions involving dates, methods for deducing dates and maintaining the date-line, and a methods for correcting the date-line if it is based on no longer believed facts. It should be remembered that enough information to deduce dates is not always available, in which case the fetcher will use other methods and organizing structures.

A) The "Know-date" Method

When the time of an event relative to another event is desired the "Know-date" method will quickly deduce the answer or fail thereby allowing other methods a chance at solving the problem. This method does a simple fetch (the same as what the "Known-already" method above does) of the dates of the event and reference event and then subtracts them. Like many others, this method

has a version which enhances this simple fetch by applying the full power of the fetcher recursively. Because the time specialist is recursed by these methods they are called recursive methods, however they need not call themselves recursively for they may not be applicable during the second call to the fetcher. "Try-finding-dates" is one such "recursive" method. It has the same logic as "Know-date"; however, in getting the dates it recurses the entire fetcher with the sub-problems of finding the dates of the event and reference event. This method recurses the whole system, but will not be called upon a second time since it is applicable only for "relative" time expression fetches (those with both an event and a reference event). Hopefully some other methods will succeed in finding the dates.

B) The "Compare-dates" Method

This method applies only when one does not care about the temporal distances between events, only about the temporal ordering between the events. Basically this method finds the date of the event and the date of the reference event and compares them to see which is later (or earlier). This is similar to "Know-dates" except that the dates are compared rather than subtracted. There are two varieties, one does a simple fetch and the other recurses the fetcher to find the dates of the events. Notice that this method could have been implemented by calling "Know-dates" and then throwing away the amount, however that didn't seem to be a very natural or efficient way to solve the problem.

C) The "Find-event-and-reference-event-times-for-date" Method

This method is invoked only if the date of an event is desired. The idea is simple; to find the date of an event, find a relative time expression of the event of the question pattern, and then

look for the date of that time expression's reference event and add or subtract the amount of the time expression. For example, suppose the system had previously been told (or deduced),

(1) A week after the sore throat I had head aches, and
(2) The sore throat was March 2f, 1979.

To answer the question, "What date were the head aches?" "Find-event-and-reference-event-times-for-date" would find the first fact, then look for the date of the sore throat and find the second fact. It would then add one week to the date of the sore throat and return the answer. There are four variations of this method. One where all the fetches are simple, one where the finding of the date of the reference event recurses the system, one where the finding of the original time expression recurses the system and one where both fetches are recursive. The latter two are of dubious value since they usually require much computation because the reference event is not specified requiring those methods that search through the entire data base to be called upon.

D) The "Use-date-line" Method

To answer an "all" events type fetch "Use-date-line" tries to calculate the dates involved in the pattern (if they are not already there) and then looks for the portion of the date-line that lies between those dates. For example, if asked, "What happened since the sore throat?" "Use-date-line" will try to find the date of the beginning of the sore throat and today's date and then reply with the events that lie between those points on the "date-line." It then throws away events that are in that portion of the "date-line" that are not of the correct group (if there is one). This is unfortunate since it is more natural and efficient never to have considered those events. One solution is to have the "Maintain-date-line" method interact with a group specialist, thereby enabling the construction of many "date-lines," each containing events only of the same (or closely related) groups.

E) The "Add-date" Method

The function of deducing and adding the date of an event to the data base is performed by the method "Add-preferred-type" which adds to the data base items corresponding to the incoming items, except they have been translated to a preferred type. If the preferred typed is "dates," the translator [2] calls the fetcher and the date is obtained by one of its methods. The preferred type is chosen by the user of the time specialist. If an "Add-date" method is desired then the preferred type merely needs to be set to "date" and the "Add-preferred-type" method activated. The function of adding preferred types could have been performed by the parser in most cases. In the interests of modularity, however, it was decided that the parser should choose the representation type on the basis of the natural language input. If, for example, the input contained the expression, "a few weeks" then "fuzzy-amount" would be a reasonable choice of representation type.

F) The "Maintain-date-line" Method

The date-line is represented by a list of lists. Each list consists of a date, a pointer to the time specification that indicated that date, and the event type. To maintain this date-line new facts when entered into the system, are checked to see if they are "date" types. If they are and they have not already been entered into the date-line then "Maintain-date-line" creates an item or two and inserts them into the date-line. When the incoming fact is fuzzy the upper and lower bounds for the date are computed and each are inserted into the date-line. Thus fuzzy dates are essentially represented as an "interval" as opposed to a point on the date-line. Items are inserted into the date-line so that the date-line is always temporally sorted.

---

2 A description of the translator can be found in Section III of Chapter II.

C) The "Fix-date-line" Method

This method is invoked by the error corrector to keep the date-line correct when a fact's status is changed to "not-to-be-believed". Since one of the elements of an entry in the data-line is a pointer to the fact that it depends on, "Fix-date-line" can easily update the date-line. It goes through the date-line looking for entries whose fact pointer corresponds to the doubted fact, removing them. These pointers were included in the date-line for ease in use, however, in this case and in many others, it is clear that keeping the basis for a deduction is very useful when old facts become doubted.

Section V: Organizing by Special Reference Events

Another way of organizing the time of events in one's memory is to link the events to "special reference events". Special reference events are events that are often referred to in giving the time of another event. Common special reference events are "birth" and "now". Each application will typically have its own special reference events, such as the onset of the illness in medical applications. The method "Try-using-special-reference-events" tries to use these common links in answering fetches. The fact organizer's method "Add-special-reference-event-link" creates these links so that they will be even more common. The choice of which events are special reference events is presently made by the user of the time specialist. The list of special reference events could, however, easily be determined by the time specialist based on how often events are referred to by other events.

There is an interesting analogy between the time specialist's special reference events and Minsky's notion of capitals discussed in his frame paper [Minsky 1974]. He makes the analogy between a frame system and the roads or airline routes. One does not in general know how to get from one place (event) to another (reference event) directly. Instead one knows how to get from one place to a major "capital" (special reference event) and how to go between major capitals.

A) The "Try-using-special-reference-events" Method

Suppose the system had be told, "Three weeks ago John had a cold, and two weeks ago it was over." To answer the question, "How long did his cold last?" it seems natural to simply subtract two weeks from three weeks and respond "one week" rather than calculating the dates of the beginning and end of the cold and then subtracting the dates. And clearly there are many cases

where one does not have a choice. Suppose in this example the day the sentence was spoken is unknown.

"Try-using-special-reference-events" tries to find a reference link (using a simple fetch like "Known-already") from both the event and the reference event to the same special reference event and then subtracts the amounts involved. If it fails in doing this, it looks for any special reference event link and then tries to find the distance between the special reference links. For example, one may know that, "Three years ago, John graduated from college, and John started college at the age of 18". Then to determine how much time John spent at college, we notice the special reference event link between the beginning of college and birth, and another link between the end of college and now. We next try to find the distance between the two special reference events, age in this case. Typically the links between special reference events do exist because of the importance of each.

There is a "recursive" version of "Try-using-special-reference-events" which is called, "Try-finding-special-reference-event-links". This one recurses the system to find the relationships between the events and the special reference events and if necessary the relationship between two of the special reference events. This could at the worst result in two times the number of special reference events plus one number of calls to the fetcher. So that this method will not be called recursively on the second call to the fetcher this method will not be applied when one of the events is a special reference event.

B) The "Add-special-reference-event" Method

This method creates links to special reference events if none exist. "Add-special-reference-

event" first checks to see if the event of new fact being entered into the system is already referencing some special reference event. If it is not, then it attempts to find the special reference event nearest to the event of the entering fact.[3] If "Add-special-reference-event" succeeds then that link is added to the data base. Notice that this method will not operate recursively after it adds the new link to the data base, since this new link already references a special reference event.

---

3 Chosing the special reference event by its vicinity to the event of the entering fact is an heuristic that is usually right, but clearly other semantic features should be taken into account.

Section VI: Organizing by "Before-after" Chains

Another form of organization employed by the time specialist uses the notion of "before-after" chains. Such a chain occurs often in a story where the principal events form a sequence. For example, suppose the system had been told

1) Event A was a week before event B.
2) Event B was three weeks before event C.
3) Event C was a month before event D.
4) Event A was two months before event D.

The time specialist would create the chain "(Event A, Event B, Event C, Event D)" and would note that in following the chain it can take the short cut from Event A to Event D.

In the time specialist the construction of chains is performed by the method "Maintain-chains". Various methods of the fetcher use these chains; they are "Try-following-chains," "Look-at-chains," and "Use-chains". The first two are able to answer questions where the event is specified, the last one tries to discover some of the events that occurred during a given time expression.

A) The "Try-following-chains" Method

"Try-following-chains" tries to use these before-after chains to find relationships between events. Its basic operation is to look for a chain that contains both the event and the reference event, then try to find short cuts in following that chain between the events, and then combine the temporal amounts between successive events in the chain. There are basically two kinds of short cuts. One is when the system had been told about, or deduced, a link between events in the chain such as the link between Event A and Event D in the previous example. The other short cut makes

use of the special reference event concept. "Try-following-chains" looks for links to special reference events along the chain and tries to find a path that links to a special reference event and then back to the chain. If the linking back to the chain fails then a link from the special reference to another special reference event is searched for and another attempt to link back to the chain is made. One reason why linking through special reference events is important is that in following long chains with fuzzy information considerable inaccuracy is introduced. Links between special reference events, however, are often known with great accuracy. (For example, age, the "distance" between birth and now, is one such link.)

B) The "Look-at-chains" Method

This method is appropriate only when the question pattern asks about the relative order of events not the temporal distance between them. This method simply looks at the before-after chains for a chain with both the event and the reference event. It then does a special check for the case that the chain consists of only events that occurred at the roughly the same time. If it is not the case, the answer to the fetch is implied by which event comes first in the before-after chain. This method is similar to "Try-following-chains" but is not concerned with computing the distance between the events.

C) The "Use-chains" Method

In a similar manner to "Use-date-line" one can use before-after chains to help find some of the events that occurred within an interval. In the case that one wants to know what happened between two events, this method finds all the chains that contain both events and then returns those events between those events (minus those of the wrong group type). If the time expression of

the question is not of the "between two events" form but is between two time expressions[4] then one could create temporary events that are defined to have occurred at those time expressions, add them to the data base and then use the above procedure. This was not implemented in the present version of the time specialist because "temporary" events would either fill up before-after chains (and other special data structures) with events of only limited interest or require the creation of temporary copies of these structures for each question of this type. This method would perform better if the chains were grouped together semantically by the following method, "Maintain-chains," interacting with a group specialist.

D) The "Maintain-chains" Method

This is the most complicated method of the fact organizer. It maintains a structure of before-after chains that is realized in the computer by a list of lists. Each list is of the form:

(<event-1> <event-2> ... <event-n>)

where event-i either immediately precedes event-i+1, temporally, or the two events occurred at the "same" time. In addition, there is a data base context called "Jump-links" that contains links between events in the same chain that are far apart. In terms of the previous example "Maintain-chains" would have created the chain, "(Event-A Event-B Event-C Event-D)". The item "(Jump-link Event-A Event-D)" would have been added to the "Jump-links" context. These jump-links are useful for finding the shortest path between two events in a chain.

Maintaining this structure is complicated, however, the task has little intellectual interest. Procedures must create new chains, merge old chains, insert events into chains, break apart old

---

4 Before (or after) one time expression can be considered a special case of two time expressions where the missing time expression is either the beginning or end of time.

chains, and add to the "Jump-links" context. As implemented this method has the property that the structure of chains it creates may differ depending on the order in which the facts are told to the system. This is a consequence of its local thinking (i.e. only what event immediately precedes or succeeds another). Were it to follow its own chains in deciding whether to link up old chains, the structure it creates would be identical for all permutations of the incoming facts. One defense of the present version is that humans, when building chains are influenced by the order of the facts. Another defense is that it avoids many costly deductions.

E) The "Fix-chains" Method

When a previously accepted fact's status is changed to "no longer believed correct," among other things, the chains should be corrected. This is handled by the error corrector's method "Fix-chains". In our previous example, suppose later that "Event-B is before Event-C" is doubted then "Fix-chains" will destroy those chains containing both Events B and C. If the chain can be easily saved as in the cases where both of the events in question are at the beginning or both are at the end of a chain, then the chain is shortened in the appropriate manner. If the events formed a "jump-link" then that also would be destroyed.

A slightly better way of correcting chains is to break each chain involved so that the chain from the beginning to the first event in the doubted fact and the chain from the second event to the end of the chain remain. In our example, this would leave the chains "(Event A, Event B)" and "(Event C, Event D)" and the jump-link would be unaffected. Were it the case that a jump-link or a resulting chain was based on a fact that was deduced using the doubted fact, then that fact would also be doubted (as described in the last section of this chapter) and the incorrect chains

and jump-links based on it would be destroyed. The simplier approach of destroying chains, however, is the one that was taken. The unnecessary destruction of a few chains will, at the worst, result in only slightly less efficient fetches.

F) The "Chain-too-vague" Method

If someone were to tell a story where one event happened after another after another to a rather long length and one expects to need to know the amount of time between two events far apart on this chain, it would seem reasonable to ask the story teller to provide a few extra links. The point at which one considers the chain too vague is dependent on the length of the chain, the fuzziness on the links between successive elements of the chain, and the accuracy that one needs to know the relation between far apart events. Also there is the question of when to ask, when the first fact comes in that makes the chain too vague, or when one needs to use the chain and decides that is is too vague, or both. As presently implemented this method is very simple, it only applies when a fact is entered or re-entered causing a chain to grow and the criterion for vagueness is only the length of the chain.

Section VII: Last Resorts

When all else fails there are two "brute-force" methods left. "All-out-search" does a "breadth-first" search to find the time of an event relative to another event. When asked for those events that occurred within a particular interval, the "Go-thru-all-known-events" method asks the fetcher whether an event occurred within the interval for all known events. Both of these methods are very slow with normal data bases. Only if an answer is needed and the other methods have failed should they be used.

A) The "All-out-search" Method

The "All-out-search" method initiates an all out breadth-first search[5] from the event, and continues until the reference event is reached (or there are no more paths to search). Were it not for the peculiarities of the "Date" representation type, this method would be adequate for all fetches.[6] If there is someway to get from here (the event) to there (the reference event), it will find it. The chain that it finds, while it will be among the shortest possible, may be less than optimal. One objection to the chain is that it may follow much more fuzzy uncertain links than other methods resulting in such a vague answer that it may be worthless. Another objection is that the chain need not be a natural path for humans. It may go from some event to another that would seem odd to any human observer. "All-out-search's" explanations may be a little bizarre if it follows a chain that is far from obvious, and the events in the chain have no natural flow from one to

---

5 A breadth-first search in this case means to find all the events that are linked to the event, and then for each of them find what they are linked to, and then repeat for each of those events found.

6 It may, however, run very slowly, exploring many dead-end paths in the graph.

another. This is the only method that seems to depart from human behavior, as a consequence its reasoning can often be peculiar. Evaluating the worth of this method brings up of the issue as to what extent the time specialist should mimic humans.[7] Clearly at times all one cares about is the answer, and if this method can deliver when others cannot, it should be used, but if explanations or efficiency are important in the application, then this method should be used sparingly.

B) The "Go-thru-all-known-events" Method

This method is quite unlike the other methods for "all" events type questions. The others are driven by the question pattern, this one is driven by the "event" pattern. It finds all events in the group subtracts those already considered by previous methods (typically "Use-date-line" and "Use-chains") and then one by one goes through that list of events recursing the fetcher with each event (or reference event) specified in turn. In the case that the number of events in the group is rather small this method is quite reasonable, however when the group is large (or unspecified indicating all known events) it is very slow and often partial computations within each individual fetch are repeated since the fetcher does not know about what is to come, and does not make much use of what came before.

For example, suppose the fetcher was asked for all symptoms that occurred in the last week. "Go-thru-all-known-events" would then recurse the system for each symptom known (typically it would also subtract those symptoms already discovered by other methods). If the remaining

---

7 It should be pointed out that I designed the time specialist to reason, when possible, in ways that I think humans do. There are several reasons for this decision; among them are intelligible explanations and justifications of answers, ease of understanding, debugging and modifiability, many problems are such that a "right" answer is judged only by whether people accept it or not, and the possibility of shedding some light on how humans perform these tasks.

symptoms were "sore throat" and "edema," then the recursive calls to the fetcher would be, "Did the "sore throat" occur within the last week?" and "Did the "edema" occur within the last week?" These recursive calls to fetcher would be handled by other methods, since "Go-thru-all-known-events" is not applicable to questions where the events are specified.

Section VIII: Organizing by Periods

One useful method of organization of temporal facts that was not incorporated into the time specialist is the use of periods. History books are a good model for this form of organization. Events are grouped into periods which in turn are often grouped into larger periods (sometimes called "ages"). Statements can be made about the temporal relation between groups at the same level. These "higher-level" temporal specifications can be used in inferences about the lower level facts. For example, one may be told that John broke his leg when he was a child and again when he was a middle-aged man. If, in addition, it is known that childhood is usually from 0 to, say, 18 years of age and that middle age occurs between 40 and 65 then some vague answers to questions about the temporal relation between the two times that John broke his leg can be made by considering the periods involved. The specialist as presently implemented can easily handle this problem without periods, however, special structures for representing the hierarchical structure of the situation should aid in avoiding unnecessary searching and deductions in many cases. The way a method designed to make use of periods would solve this problem is by noticing that both times John's leg was broken were within periods. So the problem is rephrased in terms of these periods. Since periods are often used to answer questions and they are few in number it can be profitable to have already made many deductions about the relations between the periods. In the previous example, it may be that the relation between childhood and middle age has already been discovered.

Another use of periods is to restrict the search while answering questions. If it can quickly be determined from analyzing the question that only one or two periods need be considered then

events in other periods should not be considered. People when faced with questions, for example, about their childhood do not usually even think about events that occurred later in life. Clearly there is a need for another special data structure to capture the structure of periods.

As described in Chapter IV on medical diagnosis this use of periods is very common in medicine. Symptoms occur within phases, and phases occur within diseases. Had the time specialist been extended to include methods that made use of periods and maintained special data structures for periods the system would probably have functioned much more efficiently. This extension to the time specialist, however, will not increase the scope of problems the time specialist can solve, only its performance in those it already can solve.

Section IX: More about Methods

The fetcher must invoke the appropriate methods for a particular question is some order. The problems of determining that order are discussed next. Following that is a discussion of how the methods could interact and the benefits that would result.

A) The Order of the Method's Application

When the time specialist is asked a question, various methods are available. The first decision to be made is which ones are appropriate. In the current implementation this choice is dependent on two things. One is the representation type of the question pattern and the other criterion is whether the events are specified in the pattern or if the events fitting the time expression are desired (i.e. "all" events type questions). The next problem is to determine the order in which the methods are to be tried. Presently the decision is based simply on a default order.[8] Other factors, such as regularities in the data base and which of the fact organizer's methods are in effect, would, in an ideal system, influence the default order of the methods. In a more intelligent system, the question pattern would be analyzed more fully, and based on what is known about the events in the question pattern, the order of the methods would be changed for each fetch.

Another criterion for deciding which methods should be considered appropriate is the effort the caller wants put into a task. Clearly, the time specialist can not know how important any particular fetch is. For this reason, an effort measure can be passed to the fetcher. Presently this

---

8 Modifications to the time specialist so that it changes the default order based on its estimate of the cost effectiveness of the various methods are desirable. This was not done because of the difficulties in debugging code whose behavior changes in such a way.

effort measure is approximated by the amount of CPU time necessary to complete a fetch.[9] This would clearly be useful, for example in the present illness application, since certain expectations associated with a disease are rather unimportant (such as the duration of albumia in AGN) and others are so important that the system should do all it can to confirm or disprove (such as the duration of the latent phase of AGN).

B) Possible Interactions between Methods

The various methods that the time specialist invokes to answer questions do not communicate with each other. The efficiency of the time specialist could be significantly improved, however, if subsequent methods could make use of what previous, partially successful methods had discovered. For example, methods need not recompute partial answers if previous methods already have computed them. Also, the previously-invoked methods may have discovered some characteristic of the data base indicating that the order of the method invocation should be altered for this problem.

The pitfall that must be avoided in allowing methods to communicate is that they must remain independent to preserve the modularity and modifiability of the time specialist. The ideal medium of communication is the "answer-tree" contexts. The methods normally leave comments here about their success or reasons for failure. They could easily leave comments about partial answers that other methods could look at. No method need necessarily leave comments nor look at the comments of others, and modularity is preserved since the methods can make use of the information without regard to which method discovered it.

---

9 Other more sophisticated measures can easily be imagined, taking into account other machine resources and real time response.

Section X: More about the Methods of the Fact Organizer

When a fact is added to the system, many different methods can be applied to that fact. Which ones should be determined by the application, by the relative frequency of "fetches" and "adds," and by any biases to certain representation types or organizational structures For example, an application of the time specialist that required very many fetches and where dates were very common would have the methods that maintain the date-line and deduce the date of the event of the entering time specification activated. These methods of the fact organizer are ordered, though the ordering need only be a partial ordering. The "Maintain-date-line" method, for example, should always be applied after the "Add-date" method, though "Maintain-chains" and "Add-preferred-types" can be applied in any order.

Perhaps the most important method that is applied to incoming facts is "Look-for-inconsistencies" which looks for inconsistencies that the incoming fact might have with previously accepted facts. Before-after chains, the event-list, the date-line, and statistics are maintained by their respective methods. The three different types of time specifications that can be deduced and added to the data base are the preferred type, the duration, and the event relative to the nearest special reference event. In addition there is the method "Chain-too-vague" that complains when a before-after chain becomes too vague.

While these methods were designed with the idea that they be applied when the fact is being entered into the system, this need not be the case. After each method finishes it notes that it has been applied to that fact. At any time later the higher level system can ask the time specialist to think about a fact, and those methods that were not done earlier will then be invoked. For

example, if some module decided that a date-line would be very useful, it could turn on "Maintain-date-line" and re-think all the old facts so that the date-line will be as complete as possible. This seems reasonably common in human thinking, that the implications of a fact are sometimes made long after being told about that fact, and yet not in response to any particular question. Also in an application where the system would have spare time, it could use that time to deduce things that will speed up the fetcher later. Notice that this mechanism of remembering that a method was already applied to particular fact needs some intelligence to be sure that the world has not changed sufficiently to warrant the method being reapplied. Each method is responsible for determining if it should be reapplied to a particular fact if that fact is re-entered into the system.

Section XI: Maintaining the Consistency of the Data Base

In addition to accepting temporal specifications, the time specialist attempts to assess the plausibility of an incoming statement and its consistency with statements already in the data base. Consider the following facts:

(1) The cold ended last week,
(2) Three weeks ago there was a party,
(3) A couple days after the party the cold ended.

If the same cold and party are being referred to, the "fuzziness" of the expressions is not sufficiently great to account for the discrepancy among these statements. We would like the time specialist to recognize and correct the situation if possible. In the following sections, I will describe the methods that search for inconsistencies and the way the methods of the error corrector handles those found.

A) The "Look-for-inconsistencies" Method

The detection of inconsistencies like the one above is simple to implement. All "Look-for-inconsistencies" needs do is call the fetcher with the pattern being the new fact (this must be done before the new fact has been entered into the data base or deductions are based on it). Three outcomes are possible, one that the fetcher indicates that the fact was already known, two, that it didn't know enough to answer the fetch and the fact is accepted (it may be contradicted later but that should be detected by the application of this method at a later time or by the fetcher), and the third outcome is that enough is known to determine that the new fact is inconsistent with what is already known. In this last case, the inconsistency handler is called. Notice that an inconsistency need not be a real one, it may be that the events are being referred to are different than the ones known in the data base.

B) The "Look-for-beginning-end-problems" Method

Another possible inconsistency that the above check will not catch is that the beginning of an event cannot be after the end of that event. This is handled by asking the fetcher if the beginning of the event of the new fact is before the end of that event, and the results are interpreted as with "Look-for-inconsistencies". Notice that this method, like all the methods of the fact organizer, is optional and, in the time travel story understanding application, should not be applied to time-travel trips. Also the larger problem solving system using the time specialist may decide that time should not be spent checking for inconsistencies without reason for being suspicious.

This kind of explicit consistency checking has been criticized because such problems should be discovered while doing something constructive. The author fully agrees with this view and in the case of "beginning end" problems has a reasonable alternative. There are at least three times that this kind of inconsistency can be found in the process of doing some constructive reasoning. For example, while adding a new event to the before-after chains or date-line it would be very easy to consider the event's type and check while building these structures for this problem. An even more appealing example is letting the "Add-duration" method handle this. This method tries to determine the duration of an event and could very easily call the inconsistency handling routines if it discovers a negative duration. Notice that using a scheme such as these makes it more difficult to selectively "turn off" such checking. For example, if a trip through time caused an inconsistency of this type then the "error" would be noticed. A reasonable thing for the time-travel story understanding system to do in such a case is to ignore the complaint. A harder problem is how the more general kind of inconsistency discussed in the previous section can be detected in a "constructive" manner.

C) The Handling of Inconsistencies

After an inconsistency is detected the normal course for the time specialist is to engage in a conversation with user. First it tries to ascertain whether the same event is being referred to in both cases. If they are different events then the addition of the new item into the data base is aborted, and the item is modified to have a unique event name and is then re-entered into the system.

In the case where the events are indeed the same, then the situation is explained to the user and the user is asked if the new item is wrong. If so, the addition of the new item to the data base is aborted. If the new one is right, then the user is asked about each of the facts that are responsible for the inconsistent assertion.[10] If all are fine, then the inconsistency is allowed into the data base, and a certain amount of indeterminism is introduced into the data base. If, however, one of the items are no longer to be believed, then its status is changed to "not to be believed," and the error corrector's methods are invoked to undo the consequences of having believed that fact.

D) Plausibility Checking

Ideally the time specialist should complain about implausible items being entered into the system. A cold cannot have a duration of several hundred years, someone cannot gain twenty pounds in an hour, a person cannot do something before being born (if one is not considering time travel stories). But how can the time specialist detect such implausible or impossible items?

The time specialist alone cannot be expected to discover anomalies as those above, however, it could be expected to help. To discover that a several hundred year long cold is anomalous the

---

10 Since the time specialist associates with each deduced fact those facts used in the deduction, it is easy to find those facts to ask about.

time specialist could, when thinking about the duration of an event, ask some other module, perhaps a cold expert in this case, what the longest possible duration of the event is, and then complain if the duration is longer than that. To find fault with gaining twenty pounds in one hour, a hypothetical rate specialist would ask the time specialist whether the duration of the weight gain is greater than the minimum interval possible and then complain if it is too small. To consider someone doing something before they were born as impossible, again the time specialist would be called by some other module, perhaps an expert on the activities of animate objects, to determine if the time of the person's activity is before that person's birth and if so complains.

None of this has been implemented, it should be noted. It is not clear when these plausibility checks should be done, there may be too many to handle at the time of entry of the item. If the system is having problems, strange answers are being deduced, then plausibility checks of the items responsible may be called for. Few of these plausibility checks, if any, should be initiated by the time specialist, instead the appropriate expert must do the checking, asking the time specialist for help when needed. In the case where the time specialist should initiate the checks, a method "Check-plausibility" can be provided by the user. The point is that the assessment of the plausibility of a new item is to a large extent a function of the higher level problem-solving program, and the time specialist can only be expected to provide support for this activity.

E) The "Re-think" Methods

A very difficult problem is keeping the data base consistent when old facts are doubted. Various other facts may have been deduced using the doubted fact and they should, perhaps, also be doubted. This is handled by the error corrector's method "Re-think-those-dependent-on-it".

Also the doubted fact's source might not be the user, but was itself inferred from other facts. These facts should be reconsidered and the method "Re-think-those-it-depends-on" performs that function. In addition, answers to previous fetches become circumspect when the reasoning of the answer was based on, among others, the doubted fact. This last task is handled by the method "Re-think-answers-dependent-on-it".

These tasks are greatly simplified by the fact that when the time specialist makes deductions that are added to the data base, such as those done by the methods, "Add-preferred-type," "Add-duration," and "Add-special-reference-event-link," it notes those facts that were used in deducing the deduction. When the "Re-think" methods are invoked they look to see what facts were deduced from the doubted fact and the premises of the doubted fact, and then asks "Look-for-inconsistencies" to check out these facts. It may turn out that they are all consistent with the rest of the data base and all is fine; however, often the inconsistency handler will be called again, and some of those facts will be doubted, and the process will recurse by "Re-thinking" those newly doubted facts. Quite understandably, this entire doubting process can be quite slow. However, it will keep the data base consistent.

The method "Re-think-those-answers-dependent-on-it" behaves differently. It has no facts to doubt, only answers. The time specialist cannot know what was done with its answers, but it can interact with its caller, if the time specialist no longer believes its old answer. To discover those answers that are in doubt, it uses comments the fetcher left behind in the answer context indicating what facts the answer depends on. When it finds an old answer in doubt, it checks to see who the caller was. If the caller was the user, a dialog is initiated, explaining the anomaly, asking if the fetch should be re-run, and re-running it if desired. If the caller was the time specialist, or a

higher level module, the fetch is re-run, and if the result is different than the old answer, a comment is left to the caller in a context with the same name as the caller. The various components of the time specialist that call the fetcher do not, at present, use this comment, however, the medical hypothesis matcher does.

One may wonder how reasonable it is to have the error corrector do such a thorough job. The amount of computation involved in removing all the conclusions of no longer believed items is typically very large. Maybe it would be better to do only the easy most obvious kinds of corrections and let the system detect the rest as it goes. Clearly, this matches my intuition as to how people react to such situations. The system could fix a few important aspects and "keep an eye out for" various contradictions as it performs its normal tasks. When a contradiction is discovered the first thing to do would be to see if any of the facts involved depend upon a no longer believed fact. Then the error corrector could be called and the computation resumed. This is difficult to implement well, and more incorrect answers would flow from the fetcher, however, it may be the case that with reasonable size data bases the scheme the time specialist currently follows would be too computationally expensive, requiring a compromise.

TABLE OF CONTENTS CHAPTER IV

A. Introduction 101

B. Acute Poststreptococcal Glomerulonephritis as an Example 101

C. A Scenario 103

D. The Medical Hypothesis Matcher 105

E. Some Problems with the Design of the Hypothesis Matcher 107

F. Representation of Time Courses of Diseases 108

G. An Example of a "Frame" 110

H. Evaluation and Further Work 112

A) Introduction

Time is an important component of the problem of taking the present illness of a patient. Many diseases have complicated time courses. Deciding whether or not a patient's history matches the time course of a suspected disease is a difficult problem. Of course, a physician does not isolate the temporal component when asking questions or making conclusions. For research and demonstration purposes, a medical hypothesis matcher was implemented. It uses the time specialist heavily; however, its medical knowledge is very limited. Its inputs are the times of major symptoms and phases of a disease; the problem of determining those symptoms and phases is left for others.[1] The purpose of this hypothesis matcher is to demonstrate the usefulness of the time specialist, not to propose a solution to the problem of automating medical decision making.

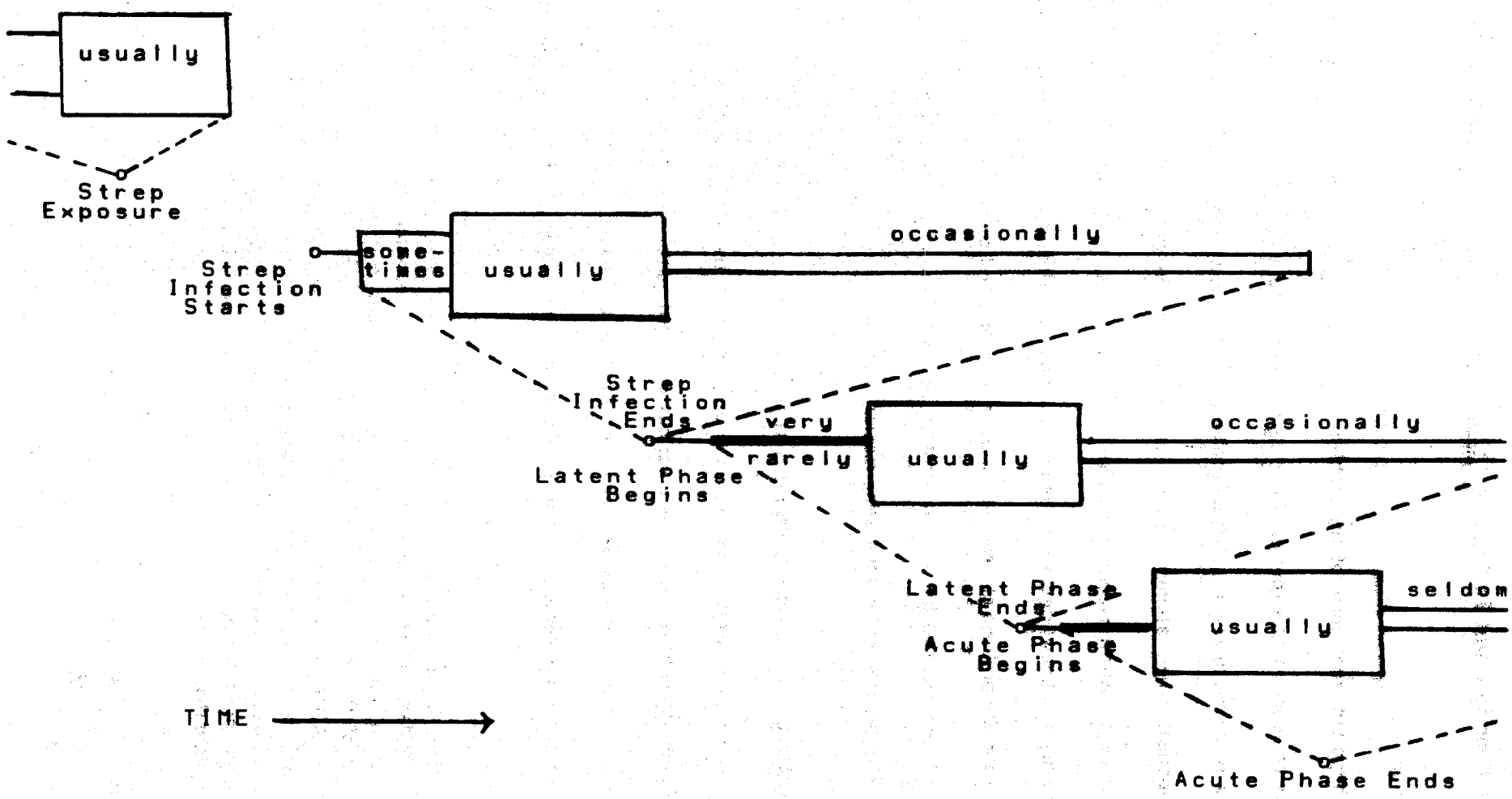
B) Acute Poststreptococcal Glomerulonephritis as an Example

To test the hypothesis matcher it was given a description of the time course of Acute Poststreptococcal Glomerulonephritis (AGN) and various concocted patient histories. AGN was chosen because it has a complex time course that is important in its diagnosis. AGN is a kidney disease that follows a prior streptococcal infection. The usual time course of the AGN can be summarized as:

> AGN usually begins with a strep infection. The strep infection usually lasts between 4 and 10 days, sometimes between 1 and 4 days, and occasionally between 10 days and 1 month. It is preceded by strep exposure that is usually between 2 and 7 days before the onset of the infection, sometimes it is as much as a month before. The "latent phase" of AGN, which begins when the strep infection ends, usually lasts a week or two, rarely lasts less than a week and occasionally lasts between 2 and 4 weeks. This phase is immediately followed by the "acute phase". The "acute phase" usually lasts several days, sometimes between one and a half and three weeks, and rarely less than 3 days or more than 3 weeks.

---

1 [Gorry 1974] describes how one might indeed attack these problems.



The Time Course of AGN

Figure 5

Medical literature describing the diseases is just as vague and fuzzy. The following are a few of the many sentences in a chapter on AGN by Schwartz and Kassirer [Schwartz 1971] that contain references to time (my comments are in italics):

> Whether or not specific antibiotic therapy is given, respiratory symptoms and fever disappear after a few days, and the patient feels entirely well. *This is a typical example of the fuzziness of the temporal aspect of medical knowledge.*

> After a few days, during which manifestations of the disease remain intense, the signs and symptoms gradually abate. *What does one mean by "gradually abate" and how should it be represented?.*

> Demonstrable type-specific antibody often disappears from the bloodstream within a few months, but it may persist for many years after the initiating infection has occurred. *The variability of the duration of signs and symptoms is often great.*

The time course of diseases is varied and complex. AGN is difficult to represent because of the many forks in the time course and its fuzziness. The problem of understanding and representing knowledge such as this is not limited to medicine but occurs in many other domains such as human social activities or long-term planning.

C) A Scenario

During diagnosis a physician must ask questions and match the responses to the time course. Decisions must constantly be made as to whether the patient's history is matching the physician's expectations, and if not whether some other hypothesis or disease should be considered. The following is a scenario (which one should imagine took place on July 27, 1975) in which a consultant is asked by a doctor to confirm or deny the diagnosis of AGN (the conversation is limited to the time aspect of the diagnosis and is in a stilted style to correspond more closely to the performance of the medical hypothesis matcher).

Consultant: What is the patient's history?

Doctor: The patient was born on May 27, 1941. The patient first noticed the symptoms associated with the "acute phase" of AGN nearly a week ago. About two weeks before the onset of the "acute phase" the patient got a strep infection.

Consultant: When did the "latent phase" begin?

Doctor: The "latent phase" began 5 or 6 days after the infection began.

Consultant: Is the AGN acute phase over?

Doctor: No

Consultant: When was the strep exposure?

Doctor: The strep exposure was two days after the patient's birthday.

Consultant: The AGN hypothesis matchs the patient's history, however strep exposure probably did not occur when stated. The fit of the duration of the "latent phase" is perfectly good. The fit of the duration of the strep infection is perfectly good. The fit of the duration of the "acute phase" is not too bad.

With the exception of understanding the English of the doctor, the hypothesis matcher can play the role of the consultant in the scenario. The hypothesis matcher relies heavily on the time specialist to match the patient's history to the time course of AGN, to make inferences as to when the various symptoms occurred and their temporal relationship and to be sure that the temporal aspect of the patient's history was consistent.

Neither the hypothesis matcher nor the time specialist is able to understand English, so the doctor's responses must be entered in a very stylized language. For example, when the doctor said, "The patient first noticed the symptoms associated with the acute phase of AGN about nearly a week ago," this was entered into the computer as,

```
(TIME-OF (BEGINNING-OF AGN-ACUTE-PHASE)
         (BEFORE (ALL-OF TODAY)
                 (FUZZY-AMOUNT (NEARLY ONE WEEKS))))
```

Sometimes the process of converting from English to the language of the time specialist, called parsing, is complicated. For example, when the doctor said, "The strep exposure was two days after the patient's birthday," the system was told,

```
(TIME-OF (ALL-OF STREP-EXPOSURE)
         (AFTER (ALL-OF (THIRTY-FOURTH BIRTHDAY))
                (BY-AMOUNT (DAYS 2) (FUZZ NIL))))
```

and

```
(TIME-OF (ALL-OF (THIRTY-FOURTH BIRTHDAY))
         (AFTER (BEGINING-OF (PATIENT'S LIFE))
                (BY-AMOUNT (YEARS 34) (FUZZ NIL))))
```

For a mechanical parser to this conversion, it would need to ask the time specialist how old the patient was, and perhaps call on a simple "birthday expert" for help.

A somewhat surprising fact is that for the hypothesis matcher to function as the consultant in the previous scenario, the time specialist is called upon about one hundred times. Some of these calls are to match particular expectations against the patient's history. Many are initiated by the time specialist itself, to check for inconsistencies or make common inferences based on the new facts. During the entire session, about seven or eight complicated methods of making temporal inferences were used repeatedly, several different organizing methods were applied to new items, and two different kinds of inconsistencies were checked for. The time specialist also recorded all these activities. These records are useful for handling inconsistencies, for improving the selection of the methods for inferencing, and for explanations.

D) The Medical Hypothesis Matcher

Clearly an understanding of the time course of a disease and the history of the patient are

important components of the task of medical diagnosis. To demonstrate the usefulness of the time specialist in this difficult domain, the simple medical hypothesis matcher was implemented. This matcher should be thought of as a component of a system that is competent in taking the present illness of a patient. This component is given an hypothesis to investigate and then its recommendation (e.g. "accept but ...", "reconsider only if nothing better turns up") is returned. Since the purpose of this implementation of the hypothesis matcher is to use the time specialist, medical knowledge that does not directly relate to time was excluded. Facts are given in terms of major symptoms, though clearly in a complete present illness program this would not be the case.

The hypothesis matcher is a process that when given an hypothesis, an effort measure, and certain minimums for the quality of matches, for the number of matches and for the importance of the expectations it should ask about, it will either accept, accept with reservations, reject the hypothesis, or suggest another hypothesis. In addition, it remembers the reasons for its recommendation in detail and a summary of its activities in a data base context. It also saves the state of the exploration so that the exploration of the hypothesis can be resumed.

The basic steps of the hypothesis matcher are:

1) initialize or resume old exploration

2) accept the patient's history

3) go through the facts of the story seeing what expectations they meet or fail to meet

4) ask about those expectations that are important and for which not enough information is available to confirm or deny

5) note those facts that are unaccounted for

6) summarize findings

If at any point during the matching a time pattern associated with a differential diagnosis is matched, then the process is stopped and a new hypothesis is suggested for further exploration. For example, if, while exploring AGN, the duration of the latent phase is two days or less then the exploration of the AGN hypothesis is stopped and the Chronic Glomerulonephritis hypothesis is suggested for exploration. When any essential expectation is not met the process is stopped with an appropriate message and it can only be resumed for that hypothesis if the facts are revised. The process is also stopped when the aggregate matching score becomes too small with the suggestion not to reconsider unless there is no better alternative. Otherwise the process continues and finally, accepts the hypothesis, with or without reservations.

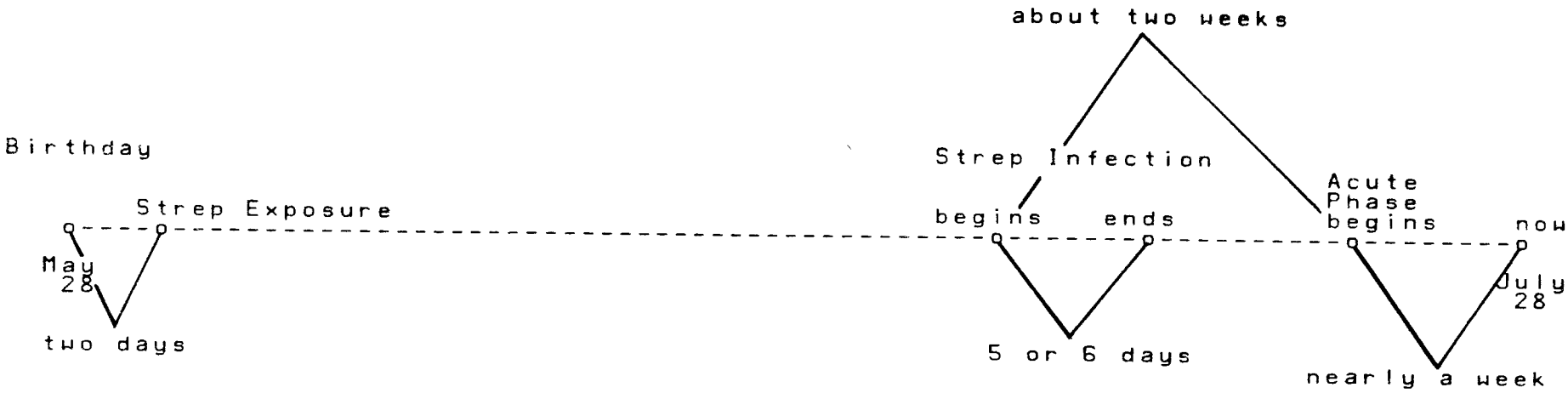
E) Some Problems with the Design of the Hypothesis Matcher

The purpose of the hypothesis matcher is to demonstrate the usefulness of the time specialist, not to do medical diagnosis. It was decided that the hypothesis matcher should be written as if it were a module of a larger present illness system. Problems of determining if, for example, the patient's brownish urine indicates hematuria, were assumed already resolved. The division between the time aspect of diagnosis, the medical aspect, and common sense is artificial. The main problem with designing the hypothesis matcher is to have it do enough to demonstrate the usefulness of the time specialist in medical diagnosis without attempting to solve the entire present illness problem.

Another major problem with the hypothesis matcher is deciding how good a match one has, or "scoring". This was done in a rather *ad hoc* manner; it was considered a problem the present illness system must resolve and that there are no special scoring features in matching the time course.

F) Representation of Time Courses of Diseases

In the previous chapters, representation schemes for temporal specifications were described. The representation of more complex objects, such as the time course of a disease, introduces new problems. Pictorially, one can visualize the time course of, say, AGN as was done in Figure 5. As can be seen the time course can easily be broken into four phases, the strep exposure, the strep infection, the latent phase and the acute phase. The length of the phases are variable, with rough probabilities assigned to various intervals. This portion of the diagnosis process can be visualized as trying to fit the patient's symptoms' course with the general time course of the disease. A pictorial representation of the history of the patient in the scenario is given in Figure 6.



The History of the Patient in the Scenario

Figure 6

Unfortunately, one cannot simply hand such diagrams to a computer program (or a physician for that matter) and ask if the patient's "diagram" fits the disease's. Some of the problems are:

1) Some of the patient's "diagram" is often unknown, or only partially known.

2) Some parts of the time course of a disease are very important, and others are only incidental. Looking at the diagrams would only be misleading in this respect.

3) Everything need not match perfectly, in order for the diagnosis to be positive, but deciding how well it need match is a complex process.

These problems were partially resolved by representing the time course as a frame system, similar to Minsky's concept of frames.[2] Frames provide a means to "chunk" knowledge, to put all the facts about, say, AGN's latent phase, into one place. Expectations, including their importance and likelihood, are the main ingredient of the medical hypothesis matcher's frames. Occasionally advice as to what to do when a mismatch occurs is provided. For example, the AGN frame, includes the following advice, "if the duration of the latent phase is too short (less than two days) then start considering Chronic Glomerulonephritis instead.

G) An Example of a "Frame"

The knowledge about the time course of AGN was structured into about 5 or 6 chunks or frames. The most important frame being the one that describes AGN using the other frames as sub frames. For example, this frame includes the statement that it is very important to confirm that the "acute phase" expectations are met. This effectively points to the sub-frame for the "acute phase" which must be confirmed. The "acute phase" frame contains statements about the likelihood

---

2 [Minsky 1974] describes these ideas in detail.

of various durations for the phase. These frames are intentionally simple, and much information that would be needed by a complete present illness system is left out.

The following is part of the frame used by the hypothesis matcher when AGN is suspected:

```
(GOBBLE AGN       ;AGN is the name of the frame being defined
        (SYNONYMS (BEGINNING-OF AGN-LATENT-PHASE)
                  (END-OF STREP-INFECTION))
        (SYNONYMS (END-OF AGN-LATENT-PHASE)
                  (BEGINNING-OF AGN-ACUTE-PHASE))
```

This says to consider "the beginning of the latent phase" synonymous with the end of the strep infection and that the end of the latent phase marks the beginning of the acute phase.

```
    (IMPORTANCE
        (PART-OF AGN AGN-LATENT-PHASE)
        ESSENTIAL)
```

This states that it is essential that one confirms that the latent phase occurred with a reasonable duration. The latent phase is itself a frame.

```
    (IMPORTANCE
        (PART-OF AGN AGN-ACUTE-PHASE)
        VERY-IMPORTANT)
```

It is very important that the acute phase of the AGN be established to confirm AGN.

```
    (LIKELIHOOD
        (* DURATION ALBUMIA
           (INTERVAL (WEEKS 3.) (MONTHS 6.)))
        USUALLY)
       (IMPORTANCE * NOT-VERY-IMPORTANT)
```

The above expressions mean that the duration of albumia is usually between 3 weeks and 6 months, however, this expectation is not very important. Please note that were it important it probably would deserve to be a sub-frame.

```
      (LIKELIHOOD
       (* TIME-OF (ALL-OF STREP-EXPOSURE)
                  (BEFORE (BEGINNING-OF STREP-INFECTION)
                          (INTERVAL (DAYS 2.) (DAYS 7.))))
       USUALLY)
      (IMPORTANCE * PRETTY-IMPORTANT)
```

The expectation, "the time of the exposure to the strep infection is usually between 2 and 7 days before the onset of symptoms" is pretty important.

```
(LIKELIHOOD
 (* TIME-OF (ALL-OF STREP-EXPOSURE)
           (BEFORE (BEGINNING-OF STREP-INFECTION)
                   (INTERVAL (WEEKS 1.) (MONTHS 1.))))
 SOMETIMES)
(IMPORTANCE * PRETTY-IMPORTANT)
```

The expectation, "the time of the exposure to the strep infection is sometimes between 1 week and 1 month before the onset of symptoms" is also pretty important. (Note that the above statements about strep exposure could have been part of the strep-infection frame or put together in a small frame.)

```
(IMPORTANCE (PART-OF AGN STREP-INFECTION) VERY-IMPORTANT)
```

It is very important to establish that a strep infection occurred at the proper time in diagnosising AGN.

```
(DIFFERENTIAL-DIAGNOSIS CHRONIC-GLOMERULONEPHRITIS
                        (DURATION AGN-LATENT-PERIOD
                                  (INTERVAL (DAYS 0) (DAYS 2))))
```

This says, "if the duration of the latent phase is less than 2 days then consider Chronic Glomerulonephritis.

H) Evaluation and Further Work

An interesting question is whether the time specialist was found adequate for the task. Some minor deficiencies and oversights were corrected. The data base is often significantly modified when an inconsistency is discovered, confirmed and corrected. So that the hypothesis matcher could be aware of such changes, a communication mechanism was implemented that leaves the higher level system a note about such changes. Minor modifications in the time specialist to improve the measure of the quality of a match were necessary. Certain inefficiencies were necessary because the time specialist, as implemented, lacks the ability to use negations and disjunctives of time

specifications, though it can answer questions consisting of these logical operators. Another deficiency of the time specialist is its lack of understanding of recurring events. As described earlier, this understanding could be incorporated into the present structure of the time specialist without significant revisions. As recurring events play a very small role in AGN this caused no problems for the test case. Because the time specialist was developed with medical diagnosis problems in mind, it is very good for this application, but this does not help support the claim that it is useful in many different domains.

The hypothesis matcher application addresses only some of the interesting roles that time plays in medicine. For example, in many diseases the recognition of recurrent symptoms is essential for diagnosis. The time specialist cannot recognize the intermittent occurrence of an event and would need to be extended for diagnosing such diseases. The determination of the time of the beginning or end of a phase based on the times of the events that compose it is a problem not dealt with by the hypothesis matcher or time specialist. For example, the beginning of the acute phase of AGN is usually marked by the onset of anorexia, hematuria, weekness, oliguria, proteinuria, hypertension, periorbital edema and headaches. The problem that the time specialist cannot handle is what if only some of these symptoms are known to be present, and those that are begin at different times. In many other domains this ability to group events into phases and make inferences about the group based on what is known about some of the events is also needed. Assumptions about the events based on the group also cannot be made by the time specialist, but are clearly useful. A deficiency of the hypothesis matcher is that it cannot recognize systematic change. For instance it is sometimes important to observe that a particular symptom is gradually abating. Perhaps this should be handled by an hypothetical "rates" specialist that is helped by the

time specialist. Despite these deficiencies the need to handle time effectively in medical diagnosis is evident, and the advantages of using a time specialist were demonstrated, though further testing is desirable.

TABLE OF CONTENTS CHAPTER V

A. The Problems of Understanding Time Travel Stories 116

B. Motivation for Exploring this Application of the Time Specialist 117

C. The Story Used to Test the Time Specialist 117

D. The Time Specialist's Understanding of the Story 120

E. Further Work 122

A) The Problems of Understanding Time Travel Stories

In order to test the various functions of the time specialist and to assess whether they are sufficient to deal with a variety of temporal references, the time specialist was applied to the problem of understanding time-travel stories. The time specialist was tested on its ability to understand a time-travel story in which time plays a very important role. Clearly a system that can understand the temporal aspect of time-travel stories can also understand more conventional stories.

Time-travel stories have one special kind of event that cannot be understood well by the time specialist as described. It is the trips through time, of course, that require special mechanism. Consider the problem of representing the statement, "On July 7, 1977, John travelled to 1066." The simplest way to handle this is to consider "John travelled to 1066" as the "name" of an event with no special characteristics. However, if the next statement was, "He then saw a battle," then the time specialist would assume incorrectly that the battle was in 1977.

Another way we can try to handle this problem is to consider the trip as beginning in 1977 and ending in 1066. We would need to be sure that the inconsistency check for occurrences beginning before they end is disabled for these events. This works fine, except what if the time specialist were asked what is John's age when he saw the battle? The time specialist might reason the following nonsense, "the date is 1066 and he was born in 1945 so he must be negative 879 years old." The problem is that from the time traveller's physical point of view,[1] very little time passed from the beginning to the end of the trip.

This last problem was resolved by extending the time specialist to take into account the point of view of a fact. The three kinds of viewpoints that the time specialist handles are:

---

1 One should not confuse the traveller's physical viewpoint with what he or she thinks. The viewpoints are primarily important for handling questions about age.

(1) the "universe" in which time flows its normal mundane course
(2) the set of time travellers taking a trip together
(3) the complement of the set of time travellers of each trip [2]

Both facts and questions have a viewpoint, the default being the "universe."

B) Motivation for Exploring this Application of the Time Specialist

Time travel stories were chosen as a test of the applicability of the time specialist to diverse problems because they contain all the elements of understanding normal stories plus the interesting addition of trips through time. People can understand time-travel stories without particular difficulty, so it was considered a deficiency that the time specialist could not. Also, the chronology of events often plays a more important and complex role in time-travel stories compared to more normal stories. One such story, "All You Zombies," by Robert Heinlein, [heinlein 1959] was chosen as a test of time specialist because the point of the story depends strongly on some complex temporal inferences.

A reason for chosing fictional stories is that the facts the system must deal with are not at the discretion of the researcher, but are given in the story. If there were any deficiencies in the time specialist's representation or inference mechanism, then they are more likely to be detected than if the facts were created by the researcher. Indeed certain deficiencies were discovered in representing "All You Zombies" that are discussed later.

C) The Story Used to Test the Time Specialist

The following is a very condensed version of the story, "All You Zombies." It is a paraphrasing of what the time specialist is told. My apologies to Robert Heinlein.

---

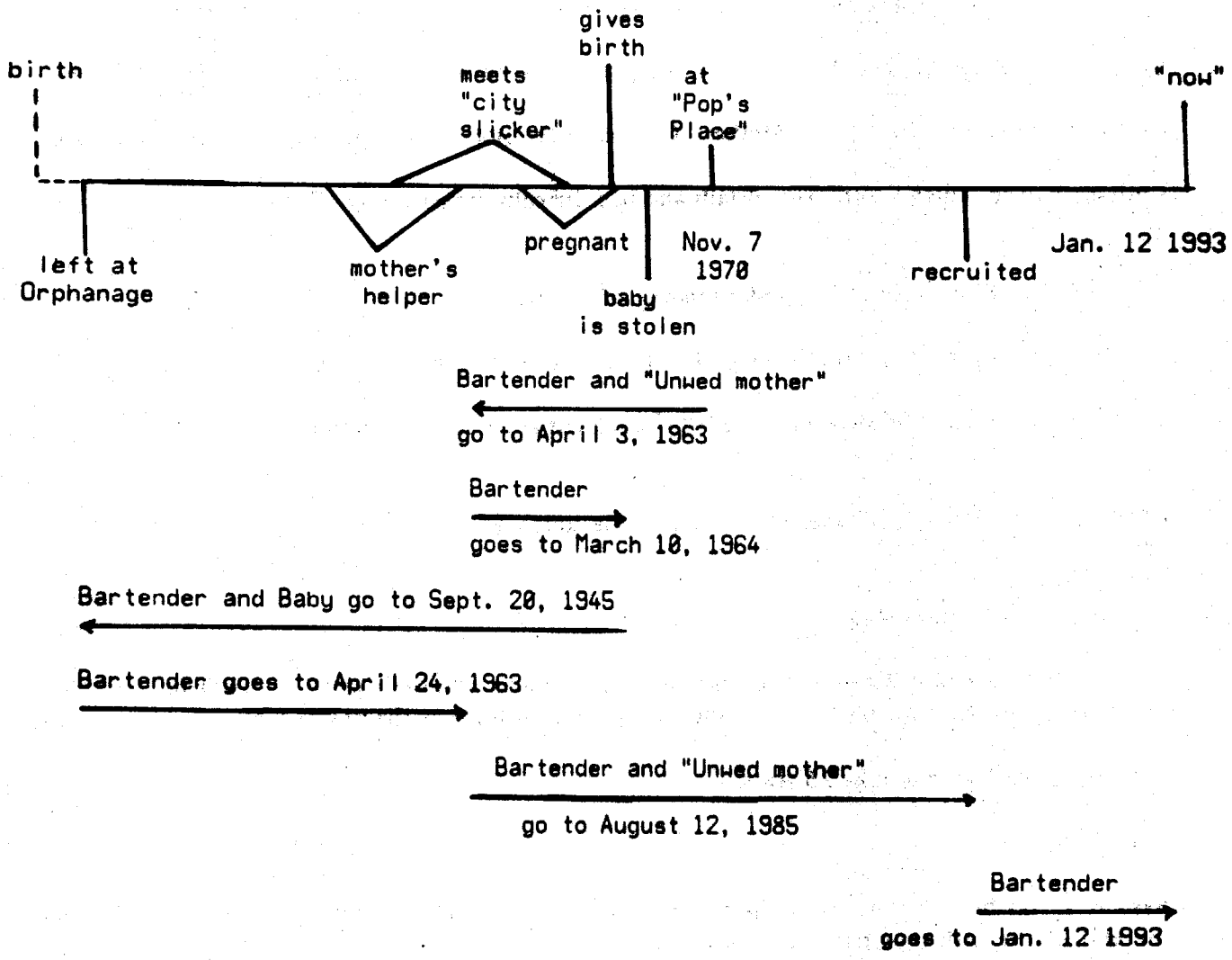
2 If the story contains only one time traveller or if it there are many and they only travel together, then this point of view is equivalent to the "universe" viewpoint.

On November 7, 1970 the Unwed Mother comes into "Pop's Place." He is 25 years old. He tells the story:

> I was left at an orphanage in 1945 at a month old. I was girl and at 18 I was placed as a "mother's helper" and then I met a "city slicker." Then one night "my pants did come off." I got pregnant and gave birth to a baby girl. I underwent a Caesarian and was changed to a male during the operation. Four weeks later the baby was snatched.

To help find the man who stole the Unwed Mother's baby, the bartender offers to take him back in time. They travel to April 3, 1963. The bartender goes off to March 10, 1964. The Unwed-mother meets a "nice girl," whom he gets pregnant. The bartender steals a baby and takes it to September 20, 1945 and abandons it at an orphanage. He then goes to April 24, 1983 and re-joins the Unwed Mother. He recruites him into the time service and leaves him off in August 12, 1985. The bartender goes back to January 12, 1993 and writes a report. It occurs to him that its been 30 years since he was recruited. (He gets undressed and looks at his Caesarian scar.)

The story is presented visually in Figure 7.



The "Time Line" of "All You Zombies"

Figure 7

The original story was read to a college class of about 25 students. Most people realized that the "nice girl" the male Unwed Mother met was himself years earlier. Fewer concluded that the bartender was the Unwed Mother after being in the time service and that the baby that the bartender stole was the Unwed Mother. Of course the story is longer, (about 14 pages) so the students had to contend with many details and had difficulty remembering all the facts.

D) The Time Specialist's Understanding of the Story

The following is a paraphrasing of a question-answering session after the story was entered into time specialist. The answers to the "why" questions are greatly pared down.

User: How old is the bartender at the end of the story? (This is from the bartender's point of view.)

System: I don't know enough.

User: Assume that the bartender's recruitment is synonymous with the Unwed Mother's recruitment and that the birth of the bartender is synonymous with the birth of the unwed mother.

System: 56 years give or take 2.5 years.

User: Why?

System: It has been 30 years since the bartender was recruited, which occurred at the same time as the end of the fifth time trip which occurred at the same time as the beginning of the fifth time trip ... which occurred at the same time as the beginning of the first time trip which was between 25 and 26 years after birth.

User: How many years have passed since he was born? (This is the same question as before except it is asked from the "universe's" viewpoint.)

System: 48 years give or take 6 months

User: Why?

System: Because birth was May 7, 1945 give or take 6 months and it is now January 12, 1993

User: Was the time of the Unwed Mother being left at the orphanage the same as the time of the bartender's abandonment of the baby?

System: Yes, the Unwed Mother was left at the orphanage 3.4 months before the bartender abandoned the baby give or take 6.3 months

User: Why?

System: The bartender abandoned the baby on September 20, 1945 and the Unwed Mother was left at the orphanage about June 7, 1945 give or take a 6.3 months.

User: What is the relation between the meeting of the (female) Unwed Mother and the "city slicker" and the meeting of the (male) Unwed Mother and the "nice girl"?

System: The Unwed Mother met the "city slicker" 1.5 give or take 2.1 years after meeting the "nice girl."

The last answer is evidence for the need for the inclusion of fuzz in the temporal representation of events. Once the hypothesis that all the characters of the story are the same person is entertained by a higher level understanding system,[3] it may want to confirm that it is possible that the "nice girl" is the Unwed Mother at an earlier age, and that the "city slicker" is also the Unwed Mother. The time that the Unwed Mother met the "city slicker," however, is vaguely given in the story as sometime after becoming a mother's helper which was when she was 18. Also until it is concluded that the baby that the bartender abandoned was the Unwed Mother, the date of birth of the Unwed Mother is only known to fall within a particular one year interval. Were one to consider only the most probable time of occurrence of these events, then she met the "city slicker" later than he met the "nice girl." It is only when the fuzz is considered that one can conclude that the events could have occurred simultaneously.

Another interesting answer the time specialist gave is to the question about how long ago the

---

3 This role was performed by the author. The problem of automating that process is beyond the scope of this research.

bartender was born. The answer is incorrect and within the framework presented it cannot be easily corrected. The problem is that the time specialist deduces that the bartender was born in 1945, since he was 25 in 1970. However, he was born in 1964 and as a one month old infant was taken to 1945.[4] The problem is that age is poorly represented for this application of the time specialist. Normally, the time specialist is justified in representing age as time since birth. Because of the limited value of a special representation of age to handle time-travel stories no changes to the time specialist were undertaken.

One problem, not peculiar to time-travel stories, that became apparent while representing the story is that one often uses real world knowledge to infer some of the intervals of time. Commonly, a time of an event is known, a series of events follow it without any information about their duration. To guess the interval between the two end events, one often fills in normal durations for the activities that intervene. Fortunately, in situations where so little information is available, only rough approximations to the interval are usually necessary. These rough, very fuzzy intervals were provided by the author in entering the story above.

E) Further Work

The time specialist was found adequate for understanding the story. The modifications to the time specialist that are desirable for this application are:

1) The addition of a special representation for age (discussed above)

2) An extension of the units of time acceptable to the time specialist to include hours and

---

4 Notice that telling the story in this manner; giving the date of the birth first, followed by the time trip, would cause this problem to vanish. It would also vanish if the question referred to the birth of the baby.

minutes.[5]

3) A mechanism to permit the time specialist to use general facts. For example, using the fact that the time between the onset of pregnancy and birth is usually nine months to infer that the onset of pregnancy was about nine months before a particular birth. The ability to recognize a particular event as an instance of a class would be needed for this and would be provided by some other mechanism that the time specialist would call upon.

If one were interested in improving the time-travel story understander, rather than the time specialist that it uses, then many problems present themselves. One problem of more general applicability is the problem of noticing temporal coincidences. The most common kind of coincidence is events that occur roughly simultaneously. In "All You Zombies," an intelligent system would have noticed the coincidence of, for example, time of the Unwed Mother meeting the "city slicker" and the time of the Unwed Mother meeting the "nice girl." This function could be performed by a method that, when desired by the higher-level understanding system, looks to see if any other event occurred at roughly the same time as the event of the incoming statement. More complex coincidences such as equivalent intervals between many events could perhaps be discovered in a similar manner. The problem here is to control such searching for coincidences so that too much time is not wasted. Of course, in a more complete time-travel understander, many of these coincidences would be suggested by other clues, and the temporal aspect would interact with the rest.

---

5 When they were needed for the understanding of the story, the appropriate fraction of a day was used.

TABLE OF CONTENTS CHAPTER VI

I. Recapitulation 125

II. Future Research 127

Section I: Recapitulation

The design and implementation of a time specialist useful in many different problem domains has been described. The thesis that a specialist whose knowledge is limited to the temporal element of problems can help larger more general problem solvers was demonstrated for two examples, medical diagnosis and time-travel story understanding. The major tasks of the time specialist are the reception of new facts, the answering of questions by making inferences, and the keeping of a consistent data base. These tasks are the same tasks required of an intelligent memory. An analogy between the time specialist's functions and AI languages which contain primitives for building intelligent memories can be drawn.

A representation of temporal statements and questions was developed. The variety of ways of expressing temporal specifications and questions is reflected in the four major representation types and the variety allowed within each type. The fuzziness or inexactness of the time of many events was handled differently depending on the representation type. The ways that the time specialist handles the interactions between the various representations was described.

Different ways of organizing temporal statements were presented. They include the use of dates, before-after chains, periods, and special reference events. Each of these methods of organization has its own special data structures and routines to maintain those structures. Many of the methods to answer questions are based on particular organizations of facts. The importance of organizing principles for facts was indicated.

The time specialist programs, which were discussed above, were designed to test the hypothesis that it is possible to encode knowledge about time "in general" in procedures which can

be invoked in the service of higher-level problem-solving systems. Certainly, the initial investigations of the use of the time specialist in several different problem settings support this hypothesis. On the other hand, it is fair to say that a final assessment of the usefulness of such an approach must await more extensive testing. This testing, in turn, will depend upon the construction of rather elaborate problem-solving programs, because it is only within the context of such programs that the time specialist can be properly exercised. The inclusion of a revised version of the time specialist in some rather large programs designed to solve medical decision-making problems is intended.[1] These programs will require extensive knowledge about time and, therefore, it should be possible to gain much more information about the long-term usefulness of this approach.

---

1 This is being planned by the Clinical Decision Making Group at MIT under which this research was carried out.

Section II: Future Research

Although a final evaluation of the time specialist must await further testing, it is possible at this point to identify a number of problem areas within the time specialist itself. Many modifications to the time specialist that would significantly improve its performance were discussed earlier. These involve extensions of the time specialist; no need to alter the basic structure was discovered. Some of these extensions to the time specialist are:

1) The extension of the representation to include recurring events. As described earlier the main problem here is how to incorporate the use of arbitrary predicates within dates into the methods and "thoughts" of the time specialist.

2) The ability to use facts about classes of events to infer the time of particular instances of the class. This requires a module separate from the time specialist that is an expert in classes and instances.

3) The integration of "periods" into the time specialist as another organizing principle. This is similar to the previous item except that time specialist should not require another module's help. The time specialist should be an expert in classes and instances when the classes are temporal.

4) The break-down of the date-line and before-after chains into smaller units whose events are semantically linked. The semantic grouping of events would be provided by a different specialist.

5) The use of more complete analysis of questions to determine the appropriate methods to be used in answering them. The problem here is to determine what additional information would be useful and how the various methods would benefit from the analysis.

6) The dynamic reordering of the methods in accordance with an estimate of their relative cost effectiveness. Statistics as to the performance of the various methods is kept so a crude estimate of the cost effectiveness of the methods is easily obtainable. A better measure would take into account the reasons for failure of a method and the type of question involved. The measure would also be improved by use of the analysis of the question discussed in the previous item.

7) Improvement of the mechanism by which methods interact so that each method can learn more about the causes of other methods failures. It was proposed earlier that this mechanism be a data base context, so that the methods remain modular and independent. One problem

here is how the methods make use of the information without becoming more complex programs.

8) A more refined representation of fuzz and better rules for combining fuzzy expressions.

The last five improvements will improve the performance of the time specialist on problems it already can solve and the others will increase the scope of the problems that the time specialist can handle. The later is essential for some applications of the time specialist that were considered, namely, the reminder system and the date-finding system.

One research question is how to handle context-dependent fuzzy expressions. How should "a while ago," for instance, be represented? How does real world knowledge constrain the meanings of such expressions? A related problem is how to handle expressions such as "the accident we saw a few minutes ago," where the time expression is used to identify an event whose time is already known to the speaker and hearer. A very interesting question is how the parser and the time specialist should interact and what sort of interface they should have. The parsing of temporal specifications and questions in a natural language into the language of the time specialist is a difficult problem that needs to be resolved if the time specialist is to be truely useful. One sub problem is to determine the linguistic clues that influence the vagueness or fuzz. For example, how does the expression "I think" preceding the statement influence the determination of the fuzz. The ways in which the use of tense influence the representation of sentences is an interesting research problem in itself.

Another, more speculative, research problem is to determine how subjective time differs from the time of the time specialist. When one remembers events that one perceived (as opposed to being told about them) are they represented in a way similar to before-after chains? If so, can the

time specialist use the length of such a chain in assessing the duration of the subjective sequence of events in question? Surely, the matter is not simple, but perhaps a good start on the problem may be made in this way.

Finally, the construction of specialists of other common elements of problems, such as distance and rates, presents many interesting possibilities. The structure and knowledge of these specialists will probably have many interesting similarities and differences. These specialists may interact in interesting ways. The most important possibility, however, is that the task of building intelligent machines will be eased significantly by the availability of a small community of specialists, experts in important aspects of problem solving.

TABLE OF CONTENTS APPENDIX A

A. Why Fuzzy Arithmetic 131

B. The Fuzzy Arithmetic Module 131

C. An Evaluation of the Fuzzy Arithmetic Module 135

A) Why Fuzzy Arithmetic

In everyday conversation one often uses "fuzzy numbers" when specifying the time of an event. Examples are, "That happened *a few* weeks ago" or "*Many* days later he died." In the medical application of time specialist,[1] for example, both the medical knowledge of the time course of diseases and the patient's history are often described using fuzzy numbers like, "a couple" and "several." The "fuzz-expression" portion of a "by-amount" type temporal specification can be used to handle these words. For example, the word "several" can be translated into "4 plus or minus 2." Or the "interval" representation type can be used, and "several" would be represented as "between 2 and 6." Indeed some people seem to deal with fuzzy numbers by translating to one of these other types.[2]

I believe that sometimes people do operate directly on these "fuzzy numbers." If John is given a few apples and then a few more, the answer, "John now has several apples," is reasonable. The problem solver who answers "several" is not aware of, nor gives any evidence that, he or she is solving this problem by converting to regular numbers, possibly maintaining an uncertainty or fuzz factor, and then converting back to fuzzy numbers. Even if this kind of processing is happening at an unconscious level, it still seems reasonable to model the gross behavior of adding "a few" to "a few" and getting "several."

B) The Fuzzy Arithmetic Module

A prototype fuzzy arithmetic module was developed, and is used by the time specialist when

---

1 See Chapter IV for more details

2 More details can be found in Appendix B.

reasoning about time-expressions of the "fuzzy-amount" representation type. This module can add and subtract "fuzzy words," it can compare "fuzzy time intervals," and it can translate between "fuzzy time intervals" and normal ones. This is accomplished by a combination of arithmetic tables and algorithms.

For example, suppose one were told, "A few days ago I had a sore throat, and nearly a half a week before that I had a cold." Then to determine how long ago the cold was, one needs to add "a few days" to "nearly a half a week." The fuzzy arithmetic module would respond, "several days" to this problem. The computation of this answer is complicated by the qualifier "nearly" and by the need to choose appropriate units to "think" in. (In this case, the choice is between "days" and "weeks.")

This problem of adding "a few days" to "nearly a half a week" is solved by the fuzzy arithmetic module as follows:

1) First the appropriate unit of time is determined. If the fuzzy number of the expression with the smaller units, after being converted to the units one greater, is less than one then the smaller units are chosen, otherwise the larger units. The expression with the smaller units, "a few days," is converted to weeks, obtaining "a half a week" by a process described later. Since "a half" is less than "one," the smaller units, days, is chosen.

2) Then the expressions are converted to this chosen unit, so that "nearly a half week" is converted to its equivalent in days, "a few days."

3) Next the qualifiers involved are considered; in this case none remain. If there were some involved, they would be combined in a manner described later.

4) Next a table of sums of fuzzy words is consulted and the entry "a few plus a few about equals several" is used.

5) The last result, "about several days" would then be modified by any qualifiers, however there were none remaining in this example.[3]

The conversion to desired units is aided by a table of each of the fuzzy words and its equivalent one unit greater. For example, "several days is equivalent to a bit more than one week" is in this table, but "many days is equivalent to about a half a month" is not. The last entry is computed, by converting "many days" to "about a couple weeks" which is turn converted to "about a half month." Here, as in the addition routines, qualifiers are not in the tables and they need to be combined after the table look up.

The combining of qualifiers such as, "nearly," "about," and "somewhat more than" is also aided by tables. If the quantities involved are close, such as "several days" and "one week," then the qualifiers are combined using a table. If, however, the qualifiers are, for example, "somewhat more than" and a few "a bit more thans" then the resulting qualifier might be "somewhat less than" and the fuzzy number is shifted up one. For example "somewhat more than a few" plus "somewhat more than a few" equals "somewhat less than many days." If one of the amounts is "much" or "very much" greater than the other then only that amount's qualifier is used. For example, "somewhat more than many days" plus "nearly one day" equals "somewhat more than many days."

This last example, points out an unusual property of fuzzy arithmetic; it is not associative. As an extreme example, suppose that ten "one day"s are added to "many days." "One" plus "many" is "many" so if the order of addition is to add each "one day" to the "many days" the result is

---

3 If different units were desired, for example, suppose the question was, "How many weeks ago was the cold?" then "several days" would be converted to "a bit more than a week."

"many days." This is not the case if the "one days" are added together first and then added to "many days" since "several" (or whatever the sum of ten ones is) plus "many" is not "many." Strange results due to this property are avoided by the time specialist. The time specialist saves up all the fuzzy expressions and then sends them all off to the fuzzy arithmetic module at once. The fuzzy arithmetic module sorts the expressions, smallest first, and then adds the smallest remaining expressions to the partial sum, thus allowing the frequent small factor to have some effect.

The comparison of fuzzy expressions is done by consulting a small table of the relationships between each fuzzy number and the fuzzy number greater than it. There are five types of relationships in this table, "very much greater than," "much greater than," "greater than," "little greater than" and "not comparable." To determine the relation between two fuzzy numbers, first the module checks to see if the numbers are identical. If not, then one of the numbers must be greater than the other and therefore in the table. The relation "not comparable" is needed because of the fuzzy number called "plural." It corresponds to the quantity in a sentence like, "I had a cold weeks ago." It is not comparable with "a few," "several" or "many." These different degrees of inequality are used by the time specialist to determine how well two fuzzy-amount expressions match.

The translation of fuzzy expressions to by-amount type expressions which consist of a mean and a fuzz factor, is accomplished by using two very small tables. One table corresponds to what the fuzzy words mean in by-amount terms. For example, an entry states that "many" translates to eight plus or minus four. The other table consists of the "meanings" of the qualifiers in terms of "by-amount" expressions. Basically the qualifiers are considered as percentages that modify the mean found in the other table. For example, "nearly a-few" translates to "3.15 plus or minus 1."

This is because "a-few" means "3.5 plus or minus 1" and "nearly" modifies the "3.5" by multiplying by .9. The qualifier "about" behaves differently, it increases the fuzz by a factor of twenty five percent.

The translation in the other direction--from by-amount expression to fuzzy expressions uses the same tables but is a little more complicated. First the translation table for fuzzy numbers is used to find the mean that differs the least (percentage-wise) from the amount in the by-amount expression. If the amount in the by-amount expression is too small or too large, then the expression is converted to better units if possible and the translation is tried again. Next the fuzzy number corresponding to the mean closest to the amount is found. The next problem is choosing an appropriate qualifier. The entry in the qualifier translation table that has a percentage, that is closest to the ratio of the amount in the by-amount expression and the mean, is looked for. If the ratio is very close to one then the problem is whether the qualifier "about" should or should not be chosen. This is decided depending on whether the fuzz of the by-amount expression is at least twenty five percent greater than the norm for the fuzzy number chosen.

C) An Evaluation of the Fuzzy Arithmetic Module

One may wonder how realistic this fuzzy arithmetic is. Does it correspond, in any way, to the way humans deal with fuzzy expressions? One criticism of this module is that it relies too heavily on tables. There are only seven fuzzy numbers and six qualifiers. Doesn't the size of the table grow very fast when new fuzzy numbers or qualifiers are added? The answer is yes they do, but very few additions are reasonable. Maybe two or three new fuzzy numbers should be added, such as "an awful lot," "a tiny bit" and "some." The number of conceivable new qualifiers is even

smaller--why add "almost" when "nearly" means about the same thing. Why add "I think that" when "about" means about the same? The tables are small and in a fuller system would not be much larger.

Another criticism of the fuzzy arithmetic module is that is doesn't use tables enough. It does too much computing to be a model of humans. It should have table entries for "almost a few" and "a bit more than one." The tables, however, would grow quite large if one were to avoid computation. The size of the tables for fuzzy expressions, as opposed to the present scheme of tables for fuzzy numbers and qualifiers separately, would be proportional to the number of fuzzy expression squared. The number of fuzzy expressions is equal to the number of fuzzy numbers times the number of qualifiers. To equal the abilities of the present system more than one thousand entries for each table would be necessary. Also, these huge tables do not match my intuition as to how I solve these problems. I think I add the fuzzy numbers independently and then combine the qualifiers.

One may criticize the fuzzy arithmetic module as not treating fuzzy numbers as more flexible, context-dependent expressions. For example, consider the expressions, "a few peas" and a "few watermelons." Most people would agree that there are more peas than watermelons. If someone said, "John just ate a few watermelons" and "George just bought a few watermelons" then one could safely assume that John ate fewer watermelons than George bought. A related problem is that fuzzy numbers are often used to identify things and when so used have even more leeway. For example, one can reasonably say, "remember that accident we saw a few minutes ago," even if it occurred a half hour ago. These are all instances of other knowledge playing an important role in the meanings of fuzzy expressions. Because of the strong dependence on other knowledge, the fuzzy arithmetic module cannot handle these kinds of expressions.

Another criticism of the fuzzy arithmetic module is that it should not be so symbolic. As pointed out in papers by Zadeh, [Zadeh 1973] one can do fuzzy arithmetic using numbers and constructs similar to probability distributions. Precise mathematical rules are available for operating on these entities and many fewer tables are necessary. My view of this approach is that it is "overkill." One does not need a precise mathematical theory of fuzzy arithmetic. Fuzzy arithmetic is only performed in situations where precision is unnecessary. If a simple, intuitive symbolic approach to fuzzy arithmetic works, why crunch arrays of numbers?

A final criticism of the fuzzy arithmetic module is that the rules and tables are too arbitrary. Why, for example, does "a few" translate into "3.5 plus or minus 1," why not "4 plus or minus 1.5"? The tables and rules, however, are <u>not</u> arbitrary, they correspond to my intuition. If the fuzzy arithmetic module performs in way similar to a human with a normal understanding of fuzzy expressions then it is a success. People do differ as to particulars of the table,[4] however, they do manage to communicate using fuzzy expressions. That is because the differences in judgements about the table entries are small compared to the leeway that is implicit in the use of fuzzy numbers. Fuzzy expressions are used because precision is not needed; being in the right ball park is all that is required.

---

4 Appendix B contains more on this.

TABLE OF CONTENTS APPENDIX B

A. People Doing Similar Problems 139

B. People Understanding Stories 139

C. Fuzzy Arithmetic Problems 141

D. Comparison with the Time Specialist's Answers 142

E. Further Work 142

A) People Doing Similar Problems

Many of the questions that the time specialist can answer do not have "correct" answers. The usual procedure for determining "correctness" of answers while doing this research was whether they "looked right." For example, if something happened a few days ago, and something else happened a few days before that, then did the latter event occur "several days ago," "quite a few days ago," "about a week ago," "a little less than a week ago," "six days ago," or what? Trusting my intuition is sufficient to determine if an answer is amiss, however a more satisfactory approach is to ask many people.

With this in mind a computerized questionaire was written.[1] One type of problem in the questionaire was to read a story and answer questions. The other type was a simple fuzzy arithmetic problem. Unfortunately, only 11 subjects took the test, all of them friends of the author, about half of which were connected to Computer Science and AI. Nonetheless, some interesting informal observations were made.

B) People Understanding Stories

Two stories were presented and after each one several questions were asked. The stories were always available to the subject. All the questions required some inference. The questions were presented one at a time and their order was randomized by the computer.

One story was intentionally full of fuzzy numbers. The two modes of organization in the story were before-after chains and the use of "now" as a special reference event. The story was:

> John met Mary several months ago. They dated a few weeks later. After a couple of months Mary moved to California. And John followed her out to California weeks later. They broke up and left California a bit more than a couple of weeks ago.

---

1 It need not have been computerized, the computer only performed bookkeeping and timing functions.

First, the most obvious finding is that this is a hard story to understand. The question, "How long ago did Mary move out to California?," for example, typically took more than half a minute and people would sometimes comment on the difficulty of these questions. It probably is not the fuzzy words that cause the difficulty, for as we shall see people have difficulty with the other story which contains only one fuzzy word. One interesting question that the answers to these questions shed light on is whether people operate upon fuzzy numbers or if they convert to numbers or intervals first. A little over a half of the answers were fuzzy expressions, very few were intervals or maximums, and the rest were normal numbers. An interesting fact is that those questions that involved operations on the fuzzy number "a-couple" had proportionally more normal numbers as answers, than the average. This happened consistently throughout the questionaire, indicating that perhaps some people do not consider "a-couple" as fuzzy, but rather operate upon it as if it were the number "2." Also the answers appeared to be rather consistent with each other, the largest variance apparently due to a difference of opinion as to what "several" means. As to be expected, those questions that required more facts to be combined, and those that required shifts from before-after chains to "before now" types of facts were more difficult to answer.

The other story consisting of many intervals is:

> When I was 8 or 9, I went to the dunes for the first time. Then for the next 5 or 6 summers I returned. The next few summers I stayed in the city. I think that the last time that I went to the dunes was when I was 20 years old. I'm 22 now.

This story appeared to be more difficult for people. It is not well organized, since there are two special reference events, "birth" and "now," and only a short before-after chain. Also the type of the amounts changes from intervals, to fuzzy numbers, to normal numbers (perhaps with implicit fuzz). The questions whose answers varied greatly, were those that required facts of different

types.[2] The question that gave the most difficulty was, "Assuming the story was written today, which summers did I spend in the city?" This problem requires many different types of facts, including dates.

C) Fuzzy Arithmetic Problems

The three kinds of fuzzy arithmetic problems that were presented to the subjects are "word problems," addition and subtraction problems, and comparison problems. An example of a "word problem" is "if I gave you a couple of apples and then gave you a few more, how many apples would you have?" The arithmetic problems were of the sort, "(almost a year) + (many months) = ?" The comparison problems asked the subject to compare say, "(a half a month) + (a couple weeks)" with "(nearly one month)." Most of the observations are the same as those given above. The answers are generally close to one another, the largest variation due to the fuzzy number, "several." One can deduce that some people consider "several" as "near 6" while others think of it as "3 or 4." For example, one subject replied, "8 days" to the problem, "(nearly a couple days)" + (about several days)." At other times the subject treated "a couple" as very close to "two," so we can reasonably conclude that "several" is roughly 6 for this subject.[3] On the other hand, a different subject answered that a couple plus a few equals 5. Since no subjects considered "a couple" as less than two, we must conclude that "several" to this subject is roughly 3.

---

2 This observation holds when the number of facts needed to infer the answer is about the same for the questions being compared.

3 When possible these conjectures as to what a fuzzy number means to a particular subject were checked for consistency with other answers of that subject. Notice that the numerical approximations of fuzzy numbers cannot be inferred when the subject answers with fuzzy numbers. In this case only consistency can be checked.

D) Comparison with the Time Specialist's Answers

The time specialist was presented with many of the problems in the questionaire. Its answers were not distinguishable from the other's answers. Some people answered with fuzzy numbers; so did the time specialist. Some people consider "several" to be between "a few" and "many," roughly 4 or 5; so does the time specialist. People had difficulty with problems where the facts were organized by different principles and were of different types; the time specialist took longer on those problems, too. The only reasonable criterion for correctness of answers is that it not disagree much with what people answer and it was fulfilled by the time specialist.

E) Further Work

Ideally one would like the observations discussed above to be based on a larger, more representative sample. Other questionaires could be designed to explore in more detail these findings. Questionaires could be written that determine the extent to which people are self-consistent in dealing with fuzzy numbers. The numerical approximations to fuzzy numbers could be determined by questionaires designed for this purpose. Clues as to how people organize facts temporally, could be found by presenting stories to people and then at various times later ask for a restatement of the story.

TABLE OF CONTENTS APPENDIX C

A. The Problem 144

B. An Implemented Partial Solution 145

C. Spin-off Benefits of the Interface Mechanism 145

D. Extensions to the Interface Mechanism 146

E. A "Better" Interface Mechanism 147

A) The Problem

One of the problems of building specialists independent of the programs that will use them is keeping the representation scheme, or language, of the specialists compatible with the overall system that uses the specialists. The usefulness of the time specialist would be lessened if it was based on a different representation scheme than the system that was using it. Of course, the time specialist must make some assumptions about the way in which facts are represented. It does assume that items in the data base are represented as lists and they can be retrieved by partially specifying the list.[1] The time specialist does not[2] make assumptions about the order of elements in any list, or sub-list. It does, unfortunately, make assumptions about the structure of the items. As shall be explained soon, this need not have been the case.

Take as an example of this problem, the top level of a temporal specification, which is defined as,

temporal-specification ==> (TIME-OF <event> <time-expression>)

However, the representation,

temporal-specification ==> (<event> OCCURS-AT <time-expression>)

may be more consistent with the representation principles of the system that is using the time specialist. The problem is how to implement the time specialist so that such changes in the representation can easily be accommodated.

---

1 The items need not be only lists, for example, packagers in PLASMA (see [Hewitt 1973] and [Smith 1975] for more details) are also compatible.

2 There are a few minor exceptions, however, these are due to historical accidents and laziness.

B) An Implemented Partial Solution

An interface mechanism enables the time specialist to be independent of the particulars of the representation's syntax. This mechanism allows one to declare the representation's syntax, and it provides macros that allows one to access the parts of a temporal specification by name. For instance, it allows access to the event of a temporal specifications using the macro called "Event." The time specialist was implemented using such macros, so that if, for example, the change in representation mentioned above is desired, one need merely make different declarations to the interface.

In addition to providing macros for selecting the desired portion of an item, the interface routines write macros that allow one to construct new items. For example, the declaration which could be paraphrased as, "to construct a temporal specification, insert the atom 'time-of' in the front followed by the 'event' and the 'time-expression'" will write three macros, "Event," "Time-expression, and "Construct-temporal-specification." "Construct-temporal-specification" expects two inputs and produces a list whose first element is the atom "Time-of" and the next two are the arguments to it. A more complete version, would also provide the two macros, "Replace-event" and "Replace-time-expression." When "Replace-event" (or "Replace-time-expression") is passed a temporal-specification and an "event" (or "time-expression") it returns a new temporal-specification with the new "event" (or "time-expression").

C) Spin-off Benefits of the Interface Mechanism

Two side effects of the use this interface mechanism are more readable code and optional type checking. The better looking code is due to the use of names to select parts, rather than using

LISP functions, such as "CADDR." The type checking is possible because the macros can be written by the interface to check if they are passed what they expect. The macros check to see if all constant atoms, like "Time-of," in the above example, are in the correct location, and that the length of the list is what is expected. The interface module can produce macros that do no checking, which is advisable after a module is debugged, since it will run faster without checking.

D) Extensions to the Interface Mechanism

One desirable modification of the time specialist would be improvement of the interface mechanism. Two problems with the present version should be resolved. One is that while the implementation is insensitive to the order of the elements, it is dependent upon the structure of the representation. For example, many routines expect, "temporal-specifications" as inputs, others expect "time-expressions" and others "amounts." Unless the alternate representation schemes have these constructs as entities, the time specialist will be unable to function.

The other problem with the interface mechanism is that it does not handle the representation of other sorts of knowledge. The answer contexts that contain the record of each attempt to answer a fetch, for example, are represented in a rather *ad hoc* manner. Many of the items are usable only for explanations to humans; in a better system, however, they would be usable by the overall system using the time specialist. It is important that the system that users of the time specialist understand its comments. The system should know, for example, why a particular fetch failed, if any inconsistencies in that data base were discovered and, if so, what was done about them. In order for these comments and notes to be usable by the overall system, they should be represented in a manner consistent with the representation scheme for the entire system. Since this

is not handled by the interface mechanism, the code that makes notes in the answer contexts would need to be re-written for each application that used a different representation scheme. This was not a problem with the applications of the time specialist described in this thesis, because they use a representation scheme consistent with the time specialist's.

The only aspect of the implementation that alleviates the problem of representing the notes and explanations of the time specialist is the use of a routine called "Remember" to add notes to an answer context. To integrate the time specialist into a larger system, there are two alternatives. One, change the code where each call to "Remember" occurs, two, modify "Remember" to translate to the desired representation before adding to the answer context.

E) A "Better" Interface Mechanism

A more radical solution to this interface problem would be to put the knowledge about the elements of a temporal specification and how to modify and construct them inside the temporal specification itself. This actor-like[3] view of temporal specifications, seems to solve most of the problems of keeping the time specialist's representation of knowledge consistent with its user's representation scheme.

---

3 See [Hewitt 1973] for more details

The Generation of English Appendix D

Clearly, when interacting with humans, the time specialist should not say things like,

```
(TIME-OF (ALL-OF COLD) (DATE (1975 6 16) (FUZZ NIL)))
NOT (DATE (1975 5 9) (FUZZ (DAYS 3)))
ARE YOU REFERRING TO THE SAME (ALL-OF COLD) ?
```

Instead it should say something like, "The date of the cold is exactly Monday, June 16, 1975 not about Friday, May 5, 1975 give or take 3 days, are you referring to the same cold?". This problem, like the inverse problem of translating from English to an internal representation, is considered at the fringes of the scope of this research.

The simple generation system incorporated into the time specialist is surprisingly adequate for most cases. The basic idea is that the relation, or the first element of every list or sub-list of an item should know how to say it's parts. It's parts in turn are either atomic, or else the first elements of them recursively know how to say their parts. For example, take the problem of "Englishifying" part of the above example,

```
(TIME-OF (ALL-OF COLD) (DATE (1975 5 9) (FUZZ (DAYS 3)))).
```

First "Time-of" is asked how to generate, and it does a little special case checking to see if it is a question or statement and if the type is "date" or not. It responds "The date of <arg> is <value>", where "<arg>" is the English version of the arg, "(all-of cold)", and "<value>" is the English version of value, "(date (1975 5 9) (fuzz (days 3)))". To get the English version of the arg, "All-of" is asked to generate and it responds with its arg, "cold". Then "date" is asked to generate, and it takes its arg without "Englishifying" it, and computes the day of the week using an algorithm for a perpetual calendar, and looks up the name of the month. It also checks to see if the "value" is nil,

in which case it precedes what the value returns with "exactly", otherwise it precedes it with "about". It lets its value, "(fuzz (days 3))" decide how to generate English on its own, and places what it returns on the end. "Fuzz" is then asked to generate English and responds with, "give or take <arg>". Then "Days" is asked to generate and it takes its "arg", "3", and turns it into a "prettier" number, that is one that has none, or only a few decimal places, and makes simple fractions like "1/2" when appropriate. "Day" also checks to see if the arg is less than 1, and if so returns the pretty number followed by "day", otherwise it is followed by "days". The result of all this is, "The date of cold is Friday, May 9, 1975 give or take 3 days". This generation scheme is both simple and modular, however, no similarity to human sentence generation is claimed.

General Works about Time:

[Fraser 1969]
Fraser, J.T. editor
The Voices of Time -- A Cooperative Survey of Man's Views of Time as Expressed by the Sciences and the Humanities
George Braziller, Inc. (New York) 1969

[Fraser 1972]
Fraser, J.T., Haber F.C., and Muller G.H.
The Study of Time -- Proceedings of the First Conference of the International Society for the Study of Time
Springer-Verlag (New York) 1972.

[Holubar 1969]
Holubar, J., translated by Barlow, J. S.
The Sense of Time -- An Electrophysical Study of Its Mechanisms in Man
MIT Press 1969

[Piaget 1971]
Piaget, J. translated by Pomerans, A.J.
The Child's Conception of Time
Ballantine Books (New York) 1971

[Whitrow 1973]
Whitrow, G.J.
The Nature of Time
Holt, Rinehart, and Winston (New York) 1973

Previous Related Work:

[Bruce 1972]
Bruce, B.
"A Model for Temporal References and Its Application in a Question Answering Program"
Artificial Intelligence 3 (1972)

[Bruce 1973a]
Bruce B.
"On the Organization of Event Memories"
Rutgers University, CBM-TM-25 (July 1973)

[Bruce 1973b]
Bruce B. et. al.
"CHRONOS User's Manual -- Version I"

Rutgers University, CBM-TM-26 (August 1973)

[Bruce 1973c]
Bruce, B.
"The Processing of Time Phrases in Chronos"
Rutgers University, CBM-TM-29 (Sept 1973)

[Findler 1971]
Findler, C. and Chen, D.
"On the Problems of Time, Retrieval of Temporal Relations, Causality, and Co-existence"
Proc. IJCAI 2 (Sept 1971)

[Findler 1973]
Findler N. and Chen D.
"On the Problems of Time, Retrieval of Temporal Relations, Causality, and Coexistence"
International Journal of Computer and Information Sciences,
Vol 2, No. 3 (1973)

[Lakoff 1970]
Lakoff, R.
"Tense: Its Relation to Speaker, Hearer, and Practically Everybody Else"
University Of Michigan Unpublished Draft (March 1970)

[Martin]
Martin, W., Krumland R. and Sunguroff A.
"More MAPL: Specifications and Basic Structures"
Project MAC, MIT, Automatic Programming Group Internal Memo 8, not available

[Mattison 1967]
Mattison, R.
"A Formal System for the Logical Analysis of Temporal Relationships between Intervals of Time"
The Rand Corporation memorandum RM-5279-PR (April 1967)

[Zadeh 1973]
Zadeh, L.
"Outline of a New Approach to the Analysis of Complex Systems and Decision Processes"
IEEE Transactions on Systems, Man, and Cybernetics,
Vol Smc-3, No. 1 (Jan 1973)

General Bibliography:

[Gorry 1974]
Gorry G.A., et. al.
"NIH Proposal" 1974

[Grief 1974]
Grief I, and Hewitt C.
"Actor Semantics of PLANNER-73"
MIT AI Laboratory Working Paper (Dec 1974)

[Heinlein 1959]
Heinlein, R.
6 x H, Pyramid Communications, Inc. (New York, 1959)

[Hewitt 1973]
Hewitt, C. et. al.
"A Universal Modular ACTOR Formalism"
Proc. IJCAI 3, (Aug 1973)

[McDermott 1974]
McDermott D. and Sussman G.
"The CONNIVER Reference Manual"
MIT AI Laboratory Memo MIT-AI-259A (Jan. 1974)

[McDermott 1974]
McDermott, D.
"Assimilation of New Information by a Natural Language-Understanding System"
MIT AI Laboratory AI-TR-291 (Feb. 1974)

[Minsky 1974]
Minsky M.
"A Framework for Representing Knowledge"
MIT AI Laboratory Memo MIT-AI-306 (June 1974)

[Pratt 1973]
Pratt V.
"A Linguistics Oriented Programming Lanuage"
MIT AI Laboratory Memo MIT-AI-277 (Feb. 1973)

[Pratt 1975]
Pratt, V.
"LINGOL - A Progress Report"

CS-TR Scanning Project
Document Control Form

Date : 11/16/95

Report # LCS-TR-155

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 154 (159-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☐ Single-sided or
☒ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☐ DOD Form ☐ Funding Agent Form ☒ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description : Page Number:

IMAGEMAP: (1-154) UNT#'D TITLE PAGE, 2-68, PAGE 68(ALSO)
FIG, PAGE
(illegible) UNT#BLANK, 69-152
(155-159) SCANCONTROL, COVER, TRGT'S(?)

Scanning Agent Signoff:
Date Received: 11/16/95 Date Scanned: 12/7/95 Date Returned: 12/7/95

Scanning Agent Signature: Michael W. Cook

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**