[illegible]

A NET STRUCTURE BASED RELATIONAL QUESTION ANSWERER: DESCRIPTION AND EXAMPLES

by

Stuart C. Shapiro & George H. Woodmansee

Technical Report #59

March 1969

Computer Sciences Department
1210 West Dayton Street
Madison, Wisconsin 53706

A NET STRUCTURE BASED RELATIONAL QUESTION ANSWERER: DESCRIPTION AND EXAMPLES.*†

ABSTRACT

A question answering system is described which uses a net structure for storage of information. The net structure consists of nodes and labelled edges, which represent relations between the nodes. The labels are also nodes, and therefore definitions of relations may be stored in the net. It is demonstrated that the generality and complexity of this memory structure allows a surprisingly powerful question answering system to be constructed using comparitively simple executive routines. Output from the question answerer, which is currently running on an interactive, time sharing system, is included, showing its range of applicability including question answering, inductive and deductive inference, simple theorem proving and problem solving.

Key words and phrases: relational question answering, question answering, memory net, memory structure, data structure, semantic memory, semantic information retrieval, deductive inference, inductive inference, problem solving, concept formation, relational logic, learning, theorem proving, fact retrieval.

*The research reported herein was partially supported by a grant from the National Science Foundation (GP-7069). Use of the University of Wisconsin Computing Center was made possible through support, in part, from the National Science Foundation, other United States Government agencies and the Wisconsin Alumni Research Foundation (WARF) through the University of Wisconsin Research committee.

†This paper is to be presented at the International Joint Conference on Artificial Intelligence in Washington D.C. (May 7-9, 1969).

1. INTRODUCTION

Our main research interest has been in the organization of data structures for question answering systems, systems that retrieve facts and have deductive and inductive capabilities to derive new information from the facts explicitly given them. Our two main aims have been to maintain as much generality as possible so that no additional programming be needed regardless of the domain of knowledge for which the system is used and to put as much question answering power as possible into the memory structure itself rather than in the executive routines. This latter aim supports the first in that it would allow special instructions for particular domains to be entered into the memory in the same way as any other information. SAMENLAQ II, the system described in this paper represents progress toward reaching these aims. Further progress is being made in a later system (see Section 5).

SAMENLAQ II is based upon binary relations. This was a natural starting point because of the generality of binary relations, and the fact that they provide a reasonable test environment for our ideas. Since our major interest is the memory structure, we have not used natural language input, thus avoiding the attendant problems. Instead, all statements input to the system are in the form x R y. We hope it will become evident that even with this restriction to binary relations and with basically simple executive routines the system attains a surprising amount of power and range of applicability. This derives from the following characteristics of the system:

1. The memory is a net structure, with the relations serving as labels on directed edges. Each statement x R y is also stored in the converse form y R(CNV) x so that all the information about a name is reachable from the node in the net which represents it.

2. The relations, though used as labels on the edges, are actually also nodes themselves, so information about them may be stored in the memory structure. The major use of this capability is to define a relation in terms of other relations.

3. The system has the ability to use the information stored about a relation when searching memory. Such information may be entered at any time and in the same manner as any other type of data or it may be constructed and entered by the system itself. In our system, any relation may be <u>used</u> as an undefined term, may be defined in terms of other relations, or may be defined recursively. Any single relation may be used in any or all of these ways. <u>Complex</u> relations may be built out of <u>simple</u> relations using the relative product operation and <u>node restrictions</u> to restrict the domains or ranges of the simple relations. Thus, quite complicated relations may be defined.

The following sections give more detailed information about and examples of the SAMENLAQ II system. The final section describes a later system which is being developed to satisfy more completely the goals discussed above.

2. IMPLEMENTATION AND OPERATION

SAMENLAQ II is a revision of SAMENLAQ, "A Semantic Association MEmory Net that Learns and Answers Questions"[10]. Both programs are written in SNOBOL3 as interactive question answering systems, but SAMENLAQ II, unlike SAMENLAQ which was run in batch mode with simulated interaction on a CDC 3600, is fully interactive and is currently running under the University of Wisconsin B5500 time sharing system. SAMENLAQ II differs from SAMENLAQ in that it provides aids to the user, who inputs data directly via a teletype, allows for storage of input files and memories in disk files, and, most importantly, allows for recursive definitions of relations and allows the user to control in real time how much effort the system should spend searching its memory to discover more information for use in answering a question.

Figures 2a and 2b depict the overall flow of control in SAMENLAQ II. Figure 1 demonstrates its operation. At the top level of operation, three types of input are allowable: statements, questions and requests to the system executive.[B1,2a]* Representative examples are given in fig. 1

*Bracketed references refer to the flow charts in fig. 2. E.g. "[B1,2a]" refers to Box 1 in fig. 2a.

```
a   ALBANY,BUFFALO,NEW.YORK,POUGHKEEPSIE IN NEW.YORK.STATE←
    STATEMENT - ALBANY,BUFFALO,NEW.YORK,POUGHKEEPSIE IN NEW.
    YORK.STATE
    OK?...YES←
     WAIT...
    READY
    BOSTON,WORCESTER IN MASSACHUSETTS←
    STATEMENT - BOSTON,WORCESTER IN MASSACHUSETTS
    OK?...YES←
     WAIT...
    READY
b   FLAG ECHO OFF@←
    REQUEST - FLAG ECHO OFF
    READY
c   BOSTON EAST.OF WORCESTER←
    READY
    WORCESTER EAST.OF ALBANY←
    READY
    ALBANY EAST.OF BUFFALO←
    READY
    ALBANY NORTH.OF POUGHKEEPSIE←
    READY
    POUGHKEEPSIE NORTH.OF NEW.YORK←
    READY
    IN MEMBER LOCATION.RELS←
    READY
d   NORTH.OF,EAST.OF MEMBER COMPASS.RELS←
    READY
e   SOUTH.OF IMPLBY NORTH.OF(CNV)←
    READY
    WEST.OF IMPLBY EAST.OF(CNV)←
    READY
    WHAT,HAS.WHAT.RELATIONSHIP.TO IS QUESTION←
    READY
f   BOSTON EAST.OF WHAT*←
    I WILL USE THE FOLLOWING RELATIONS:
      EAST.OF/
    I FIGURE BOSTON EAST.OF/ WORCESTER
    IF THIS IS SUFFICIENT, RESPOND - OK
    OTHERWISE TYPE AN INTEGER FOR FURTHER SEARCH
    OR - LIST TROUBLES OR AN INPUT STATEMENT OR - EXAMPLES
g   1←
    I WILL USE THE FOLLOWING RELATIONS:
      NO FURTHER RELATIONS FOUND.
```

Fig. 1a A Conversation With SAMENLAQ II.

```
    I FIGURE BOSTON EAST.OF/ WORCESTER
    IF THIS IS SUFFICIENT, RESPOND - OK
    OTHERWISE TYPE AN INTEGER FOR FURTHER SEARCH
    OR - LIST TROUBLES OR AN INPUT STATEMENT OR - EXAMPLES
h   EAST.OF IMPLBY EAST.OF/EAST.OF←
    STATEMENT - EAST.OF IMPLBY EAST.OF/EAST.OF
    OK?...YES←
    WAIT
    ENTER ANY OF THE ABOVE OPTIONS.
    2←
    I WILL USE THE FOLLOWING RELATIONS:
       EAST.OF/EAST.OF/
    AND EAST.OF/EAST.OF/EAST.OF/
    I FIGURE BOSTON EAST.OF/WORCESTER AND ALBANY AND BUFFALO
    IF THIS IS SUFFICIENT, RESPOND - OK
    OTHERWISE TYPE AN INTEGER FOR FURTHER SEARCH
    OR - LIST TROUBLES OR AN INPUT STATEMENT OR - EXAMPLES
    OK←
    WAIT...
i   ANSWER - WORCESTER       [Answer to Question posed at line "f".]
       AND   ALBANY
       AND   BUFFALO
    READY
j   BUFFALO HAS.WHAT.RELATIONSHIP.TO NEW.YORK*2←
    ENTER COMMA LIST OF RELATION CLASSES TO BE USED OR - " ANY."
    ANY←
    BUFFALO IN NEW.YORK.STATE IN(CNV) NEW.YORK
    ENTER - OK - OR AN INTEGER INDICATING NUMBER OF ADDITIONAL
    PATH LINKS.
    1←
    WAIT...
     BUFFALO EAST.OF(CNV) ALBANY NORTH.OF POUGHKEEPSIE NORTH.OF
    NEW.YORK
    ENTER - OK - OR AN INTEGER INDICATING NUMBER OF ADDITIONAL
    PATH LINKS.
    OK←
    READY
    FLAG ASK OFF@←
    READY
    FLAG TRACE OFF@←
    READY
k   WHAT (IN-NEW.YORK.STATE,)SOUTHWEST.OF BOSTON*←
    I FIGURE BOSTON SOUTHWEST.OF(CNV)/(IN-NEW.YORK.STATE,)
    UNKNOWN
```

Fig. 1b A Conversation With SAMENLAQ II

```
WHAT NOW?
LIST TROUBLES←
AT THE FOLLOWING NAMES COULD NOT APPLY THE LISTED RELATIONS.
   BOSTON - SOUTHWEST.OF(CNV)
WHAT NOW?
SOUTHWEST.OF IMPLBY SOUTH.OF/WEST.OF,WEST.OF/SOUTH.OF←
STATEMENT - SOUTHWEST.OF IMPLBY SOUTH.OF/WEST.OF,WEST.
OF/SOUTH.OF
OK?...YES←
WAIT...
WHAT NOW?
3←
I FIGURE BOSTON SOUTHWEST.OF(CNV)/(IN-NEW.YORK.STATE,)
POUGHKEEPSIE
WHAT NOW?
SOUTH.OF IMPLBY SOUTH.OF/SOUTH.OF←
STATEMENT - SOUTH.OF IMPLBY SOUTH.OF/SOUTH OF
OK?...YES←
WAIT...
WHAT NOW?
3←
I FIGURE BOSTON SOUTHWEST.OF(CNV)/(IN-NEW.YORK.STATE,)
POUGHKEEPSIE
AND NEW.YORK
WHAT NOW?
OK←
ANSWER - POUGHKEEPSIE  [Answer to question posed at line "k".]
   AND   NEW.YORK
READY
```

Fig. 1c A Conversation With SAMENLAQ II.

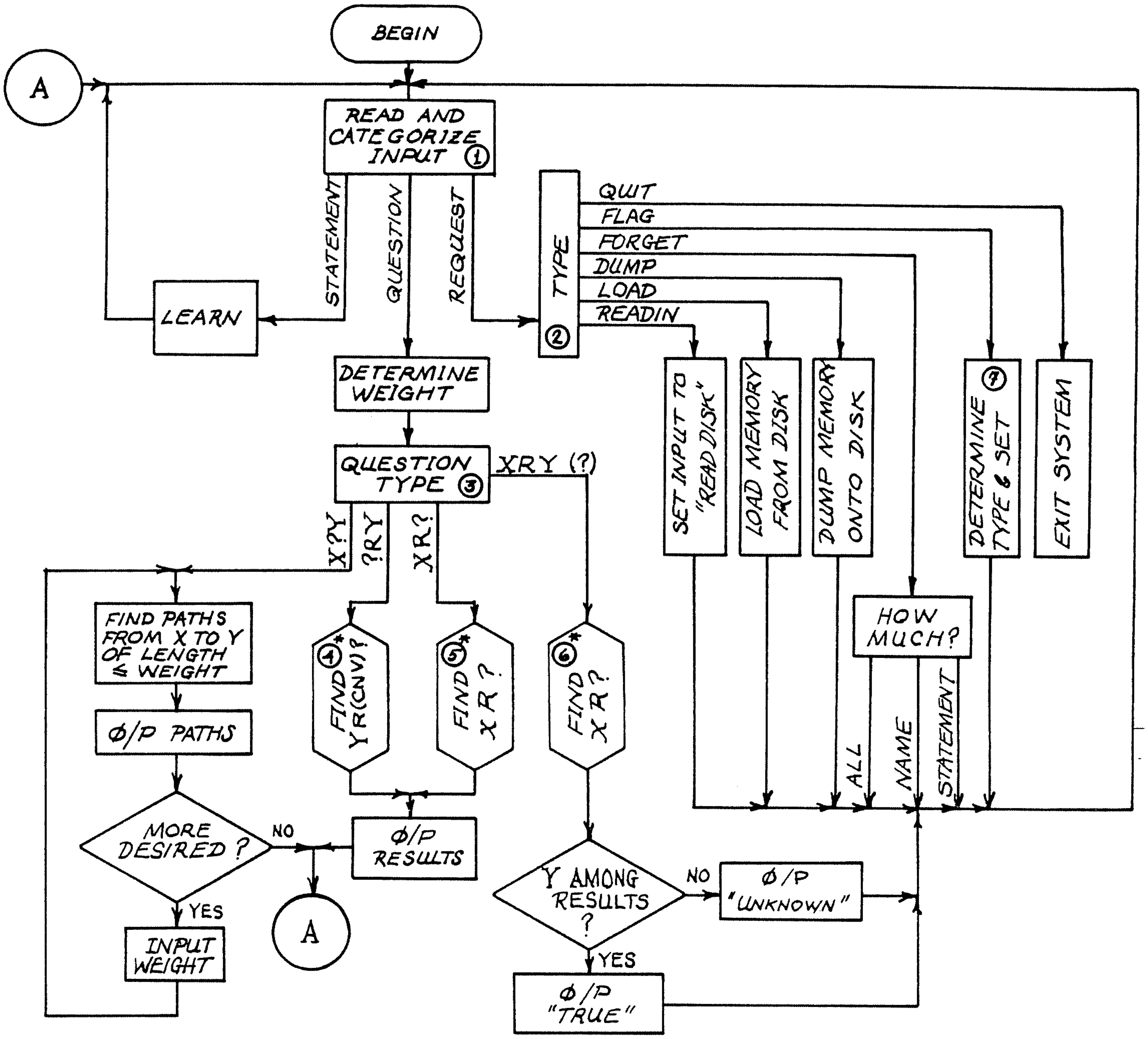

Fig. 2a Overall Flow of Control

*See figure 2b.

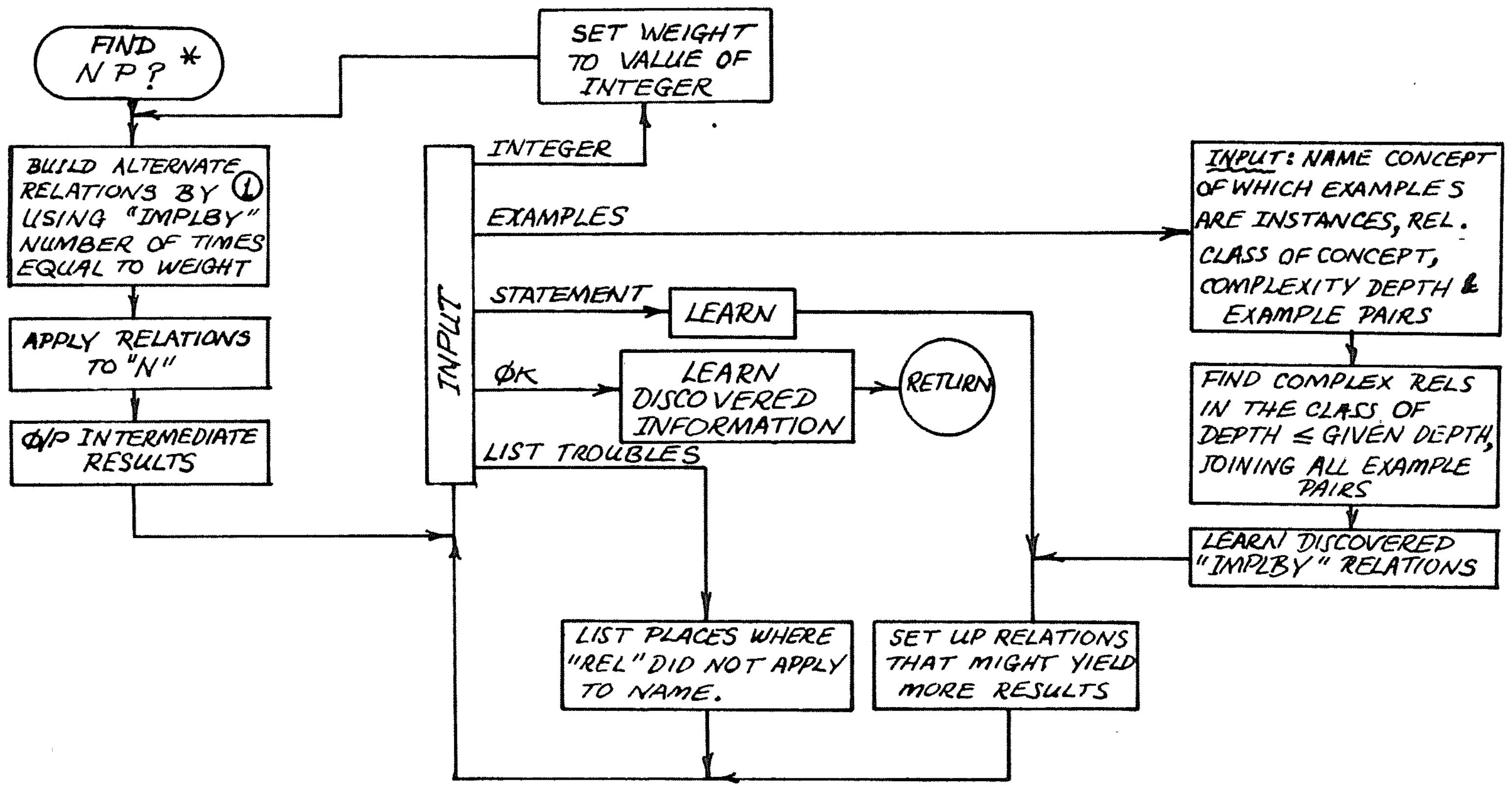
FIND NP? *
BUILD ALTERNATE RELATIONS BY USING "IMPLBY" NUMBER OF TIMES EQUAL TO WEIGHT
APPLY RELATIONS TO "N"
O/P INTERMEDIATE RESULTS
SET WEIGHT TO VALUE OF INTEGER
INPUT
INTEGER
EXAMPLES
STATEMENT
LEARN
ØK
LEARN DISCOVERED INFORMATION
RETURN
LIST TROUBLES
LIST PLACES WHERE "REL" DID NOT APPLY TO NAME.
SET UP RELATIONS THAT MIGHT YIELD MORE RESULTS
INPUT: NAME CONCEPT OF WHICH EXAMPLES ARE INSTANCES, REL. CLASS OF CONCEPT, COMPLEXITY DEPTH & EXAMPLE PAIRS
FIND COMPLEX RELS IN THE CLASS OF DEPTH ≤ GIVEN DEPTH, JOINING ALL EXAMPLE PAIRS
LEARN DISCOVERED "IMPLBY" RELATIONS

Fig. 2b Flow Chart of Main Question Answering Routines

*This routine is used at [B4,2a], [B5,2a] and [B6,2a].

by lines "a," "f," and "b" respectively. Notice that user inputs are indicated by terminal left arrows whereas unterminated lines indicate SAMENLAQ II responses. The simple statement at line "c" results in the construction of a net substructure containing the nodes "BOSTON", "EAST.OF", "EAST.OF(CNV)", and "WORCESTER" in which "BOSTON" and "WORCESTER" are tied together via the "EAST.OF" node and "WORCESTER" and "BOSTON" are tied together via the "EAST.OF(CNV)" node. This structure is considered in more detail in the next section. More complicated statements such as line "a" are interpreted by the system as a series of simple statements.

The system provides the user with various types of feedback, some of which may be turned off by appropriate requests to the executive.[B7,2a] Lines prior to "f" demonstrate input in the full and limited response modes. Line "b" requests that the full response mode be turned off. Requests are identified by a terminating "@". Other request options are indicated in fig. 2.[B2,2a]

Several relation words are built into the system. "MEMBER" allows a particular net search to be limited to a subclass of all the relations represented in the net. Subclass definition may take place at any point during a conversation and is determined solely by the user. Line "d" represents the introduction of the subclass "COMPASS.RELS". Such classes are useful for handling questions involving paths in the net which connect prescribed nodes.

"IMPLBY" allows a given relation to be defined in terms of other relations. It is one of the most important features of SAMENLAQ II. The system has the ability to utilize "IMPLBY" information about a relation during the question answering process by using it as a generalized substitution rule.

Line "e" demonstrates the use of IMPLBY to introduce NORTH.OF(CNV) as an acceptable replacement for SOUTH.OF. The line also results in SOUTH.OF(CNV) IMPLBY NORTH.OF being incorporated in memory.

To enhance readability, the system allows the user to introduce his own interrogatives by means of the form "x IS QUESTION".

Questions are terminated by "*". There are four possible types -- one verification type (x R y *) and three fill in the blank types (x R _ *), (_ R y *) and (x _ y *).[B3,2a] Line "f" illustrates the x R _ * type, line "j" the X _ Y* type and line "k" the _ R Y* type. Notice that non-simple relations can be handled by the system. There are three types of relations used in SAMENLAQ II, <u>simple</u>, <u>compound</u> and <u>complex</u>. A <u>simple</u> relation is of the form R or R(CNV) where the character string R is not meaningfully decomposable and Y R(CNV) X if and only if X R Y. A <u>compound</u> relation is a simple relation or a relative product of simple relations and is of the form Rl/R2/R3, (or sometimes, as a stylistic variant, Rl/R2/R3/). X Rl/R2/R3 Y holds if and only if there exists some z and w such that X R1 z, z R2 w and w R3 Y. A complex relation is a compound relation or a compound relation with <u>node restrictions</u>. A <u>node restriction</u> is of the form (R-Ø) where R is a simple relation and Ø is a string of names, each one followed by a comma. Node restrictions are used to restrict the domain or range of a simple relation which forms part of the complex relation. For example, (R-Ø)R1 is the relation R1 with a restricted domain, and R1(R-Ø) is R1 with a restricted range.

$$x\ \overbrace{(R1-\emptyset 1)(R2-\emptyset 2)}^{\text{Domain of R3}}R3/\overbrace{\underbrace{(R4-\emptyset 3)(R5-\emptyset 4)}_{\text{Range of R3}}}^{\text{Domain of R6}}R6/\underbrace{(R7-\emptyset 5)}_{\text{Range of R6}}\ y$$

A name ,x, <u>satisfies</u> the node restriction (R-Ø) if for all y in the string Ø xRy is explicitly stored in memory. Thus the relational statement above holds if x satisfies (R1-Ø1) and (R2-Ø2), y satisfies (R7-Ø5) and there exists some z such that z satisfies (R4-Ø3) and (R5-Ø4) , xR3z, and zR6y.

A complex relation may also consist solely of node restrictions, in which case it is an identity relation on a restricted domain (viz. the set of all names which satisfy all the node restrictions) and is the statement of a conjunctive concept. The executive routines have built into them the ability to deal with converse relations and with compound and complex relations. When a relation serves to label an edge of the net, i.e. in its appearance in the value of a name, it is treated as a simple relation. Compound and complex relations are used when defining other relations and may themselves have definitions. Lines beginning at "k" illustrate these ideas.

Line "f" represents one of the four question types. To answer it, SAMENLAQ II attempts to apply "EAST.OF" to "BOSTON". Since the system has the statement "BOSTON EAST.OF WORCESTER" represented explicitely in its memory, EAST.OF can be successfully applied - yielding "WORCESTER." Since the system does not know explicitly that "BOSTON EAST.OF ALBANY" or that EAST.OF is a transitive relation, it is incapable of finding further nodes satisfying "BOSTON EAST.OF X". This is illustrated in line "g" where "1" indicates that SAMENLAQ II is to execute one substitution cycle - i.e. substitute for each relation it is currently attempting to apply to a node, all acceptable replacements contained on the relations IMPLBY list.[B1,2b] (The IMPLBY list for a relation R is a list of all relations, X, such that R IMPLBY X.) In line "h" the system is informed that EAST.OF is transitive. Two additional IMPLBY substitution cycles are then requested by typing in "2".

Figure 3 illustrates an application of SAMENLAQ to rudimentary conjunctive concept formation. The statements presented in Fig. 3a supply the system with a small data base concerning the relations of various objects in a room. Note, in line "a" two of these are grouped into the class "SPLREL". In line "b" SAMENLAQ is asked to answer a question concerning a relation it has never seen before. Failing to

```
CHAIR,SHELF,TABLE HAS.PART ELEVATED.HORIZONTAL.SUPPORTING.
SURFACE←
READY
DRESSER,BENCH HAS.PART ELEVATED.HORIZONTAL.SUPPORTING.SURFACE←
READY
SHELF HAS.PART BRACKET.SUPPORTS←
READY
CHAIR,TABLE,DRESSER,BENCH HAS.PART LEGS←
READY
DRESSER HAS.PART DRAWERS←
READY
BENCH,CHAIR,SHELF,TABLE,DRESSER ELEMENT FURNITURE←
READY
TV1,TTY1,TEL1,TEL3,PEN1,SIGNAL.LIGHT1,RADIO1 USED.FOR
COMMUNICATION←
READY
TV1,TTY1,TEL3,TEL1,TEL2,SIGNAL.LIGHT1,RADIO1 CONSTRUCTION
ELECTRICAL←
READY
PEN1 CONSTRUCTION MECHANICAL←
READY
CHAIR1,CHAIR2 ELEMENT CHAIR←
READY
SHELF1,SHELF2,SHELF3 ELEMENT SHELF←
READY
TABLE1,TABLE2 ELEMENT TABLE←
READY
DRESSER1 ELEMENT DRESSER←
READY
CHAIR1,CHAIR2,SHELF2 CONSTRUCTION METAL←
READY
BENCH1,SHELF1,TABLE1,TABLE2,DRESSER1 CONSTRUCTION WOOD←
READY
RADIO1,PEN1 ON.TOP DRESSER1←
READY
SIGNAL.LIGHT1,PICTURE1 ON.TOP SHELF1←
READY
TV1 ON.TOP TABLE2←
READY
TTY1 ON.TOP BENCH1←
READY
BENCH1 ELEMENT BENCH←
READY
```

Fig. 3a A Conjunctive Concept Formation Example: Data Base Input.

```
    PICTURE3,TEL3 ATTACHED.TO WALL1←
    READY
    TEL1 ON.TOP TABLE1←
    READY
    WHAT IS QUESTION←
    READY
a   ON.TOP,ELEMENT MEMBER SPLREL←
    READY
b   WHAT IS.AN.ELECTRICAL.COMMUNICATION.DEVICE.THAT.IS.ON.
    TOP.OF.A.PIECE.OF.WOODEN FURNITURE*←
     I FIGURE FURNITURE
    IS.AN.ELECTRICAL.COMMUNICATION.DEVICE.THAT.IS.ON.TOP.OF.
    A.PIECE.OF.WOODEN(CNV)/ UNKNOWN
    WHAT NOW?
    EXAMPLES←
    RELATION WHOSE DEFINITION IS TO BE FOUND...CONCEPT1←
    RELATION CLASS CONCEPT1 IS A MEMBER OF...SPLREL←
    DEPTH OF ALTERNATE DEFINITIONS...2←
    TYPE PAIRS  X Y  SUCH THAT X CONCEPT1 Y.
    WHEN FINISHED, TYPE - END.
    TEL1 TABLE←
    RADIO1 DRESSER←
    SIGNAL.LIGHT1 SHELF←
    END←
    OK?...YES←
    THANK YOU.
    WAIT...
    FROM THE EXAMPLES YOU HAVE GIVEN ME,I WOULD GUESS THAT
    CONCEPT1/ IS THE SAME AS
    (CONSTRUCTION-ELECTRICAL,)(USED.FOR-COMMUNICATION,)ON.TOP/
    (CONSTRUCTION-WOOD,)ELEMENT/HAS.PART-ELEVATED.HORIZONTAL.
    SUPPORTING.SURFACE,)
    WHAT NOW?
    IS.AN.ELECTRICAL.COMMUNICATION.DEVICE.THAT.IS.ON.TOP.OF.
    A.PIECE.OF.WOODEN IMPLBY CONCEPT1/ELEMENT←
     STATEMENT -
    IS.AN.ELECTRICAL.COMMUNICATION.DEVICE.THAT.IS.ON.TOP.OF.
    A.PIECE.OF.WOODEN IMPLBY CONCEPT1/ELEMENT
    OK?...YES←
    WAIT...
    WHAT NOW?
    2←
```

Fig. 3b A Conjunctive Concept Formation Example: Interrogation.

```
 I FIGURE FURNITURE
IS.AN.ELECTRICAL.COMMUNICATION.DEVICE.THAT.IS.ON.TOP.OF.
A.PIECE.OF.WOODEN(CNV)/ TTY1 AND SIGNAL.LIGHT1 AND TEL1
AND TV1 AND RADIO1
WHAT NOW?
OK←
WAIT...
ANSWER - TTY1      [Answer to Question posed at line "b".]
    AND  SIGNAL.LIGHT1
    AND  TEL1
    AND  TV1
    AND  RADIO1
READY
```

Fig. 3c A Conjunctive Concept Formation Example: Interrogation.

apply this relation, it is given a series of examples whose relationship to one another is arbitrarily designated CONCEPT1. Restricting its search to the relation class SPLREL, the system obtains all paths of length 2 or less connecting the example pairs. The properties common to each class of nodes at the same level along the path are also calculated. For example (CONSTRUCTION-ELECTRICAL,), (CONSTRUCTION-WOOD,) and (HAS.PART-ELEVATED.HORIZONTAL.SUPPORTING SURFACE,) are node properties common to the first, second, and third levels respectively. The path connecting the example pairs is "ON.TOP/ELEMENT". This relation is then placed upon the IMPLBY list for CONCEPT1. Finally CONCEPT1 is used to define the original concept IS.AN.ELECTRICAL. COMMUNICATION.DEVICE.THAT.IS.ON.TOP.OF.A.PIECE.OF.WOODEN. The system then has sufficient information to answer the question and responds with the correct answer.

3. SAMENLAQ II DESCRIPTION

Statements entered into the memory are of the form "NAME1 RELATION NAME2" where NAME1 and NAME2 are non-decomposable names and RELATION is a simple relation. Information contained in such statements is stored on _paren lists_ associated with NAME1 and NAME2. Thus, the above statement produces the following paren list for "NAME1": "(RELATION-/1)" where the contents of the _slash name_ "/1" is the _comma list_ "NAME2,". In the example below, the _paren pair_ (WEST.OF-/3) on the paren list for

WORCESTER indicates that Worcester is West of each of the elements found on the comma list named /3. Note that the value of a slash name is a comma list, similarly the value of a name is its paren list.

```
BOSTON      =  (EAST.OF-/1)
/1          =  WORCESTER,ALBANY,BUFFALO,
WORCESTER   =  (EAST.OF-/2)(WEST.OF-/3)
/2          =  ALBANY,BUFFALO,
/3          =  BOSTON,
EAST.OF     =  (IMPLBY-/4)(MEMBER-/5)
/4          =  EAST.OF/EAST.OF/,
/5          =  COMPASS.RELS,
```

Although the paren list looks like a conventional attribute - value list it differs in that both the relations (attributes) and the names on the comma lists (values) are themselves names of paren lists and these various paren lists mutually occur as elements of each other. The memory may be thought of as a directed graph whose nodes are the names (BOSTON,WORCESTER, etc.) and whose edges are labelled by the relations. Since, however, the relations are also names and thus should be thought of as nodes in the graph, we should, perhaps, think of the edges as being labelled by passing through a node, and all edges bearing the same label as passing through the same node.

The statement NAME1 RELATION NAME2 is not only stored as such on NAME1's paren list, but its converse, NAME2 RELATION(CNV) NAME1 is stored on NAME2's paren list. This is done so that the information contained in the statement is recoverable from either name. Although this involves duplicate storage of information, changing the statement to its converse form for storage under the second argument allows all statements about a name to be stored in the same place (the name's value) regardless of whether the name was the first or second argument in the original statements. This contrasts with the methods for retrieving a relational statement from either argument used by the Relational Data File[4] (RDF) and by DEACON[1,6]. In RDF, statements are stored in only one direction, but in dirierent files which are ordered on different parts of the statement. Thus to get all information about a single name, either one file must be searched exaustively or the name must be looked up in all files. In DEACON, the statements are stored in the form of closed "connecting rings" through the three parts of the statement. Thus the statement is reachable from any part of it without recourse to several files, but it is impossible to tell from a connecting ring where the statement should begin, i.e. whether the ring through x, R, and y represents the statement xRy, Ryx, or yxR.

The generality of SAMENLAQ derives largely from the ability to introduce new relations at any time, to introduce definitions of new relations or relations that had previously been undefined and to extend the definition of a relation. Definitions may also be added at any time and are stored in the memory net structure just like any other data. Definitions are not given in terms of relation properties that have been built into the system, but in terms of other relations. Nevertheless, various standard relation properties can be dealt with, for example:

1. Entering the statement, "R1 IMPLBY R1/R1 causes R1 to be transitive.
2. Entering "R1 IMPLBY R1(CNV)" causes R1 to be symmetric.
3. It was previously pointed out (section 2) that the complex relation consisting only of a node restriction (R-∅) serves as an identity relation for all names x such that for every name y in the string ∅, xRy is explicitly in memory. If ∅ were the string consisting only of the delimiter ",", (R-∅) would be the identity relation for all x such that for any y, xRy were explicitly stored. If this were true for all x in memory, (R-,) would be the universal identy relation. In that case entering, the statement "R1 IMPLBY (R-,)" would cause R1 to be reflexive.

Thus, R1 might be defined to be an equivalence relation by entering "R1 IMPLBY (MEMBER-,), R1(CNV), R1/R1/". In addition to these forms of definition, a relation may be defined a) by using only other relations, and b) by combining other relations along with the relation being defined thus forming a general recursive definition. A wide range of relations may thus be used without programming them into the executive routines as it is done in Raphael's SIR[8]. This method of defining relations also contrasts with that used by Elliott[2] in "GRAIS". In "GRAIS" relational properties are built into the executive routines. In fact, this is done in such a way as to provide specific routines for 32 classes of relations. The user introduces a new relation by specifying which of the 32 classes it belongs in and this determines how it will be handled. This does not allow a user to use a relation whose properties he either does not know completely or does not wish to make specific _initially_. A user may also define a relation in terms of a Boolean function of previously introduced relations. However, these relations may not be stored in the data structure, only used for question answering.

It is interesting to note that the user builds his own logic system into SAMENLAQ II when he specifies IMPLBY information (used as the rules of inference) and other statements (the axioms). The only logical structure imposed on the user is the metatheoretic substitution rule embodied in the procedures which apply IMPLBY, and the limits on the form of a rule of inference imposed by the syntax of _complex relations_. If the user specifies a strange or even self-contradictory "logic" SAMENLAQ II will produce deductions that are equally strange or contradictory; interpretation is in the mind of the user. (For example, in section 4, deductions arising from the relation "IS.PART.OF" only make sense if the interpretation of "x IS.PART.OF y" motivating the rule of inference "IS.PART.OF IMPLBY IS.PART.OF/IS(CNV)" is that every member of y has a part which is a member of x and the interpretation of "IS" is "is subset of.")

It would be possible to append an executive to SAMENLAQ II which would constrain the type of logical system to one with certain prescribed properties.

4. SOME VARIED APPLICATIONS OF SAMENLAQ II

The following sample conversations demonstrate SAMENLAQ II's ability to deal with relations arising from a variety of problem areas.

The first conversation (Figure 4a-d) involves a modified subdialogue from SIR[8]. It demonstrates SAMENLAQ's ability to handle relations such as part, subset, owns and element and the interdependence between such relations. Although specific relations and relational properties are not built into the system, SAMENLAQ can utilize the information (x)(Y)(Z) (x IS.PART.OF Y & $Z \subset Y \rightarrow$ x IS.PART.OF Z) via the IMPLBY statement IS.PART.OF IMPLBY IS.PART.OF/IS(CNV). (See comments at the end of section 3.)

In attempting to answer a question, it may be necessary to supply further information to the system. Such a situation is illustrated by line "b" in the conversation starting at line "a".

Notice that in the case of a "WHAT R Y" question, SAMENLAQ II proceeds by attempting to apply the R(CNV) relation to the node Y. [B4,2a]

Figure 5 shows an application of SAMENLAQ II to census, airplane and airline flight data. Note especially that even though the

```
IS.PART.OF IMPLBY IS.PART.OF/IS.PART.OF,IS.PART.OF/IS(CNV)←
READY
SOMETIMES IMPLBY IS(CNV)←
READY
IS IMPLBY IS/IS←
READY
NOSE IS.PART.OF PERSON←
READY
NOSTRIL IS.PART.OF NOSE←
READY
PROFESSOR IS TEACHER←
READY
TEACHER IS PERSON←
READY
NOSTRIL IS.PART.OF PROFESSOR*1←
I FIGURE NOSTRIL IS.PART.OF/ NOSE AND PERSON
WHAT NOW?
1←
I FIGURE NOSTRIL IS.PART.OF/ NOSE AND PERSON AND TEACHER
WHAT NOW?
1←
I FIGURE NOSTRIL IS.PART.OF/ NOSE AND PERSON AND TEACHER
AND PROFESSOR
WHAT NOW?
OK←
WAIT...
ANSWER - TRUE
READY
PERSON IS LIVING CREATURE←
BAD INPUT.  TRY AGAIN.
READY
PERSON IS LIVING.CREATURE←
READY
HAS.AS.PART IMPLBY IS.PART.OF(CNV)←
READY
LIVING.CREATURE SOMETIMES/HAS.AS.PART NOSTRIL*3←
I FIGURE LIVING.CREATURE SOMETIMES/HAS.AS.PART/ NOSTRIL
WHAT NOW?
OK←
ANSWER - TRUE
READY
CRT IS DISPLAY.DEVICE←
READY
```

Fig. 4a   Learning and Deduction Using Several Relations From SIR

```
    CRT IS.PART.OF B5500←
    READY
    BRUTUS IS B5500←
    READY
    SCREEN IS.PART.OF DISPLAY.DEVICE←
    READY
d   SCREEN IS.PART.OF BRUTUS*1←
    I FIGURE SCREEN IS.PART.OF/ DISPLAY.DEVICE AND CRT
    WHAT NOW?
    1←
    I FIGURE SCREEN IS.PART.OF/ DISPLAY.DEVICE AND CRT AND B5500
    WHAT NOW?
    1←
     I FIGURE SCREEN IS.PART.OF/ DISPLAY.DEVICE AND CRT AND
    B5500 AND BRUTUS
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - TRUE  [Answer to question posed at line "d" above]
    READY
    OWNS IMPLBY IS/OWNS←
    READY
    FIREMAN OWNS PAIR.OF.RED.SUSPENDERS←
    READY
    DOCTOR OWNS PAIR.OF.RED.SUSPENDERS*1←
    I FIGURE DOCTOR OWNS/ UNKNOWN
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - UNKNOWN
    READY
    FIRECHIEF IS FIREMAN←
    READY
    FIRECHIEF OWNS PAIR.OF.RED.SUSPENDERS*1←
    I FIGURE FIRECHIEF OWNS/ PAIR.OF.RED.SUSPENDERS
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - TRUE
    READY
a   EXAMPLE.OF IMPLBY EXAMPLE.OF/IS←
    READY
    A IMPLBY EXAMPLE.OF←
    READY
```

Fig. 4b Learning and Deduction Using Several Relations From SIR

```
     STU OWNS LOG.LOG.DECITRIG1←
     READY
     LOG.LOG.DECITRIG1 EXAMPLE.OF LOG.LOG.DECITRIG←
     READY
     LOG.LOG.DECITRIG IS SLIDE.RULE←
     READY
     STU OWNS/A SLIDE RULE*1←
     I FIGURE STU OWNS/A/ LOG.LOG.DECITRIG
     WHAT NOW?
     1←
     I FIGURE STU OWNS/A/ LOG.LOG.DECITRIG AND SLIDE.RULE
     WHAT NOW?
     OK←
     ANSWER TRUE
     READY
     ENGINEERING.STUDENT OWNS SLIDE.RULE←
     READY
     GEORGE EXAMPLE.OF TECH.MAN←
     READY
     TECH.MAN IS ENGINEERING.STUDENT←
     READY
     GEORGE OWNS/A SLIDE.RULE*1←
     I FIGURE GEORGE OWNS/A/ UNKNOWN
     WHAT NOW?
     1←
     I FIGURE GEORGE OWNS/A/ UNKNOWN
     WHAT NOW?
     OK←
     ANSWER - UNKNOWN
     READY
c    ENGINEERING.STUDENT EXAMPLE.OF(CNV)/OWNS/EXAMPLE.OF LOG.
     LOG.DECITRIG*1←
     I FIGURE ENGINEERING.STUDENT EXAMPLE.OF(CNV)/OWNS/EXAMPLE.
     OF/ UNKNOWN
     WHAT NOW?
b    STU EXAMPLE.OF TECH.MAN←  [Provide additional information necessary
     STATEMENT - STU EXAMPLE.OF TECH.MAN        to answer question.]
     OK?...YES←
     WAIT...
     WHAT NOW?
     2←
      I FIGURE ENGINEERING.STUDENT EXAMPLE.OF(CNV)/OWNS/EXAMPLE.
     OF/ LOG.LOG.DECITRIG AND SLIDE.RULE
```

Fig. 4c   Learning and Deduction Using Several Relations From SIR.

```
WHAT NOW?
OK←
ANSWER - TRUE  [Answer to question posed at line "c".]
READY
```

Fig. 4d Learning and Deduction Using Several Relations From SIR

```
  NEW.YORK,LOS.ANGELES,SANTA.BARBARA,ORLANDO IS CITY←
  READY
  NEW.YORK HAS.POPULATION 7781984←
  READY
  LOS.ANGELES HAS.POPULATION 2479015←
  READY
  ORLANDO HAS.POPULATION 88135←
  READY
  SANTA.BARBARA HAS.POPULATION 58768←
  READY
  7781984,2479015 IS.GREATER.THAN 100000←
  READY
  88135,58768 IS.LESS.THAN 100000←
  READY
  BOEING.707,BOEING.727,DC8 IS JET.PLANE←
  READY
  CONVAIR.240 IS PROP.PLANE←
  READY
  JET.PLANE,PROP.PLANE SUBSET AIRPLANE←
  READY
  IS IMPLBY IS/SUBSET←
  READY
  BOEING.707,DC.8 CARRIES 150←
  READY
  BOEING.727 CARRIES 120←
  READY
  CONVAIR.240 CARRIES 50←
  READY
  150 IS.GREATER.THAN 120←
  READY
  50 IS.LESS.THAN 120←
  READY
a IS.LARGE IMPLBY (IS-CITY,)HAS.POPULATION/(IS.GREATER.THAN-
  100000,)HAS.POPULATION(CNV)/IS←
  READY
b IS.LARGE IMPLBY (IS-AIRPLANE,)CARRIES/(IS.GREATER.THAN-120,)
  CARRIES(CNV)/IS←
  READY
  WHAT IS QUESTION←
  WHAT IS AIRPLANE*1←
   I FIGURE AIRPLANE IS(CNV)/ BOEING.707 AND BOEING.727 AND
  DC.8 AND CONVAIR.240
```

Fig. 5a Application to census, airplane and airline flight data with an ambiguous relation.

```
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - BOEING.707
       AND   BOEING.727
       AND   DC.8
       AND   CONVAIR.240
    READY
c   DC.8 IS.LARGE WHAT*1←
    I FIGURE DC.8 IS.LARGE/ JET.PLANE AND AIRPLANE
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - JET.PLANE
       AND   AIRPLANE
    READY
    WHAT IS.LARGE CITY*1←
    I FIGURE CITY IS.LARGE(CNV)/ NEW.YORK AND LOG.ANGELES
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - NEW.YORK
       AND   LOS.ANGELES
    READY
    SANTA.BARBARA,ORLANDO HAS.POPULATION WHAT*←
    I FIGURE SANTA.BARBARA,ORLANDO HAS.POPULATION/ 58768 AND 88135
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - 58768
       AND   88135
    READY
    FLT.207 FLIES.FROM NEW.YORK←
    READY
    FLT.207 FLIES.TO LOS.ANGELES←
    READY
    DC.8 USED.ON FLT.207←
    READY
    FLT.207 DEPARTS.AT 10:00AM←
    READY
    FLT.207 ARRIVES.AT 12:30PM←
    READY
    FLT.308 FLIES.FROM NEW.YORK←
```

Fig. 5b Application to census, airplane and airline flight data with an ambiguous relation.

```
READY
FLT.308 FLIES.TO ORLANDO←
READY
BOEING.727 USED.ON FLT.308←
READY
FLT.45 FLIES.FROM LOS.ANGELES←
READY
FLT.45 FLIES.TO SANTA.BARBARA←
READY
CONVAIR.240 USED.ON FLT.45←
READY
FLT.45 DEPARTS.AT 1:30PM←
READY
FLT.45 ARRIVES.AT 2:15PM←
READY
1:30PM IS.LATER.THAN 12:30PM←
READY
CONNECTS.WITH IMPLBY ARRIVES.AT/IS.LATER.THAN(CNV)/DEPARTS.
AT(CNV)←
READY
WHAT (FLIES.FROM-NEW.YORK,)CONNECTS.WITH/FLIES.TO SANTA.
BARBARA*1←
 I FIGURE SANTA.BARBARA
FLIES.TO(CNV)/CONNECTS.WITH(CNV)/(FLIES.FROM-NEW.YORK,)
FLT.207
WHAT NOW?
OK←
ANSWER - FLT.207
READY
FLT.207 USED.ON(CNV)/IS.LARGE AIRPLANE*←
I FIGURE FLT.207 USED.ON(CNV)/IS.LARGE/ JET.PLANE AND AIRPLANE
WHAT NOW?
OK←
ANSWER - TRUE
READY
```

Fig. 5c Application to census, airplane and airline flight data with an ambiguous relation.

relation "IS.LARGE" is defined ambiguously in lines "a" and "b" as to its application to cities or airplanes, SAMENLAQ II can disambiguate it from context (lines "c" and "d").

In figure 6, SAMENLAQ II works with simple logic and set theory. The rule of inference Modus Ponens is entered in 6a line "a", and with this and the transitivity of IMPLIES given in line "b", simple "chain implication" problems can be solved. In 6b the relations SUBSET and ELEMENT are introduced along with the rule $(x)(x \in A \;\&\; A \subset B \rightarrow x \in B)$. Then some set membership problems are solved, and finally, a simple proof is constructed.

Figure 7 shows SAMENLAQ II being taught its first lesson in arithmetic. Although neither numbers nor arithmetic functions have been built into the SAMENLAQ II structure or executive routines, SAMENLAQ II is capable of being taught arithmetic the way school children used to be taught: by first memorizing tables, and then being taught certain rules. Notice especially that divisibility by 2 was defined recursively in lines "a", "b" and "c". Similarly, SAMENLAQ II could have been taught multiplication, division, the recursive definitions for less than and greater than as well as other arithmetic relations.

Figure 8 shows SAMENLAQ II solving the Missionary - Cannibal Problem, with three missionaries, three cannibals and a boat that holds a maximum of two people. The problem was described to SAMENLAQ II as a set of all the legal states in the problem with all the possible

```
a   TRUE IMPLBY IMPLIES(CNV)/TRUE←
    READY
    FALSE IMPLBY IMPLIES/FALSE←
    READY
    A IMPLIES B←
    READY
    A TRUE PROPOSITION←
    READY
    B TRUE PROPOSITION*1←
    I FIGURE B TRUE/ PROPOSITION
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - TRUE
    READY
    FOLLOWS.FROM IMPLBY IMPLIES(CNV)←
    READY
b   IMPLIES IMPLBY IMPLIES/IMPLIES←
    READY
    C IMPLIES D←
    READY
    D FALSE PROPOSITION←
    READY
    C FALSE PROPOSITION*1←
    I FIGURE C FALSE/ PROPOSITION
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - TRUE
    READY
    FORGET PROPOSITION@←
    FORGET - PROPOSITION
    OK?...YES←
     WAIT...
    READY
    B IMPLIES C←
    READY
    D FALSE PROPOSITION←
    READY
    A FALSE PROPOSITION*1←
    I FIGURE A FALSE/ UNKNOWN
    WHAT NOW?
    2←
```

Fig. 6a   A Simple Problem in Logic Using Modus Ponens

```
    I FIGURE A FALSE/ PROPOSITION
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - TRUE
    READY
    ELEMENT IMPLBY ELEMENT/SUBSET←
    READY
    S1 SUBSET S2←
    READY
    S2 SUBSET S3←
    READY
    E1 ELEMENT S1←
    READY
    WHAT IS QUESTION←
    READY
    E1 ELEMENT WHAT*1←
    I FIGURE E1 ELEMENT/ S1 and S2
    WHAT NOW?
    1←
    I FIGURE E1 ELEMENT/ S1 AND S2 AND S3
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - S1
       AND   S2
       AND   S3
    READY
c   D FOLLOWS.FROM A*2←
    I FIGURE D FOLLOWS.FROM/ C AND B
    WHAT NOW?
    1←
    I FIGURE D FOLLOWS.FROM/ C AND B AND A
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - TRUE  [Answer to question posed at line "c".]
    READY
    F IS AXIOM←
    READY
    FIND.PROOF.FROM IS QUESTION←
    READY
```

Fig. 6b   Set Theory and Simple Theorem Proving

```
A DERIVABLE.FROM F←
READY
D FIND.PROOF.FROM AXIOM*6←
ENTER COMMA LIST OF RELATION CLASSES TO BE USED OR - " ANY "
ANY←
D FOLLOWS.FROM A DERIVABLE.FROM F IS AXIOM
WHAT NOW?
OK←
READY
```

Fig. 6c Set Theory and Simple Theorem Proving

```
    1 PLUS.1.IS 2←
    READY
    2 PLUS.1.IS 3←
    READY
    3 PLUS.1.IS 4←
    READY
    PLUS.2.IS IMPLBY PLUS.1.IS/PLUS.1.IS←
    READY
    WHAT IS QUESTION←
    READY
    2 PLUS.2.IS WHAT*1←
    I FIGURE 2 PLUS.2.IS/ 4
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - 4
    READY
    MINUS.2.IS IMPLBY PLUS.2.IS(CNV)←
    READY
a   1 DIVISIBLE.BY.2 FALSE←
    READY
b   2 DIVISIBLE.BY.2 TRUE←
    READY
c   DIVISIBLE.BY.2 IMPLBY MINUS.2.IS/DIVISIBLE.BY.2←
    READY
    4 DIVISIBLE.BY.2 WHAT*2←
    I FIGURE 4 DIVISIBLE.BY.2/ TRUE
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - TRUE
    READY
    3 DIVISIBLE.BY.2 WHAT*3←
    I FIGURE 3 DIVISIBLE.BY.2/ FALSE
    WHAT NOW?
    OK←
    WAIT...
    ANSWER - FALSE
    READY
```

Fig. 7 Arithmetic and Handling Recursive Definitions

```
3M3CL 1C 0M1CR←
READY
3M3CL 1M1C 1M1CR←
READY
3M3CL 2C 0M2CR←
READY
3M3CL 1C 0M1CR←
READY
3M2CL 1M 1M1CR←
READY
3M2CL 2C 0M3CR←
READY
3M1CL 1C 0M3CR←
READY
3M1CL 2M 2M2CR←
READY
2M2CL 1M1C 2M2CR←
READY
2M2CL 2M 3M1CR←
READY
1M1CL 1M 3M2CR←
READY
1M1CL 1M1C 3M3CR←
READY
0M3CL 1C 3M1CR←
READY
0M3CL 2C 3M2CR←
READY
0M2CL 1C 3M2CR←
READY
0M2CL 2C 3M3CR←
READY
0M1CL 1C 3M3CR←
READY
GET.TO IS QUESTION←
READY
3M3CL GET.TO 3M3CR*11←
ENTER COMMA LIST OF RELATION CLASSES TO BE USED OR - " ANY "
ANY←     [First solution follows:]
 3M3CL 2C 0M2CR 1C(CNV) 3M2CL 2C 0M3CR 1C(CNV) 3M1CL 2M 2M2CR
1M1C(CNV) 2M2CL 2M 3M1CR 1C(CNV) 0M3CL 2C 3M2CR 1M(CNV) 1M1CL
1M1C 3M3CR   [Second solution follows:]
 3M3CL 2C 0M2CR 1C(CNV) 3M2CL 2C 0M3CR 1C(CNV) 3M1CL 2M 2M2CR
1M1C(CNV) 2M2CL 2M 3M1CR 1C(CNV) 0M3CL 2C 3M2CR 1C(CNV) 0M2CL
2C 3M3CR
```

Fig. 8a Solving the Missionary-Cannibal Problem

```
WHAT NOW?
OK←
READY
```

Fig. 8b Solving the Missionary-Cannibal Problem

transitions between the states. For example, the first line represents the fact that if 3 missionaries and 3 cannibals are on the left bank with the boat on the left bank, then 1 cannibal can take the boat to the right bank, which will result in there being 0 missionaries and 1 cannibal on the right bank with the boat on the right bank. SAMENLAQ II solves the problem by showing how the boat should be used to get from the initial state, 3M3CL, to the final state, 3M3CR. Any problem solving task that can be represented as finding a path from an initial state to a final state through a state transition graph can in theory be solved similarly by SAMENLAQ II.*

5. EXTENSION OF THE SAMENLAQ STRUCTURE

Work is now proceeding on the design and implementation of a memory net structure, MENS[9], which goes further than SAMENLAQ II toward satisfying goals discussed in the first section of this paper. The two major improvements needed in the SAMENLAQ structure are:

1. the ability to deal with a name which is itself a statement
2. the ability to store names which represent some unspecified other names, i.e. act as variables.

*Since SAMENLAQ II exhaustively searches the state transition graph, the threat of exponential growth is ever present. Thus in certain interesting problems exhaustive search would be infeasible and heuristic search techniques would be necessary.

The first would facilitate the handling of n-ary relations and statements which serve to modify or give further information about other statements. The second would allow generalizations to be stored, and would also permit the storage of statements of the predicate calculus directly in the memory structure. These statements could then be interpreted by the executive and used as rules of inference to direct the memory search routines in a manner similar to the way SAMENLAQ II deals with IMPLBY definitions. The currently extant implementation of MENS (which is programmed in Burroughs Extended ALGOL and uses ASLIP, a SLIP like package of list processing routines) incorporates the first improvement, and work is progressing on the design of the implementation of the second improvement.

The main generalization involved in going from SAMENLAQ to MENS was to let xRy statements be nodes in the net along with arguments and relations. The basic element of the MENS structure is called an <u>item</u>, which may be an unstructured unit or may be a structure consisting of a pair or triple of items. Thus, items are similar to the "events" used by Simmons et al in Protosynthex II[11]. As in SAMENLAQ, a major characteristic of the MENS structure is that there is no duplication of items or structures; the physically same item is used everywhere that the structure it represents is referred to in a containing structure. Several implications of this uniqueness of items are: (1) two structures which have a substructure in common actually overlap in the net, (2) if there is an item representing logical implication, all structures interpretable as rules of inference will be discoverable directly from that item since it will be a central substructure of all of them, (3) in general statements involving a quantified variable, the separate occurrences of the variable will all be pointers to a single item, so that a substitution attached to that item will serve as a substitution for all occurrences of the variable.

Allowing a structure to be formed from a pair of substructures provides for the representation of unary relations such as negation and quantification. Allowing a structure to be formed from a triple of substructures provides for the representation of binary relation, and since any of the substructures may in fact be structures as well as unstructured items provides for the representation of n-ary relations. A more direct representation of n-ary relations is provided in another version of MENS being implemented*, which will allow any structure to consist of any number of substructures.

It is demonstrated in the examples given in Section 4 that SAMENLAQ II is capable of answering questions in formal logic involving simple chains of inference and basic set theory. With its ability to store statements of the predicate calculus, the MENS structure

*A paper describing this version of MENS and the natural language question answering system which will use it is forthcoming from the RAND Corporation as a RAND Memorandum by Martin Kay, Ronald M. Kaplan, and Stuart C. Shapiro.

should enable simple question answering routines to perform more complicated theorem proving. Although we should perhaps not expect a high powered theorem prover to be developed in this way, the MENS extension of SAMENLAQ will provide an interesting contrast to systems, such as Green and Raphael's QA2[3], which use theorem proving techniques to answer questions.

APPENDIX - MEMORY STRUCTURE

```
READY

DUMP MEMORY ON TELETYPE@←

THE MEMORY IS ---

2 = (DIV.BY.2-/20)(PLUS.2.IS-/13)(PLUS.1.IS-/4)(PLUS.1.IS(CNV)-/1)
/1 = 1,
1 = (DIV.BY.2-/18)(PLUS.1.IS-/2)
/2 = 2,
3 = (DIV.BY.2-/25)(PLUS.1.IS-/6)(PLUS.1.IS(CNV)-/3)
/3 = 2,
/4 = 3,
4 = (DIV.BY.2-/24)(PLUS.2.IS(CNV)-/12)(PLUS.1.IS(CNV)-/5)
/5 = 3,
/6 = 4,
PLUS.1.IS/PLUS.1.IS = (IMPLBY(CNV)-/7)
/7 = PLUS.2.IS,
PLUS.2.IS(CNV) = (IMPLBY(CNV)-/14)(IMPLBY-/8)
/8 = PLUS.1.IS(CNV)/PLUS.1.IS(CNV)/,
PLUS.2.IS = (IMPLBY-/9)
/9 = PLUS.1.IS/PLUS.1.IS/,
QUESTION = (IS(CNV)-/10)
/10 = WHAT
WHAT = (IS-/11)
/11 = QUESTION,
/12 = 2,
/13 = 4,
/14 = MINUS.2.IS,
MINUS.2.IS(CNV) = (IMPLBY-/15)
/15 = PLUS.2.IS/,
MINUS.2.IS = (IMPLBY-/16)
/16 = PLUS.2.IS(CNV)/,
FALSE = (DIV.BY.2(CNV)-/17)
/17 = 1,3,
/18 = FALSE,
TRUE = (DIV.BY.2(CNV)-/19)
/19 = 2,4,
/20 = TRUE,
```

Fig. 9 Shows SAMENLAQ II's actual memory structure after its arithmetic lesson, which was shown in figure 7. The underlined material was learned after being discovered as implicit information while answering questions.

```
MINUS.2.IS/DIV.BY.2 = (IMPLBY(CNV)-/21)
/21 = DIV.BY.2,
DIV.BY.2(CNV) = (IMPLBY-/22)
/22 = DIV.BY.2(CNV)/MINUS.2.IS(CNV)/,
DIV.BY.2 = (IMPLBY-/23)
/23 = MINUS.2.IS/DIV.BY.2/,
/24 = TRUE,
/25 = FALSE,
READY
```

Fig. 9 (Cont.)

References

1. Craig, J. A., Berezner, S. C., Carney, H. C., Longyear, C. R., DEACON: Direct English Access and CONtrol. Proc. AFIPS FJCC (1966), 365-380.

2. Elliott, R. W., A Model for a Fact Retrieval System, unpublished Ph.D. dissertation, University of Texas, Austin, Texas, 1965.

3. Green, C. C., Raphael, B., Research on Intelligent Question-Answering System. AFCRL-67-0370, Stanford Research Institute, Menlo Park, California, May, 1967.

4. Levien, R., Maron, M. E., Relational Data File: A Tool for Mechanized Inference execution and Data Retrieval. RM-4793-PR, The RAND Corporation, Santa Monica, California, 1965.

5. _________________ A Computer System for Inference Execution and Data Retrieval. RM-5085-PR, The RAND Corporation, Santa Monica, California, Sept., 1966. Also C.ACM 10,11 (Nov. 1967), 715-721.

6. Longyear, C. R. Memory Structure in DEACON Natural Language Question-Answering Systems. P-129, General Electric Company, TEMPO, Santa Barbara, California, 1966.

7. Quillian, M. R., Semantic Memory, unpublished Ph.D. dissertation, Carnegie Institute of Technology, Pittsburgh, Pennsylvania, 1966. Also AFCRL-66-189, Bolt Beranek and Newman, Inc., Cambridge, Massachusetts, 1966.

8. Raphael, B., Semantic Information Retrieval, unpublished Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1964. Also TR-2, Project MAC, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1964.

9. Shapiro, S. C., A Memory Net Structure: Present Implementation and a Proposed Language, Technical Report #53, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, Dec. 1968.

10. ________________, Woodmansee, G. H., Krueger, M. W., A Semantic Associational Memory Net That Learns and Answers Questions (SAMENLAQ). Technical Report #8, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, Jan., 1968.

11. Simmons, R. F., Burger, J. F., A Semantic Analyzer for English Sentences. SP-2987, Systems Development Corp., Santa Monica, California, Jan., 1968.