IMPLEMENTATION OF FORWARD INFERENCE IN SNePS

by

Joao P. Martins

Department of Computer Science

State University of New York at Buffalo

4226 Ridge Lea Road, Amherst, N.Y. 14226

November 1980

Abstract

This paper describes the details of the implementation of a forward inference system in SNePS and discusses the problems encontered during the implementation and the design decisions taken to solve those problems.

1. Introduction

Since the late 50's there has been a large amount of work done in AI concerning the design and implementation of systems capable of performing deduction. Early work on deduction systems focused mainly in the proof of theorems [Chang and Lee 73; Coelho and Pereira 75; Gelernter 63; Hewitt 72; Kowalski 70; 79; Newell et. al. 63; Nilsson 71; 80; Robinson 65; Sussman et. al. 71], later on AI researchers found it useful to apply deduction methods to Natural Language Understanding systems, Question-Answering systems and Data Base systems [Bobrow 68; Bruce 72; Fikes and Hendrix 78; Green 69; Hendrix 75; 78; 79; McCarty 68; McCarty and Hayes 69; Rieger 75; Shapiro 76; 78; 79; Shapiro and McKay 80].

Several formalisms have been developed to implement deductive systems: the Refutation Method [Chang and Lee 73; Nilsson, 71], Production Systems [Davis and King 77] and Semantic Networks [Deliyanni and Kowalski 79; Fikes and Hendrix 78; Hendrix 75; 78; 79; Schubert et. al. 79; Shapiro 76; 78; 79; Shapiro and Mckay 80]. If one has a system capable of performing inference there are basicaly two types of inference modes that can be implemented:

1. Backward Inference or Inference at Question Time: starting with the goal that has to be proved the system searches for facts or rules that will help prove the desired goal; the advantage of backward inference is that it permits the maintenance of a relatively small data base of facts and rules for use when deducing new facts when they are

needed. The disadvantage of backward inference is the need to do some processing when ever a new new fact is requested, i.e. one has to wait until the fact is deduced.

2. Forward Inference or Inference at Knowledge Acquisition Time: when some new fact is added to the system it tries to use it, together with all other stored information to infer new facts. Starting with the fact that has just been added the system searches for all facts and rules that would enable it to deduce new facts. The advantage of forward inference is that everything that the system knows is explicitly stored. The disadvantage of forward inference is the fact that the database will be full of information that may never be needed.

This paper describes the implementation of forward inference in the SNePS deduction system [Shapiro 76; 79]. This implementation grew out of the work done for an independent study course taken during the Spring '80 semester under the supervision of Dr. Stuart Shapiro.

2. Goals of the forward inference implementation

A SNePS semantic network [Shapiro 76; 79] is a labeled directed graph in which nodes represent concepts and arcs represent non-conceptual binary relations between concepts.

The SNePS network permits the representation of deduction rules. A deduction rule is represented in the network by a rule node.

Before the work being reported in this paper had started, the SNePS deduction system basically only allowed backward inference. Backward inference was done by setting up a set of processes, each one with certain duties, that run under the MULTI multiprocessing sytem [McKay and Shapiro 80]. The processes built during the deduction of some fact are not erased after the fact is deduced and may be used later on while deducing new facts (answering further questions).

Forward Inference has been implemented using the MULTI system and one of the goals kept in mind during the "working out" of the implementation details was that the set of processes left behind by the system after forward inferencing was done should be similar to the set of processes left behind by backward inference if it was trying to deduce the same results. This would mean that after the set of processes is built it would no longer be possible to tell whether any process was built by forward or backward inference. The processes built in either type of inference should also be able to intersect smoothly with each other, i.e. if some process built during one kind of inference needs a process that has already been built while deducing some fact using the other kind

of inference then the system should be able to find this process and use it, instead of creating a new one.

One of the earliest questions raised during this work was when to do (and when not to do) forward inferencing. It was clear that forward inference should only be triggered when the newly asserted node matches a node which is in antecedent position of some rule, but should it be triggered everytime this situation would happen? Perhaps the best way to analyse this problem is by looking at some examples.

Suppose that in the network there is the rule represented in Figure 1. Roughly we can read this rule as "if M1 then M2". In

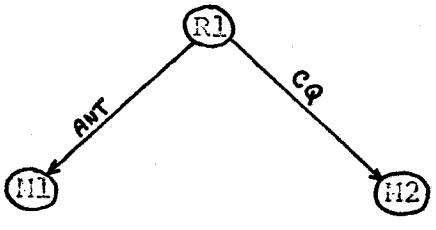

Figure 1
Simple rule

the typical case both M1 and M2 contain variables and the rule should be interpreted in the following way: "If there is a node in the network, say A, which is a fully ground instance of the node M1, with b being the substitution that when applied to M1 produces A, then we can infer the node resulting from applying the substitution b to M2". We will henceforth refer to this by saying that if there is a node, say A, that matches M1b then we can assert M2b.

With rule R1 and the assertion in the network of some node

which matches M1b forward inference should deduce M2b. This corresponds to the well known "modus ponens" rule of inference.

But now, suppose that rule R1 is not asserted in the network but rather embeded within some other rule. There are two possible cases for this rule embeding:

1. R1 is in antecedent position of some other rule (Fig.2a).

2. R1 is in consequent position of some other rule (Fig.2b).

Figure 2
Simple cases of rule embeding

Suppose that, in the first case, we assert a node that matches M1b. Then what? In this case since R1 is not asserted in the network we can not infer M2b since we simply don't know if the rule R1 "holds". The first step that we would have to do is to prove that rule R1 holds and in that case assert M3b since the antecedent of R2b would then hold. The proof that rule R1b holds would have to be done in the following way: assume M1b and then using only sound rules of inference deduce M2b. The SNePS inference system is not able to do this (yet), and therefore when

some newly asserted node matches a node which is in antecedent position of some rule which is in turn in antecedent position of other rule no forward inferencing is done.

Suppose now that in the case represented in Figure 2b we assert a node matching M1b. Again, we can not use rule R1 unless we prove that it holds. But in this case it is quite simple: the rule of inference "modus ponens" enables us to infer R1b provided that M3b holds. What the forward inference system should then do is to start working on rule R2b, trying to prove its antecedent, i.e. trying to show that an instance of M3b exists in the network or that we can deduce it with the existing facts and rules. In the case of such a proof being successfull modus ponens enables us to infer R1b and therefore since we know M1b again by modus ponens we can assert M2b.

From this example it is easy to see the cases where forward inference should be done: first, forward inference is only triggered when some newly asserted node matches a node in antecedent position of some rule; next, this rule should be either asserted or in consequent position of some other rule. The forward inference system should then start to work on the embedding rule and again, either the rule is asserted and it will try to prove its antecedents or the rule is in consequent position of some other rule and the process recurs. If during this procedure we encounter one rule in antecedent position of some other rule the whole process stops.

This line of reasoning can be directly applied to the SNePS connectives v-> and &->, the only difference between them being

that for v-> we can infer all the consequents of the rule as soon as we can prove one single antecedent whereas for the &-> we need to prove all the antecedents of the rule before we can infer the consequents.

But, "what about the other SNePS connectives?" one may ask. Well, AND-OR and THRESH made no distinction among their arguments. This means that one particular argument may be looked at as being an antecedent or as being a consequent depending on the case in which the connective is being used. In the case of THRESH and AND-OR, if we assert in the network a node which is an ARGument of one of these connectives, the node matched will be looked at as being one antecedent and, depending on the parameters of the connective, to use the rule we may either need to find more antecedents or not, in either case all the nodes that are not used as antecedents are looked at as being consequents.

The cases in which forward inference will be triggered can then be summarized in the following way: "forward inference is done when a node is asserted which matches some node in antecedent (ANT or &ANT) or argument (ARG) position of some rule. If the rule is asserted it will try to use the rule. If the rule is not asserted and is in consequent (CQ) or argument (ARG) position of some other rule it will follow the CQ^c or ARG^c arc and apply the same reasoning."

The path of arcs to be followed during forward inference is represented in Figure 3 where "top?" means that an asserted node is found.

$$\left\{ \begin{array}{l} \overset{c}{\text{ANT}} \\ \overset{c}{\text{\&ANT}} \\ \overset{c}{\text{ARG}} \end{array} \right\} . \left[\left[\overset{c}{\text{CQ}} \right]^{*} . \left[\overset{c}{\text{ARG}} \right]^{*} \right]^{*} . \text{ top?}$$

Figure 3
path of arcs to be followed during forward inference

Having determined when to do forward inference the next step will be to try to answer the following questions: "What processes should be responsible for forward inference?" and "What should these processes do?". These questions are answered in the next section which deals with the implementation details.

3. Forward Inference Processes

The forward inferencing mechanism is initiated through the use of the function ADD: (ADD R_1 NS_1 ... R_k NS_k) instructs the system to add to the network the node with R_1 to NS_1, ..., R_k to NS_k and do forward inferencing with that node. As we saw before the first step in forward inferencing will be to match the given node against the network and check if any of the matched nodes is in antecedent or argument position of some rule, in which case further processing has to be done.

A new process was created to accomplish this job: the process F-INFER is given a newly asserted node and its task is to match the node against the network and for each of the matched nodes and respective bindings check if they are in antecedent (or argument) position of some rule(s) and if it is the case set up the appropriate set of processes (described later on) to use such rule(s).

Besides F-INFER another process was created, the process IMPLY. IMPLY is a process which is responsible for a given rule: it receives instances of the consequents of the rule and determines what to do with them. IMPLY receives messages of the kind "the consequents of the rule you are working on hold with binding b". Upon receiving such a message IMPLY looks at each consequent of the rule in turn. If the consequent is a rule it means that such a rule holds with binding b, in which case IMPLY sets up the appropriate structure of processes to use the rule; if the consequent is not a rule it means that the node resulting from the application of the binding b to the consequent can be

deduced and that is exactly what IMPLY does in this case: asserts the instance of the consequent in the network and sets up a new F-INFER process to do further forward inference on such a node.

4.How it all works

In this section an hypothetical example of how forward inference works will be fully developed.

Suppose that the rule represented in Figure 4 exists in the network. Suppose also that the user uses the function ADD to

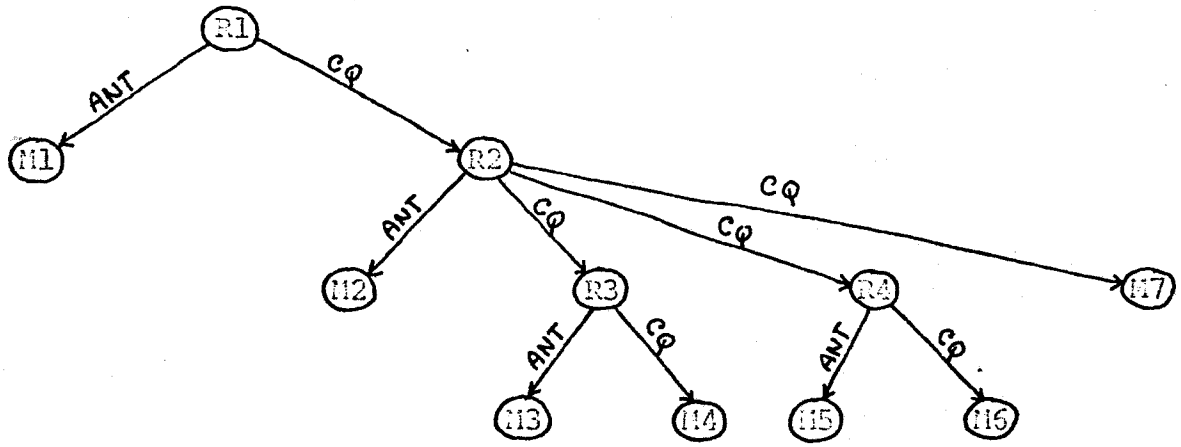

Figure 4
rule existing in the network

build a node M2 which is a fully ground node matching M2b. The function ADD creates an F-INFER process to work on the node N2. This process does a match in the network and finds M2, which is in antecedent position of rule R2, and this is enough for the system to try to do some forward inferencing. Process F-INFER builds two processes: an INPLY process which will be responsible for rule R2 and a GO-UPI1 process that checks if the rule R2 is asserted and if not crawls up on CQ^C or ARG^C arcs (if such arcs exist) to take care of the next higher up rule. The process GO-UPI1 is scheduled. Figure 5 shows the processes built so far. In this Figure and in the ~~forthcoming~~ subsequent Figures a solid arow going from process A to process B means that A reports (sends messages)

to process B (sometimes said that B is the "boss" of A); a dashed arrow going from process A to process B means that process A creates process B. If process A also schedules process B, the arrow head will be solid. The job of the F-INFER process is now

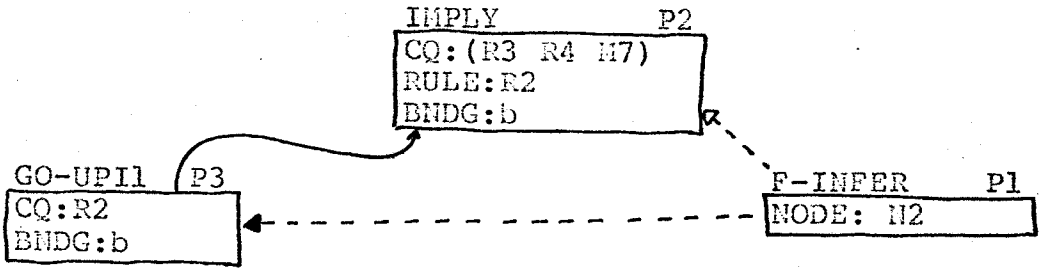

Figure 5
Complete set of processes - phase 1

completed, it stops execution and the scheduler initiates the process GO-UPI1. This process crawls up on the CQ^c arc leading to rule R1. Its being able to do this means that the rule R2 is not asserted in the network and therefore before we can use it we have to prove that the rule holds with binding b, i.e. first we have to be able to use rule R1b. The job of GO-UPI1, in this case, is to create an IMPLY process that will be responsible for rule R1 and which will be reporting (sending messages) to a data-colletor (ANS-CATCH process) which, in turn, reports to the process P2 (Fig.6). Furthermore process P3 creates another GO-UPI1 process which will check whether or not rule R1 is asserted and schedules this process. The set of processes built up to this point is shown in Figure 6. Since process P3 has completed its job it stops execution and the scheduler initiates the process P6. Since the rule R1 is asserted in the network the process P6 creates the structure necessary to use the rule, i.e.

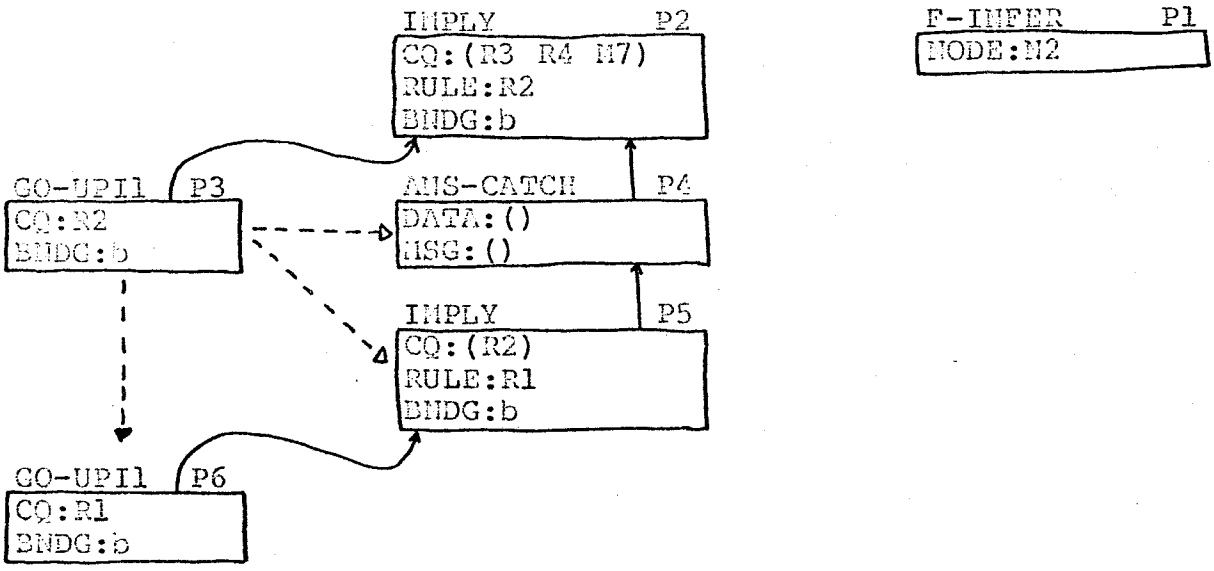

Figure 6
Complete set of processes - phase 2

the processes ANS-CATCH (P7) and USE (P8) and schedules P8 for execution. The set of processes built up to this point is shown

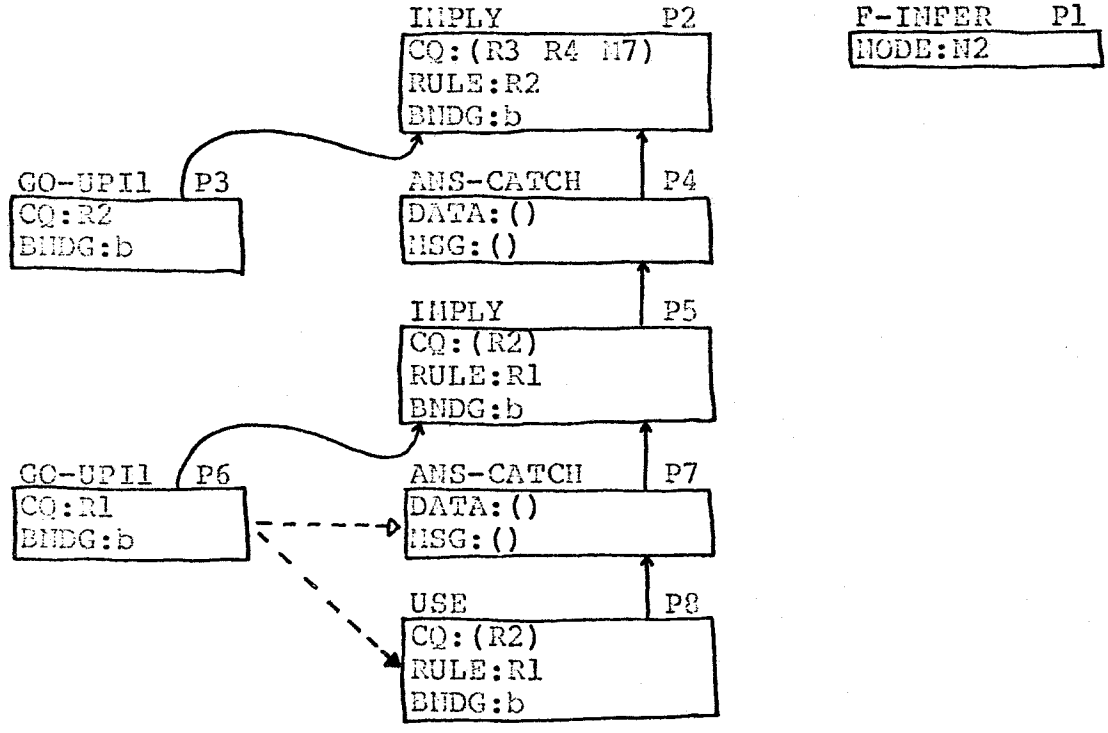

Figure 7
Complete set of processes - Phase 3

in Figure 7. Process P6 stops execution and process P8 is

initiated. This process will, in turn, initiate some other processes whose joint task is to check if there is any node in the network that matches M1b (the antecedent of R1). If such is the case modus ponens permits us to infer that the consequent of R1b holds. Let us suppose that there exists in the network a node satisfying such a condition, say M1. The processes that USE initiates will find this node and will report such a node to the ANS-CATCH above the USE (P8). ANS-CATCH compares the answer received with the data stored in its DATA: register (containing a record of all the messages received so far) and if a similar message has not been received before, as is the case, since no messages at all were received, it reports the message to its bosses, in this case IMPLY (P5). When IMPLY gets the message stating something like "the rule that you are interested in (R1) holds and an instance of its antecedent was found with binding b". IMPLY looks at the consequent of R1 and since it is a rule (R2) it sets up the structure of processes necessary to use the rule, which, as we have already seen before, is a USE process which reports to an ANS-CATCH. The job of the IMPLY process is now done, it schedules the newly created USE process and stops execution. The structure of the processes built so far is represented in Figure 8. The scheduler now initiates process P10 (USE) which, as before, will set up some structure of processes to use the rule R2, i.e. find an instance of M2b. In this case, however, since an instance of the antecedent of R2 was the node whose assertion in the network triggered the forward inference process, part of the structure to use the rule is already there. This piece of structure was built by the F-INFER (P1) which was

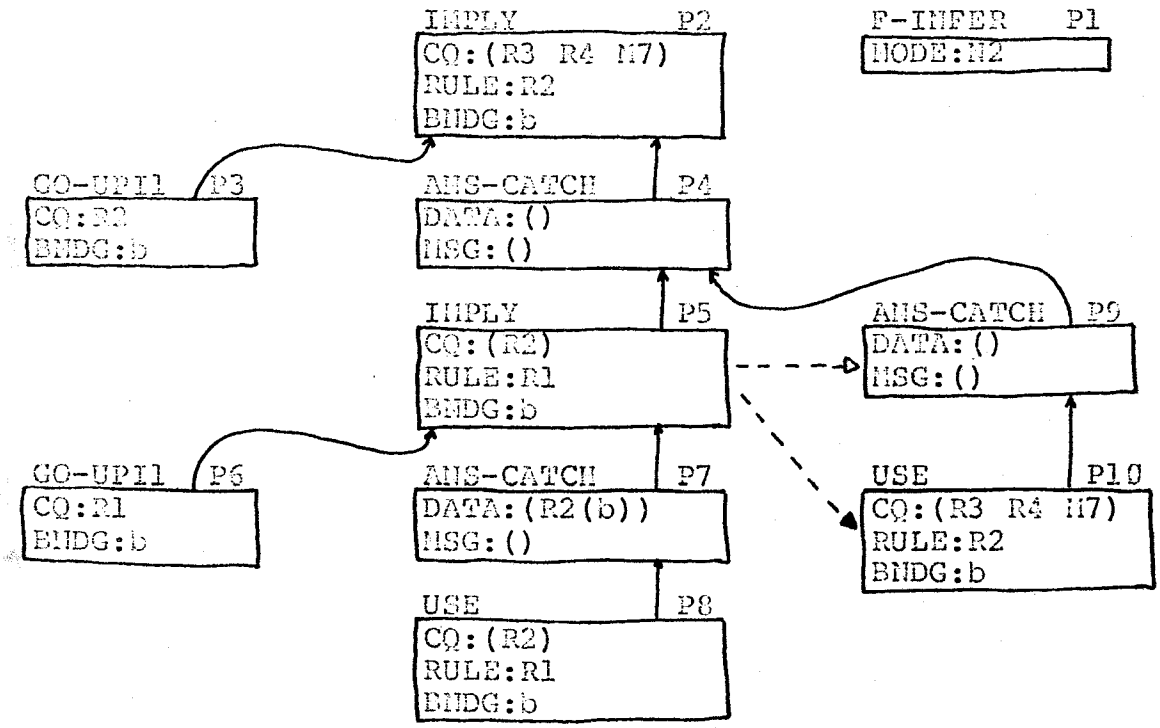

Figure 8
Complete set of processes - phase 4

foreseeing the possibility that later on such set of processes would be needed. This set of processes is linked to the processes created by USE. The advantage of having F-INFER create these processes is that the system doesn't need to look at the network and check if there is a node which matches N2b, because in this case it knows that N2 is a node in such a situation. The execution of these processes results in P9 receiving a message stating that N2 is a node that matches N2b. This message is propagated through the chain of processes until it reaches P2. P2 gets a message stating that the rule it is interested in (R2) holds and that an instance of its antecedent with binding b was found, therefore by modus ponens it can assert the consequents of the rule with binding b. What IMPLY does is looking at each consequent of R2 in turn: both consequents R3 and R4 are rules and therefore these rules hold with binding b. P2 creates an

IMPLY process that will be responsible for each of these rules and since the rules hold it also creates the structure to use the rules, i.e. USE processes P14 and P17 and schedules these processes for execution (Fig.9); the consequent M7 is not a rule,

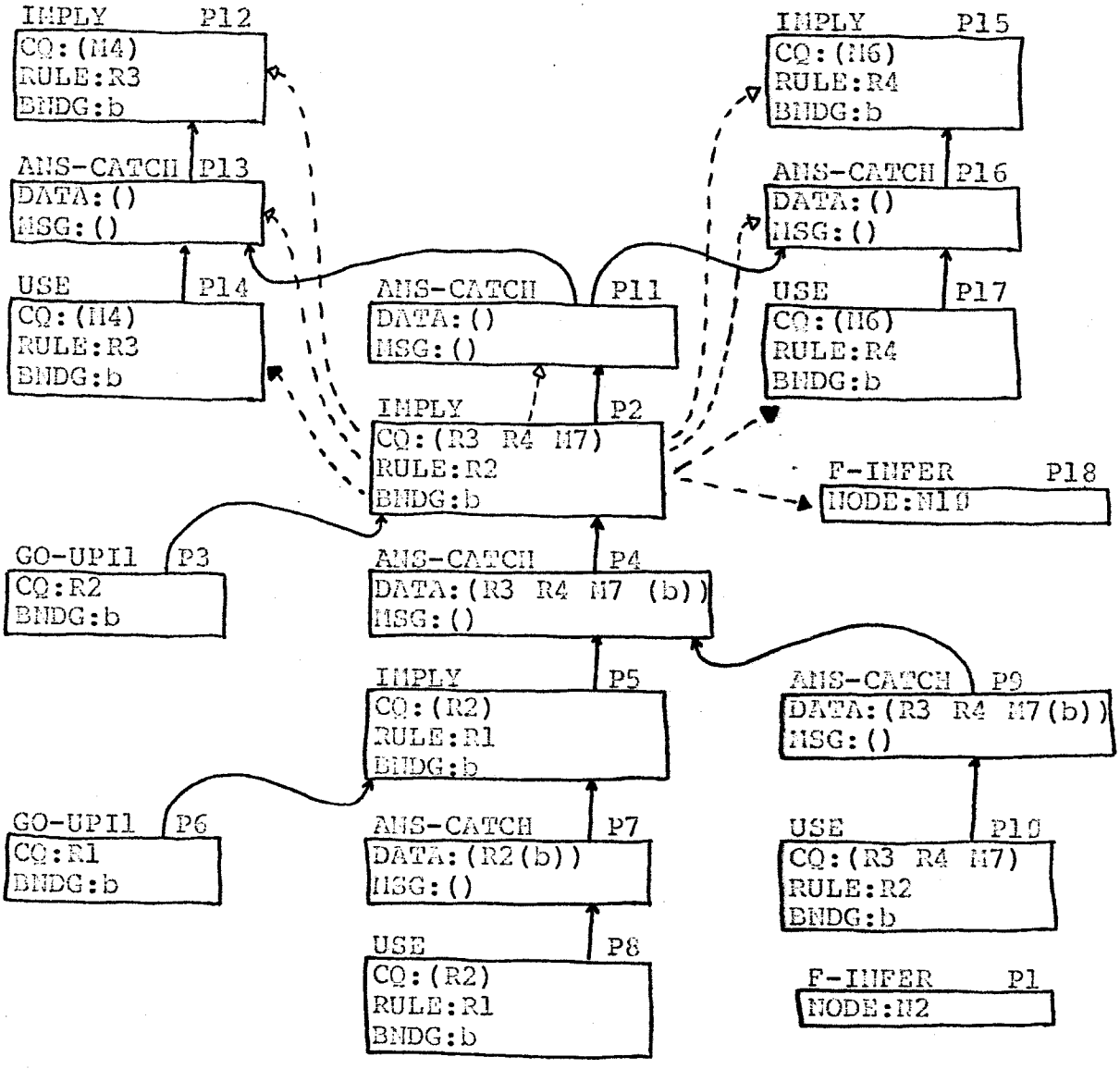

Figure 9
Complete set of processes - phase 5

therefore an instance of M7b is asserted in the network, suppose it results in the assertion of node N10, an F-INFER process (P18) is created to do further forward inference on the node N10. The

task of process P2 is now completed, it stops execution and the scheduler picks up one of the processes P14, P17 or P18 for execution. Let us suppose that P14 is initiated first. The same procedure is repeated: USE (P14) creates the set of processes to use rule R3, the execution of these processes results in doing a match in the network to check if there is any node matching M3b. If that is the case messages are propagated to the IMPLY process P12 which will assert in the network an instance of node M4b and create an F-INFER to work on such a node. And the same holds for the task of process P17.

In the next section and in appendix 2 are presented further examples of use of forward inference.

5.ADDing rules to the network

The example presented in the previous section showed the case where we have rule(s) in the network and ADD a node that matches the antecedent of some rule. In this section is presented the case of having a network with rule and non-rule nodes and ADDing a rule whose antecedent(s) are matched by nodes existing in the network. In this case not only the rule should be asserted in the network but it should also be used to derive new facts.

In this case the function ADD does not initiate an F-INFER process that matches the newly asserted (rule) node against the network. The reason for this is that, as said before in section 2, we don't allow inference on rules which have other rules in antecedent position. Instead, the function ADD creates an IMPLY process that will be responsible for the new rule and a GO-UPI1 process that, when scheduled, will create an USE process to use the rule. We present next a detailed and fully developed example of how this case works. Suppose that we have in the network a node that asserts that "Socrates is a man" (Fig.10) Suppose now

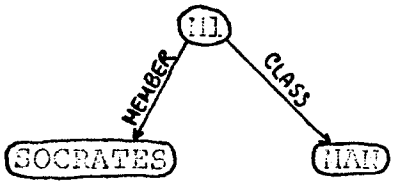

Figure 10
Network representation of "Socrates is a man"

that using the function ADD we add to the network the rule that "All men are mortal" (Fig.11). As we said, the function ADD creates an IMPLY process responsible for rule M4 and a GO-UPI1

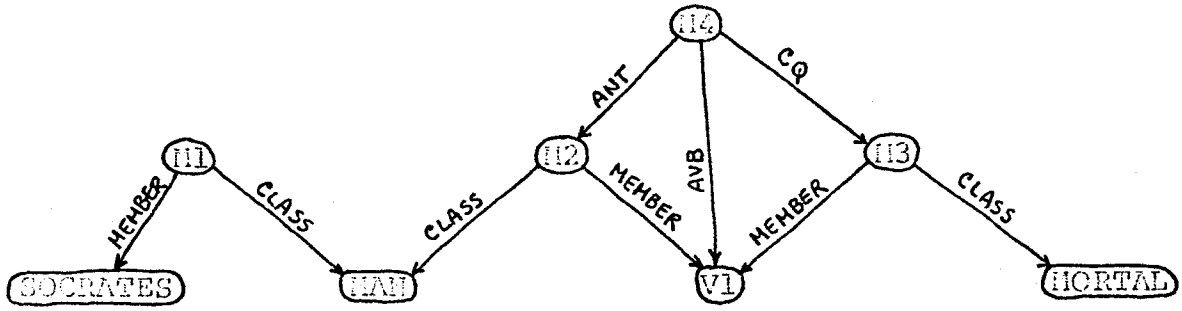

Figure 11
Network built so far

process to work on such a rule. GO-UPI1 is initiated and since M4 is asserted it creates an USE process to use the rule (Fig.12).

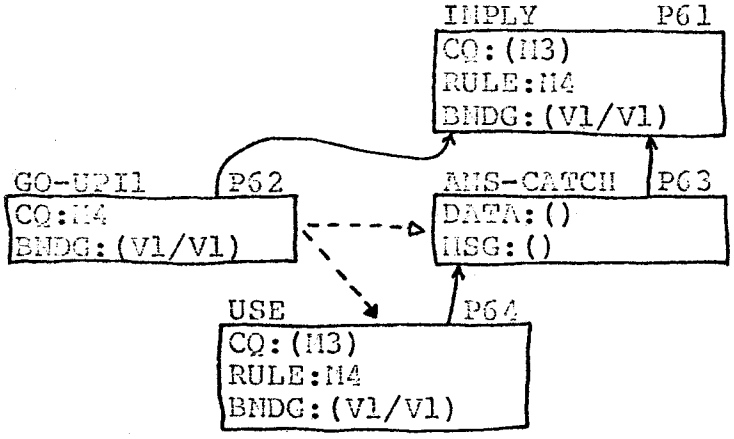

Figure 12
Structure of processes - phase 1

At this point process P62 stops execution and process P64 (USE) is initiated. The USE process creates the set of processes that use the rule, such set of processes is represented in Figure 13 (for a description of these processes refer to Appendix 1). Process P64 schedules P68 for execution and halts. The scheduler initiates process P68 which does a match in the network to see if there is any node matching M2 and finds M1. INFER informs its boss (TOPINF) about this fact, which in turn informs process P65 (which is now NAME:ed CHENT-R because it is receiving messages --

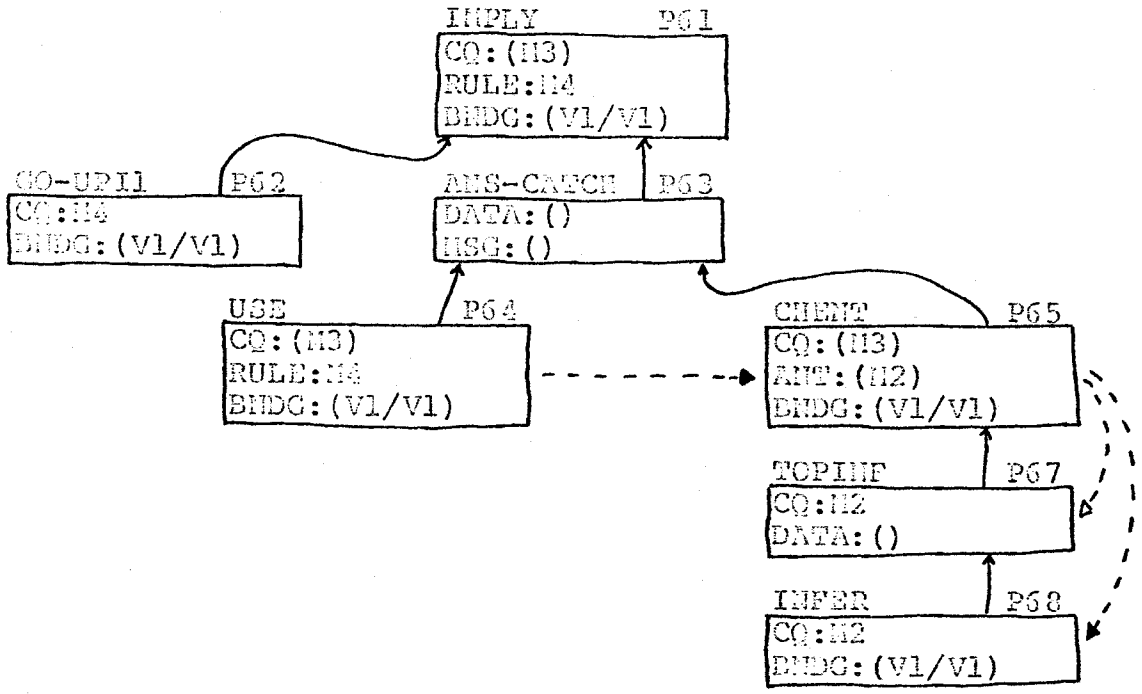

Figure 13
Structure of processes - phase 2

refer to Appendix 1) and so on. The message finally reaches P61 which asserts in the network the node resulting from applying the binding (SOCRATES,V1) to node H3 (Fig.14) and creates an F-INFER

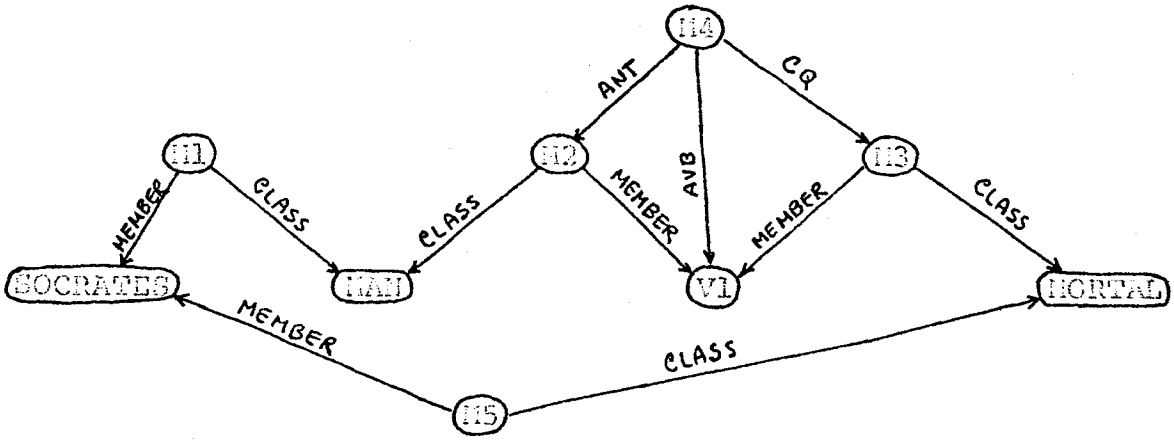

Figure 14
Network after forward inference

process which tries to do further forward inference with such a node (H5), which is not possible since there are no rules in the

network whose antecedent(s) are matched by M5, and the whole process stops. The final structure of processes after forward

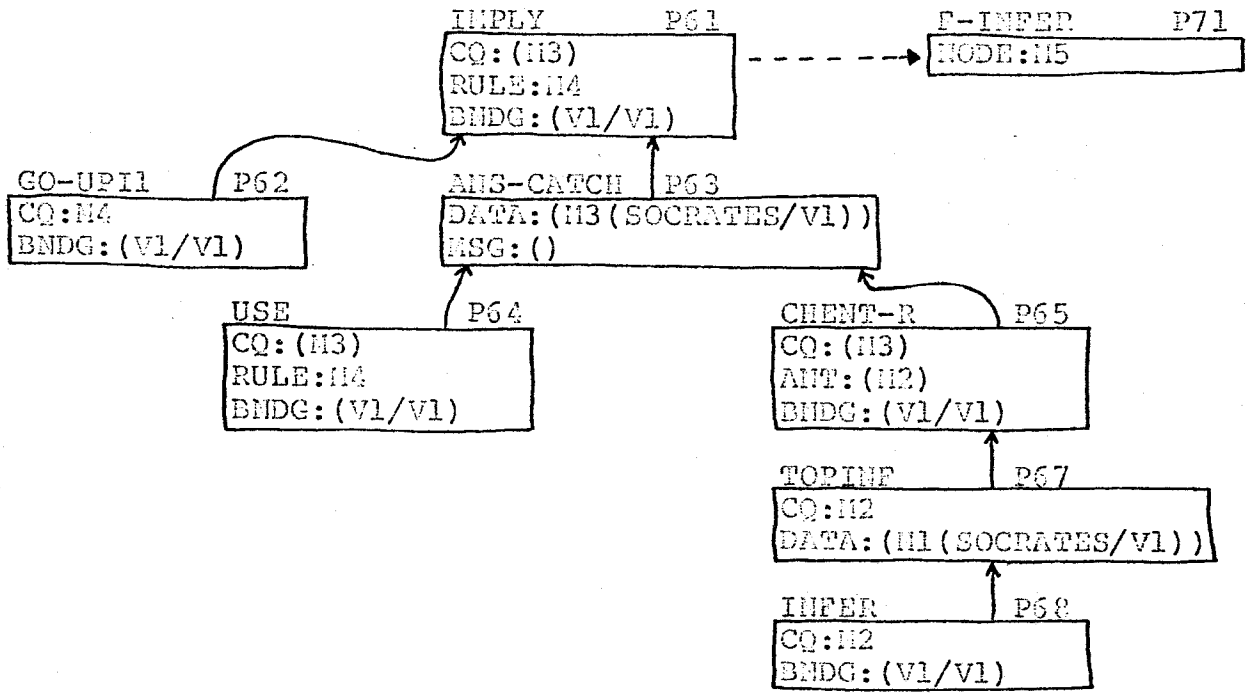

Figure 15
Structure of processes - last phase

inference is done is represented in Figure 15.

6. References

1. Bobrow D., "Natural Language Input for a Computer Problem Solving System", in Semantic Information Processing, M. Minsky (ed.), MIT Press, 1968

2. Bruce B., "A model for temporal references and its application in a question answering program", Artificial Intelligence, Vol.3, pp.1-25, 1972.

3. Chang C. and Lee R., Symbolic Logic and Mechanical Theorem Proving, Academic Press, 1973.

4. Coelho H. and Pereira L.M., "The dialetic development of GEOM, a PROLOG Geometry Theorem Prover", Department of Artificial Intelligence, Edinburgh University, 1975.

5. Davis R. and King J., "An overview of production systems", in Machine Intelligence 8, Elcock and Michie (eds.), Halsted Press, 1977.

6. Deliyanni A. and Kowalski R., "Logic and Semantic Networks", Comm. ACM, 22, pp.184-192, 1979.

7. Duda R., Hart P., Nilsson N. and Sutherland G., "Semantic Network Representation in Rule-Based Inference Systems", in Pattern Directed Inference Systems, Waterman and Hayes-Roth (eds.), Academic Press, 1978.

8. Fikes R. and Hendrix G., "The Deduction Component", in Understanding Spoken Language, D.E. Walker (ed.), Elsevier North-Holland, 1978.

9. Gelernter H., "Realization of a Geometry-theorem Proving

Machine", in Computers and Thought, Feigenbaum and Feldman (eds.), McGraw-Hill, 1963.

10. Green C., "Theorem Proving by Resolution as a basis for Question-Answering Systems", in Machine Intelligence 4, Michie and Meltzer (eds.), Edinburgh University Press, 1969.

11. Hendrix G., "Expanding the utility of Semantic Networks through partitioning", Proc. IJCAI-75, pp.115-121.

12. Hendrix G, "The Representation of Semantic Knowledge", in Understanding Spoken Language, D.E. Walter (ed.), Elsevier North-Holland, 1978.

13. Hendrix G., "Encoding knowledge in partitioned networks" in Associative Networks, N.V. Findler (ed.), Academic Press, 1979.

14. Hewitt C., "Description and theoretical analysis of PLANNER: A language for proving theorems and manipulating models in a robot", Tech. Report No. AI-TR-258, MIT AI-Lab, 1972.

15. Kowalski R., "Search Strategies for Theorem Proving", in Machine Intelligence 5, Michie and Meltzer (eds.), Edinburgh University Press, 1970.

16. Kowalski R., Logic for Problem Solving, Elsevier North-Holland, 1979.

17. McCarty J., "Programs with Common Sense" in Semantic

Information Processing, M. Minsky (ed.), MIT Press, 1968.

18. McCarty J. and Hayes P., "Some philosophical problems from the standpoint of Artificial Intelligence", in Machine Intelligence 4, Michie and Meltzer (eds.), Edinburgh University Press, 1969.

19. McKay D. and Shapiro S., "MULTI a LISP-Based Multiprocessing System", Technical Report No.164, Department of Computer Science, SUNY at Buffalo, 1980.

20. Newell A., Shaw J. and Simon H., "Empirical Explorations with the Logic Theory Machine: a case study in heuristics", in Computers and Thought, Feigenbaum and Feldman (eds.), McGraw-Hill, 1963.

21. Nilsson N., Problem Solving Methods in Artificial Intelligence, McGraw-Hill, 1971.

22. Nilsson N., Principles of Artificial Intelligence, Tioga, 1980.

23. Raphael B., "SIR: Semantic Information Retrieval", in Semantic Information Processing, M. Minsky (ed.), MIT Press, 1968.

24. Rieger C., "Conceptual Memory and Inference", in Conceptual Information Processing, R.C. Schank (ed.), Elsevier North-Holland, 1975.

25. Robinson J. A., "A Machine-Orienteed logic based on the Resolution Principle", Journal of the ACM, Vol.12, No.1, 1965.

26. Schubert L., Goebel R. and Cercone N., "The structure and organization of a Semantic Net for Compreension and Inference", in Associative Networks, N.V. Findler (ed.), Academic Press, 1979.

27. Shapiro S.C., "An introduction to SNePS", Technical report No.31, Computer Science Department, Indiana University, 1976.

28. Shapiro S.C., "Path-based and Node-based inference in semantic networks", Proc. TINLAP2, 1978.

29. Shapiro S.C., "The SNePS semantic network processing system", in Associative Networks, N.V. Findler (ed.), Academic Press, 1979.

30. Shapiro S.C. and McKay D.P., "Inference with recursive rules", Proc. First AAAI Conference, pp.151-153, 1980.

31. Sussman G., Winograd T. and Charniak E., MICRO-PLANNER reference manual, Tech. Report No. 203a, MIT AI-Lab, 1971.

APPENDIX 1: Some particular processes

The material presented in this appendix was adapted from a Grant Proposal submited by Dr. Stuart Shapiro to N.S.F. in November 1979.

1. INFER is the process which matches a network structure in a particular binding within the current data base. The result of the match yelds assertions and patterns which are possible rule structures. Assertions are immediately sent to INFER's boss as instances of the original structure. Possible rule structures are investigated further. Each INFER process which is created is remembered. Whenever a new INFER process is to be constructed a check is first made to determine if some other INFER is already working on the same problem. If no such INFER process exists a new INFER with a TOPINF as its boss (see below for a description of TOPINF) is created. If an already existing INFER is acceptable then the process trying to create the INFER is added to the bosses of the existing INFER's TOPINF and is immediately sent any results the TOPINF has recorded.

2. TOPINF acts as data collector for INFER processes. TOPINF receives messages from processes created below it and remembers each different message. When a message is received that has not been propagated previously TOPINF sends the message to all its bosses.

3. CHENT is the process which performs or-entailment (v->).

CHENT attempts to establish any of the antecedents of the or-entailment so that a given set of consequents (eventually containing all of the consequents) can be instantiated in a particular binding b. For each antecedent an INFER process is created which is to verify the antecedent in the binding b. If a suitable INFER process already exists, a new one is not created. (see above) The CHENT changes its name to CHENT-R to receive answers. When an INFER process reports an answer consisting of a particular antecedent and a particular binding and meaning that an instance of such antecedent with such a binding was found, the message that the set of consequents in which the CHENT-R process is interested holds in such a binding is sent to the CHENT-R's boss.

4. CHSENT is the process which performs and-entailment (&->). It works similarly to CHENT but it tries to establish all the antecedents of the and-entailment in a particular binding.

5. CHANDOR is the process that implements the and-or connective ($_{n}\mathbf{X}_{1}^{j}$). One of the particularities of this connective is that any argument can be either antecedent or consequent. CHANDOR tries to create or find INFER processes for all its arguments. CHANDOR changes its name to CHANDOR-R to receive answers. When the answers are returned CHANDOR-R maintains counts of the answers indexed on the bindings in the result. Those arguments for which answers are supplied are considered antecedents and the remaining the consequents. If j antecedents are ever derived for a

particular binding b a message is sent to CHANDOR-R's boss stating that the negation of the consequents (all ARGuments of the AND-OR except the j antecedents derived) hold with binding b.

6. CHTHRESH is the process that implements the thresh connective. It works very similarly to CHANDOR.

7. USE the process USE tries to deduce a set of consequents using a given rule with a given binding. It creates a CH-process (CHENT, CH&ENT, CHANDOR or CHTHRESH, depending on the connective of the rule) that will work on the rule.

8. F-INFER is a process which triggers forward inference. It is given a newly asserted node and its task is to match the node against the network and for each of the matched node(s) and respective bindings check if they are in antecedent (or argument) position of some rule(s) and if it is the case set up the appropriate set of processes to use the rule(s).

9. IMPLY is a process which is responsible for a given rule. It receives messages stating that certain consequents of the rule hold in a given binding and determines what to do with them: if the consequent is a rule it sets up the processes necessary to use the rule; if the consequent is not a rule it asserts in the network the node resulting from applying the binding to the consequent.

APPENDIX 2: Fully developed example

In this appendix is presented a complete example of forward inference which involves several of the connectives available in SNePS (&->, AND-OR and THRESH). Also shown here are some of the options available to the users of the inference system regarding the form of the final output.

Consider the network represented in Figure 16. The network

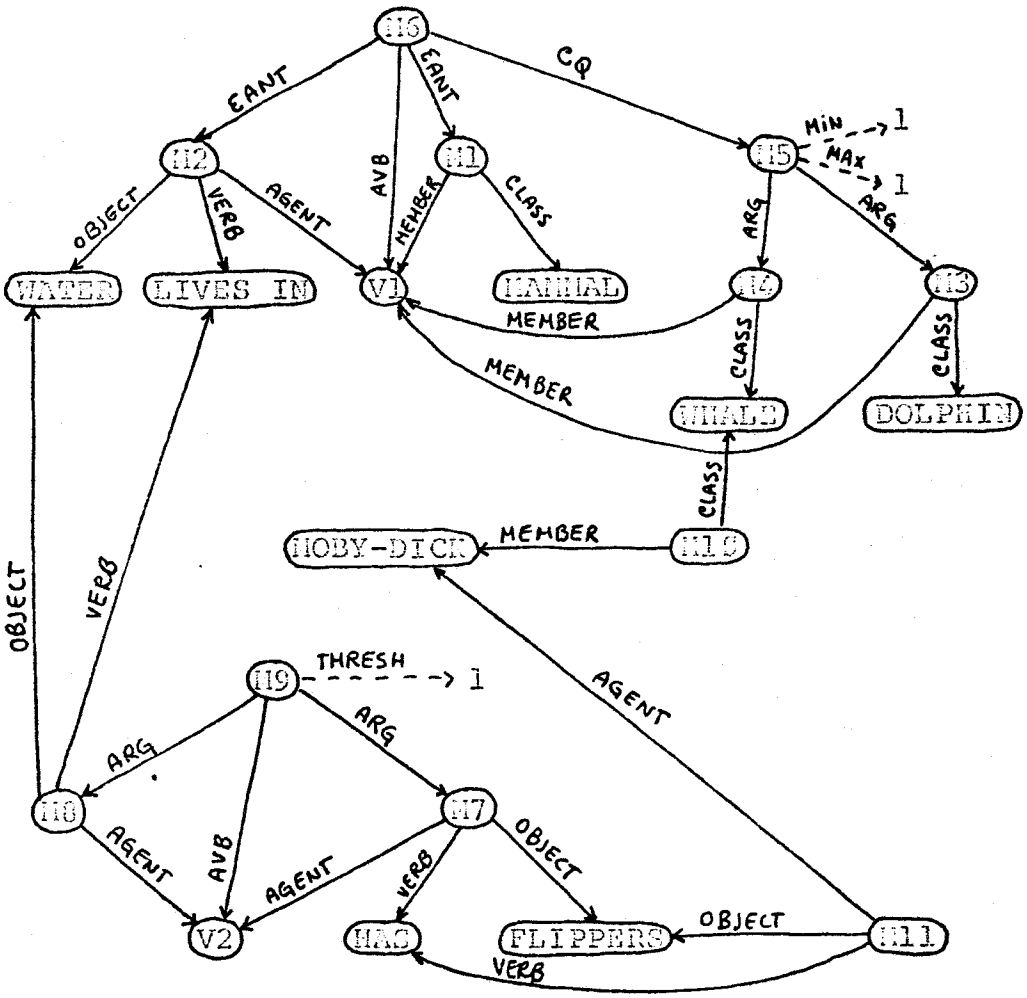

Figure 16
Network before forward inference

contains two rules: "Every mammal which lives in water is either a whale or a dolphin" (M6) and "An animal lives in water iff it

has flippers" (M9). It also contains two ground assertions: "Moby-Dick is a whale" (M10) and "Moby-Dick has flippers" (M11). The sample run presented in this appendix consists of first building such a network and then using the function ADD to add the node "Moby-Dick is a mammal" (M12) to the network.

The first run, presented next, uses an ATN grammar generator to output the inferences done by the system in a natural English way:

```
? (SHEPS)
SHEPS
* (DEFINE MEMBER MEMBER-1 CLASS CLASS-1 AGENT AGENT-
*          VERB VERB- OBJECT OBJECT-)
(MEMBER MEMBER-1)
(CLASS CLASS-1)
(AGENT AGENT-)
(VERB VERB-)
(OBJECT OBJECT-)
(DEFINED)
34 MSECS

* (SURFACE (BUILD AVB $X
*                 &ANT ((BUILD MEMBER *X CLASS MAMMAL)
*                       (BUILD AGENT *X
*                              VERB LIVES/ IN
*                              OBJECT WATER))
*                 CQ   (BUILD MIN 1
*                             MAX 1
*                             ARG ((BUILD MEMBER *X CLASS WHALE)
*                                  (BUILD MEMBER *X
*                                         CLASS DOLPHIN)))))
IF V1 LIVES IN WATER AND
   V1 IS A MAMMAL
   THEN EITHER V1 IS A DOLPHIN OR
               V1 IS A WHALE
(DUMPED)
3329 MSECS

* (SURFACE (BUILD AVB $Y
*                  THRESH 1
*                  ARG    ((BUILD AGENT *Y
*                                 VERB  HAS
*                                 OBJECT FLIPPERS)
*                           (BUILD AGENT *Y
*                                  VERB LIVES/ IN
*                                  OBJECT WATER))))
```

```
V2 LIVES IN WATER IF AND ONLY IF
   V2 HAS FLIPPERS
(DUMPED)
1156 MSECS

* (SURFACE (BUILD MEMBER MOBY-DICK CLASS WHALE))
MOBY-DICK IS A WHALE
(DUMPED)
1102 MSECS

* (SURFACE (BUILD AGENT MOBY-DICK VERB HAS OBJECT FLIPPERS))
MOBY-DICK HAS FLIPPERS
(DUMPED)
323 MSECS

* (SURFACE (ADD MEMBER MOBY-DICK CLASS MAMMAL))

SINCE
MOBY-DICK HAS FLIPPERS

WE INFER
MOBY-DICK LIVES IN WATER

SINCE
MOBY-DICK LIVES IN WATER AND
MOBY-DICK IS A MAMMAL

WE INFER
EITHER MOBY-DICK IS A WHALE OR
       MOBY-DICK IS A DOLPHIN

SINCE
MOBY-DICK IS A WHALE

WE INFER
MOBY-DICK IS NOT A DOLPHIN


MOBY-DICK IS A MAMMAL AND
MOBY-DICK LIVES IN WATER AND
MOBY-DICK IS NOT A DOLPHIN
(DUMPED)
10244 MSECS


* (LISP)
END SNEPS
```

After forward inference is completed the network has three more nodes: "Moby-Dick is a mammal" (M12), "Moby-Dick lives in water" (M13) and "Moby-Dick is not a dolphin" (M15). Such a network is presented in Figure 17.

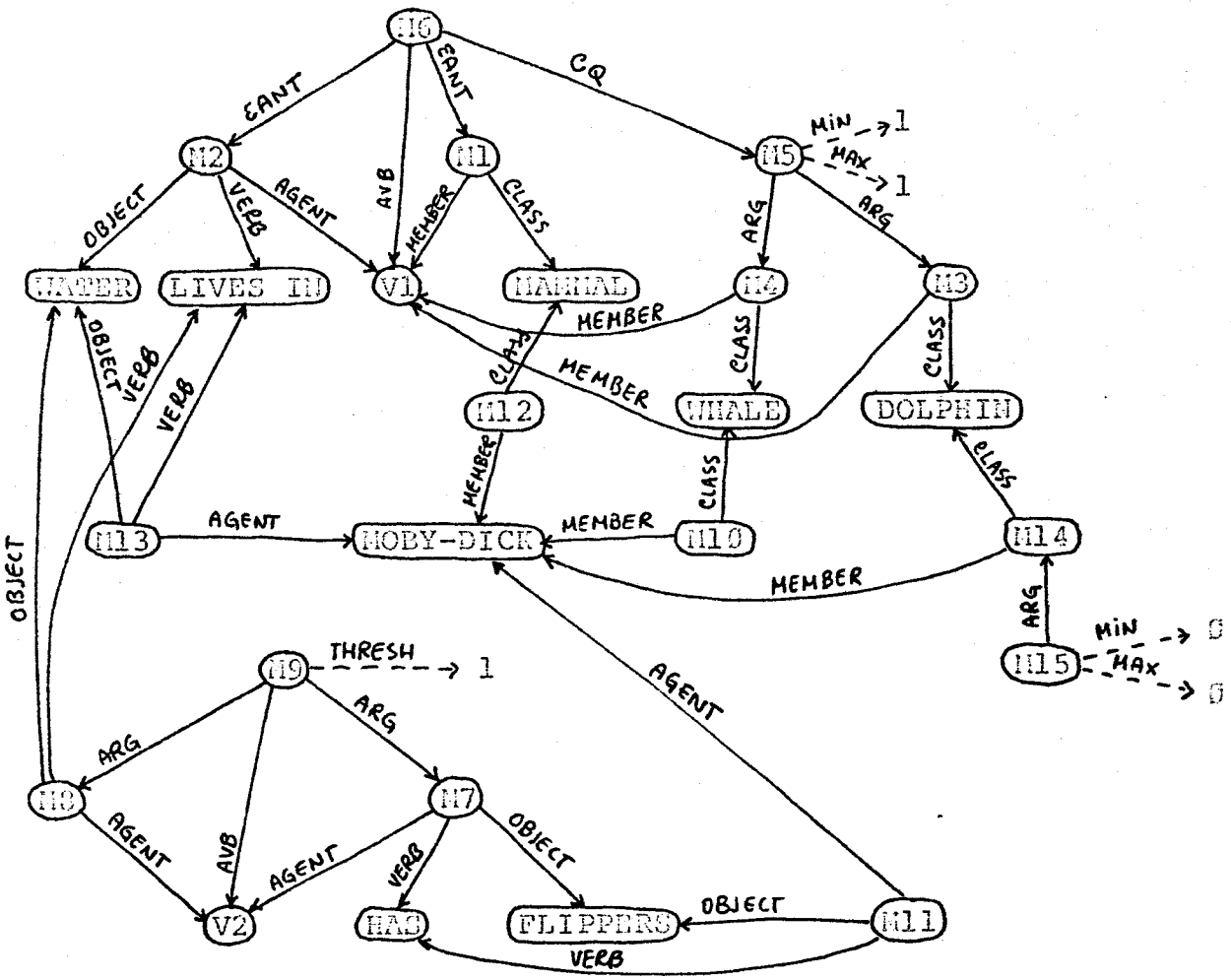

Figure 17
Network after forward inference

The second run, presented next, traces the execution of all the processes and does not use the ATN grammar generator.

```
? (SNEPS)
SNEPS
* (DEFINE MEMBER MEMBER-1 CLASS CLASS-1 AGENT AGENT-
*          VERB VERB- OBJECT OBJECT-)
(MEMBER MEMBER-1)
(CLASS CLASS-1)
(AGENT AGENT-)
(VERB VERB-)
(OBJECT OBJECT-)
(DEFINED)
35 MSECS

* (DESCRIBE (BUILD AVB  $X
*                  &ANT ((BUILD MEMBER *X CLASS MAMMAL)
*                        (BUILD AGENT *X
*                               VERB LIVES/ IN
*                               OBJECT WATER))
```

```
*                    CQ    (BUILD MIN 1
*                                 MAX 1
*                                 ARG ((BUILD MEMBER *X
*                                             CLASS  WHALE)
*                                      (BUILD MEMBER *X
*                                             CLASS  DOLPHIN)))))
(M6
 (CQ
  (M5
   (:SVAR (V1 (:VAR (T))))
   (ARG
    (M4 (CLASS (DOLPHIN))
        (:SVAR (V1 (:VAR (T))))
        (MEMBER (V1 (:VAR (T)))))
    (M3 (CLASS (WHALE))
        (:SVAR (V1 (:VAR (T))))
        (MEMBER (V1 (:VAR (T))))))
   (MAX (1))
   (MIN (1))))
 (&ANY
  (M2 (OBJECT (WATER))
      (VERB (LIVES IN))
      (:SVAR (V1 (:VAR (T))))
      (AGENT (V1 (:VAR (T)))))
  (M1 (CLASS (MAMMAL))
      (:SVAR (V1 (:VAR (T))))
      (MEMBER (V1 (:VAR (T))))))
 (AVB (V1 (:VAR (T)))))
(DUMPED)
1253 MSECS

* (DESCRIBE (BUILD AVB    $Y
*                  THRESH 1
*                  ARG    ((BUILD AGENT *Y
*                                 VERB  HAS
*                                 OBJECT FLIPPERS)
*                           (BUILD AGENT *Y
*                                 VERB LIVES/ IN
*                                 OBJECT WATER))))
(M9
 (ARG
  (M8 (OBJECT (WATER))
      (VERB (LIVES IN))
      (:SVAR (V2 (:VAR (T))))
      (AGENT (V2 (:VAR (T)))))
  (M7 (OBJECT (FLIPPERS))
      (VERB (HAS))
      (:SVAR (V2 (:VAR (T))))
      (AGENT (V2 (:VAR (T))))))
 (AVB (V2 (:VAR (T))))
 (THRESH (1)))
(DUMPED)
338 MSECS

* (DESCRIBE (BUILD MEMBER MOBY-DICK CLASS WHALE))
(M10 (CLASS (WHALE)) (MEMBER (MOBY-DICK)))
(DUMPED)
```

```
57 MSECS

* (DESCRIBE (BUILD AGENT MOBY-DICK VERB HAS OBJECT FLIPPERS))
(NIL (OBJECT (FLIPPERS)) (VERB (HAS)) (AGENT (MOBY-DICK)))
(DUMPED)
75 MSECS

* (DESCRIBE (ADD MEMBER MOBY-DICK CLASS MAMMAL))

          ** NEW - F-INFER ID - P60 **

          *****   "ENTERING"   PROCESS :  P60   *****
          NAME: F-INFER
          CLINK: NIL
          MODE: N12

          ** NEW - IMPLY ID - P61 **

          ** NEW - TOPINF ID - P63 **

          ** NEW - INFER ID - P64 **

          ** NEW - GO-UPIl ID - P65 **

          ** INITIATE - GO-UPIl ID - P65 INITIATED BY - P60

          *****   "LEAVING"   PROCESS :  P60   *****
          NAME: F-INFER
          CLINK: NIL
          MODE: N12

          *****   "ENTERING"   PROCESS :  P65   *****
          NAME: GO-UPIl
          CLINK: P61
          CQ: N6
          BNDG: ((V1,MOBY-DICK))

          ** NEW - AMS-CATCH ID - P66 **

          ** NEW - USE ID - P67 **

          ** INITIATE - USE ID - P67 INITIATED BY - P65

          *****   "LEAVING"   PROCESS :  P65   *****
          NAME: GO-UPIl
          CLINK: P61
          CQ: N6
          BNDG: ((V1,MOBY-DICK))

          *****   "ENTERING"   PROCESS :  P67   *****
          NAME: USE
          CLINK: P66
          CQ: (N5)
          RULE: N6
          BNDG: ((V1,MOBY-DICK))
          MSG: NIL
```

```
** NEW - CH&ENT ID - P66 **

** INITIATE - CH&ENT ID - P66 INITIATED BY - P67

*****   "LEAVING"   PROCESS :  P67   *****
NAME: USE
CLINK: P66
CQ: (M5)
RULE: R6
BNDG: ((V1,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P66   *****
NAME: CH&ENT
CLINK: P66
CQ: (M5)
ANT: (M2 M1)
TOT: 2
BNDG: ((V1,MOBY-DICK))
DATA: NIL
MSG: NIL

** NEW - TOPINF ID - P70 **

** NEW - INFER ID - P71 **

** INITIATE - INFER ID - P71 INITIATED BY - P68

** INITIATE - CH&ENT ID - P68 INITIATED BY - P68

*****   "LEAVING"   PROCESS :  P68   *****
NAME: CH&ENT-R
CLINK: P66
CQ: (M5)
ANT: (M2 M1)
TOT: 2
BNDG: ((V1,MOBY-DICK))
DATA: NIL
MSG: ((M1 (M12 ((V1,MOBY-DICK)))))

*****   "ENTERING"   PROCESS :  P68   *****
NAME: CH&ENT-R
CLINK: P66
CQ: (M5)
ANT: (M2 M1)
TOT: 2
BNDG: ((V1,MOBY-DICK))
DATA: NIL
MSG: ((M1 (M12 ((V1,MOBY-DICK)))))

*****   "LEAVING"   PROCESS :  P68   *****
NAME: CH&ENT-R
CLINK: P66
CQ: (M5)
ANT: (M2 M1)
TOT: 2
BNDG: ((V1,MOBY-DICK))
```

```
DATA: ((((V1,MOBY-DICK)) (M1) 1 NIL 0))
MSG: NIL

*****   "ENTERING"   PROCESS :  P71   *****
NAME: INPER
CLINK: P70
CQ: M2
BNDG: ((V1,MOBY-DICK))
MSG: NIL

** NEW - SWITCH ID - P72 **

** NEW - GO-UPIl ID - P73 **

** INITIATE - GO-UPIl ID - P73 INITIATED BY - P71

*****   "LEAVING"   PROCESS :  P71   *****
NAME: INPER
CLINK: P70
CQ: M2
BNDG: ((V1,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P73   *****
NAME: GO-UPIl
CLINK: P72
CQ: M8
BNDG: ((V2,MOBY-DICK))

** NEW - ANS-CATCH ID - P74 **

** NEW - IMPLY ID - P75 **

** NEW - GO-UPIl ID - P76 **

** INITIATE - GO-UPIl ID - P76 INITIATED BY - P73

*****   "LEAVING"   PROCESS :  P73   *****
NAME: GO-UPIl
CLINK: P72
CQ: M8
BNDG: ((V2,MOBY-DICK))

*****   "ENTERING"   PROCESS :  P76   *****
NAME: GO-UPIl
CLINK: P75
CQ: M9
BNDG: ((V2,MOBY-DICK))

** NEW - ANS-CATCH ID - P77 **

** NEW - USE ID - P78 **

** INITIATE - USE ID - P78 INITIATED BY - P76

*****   "LEAVING"   PROCESS :  P76   *****
NAME: GO-UPIl
```

```
CLINK: P75
CQ: M9
BNDG: ((V2,MOBY-DICK))

*****   "ENTERING"   PROCESS :  P78   *****
NAME: USE
CLINK: P77
CQ: (M8)
RULE: M9
BNDG: ((V2,MOBY-DICK))
MSG: NIL

** NEW - CHTHRESH ID - P79 **

** INITIATE - CHTHRESH ID - P79 INITIATED BY - P78

*****   "LEAVING"   PROCESS :  P78   *****
NAME: USE
CLINK: P77
CQ: (M8)
RULE: M9
BNDG: ((V2,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P79   *****
NAME: CHTHRESH
CLINK: P77
THRESH: 1
TOT: 2
CQ: (M8 M7)
ANT: (M8 M7)
BNDG: ((V2,MOBY-DICK))
DATA: NIL
MSG: NIL

** NEW - TOPINF ID - P81 **

** NEW - INFER ID - P82 **

** INITIATE - INFER ID - P82 INITIATED BY - P79

** NEW - TOPINF ID - P84 **

** NEW - INFER ID - P85 **

** INITIATE - INFER ID - P85 INITIATED BY - P79

*****   "LEAVING"   PROCESS :  P79   *****
NAME: CHTHRESH-R
CLINK: P77
THRESH: 1
TOT: 0
CQ: (M8 M7)
ANT: (M8 M7)
BNDG: ((V2,MOBY-DICK))
DATA: NIL
MSG: NIL
```

```
*****   "ENTERING"   PROCESS :  P82   *****
NAME: INFER
CLINK: P81
CQ: M8
BNDG: ((V2,MOBY-DICK))
MSG: NIL

** NEW - SWITCH ID - P86 **

** NEW - GO-UP11 ID - P87 **

** INITIATE - GO-UP11 ID - P87 INITIATED BY - P82

*****   "LEAVING"   PROCESS :  P82   *****
NAME: INFER
CLINK: P81
CQ: M8
BNDG: ((V2,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P85   *****
NAME: INFER
CLINK: P84
CQ: M7
BNDG: ((V2,MOBY-DICK))
MSG: NIL

** INITIATE - TOPINF ID - P84 INITIATED BY - P85

*****   "LEAVING"   PROCESS :  P85   *****
NAME: INFER
CLINK: P84
CQ: M7
BNDG: ((V2,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P84   *****
NAME: TOPINF
CLINK: P79
CQ: M7
BOSSES: (P79)
DATA: NIL
MSG: ((M7 (M11 ((V2,MOBY-DICK)))))
MTR: P83

** INITIATE - CHTHRESH-R ID - P79 INITIATED BY - P84

*****   "LEAVING"   PROCESS :  P84   *****
NAME: TOPINF
CLINK: P79
CQ: M7
BOSSES: (P79)
DATA: ((M11 ((V2,MOBY-DICK))))
MSG: NIL
MTR: P83
```

```
            *****   "ENTERING"   PROCESS :  P79   *****
            NAME: CHTHRESH-R
            CLINK: P77
            THRESH: 1
            TOT: 0
            CQ: (M8 M7)
            ANT: (M8 M7)
            BNDG: ((V2,MOBY-DICK))
            DATA: NIL
            MSG: ((M7 (M11 ((V2,MOBY-DICK)))))

 SINCE
(M7 (OBJECT (FLIPPERS))
    (VERB (HAS))
    (:SVAR (V2 (:VAR (T)) (:VAL (MOBY-DICK))))
    (AGENT (V2 (:VAR (T)) (:VAL (MOBY-DICK)))))

 WE INFER
(M8 (OBJECT (WATER))
    (VERB (LIVES IN))
    (:SVAR (V2 (:VAR (T)) (:VAL (MOBY-DICK))))
    (AGENT (V2 (:VAR (T)) (:VAL (MOBY-DICK)))))

            ** INITIATE - ANS-CATCH ID - P77 INITIATED BY - P79

            *****   "LEAVING"   PROCESS :  P79   *****
            NAME: CHTHRESH-R
            CLINK: P77
            THRESH: 1
            TOT: 0
            CQ: (M8 M7)
            ANT: (M8 M7)
            BNDG: ((V2,MOBY-DICK))
            DATA: ((((V2,MOBY-DICK)) (M7) 1 NIL 0))
            MSG: NIL

            *****   "ENTERING"   PROCESS :  P77   *****
            NAME: ANS-CATCH
            CLINK: P75
            BOSSES: (P75)
            DATA: NIL
            MSG: (((M8 ((V2,MOBY-DICK)))))

            ** INITIATE - IMPLY ID - P75 INITIATED BY - P77

            *****   "LEAVING"   PROCESS :  P77   *****
            NAME: ANS-CATCH
            CLINK: P75
            BOSSES: (P75)
            DATA: ((M8 ((V2,MOBY-DICK))))
            MSG: NIL

            *****   "ENTERING"   PROCESS :  P75   *****
            NAME: IMPLY
            CLINK: P74
            CQ: (M8)
            RULE: M9
```

```
BNDG: ((V2,MOBY-DICK))
MSG: (((M8 ((V2,MOBY-DICK)))))

** NEW - F-INFER ID - P68 **

** INITIATE - F-INFER ID - P68 INITIATED BY - P75

** INITIATE - ANS-CATCH ID - P74 INITIATED BY - P75

*****   "LEAVING"   PROCESS :  P75   *****
NAME: IMPLY
CLINK: P74
CQ: (M8)
RULE: M9
BNDG: ((V2,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P74   *****
NAME: ANS-CATCH
CLINK: P72
BOSSES: (P72)
DATA: NIL
MSG: (((M8 ((V2,MOBY-DICK)))))

** INITIATE - SWITCH ID - P72 INITIATED BY - P74

*****   "LEAVING"   PROCESS :  P74   *****
NAME: ANS-CATCH
CLINK: P72
BOSSES: (P72)
DATA: ((M8 ((V2,MOBY-DICK))))
MSG: NIL

*****   "ENTERING"   PROCESS :  P72   *****
NAME: SWITCH
CLINK: P70
CQ: M2
BNDG: ((V1,MOBY-DICK))
MSG: (((M8 ((V2,MOBY-DICK)))))

** INITIATE - TOPINF ID - P70 INITIATED BY - P72

*****   "LEAVING"   PROCESS :  P72   *****
NAME: SWITCH
CLINK: P70
CQ: M2
BNDG: ((V1,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P70   *****
NAME: TOPINF
CLINK: P68
CQ: M2
BOSSES: (P68)
DATA: NIL
MSG: ((M2 (M8 ((V1,MOBY-DICK)))))
MTR: P69
```

```
            ** INITIATE - CH&ENT-R ID - P68 INITIATED BY - P70

            *****   "LEAVING"   PROCESS :  P70   *****
            NAME: TOPINF
            CLINK: P68
            CQ: N2
            BOSSES: (P68)
            DATA: ((N8 ((V1,MOBY-DICK))))
            MSG: NIL
            MTR: P69

            *****   "ENTERING"   PROCESS :  P68   *****
            NAME: CH&ENT-R
            CLINK: P66
            CQ: (N5)
            ANT: (N2 N1)
            TOT: 2
            BNDG: ((V1,MOBY-DICK))
            DATA: ((((V1,MOBY-DICK)) (N1) 1 NIL 0))
            MSG: ((N2 (N8 ((V1,MOBY-DICK)))))

 SINCE
(N2 (OBJECT (WATER))
    (VERB (LIVES IN))
    (:SVAR (V1 (:VAR (T)) (:VAL (MOBY-DICK))))
    (AGENT (V1 (:VAR (T)) (:VAL (MOBY-DICK)))))
(N1 (CLASS (MAMMAL))
    (:SVAR (V1 (:VAR (T)) (:VAL (MOBY-DICK))))
    (MEMBER (V1 (:VAR (T)) (:VAL (MOBY-DICK)))))

 WE INFER
(N5
 (:SVAR (V1 (:VAR (T)) (:VAL (MOBY-DICK))))
 (ARG
  (N4 (CLASS (DOLPHIN))
      (:SVAR (V1 (:VAR (T)) (:VAL (MOBY-DICK))))
      (MEMBER (V1 (:VAR (T)) (:VAL (MOBY-DICK)))))
  (N3 (CLASS (WHALE))
      (:SVAR (V1 (:VAR (T)) (:VAL (MOBY-DICK))))
      (MEMBER (V1 (:VAR (T)) (:VAL (MOBY-DICK))))))
 (MAX (1))
 (MIN (1)))

            ** INITIATE - ANS-CATCH ID - P66 INITIATED BY - P68

            *****   "LEAVING"   PROCESS :  P68   *****
            NAME: CH&ENT-R
            CLINK: P66
            CQ: (N5)
            ANT: (N2 N1)
            TOT: 2
            BNDG: ((V1,MOBY-DICK))
            DATA: ((((V1,MOBY-DICK)) (N2 N1) 2 NIL 0))
            MSG: NIL

            *****   "ENTERING"   PROCESS :  P66   *****
```

```
NAME: ANS-CATCH
CLINK: P61
BOSSES: (P61)
DATA: NIL
MSG: (((M5 ((V1,MOBY-DICK)))))

** INITIATE - IMPLY ID - P61 INITIATED BY - P66

*****   "LEAVING"   PROCESS :  P66   *****
NAME: ANS-CATCH
CLINK: P61
BOSSES: (P61)
DATA: ((M5 ((V1,MOBY-DICK))))
MSG: NIL

*****   "ENTERING"   PROCESS :  P61   *****
NAME: IMPLY
CLINK: NIL
CQ: (M5)
RULE: M6
BNDG: ((V1,MOBY-DICK))
MSG: (((M5 ((V1,MOBY-DICK)))))

** NEW - ANS-CATCH ID - P89 **

** NEW - ANS-CATCH ID - P90 **

** NEW - USE ID - P91 **

** NEW - IMPLY ID - P92 **

** INITIATE - USE ID - P91 INITIATED BY - P61

*****   "LEAVING"   PROCESS :  P61   *****
NAME: IMPLY
CLINK: P89
CQ: (M5)
RULE: M6
BNDG: ((V1,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P87   *****
NAME: GO-UP11
CLINK: P86
CQ: M8
BNDG: ((V2,MOBY-DICK))

** INITIATE - SWITCH ID - P86 INITIATED BY - P87

*****   "LEAVING"   PROCESS :  P87   *****
NAME: GO-UP11
CLINK: P86
CQ: M8
BNDG: ((V2,MOBY-DICK))

*****   "ENTERING"   PROCESS :  P86   *****
NAME: SWITCH
```

```
CLINK: P81
CQ: M8
BNDG: ((V2,MOBY-DICK))
MSG: (((M8 ((V2,MOBY-DICK)))))

** INITIATE - TOPINF ID - P81 INITIATED BY - P86

*****   "LEAVING"   PROCESS :  P86   *****
NAME: SWITCH
CLINK: P81
CQ: M8
BNDG: ((V2,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P81   *****
NAME: TOPINF
CLINK: P79
CQ: M8
BOSSES: (P79)
DATA: NIL
MSG: ((M8 (M8 ((V2,MOBY-DICK)))))
MTR: P80

** INITIATE - CHTHRESH-R ID - P79 INITIATED BY - P81

*****   "LEAVING"   PROCESS :  P81   *****
NAME: TOPINF
CLINK: P79
CQ: M8
BOSSES: (P79)
DATA: ((M8 ((V2,MOBY-DICK))))
MSG: NIL
MTR: P80

*****   "ENTERING"   PROCESS :  P79   *****
NAME: CHTHRESH-R
CLINK: P77
THRESH: 1
TOT: 0
CQ: (M8 M7)
ANT: (M8 M7)
BNDG: ((V2,MOBY-DICK))
DATA: ((((V2,MOBY-DICK)) (M7) 1 NIL 0))
MSG: ((M8 (M8 ((V2,MOBY-DICK)))))

*****   "LEAVING"   PROCESS :  P79   *****
NAME: CHTHRESH-R
CLINK: P77
THRESH: 1
TOT: 0
CQ: (M8 M7)
ANT: (M8 M7)
BNDG: ((V2,MOBY-DICK))
DATA: ((((V2,MOBY-DICK)) (M8 M7) 2 NIL 0))
MSG: NIL

*****   "ENTERING"   PROCESS :  P88   *****
```

```
NAME: F-INFER
CLINK: NIL
NODE: N13

** INITIATE - INFER ID - P82 INITIATED BY - P88

** INITIATE - INFER ID - P71 INITIATED BY - P88

*****   "LEAVING"   PROCESS :  P88   *****
NAME: F-INFER
CLINK: NIL
NODE: N13

*****   "ENTERING"   PROCESS :  P91   *****
NAME: USE
CLINK: P90
CQ: (N4 N3)
RULE: N5
BNDG: ((V1,MOBY-DICK))
MSG: NIL

** NEW - CHANDOR ID - P93 **

** INITIATE - CHANDOR ID - P93 INITIATED BY - P91

*****   "LEAVING"   PROCESS :  P91   *****
NAME: USE
CLINK: P90
CQ: (N4 N3)
RULE: N5
BNDG: ((V1,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P82   *****
NAME: INFER
CLINK: P81
CQ: N8
BNDG: ((V2,MOBY-DICK))
MSG: ((N13 NIL ((V2,MOBY-DICK))))

** INITIATE - TOPINF ID - P81 INITIATED BY - P82

*****   "LEAVING"   PROCESS :  P82   *****
NAME: INFER
CLINK: P81
CQ: N8
BNDG: ((V2,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P81   *****
NAME: TOPINF
CLINK: P79
CQ: N8
BOSSES: (P79)
DATA: ((N8 ((V2,MOBY-DICK))))
MSG: ((N8 (N13 ((V2,MOBY-DICK)))))
NTR: P80
```

```
*****   "LEAVING"   PROCESS :  P81   *****
NAME: TOPINF
CLINK: P79
CQ: M8
BOSSES: (P79)
DATA: ((M8 ((V2,MOBY-DICK))))
MSG: NIL
MTR: P80

*****   "ENTERING"   PROCESS :  P71   *****
NAME: INFER
CLINK: P70
CQ: M2
BNDG: ((V1,MOBY-DICK))
MSG: ((M13 NIL ((V1,MOBY-DICK))))

** INITIATE - TOPINF ID - P76 INITIATED BY - P71

*****   "LEAVING"   PROCESS :  P71   *****
NAME: INFER
CLINK: P70
CQ: M2
BNDG: ((V1,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P76   *****
NAME: TOPINF
CLINK: P68
CQ: M2
BOSSES: (P68)
DATA: ((M8 ((V1,MOBY-DICK))))
MSG: ((M2 (M13 ((V1,MOBY-DICK)))))
MTR: P69

*****   "LEAVING"   PROCESS :  P76   *****
NAME: TOPINF
CLINK: P68
CQ: M2
BOSSES: (P68)
DATA: ((M8 ((V1,MOBY-DICK))))
MSG: NIL
MTR: P69

*****   "ENTERING"   PROCESS :  P93   *****
NAME: CHANDOR
CLINK: P90
MIN: 1
MAX: 1
TOT: 2
CQ: (M4 M3)
ANT: (M4 M3)
BNDG: ((V1,MOBY-DICK))
DATA: NIL
MSG: NIL

** NEW - TOPINF ID - P95 **
```

```
** NEW - INFER ID - P96 **

** INITIATE - INFER ID - P96 INITIATED BY - P93

** NEW - TOPINF ID - P98 **

** NEW - INFER ID - P99 **

** INITIATE - INFER ID - P99 INITIATED BY - P93

*****   "LEAVING"   PROCESS :  P93   *****
NAME: CHANDOR-R
CLINK: P90
MIN: 1
MAX: 1
TOT: 2
CQ: (M4 M3)
ANT: (M4 M3)
BNDG: ((V1,MOBY-DICK))
DATA: NIL
MSG: NIL

*****   "ENTERING"   PROCESS :  P96   *****
NAME: INFER
CLINK: P95
CQ: M4
BNDG: ((V1,MOBY-DICK))
MSG: NIL

** NEW - SWITCH ID - P100 **

** NEW - GO-UP11 ID - P101 **

** INITIATE - GO-UP11 ID - P101 INITIATED BY - P96

*****   "LEAVING"   PROCESS :  P96   *****
NAME: INFER
CLINK: P95
CQ: M4
BNDG: ((V1,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P99   *****
NAME: INFER
CLINK: P98
CQ: M3
BNDG: ((V1,MOBY-DICK))
MSG: NIL

** INITIATE - TOPINF ID - P98 INITIATED BY - P99

*****   "LEAVING"   PROCESS :  P99   *****
NAME: INFER
CLINK: P98
CQ: M3
BNDG: ((V1,MOBY-DICK))
```

```
            MSG: NIL

            *****   "ENTERING"   PROCESS :  P98   *****
            NAME: TOPINF
            CLINK: P93
            CQ: M3
            BOSSES: (P93)
            DATA: NIL
            MSG: ((M3 (M10 ((V1,MOBY-DICK)))))
            MTR: P97

            ** INITIATE - CHANDOR-R ID - P93 INITIATED BY - P98

            *****   "LEAVING"   PROCESS :  P98   *****
            NAME: TOPINF
            CLINK: P93
            CQ: M3
            BOSSES: (P93)
            DATA: ((M10 ((V1,MOBY-DICK))))
            MSG: NIL
            MTR: P97

            *****   "ENTERING"   PROCESS :  P93   *****
            NAME: CHANDOR-R
            CLINK: P90
            MIN: 1
            MAX: 1
            TOT: 2
            CQ: (M4 M3)
            ANT: (M4 M3)
            BNDG: ((V1,MOBY-DICK))
            DATA: NIL
            MSG: ((M3 (M10 ((V1,MOBY-DICK)))))

 SINCE
(M3 (CLASS (WHALE))
    (:SVAR (V1 (:VAR (T)) (:VAL (MOBY-DICK))))
    (MEMBER (V1 (:VAR (T)) (:VAL (MOBY-DICK)))))

 WE INFER
(T102
 (:SVAR (V1 (:VAR (T)) (:VAL (MOBY-DICK))))
 (ARG (M4 (CLASS (DOLPHIN))
          (:SVAR (V1 (:VAR (T)) (:VAL (MOBY-DICK))))
          (MEMBER (V1 (:VAR (T)) (:VAL (MOBY-DICK))))))
 (MAX (0))
 (MIN (0)))

            ** INITIATE - ANS-CATCH ID - P90 INITIATED BY - P93

            *****   "LEAVING"   PROCESS :  P93   *****
            NAME: CHANDOR-R
            CLINK: P90
            MIN: 1
            MAX: 1
            TOT: 2
            CQ: (M4 M3)
```

```
ANT: (M4 M3)
BNDG: ((V1,MOBY-DICK))
DATA: ((((V1,MOBY-DICK)) (M3) 1 NIL 0))
MSG: NIL

*****   "ENTERING"   PROCESS :  P90   *****
NAME: ANS-CATCH
CLINK: P92
BOSSES: (P92)
DATA: NIL
MSG: (((T102 ((V1,MOBY-DICK)))))

** INITIATE - IMPLY ID - P92 INITIATED BY - P90

*****   "LEAVING"   PROCESS :  P90   *****
NAME: ANS-CATCH
CLINK: P92
BOSSES: (P92)
DATA: ((T102 ((V1,MOBY-DICK))))
MSG: NIL

*****   "ENTERING"   PROCESS :  P92   *****
NAME: IMPLY
CLINK: NIL
CQ: (M4 M3)
RULE: M5
BNDG: ((V1,MOBY-DICK))
MSG: (((T102 ((V1,MOBY-DICK)))))

** NEW - ANS-CATCH ID - P103 **

** NEW - P-INFER ID - P104 **

** INITIATE - P-INFER ID - P104 INITIATED BY - P92

*****   "LEAVING"   PROCESS :  P92   *****
NAME: IMPLY
CLINK: NIL
CQ: (M4 M3)
RULE: M5
BNDG: ((V1,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P101   *****
NAME: GO-UP11
CLINK: P100
CQ: M4
BNDG: ((V1,MOBY-DICK))

** NEW - ANS-CATCH ID - P105 **

** NEW - IMPLY ID - P106 **

** NEW - GO-UP11 ID - P107 **

** INITIATE - GO-UP11 ID - P107 INITIATED BY - P101
```

```
*****   "LEAVING"   PROCESS :  P101   *****
NAME: GO-UP11
CLINK: P100
CQ: N4
BNDG: ((V1,MOBY-DICK))

*****   "ENTERING"   PROCESS :  P104   *****
NAME: F-INFER
CLINK: NIL
NODE: N15

** INITIATE - INFER ID - P96 INITIATED BY - P104

*****   "LEAVING"   PROCESS :  P104   *****
NAME: F-INFER
CLINK: NIL
NODE: N15

*****   "ENTERING"   PROCESS :  P107   *****
NAME: GO-UP11
CLINK: P106
CQ: N5
BNDG: ((V1,MOBY-DICK))

*****   "LEAVING"   PROCESS :  P107   *****
NAME: GO-UP11
CLINK: P106
CQ: N5
BNDG: ((V1,MOBY-DICK))

*****   "ENTERING"   PROCESS :  P96   *****
NAME: INFER
CLINK: P95
CQ: N4
BNDG: ((V1,MOBY-DICK))
MSG: ((N14 NIL ((V1,MOBY-DICK))))

** NEW - SWITCH ID - P108 **

** NEW - GO-UP11 ID - P109 **

** INITIATE - GO-UP11 ID - P109 INITIATED BY - P96

*****   "LEAVING"   PROCESS :  P96   *****
NAME: INFER
CLINK: P95
CQ: N4
BNDG: ((V1,MOBY-DICK))
MSG: NIL

*****   "ENTERING"   PROCESS :  P109   *****
NAME: GO-UP11
CLINK: P108
CQ: N14
BNDG: NIL

** NEW - AMS-CATCH ID - P110 **
```

```
** NEW - IMPLY ID - P111 **

** NEW - GO-UP11 ID - P112 **

** INITIATE - GO-UP11 ID - P112 INITIATED BY - P109

*****   "LEAVING"   PROCESS :  P109   *****
NAME: GO-UP11
CLINK: P108
CQ: M14
BNDG: NIL

*****   "ENTERING"   PROCESS :  P112   *****
NAME: GO-UP11
CLINK: P111
CQ: M15
BNDG: NIL

** NEW - ANS-CATCH ID - P113 **

** NEW - USE ID - P114 **

** INITIATE - USE ID - P114 INITIATED BY - P112

*****   "LEAVING"   PROCESS :  P112   *****
NAME: GO-UP11
CLINK: P111
CQ: M15
BNDG: NIL

*****   "ENTERING"   PROCESS :  P114   *****
NAME: USE
CLINK: P113
CQ: (M14)
RULE: M15
BNDG: NIL
MSG: NIL

** NEW - CHANDOR ID - P115 **

** INITIATE - CHANDOR ID - P115 INITIATED BY - P114

*****   "LEAVING"   PROCESS :  P114   *****
NAME: USE
CLINK: P113
CQ: (M14)
RULE: M15
BNDG: NIL
MSG: NIL

*****   "ENTERING"   PROCESS :  P115   *****
NAME: CHANDOR
CLINK: P113
MIN: 0
MAX: 0
TOT: 1
```

```
CQ: (M14)
ANT: (M14)
BNDG: NIL
DATA: NIL
MSG: NIL

** INITIATE - ANS-CATCH ID - P113 INITIATED BY - P115


*****   "LEAVING"   PROCESS :  P115   *****
NAME: CHANDOR
CLINK: P113
MIN: 0
MAX: 0
TOT: 1
CQ: (M14)
ANT: (M14)
BNDG: NIL
DATA: NIL
MSG: NIL

*****   "ENTERING"   PROCESS :  P113   *****
NAME: ANS-CATCH
CLINK: P111
BOSSES: (P111)
DATA: NIL
MSG: (((T116 NIL)))

** INITIATE - IMPLY ID - P111 INITIATED BY - P113

*****   "LEAVING"   PROCESS :  P113   *****
NAME: ANS-CATCH
CLINK: P111
BOSSES: (P111)
DATA: ((T116 NIL))
MSG: NIL

*****   "ENTERING"   PROCESS :  P111   *****
NAME: IMPLY
CLINK: P110
CQ: (M14)
RULE: M15
BNDG: NIL
MSG: (((T116 NIL)))

*****   "LEAVING"   PROCESS :  P111   *****
NAME: IMPLY
CLINK: P110
CQ: (M14)
RULE: M15
BNDG: NIL
MSG: NIL


(M12 (CLASS (MAMMAL)) (MEMBER (MOBY-DICK)))
(M13 (OBJECT (WATER)) (VERB (LIVES IN)) (AGENT (MOBY-DICK)))
(M15 (ARG (M14 (CLASS (DOLPHIN)) (MEMBER (MOBY-DICK))))
```

```
      (MAX (0))
      (MIN (0)))
(DUMPED)
9705 MSECS

* (LISP)
END SHEPS
```

In Figures 18 and 19 are presented the processes left behind by the system after forward inference is completed. In both Figures the word 'MOBY-DICK' has been abbreviated to 'M-D'. Hopefully this would not cause confusion to the reader.

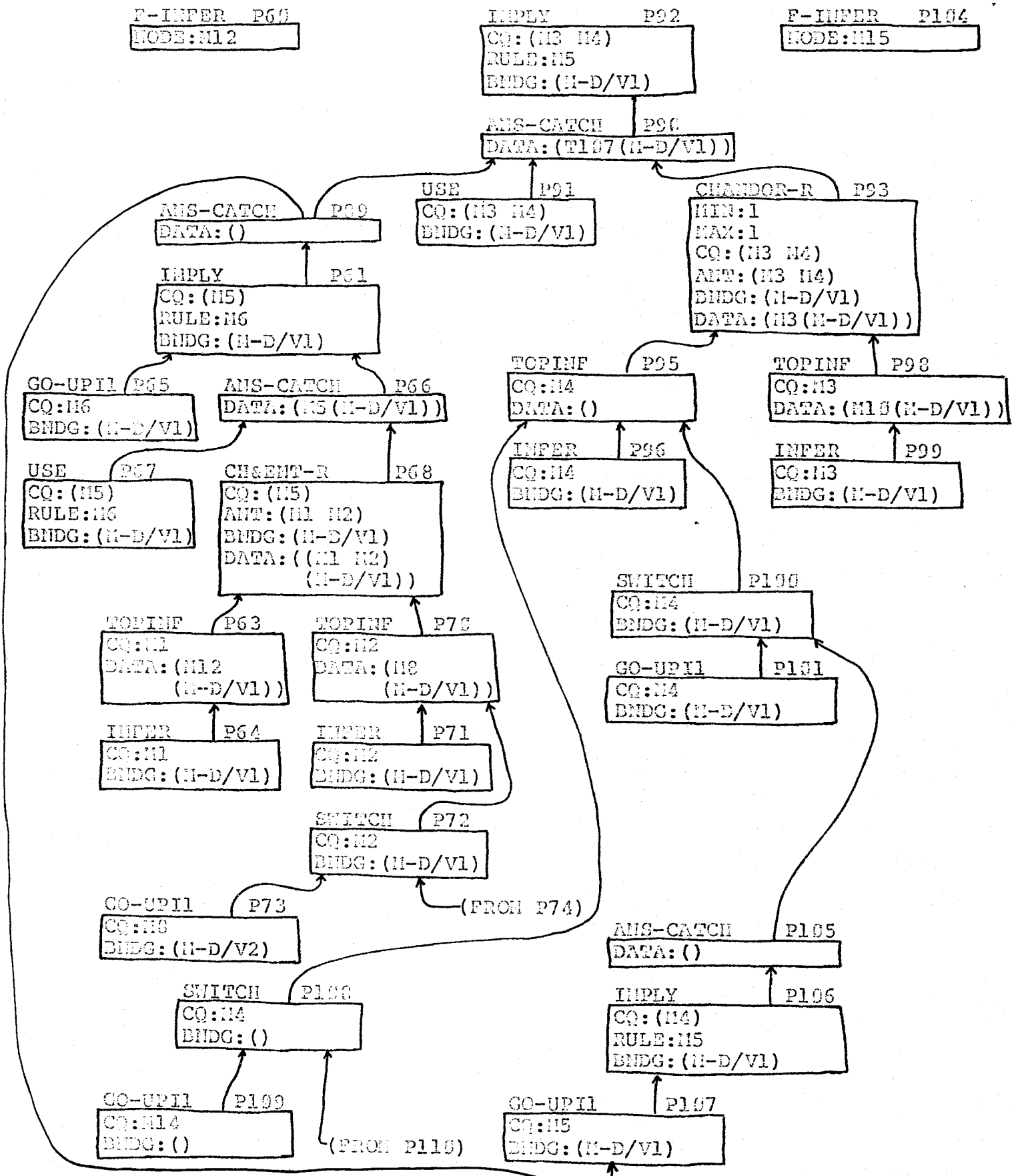

Figure 16
Set of processes

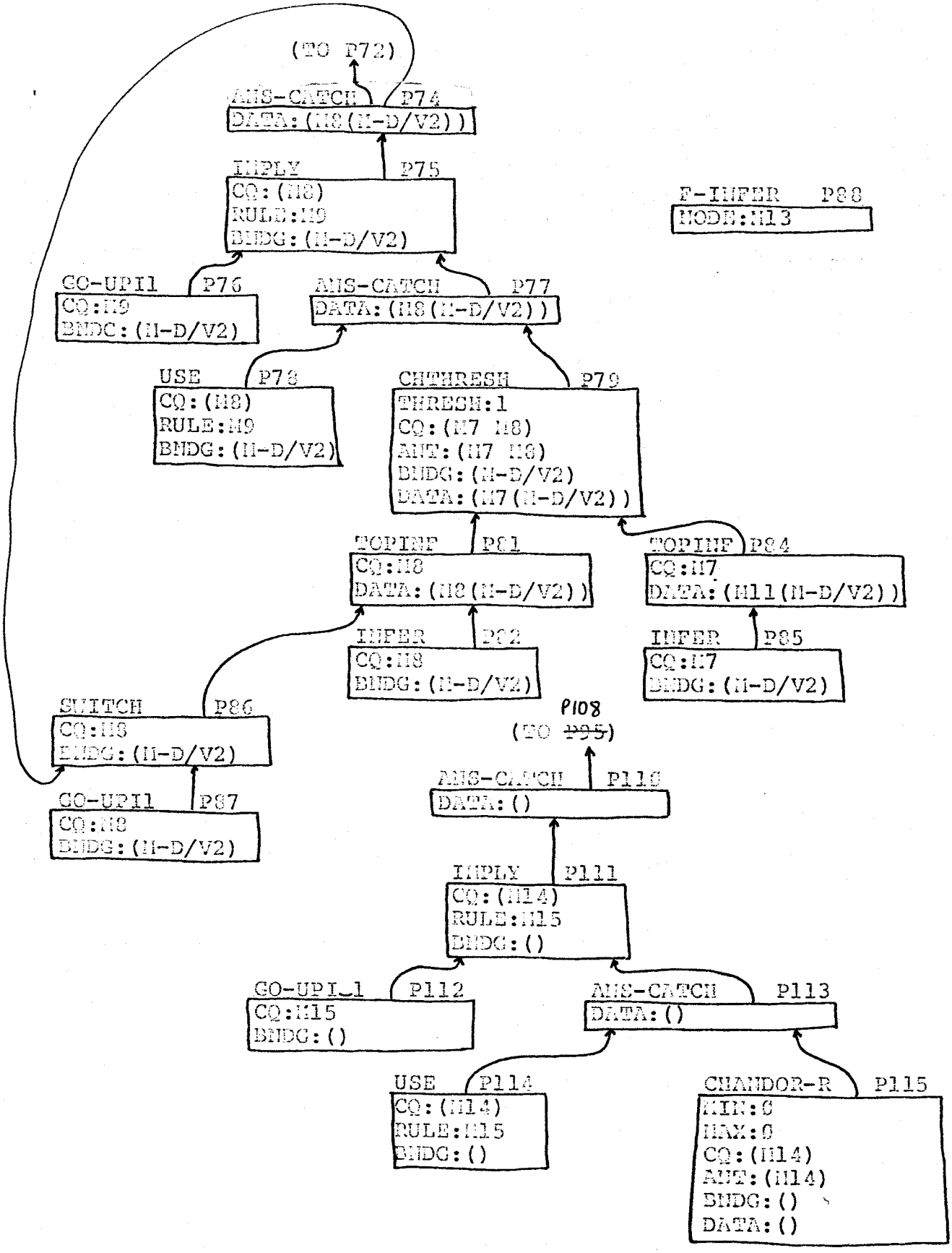

Figure 19
Set of processes (contd.)