Department of Computer Science

STATE UNIVERSITY OF NEW YORK AT BUFFALO

COCCI: A DEDUCTIVE SEMANTIC NETWORK PROGRAM FOR SOLVING MICROBIOLOGY UNKNOWNS

by

Stuart C. Shapiro

March 1981

Technical Report Number 173

COCCI: A Deductive Semantic Network Program for Solving Microbiology Unknowns

Stuart C. Shapiro
Department of Computer Science
State University of New York at Buffalo
Amherst, New York 14226

February, 1981

Abstract

"You have been given a culture of one of nine cocci. Identify it." COCCI is a program to solve this problem. To identify the unknown, COCCI requests a human assistant to perform tests and make observations and report the results to it. COCCI consists of 14 specific facts and 17 deduction rules stored in SNePS, a general purpose deductive semantic network processing system, and a small ATN grammar for parsing and for generating English from the semantic network. All of COCCI's reasoning and interaction with humans is driven by the general purpose bi-directional inference sub-system of SNePS. This paper describes COCCI as an illustration of deductive semantic networks in general and SNEPS in particular. Specific points covered include structure sharing, procedural attachment, generation grammars and the structure of deduction rules.

Acknowledgements

A number of people have helped with the development of SNePS over the years. I am particularly grateful to Don McKay, Joao Martins and Gerard Donlon for their current work. Caren Shapiro provided COCCI's domain, and a domain expert's critique.

This work was supported in part by the National Science Foundation under grants MCS78-Ø2274 and MCS8Ø-Ø6314.

Introduction

SNePS [Shapiro 79a] is a general purpose deductive semantic network processing system. It is called a semantic network partly for historical reasons, but also because it was designed to reflect certain philosophical positions on the representation of knowledge derived from understanding natural language. Some of these positions will not concern us in this paper (but see [Maida and Shapiro 81]). Two that will are the following.

It is my position that semantic networks use nodes exclusively to represent "concepts" -- every individual, class, proposition, etc. the system "knows" and about which it has information and can reason. Although a labelled arc represents a relationship between the two nodes it connects, it does not represent a proposition the system can discuss with a user or about which it can reason. I have called such relations, represented by arcs, "system relations" [Shapiro 71] to distinguish them from the "conceptual relations" represented by nodes. This stance is relevant to this paper because among the propositions that can be stored in a semantic network are general propositions such as, "Every organism is either gram positive or gram negative" as opposed to specific facts such as "Yellow is a color". General propositions can not only be known, they can be used for reasoning. Since the reasoning I have been concerned with is deductive rather than inductive, I call the network representations of general propositions deduction rules. Since general propositions are composed from other propositions, and all are represented by nodes, SNePS deduction rules can be composed of other deduction rules to any level or degree of complexity. In this way, SNePS differs from other reasoning systems that limit their rules to Horn clauses, production rules, or the like.

Another philosophical stance is that the identifiers of nodes are uninterpretable symbols. The meaning of a concept is embedded in the entire structure of the network connected to the node which represents it [Quillian 68]. In this way, SNePS differs from the "semantic networks" of [Deliyanni and Kowalski 79] in which the node labelled "w(x)" is a Skolem function of the node labelled "x" by dint of the syntactic structure of its label.

SNePS is a general semantic network in the sense that it was designed to be a system that could be used for experimenting with various knowledge representations. The only system relations built in are those used to represent deduction rules, and they were built in only so that an inference system could be written to use the rules. (I still consider the inference system to be a layer outside the core, basic, SNePS.) Generality extends to the rules, so the representation of deduction rules was designed to be as powerful as predicate calculus. In this way SNePS differs from ACT [Anderson 76], NETL [Fahlman 79], and most representation systems based on "IS-A" hierarchies.

Since SNePS was designed with natural language understanding in mind, I was disturbed by the structural complexity needed to represent general propositions such as, "The organism is exactly one of the following nine: ... " using the standard binary logical connectives, so SNePS uses set oriented, non-standard connectives, some of which will be shown in this paper. In this way, SNePS differs from other logically adequate semantic networks such as those of Hendrix [Hendrix 79] and Schubert [Schubert et al. 79].

This paper presents an example of using SNePS in a particular domain, that of solving microbiology laboratory unknowns. I have named this instance of SNePS _cum_ facts and rules, COCCI. Although COCCI's domain and performance is somewhat reminiscent of MYCIN [Shortliffe 76], it is far less ambitious. COCCI is not meant to be used by physicians (or even microbiologists), does not have certainty factors, and does not have a well developed explanation facility. (However, it does exceed MYCIN in its rule syntax and the flexibility of its inference control structure.) Indeed, the domain was chosen only because of a locally available expert. COCCI is intended to be a demonstration of the features and facilities of SNePS as a general deductive semantic network, designed on the basis of the principles outlined above.

The Domain

Figure 1 shows COCCI's domain. This figure was adapted from a handout given to her laboratory class by Dr. Caren D. Shapiro of the Biology Department of D'Youville College, Buffalo, New

```
                    IDENTIFICATION OF COCCI
GR
RE
AA
MC  ARRANGEMENT      OXYGEN      PIGMENTATION, COLOR   ORGANISM
 T                 REQUIREMENT
 I
 O
 N
   |->IN TETRADS------------------>NON-PIGMENTED----->GAFFKYA
GP |
RO |->IN PACKETS------------------>PIGMENTED--------->SARCINA
AS |
MI-|->IN CHAINS------------------------------------>STREPTOCOCCUS
 T |                                                  PYOGENES
 I |
 V |->IN PAIRS------------------------------------->STREPTOCOCCUS
 E |                                                  PNEUMONIAE
   |
   |                          |->NON-PIGMENTED-----|
   |            |-->AEROBIC---|                    |->MICROCOCCUS
   |            |             |->PIGMENTED, YELLOW-|
   |            |
   |->IN MASSES-|                |->PIGMENTED, GOLD->STAPHYLOCOCCUS
                |                |                       AUREUS
                |                |
                |->FACULTATIVE-|->NON-PIGMENTED--->STAPHYLOCOCCUS
                               |                    EPIDERMIDIS
                               |
GN                             |->PIGMENTED, YELLOW-->MICROCOCCUS
RE                                                       CITREUS
AG
MA--->IN PAIRS--------------------------------------->NEISSERIA
 T
 I
 V
 E
```

Figure 1
Chart showing the possible organisms in the unknown problem

York. The students are given an unknown culture in a petri dish, and are required to figure out which of the nine microorganisms listed in the figure is in the culture. The students make observations and perform tests on the cultures, guided by the information in the figure. To determine the gram reaction of the organism and its arrangement (how cells clump together), the student must first apply a gram stain. Oxygen requirement indicates the atmosphere required for the organism to grow.

Aerobic organisms require the presence of oxygen. Anaerobic organisms (not included in this assignment) require the absence of oxygen. Facultative organisms can grow in either environment. Pigmentation of the cultures can be determined by looking at them in the petri dish. Notice that some data is missing from the chart. These observations are irrelevant for determining the unknown. Also note two species of aerobic micrococci which the students needn't differentiate more precisely.

COCCI simulates a laboratory student's decision making processes, but requires a human assistant to perform the tests and make the observations. In designing COCCI, it was important that the program not ask for the same observation more than once (e.g. "Is it in masses?" and later "Is it in pairs?"), not ask for an observation permaturely (e.g. asking for the gram reaction before asking for a gram stain to be done), and not ask for information it should already know is irrelevant (asking for the color after being told the organism is non-pigmented). We will see that these dependencies did not require special purpose control structures to be programmed, but were embedded in the way the deduction rules were expressed. This relies on SNePS' lack of restrictions on the structure of deduction rules. Essentially, a SNePS deduction rule can be a rule about rules.

Another design criterion of COCCI was that a human not expert in SNePS could follow the reasoning. The SNePS system does not yet have a fully developed explanation facility like MYCIN's [Shortliffe 76], but it has long had a trace facility for use in debugging deduction rules and the inference system itself. This trace facility prints the relevant antecedents and consequents of a rule whenever an inference is made. By making this trace conditional, so that not all inferences need be traced, and by sending the output through a generating Augmented Transition Network [Shapiro, 79b], the reasoning can easily be followed. Figure 2 shows a session with COCCI. In subsequent sections, we will explain how this session came to be.

Representation Schemata

The semantic network representation of the information in the domain was designed specifically for this domain. The ad hoc design criteria were: 1) using a minimal number of arcs per assertion; 2) using different case frames for different underlying predicates to simplify the deduction rules; 3) using a small set of different arc labels to simplify the generation grammar, yet big enough so that different sentence frames are distinguished. Alternatively, we could have used a representation designed for general English, but decided not to pursue that for this demonstration project.

Figure 3 shows the six different case frames used for specific information, and the sentences generated for each one by the generator. Node M1 asserts that an organism has a certain property. M2 asserts that an organism is in a certain class of organisms. Note that the organism is a mass, not an individual. M3 asserts that an individual property is a member of a class of properties. M4 asserts that a culture is prepared for making an observation. This proposition is crucial for COCCI's ordering of its requests. M5 and M6 are <u>function nodes</u>, our version of

```
** (: UNKNOWN1 IS ASSIGNED)

 PLEASE DO A GRAM STAIN ON UNKNOWN1
 PRESS CARRIAGE RETURN WHEN YOU'RE READY TO GO ON.
**
 SINCE
 A GRAM STAIN IS DONE ON UNKNOWN1
 WE INFER
 UNKNOWN1 IS PREPARED FOR DETERMINING ARRANGEMENT

 WHAT IS THE ARRANGEMENT OF UNKNOWN1 ?
 PLEASE TYPE THE APPROPRIATE NUMBER BETWEEN 1 AND 6 --
      1 )    IN MASSES
      2 )    IN PAIRS
      3 )    IN CHAINS
      4 )    IN PACKETS
      5 )    IN TETRADS
      6 )    NOT APPLICABLE OR UNKNOWN
 ** 1
 SINCE
 IN PAIRS IS AN ARRANGEMENT
 WE INFER
 THE ARRANGEMENT OF UNKNOWN1 IS SAID NOT TO BE IN PAIRS
 SINCE
 IN CHAINS IS AN ARRANGEMENT
 WE INFER
 THE ARRANGEMENT OF UNKNOWN1 IS SAID NOT TO BE IN CHAINS
 SINCE
 IN PACKETS IS AN ARRANGEMENT
 WE INFER
 THE ARRANGEMENT OF UNKNOWN1 IS SAID NOT TO BE IN PACKETS
 SINCE
 IN TETRADS IS AN ARRANGEMENT
 WE INFER
 THE ARRANGEMENT OF UNKNOWN1 IS SAID NOT TO BE IN TETRADS
 SINCE
 THE ARRANGEMENT OF UNKNOWN1 IS SAID TO BE IN MASSES
 WE INFER
 UNKNOWN1 IS IN MASSES
 SINCE
 A GRAM STAIN IS DONE ON UNKNOWN1
 WE INFER
 UNKNOWN1 IS PREPARED FOR DETERMINING GRAM REACTION


 WHAT IS THE GRAM REACTION OF UNKNOWN1 ?
 PLEASE TYPE THE APPROPRIATE NUMBER BETWEEN 1 AND 3 --
      1 )    GRAM NEGATIVE
      2 )    GRAM POSITIVE
      3 )    NOT APPLICABLE OR UNKNOWN
 ** 2
 SINCE
 GRAM NEGATIVE IS A GRAM REACTION
 WE INFER
 THE GRAM REACTION OF UNKNOWN1 IS SAID NOT TO BE GRAM NEGATIVE
 SINCE
 THE GRAM REACTION OF UNKNOWN1 IS SAID TO BE GRAM POSITIVE
```

Figure 2
A sample session part 1

```
WE INFER
UNKNOWN1 IS GRAM POSITIVE
SINCE
UNKNOWN1 IS ASSIGNED
WE INFER
UNKNOWN1 IS PREPARED FOR DETERMINING PIGMENTATION

WHAT IS THE PIGMENTATION OF UNKNOWN1 ?
PLEASE TYPE THE APPROPRIATE NUMBER BETWEEN 1 AND 3 --
     1 )   NON-PIGMENTED
     2 )   PIGMENTED
     3 )   NOT APPLICABLE OR UNKNOWN
** 1
SINCE
PIGMENTED IS A PIGMENTATION
WE INFER
THE PIGMENTATION OF UNKNOWN1 IS SAID NOT TO BE PIGMENTED
SINCE
THE PIGMENTATION OF UNKNOWN1 IS SAID TO BE NON-PIGMENTED
WE INFER
UNKNOWN1 IS NON-PIGMENTED
SINCE
UNKNOWN1 IS ASSIGNED
WE INFER
UNKNOWN1 IS PREPARED FOR DETERMINING OXYGEN REQUIREMENT


WHAT IS THE OXYGEN REQUIREMENT OF UNKNOWN1 ?
PLEASE TYPE THE APPROPRIATE NUMBER BETWEEN 1 AND 4 --
     1 )   FACULTATIVE
     2 )   ANAEROBIC
     3 )   AEROBIC
     4 )   NOT APPLICABLE OR UNKNOWN
** 1
SINCE
ANAEROBIC IS AN OXYGEN REQUIREMENT
WE INFER
THE OXYGEN REQUIREMENT OF UNKNOWN1 IS SAID NOT TO BE ANAEROBIC
SINCE
AEROBIC IS AN OXYGEN REQUIREMENT
WE INFER
THE OXYGEN REQUIREMENT OF UNKNOWN1 IS SAID NOT TO BE AEROBIC
SINCE
THE OXYGEN REQUIREMENT OF UNKNOWN1 IS SAID TO BE FACULTATIVE
WE INFER
UNKNOWN1 IS FACULTATIVE
SINCE
UNKNOWN1 IS NON-PIGMENTED AND
UNKNOWN1 IS FACULTATIVE AND
UNKNOWN1 IS IN MASSES AND
UNKNOWN1 IS GRAM POSITIVE
WE INFER
UNKNOWN1 IS STAPHYLOCOCCUS EPIDERMIDIS
SINCE
UNKNOWN1 IS STAPHYLOCOCCUS EPIDERMIDIS
WE INFER
UNKNOWN1 IS NOT GAFFKYA AND
```

Figure 2
A sample session part 2

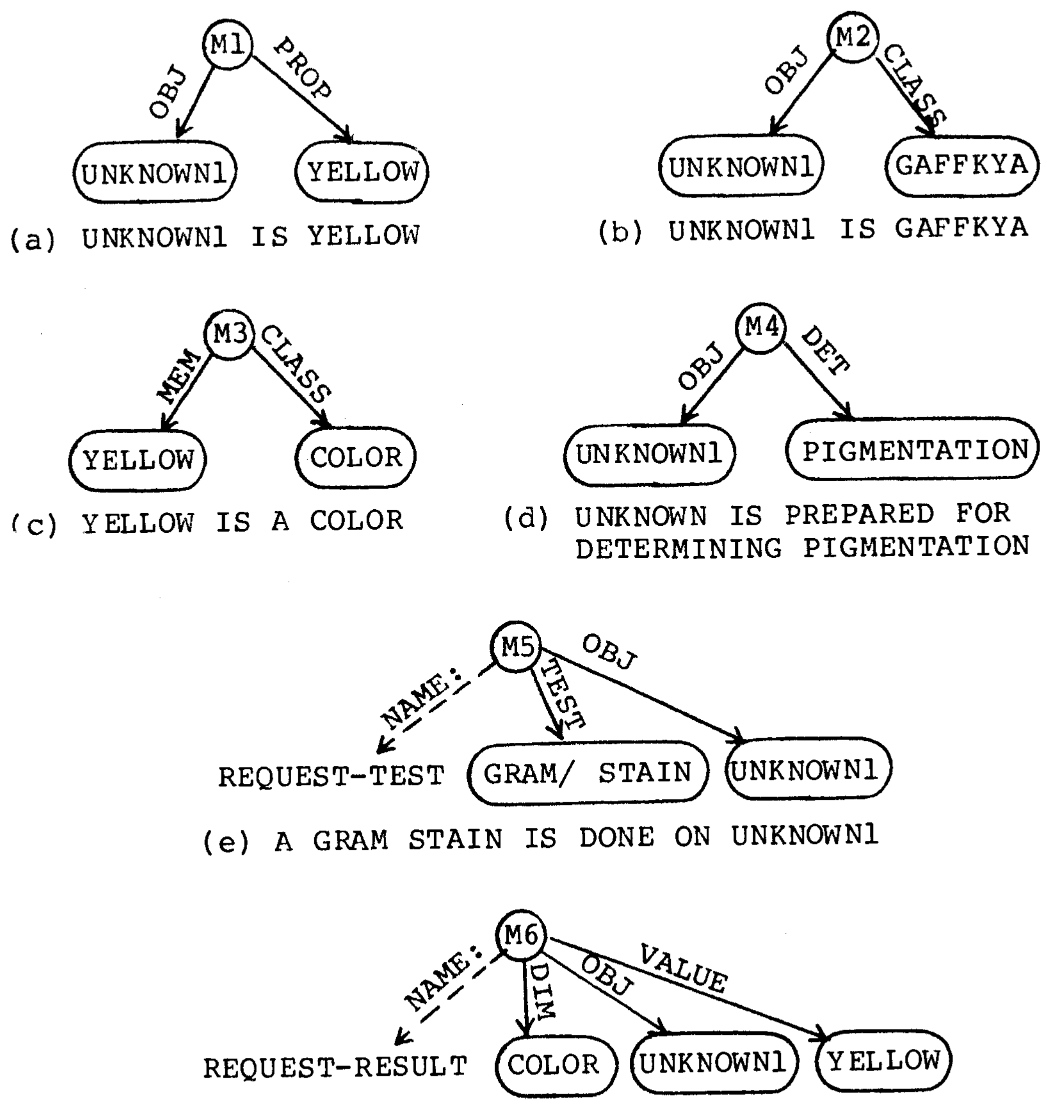
```
UNKNOWN1 IS NOT SARCINA AND
UNKNOWN1 IS NOT STREPTOCOCCUS PYOGENES AND
UNKNOWN1 IS NOT STREPTOCOCCUS PNEUMONIAE AND
UNKNOWN1 IS NOT MICROCOCCUS AND
UNKNOWN1 IS NOT STAPHYLOCOCCUS AUREUS AND
UNKNOWN1 IS NOT MICROCOCCUS CITREUS AND
UNKNOWN1 IS NOT NEISSERIA

UNKNOWN1 IS ASSIGNED AND
UNKNOWN1 IS PREPARED FOR DETERMINING ARRANGEMENT AND
UNKNOWN1 IS IN MASSES AND
THE ARRANGEMENT OF UNKNOWN1 IS SAID NOT TO BE IN PAIRS AND
THE ARRANGEMENT OF UNKNOWN1 IS SAID NOT TO BE IN CHAINS AND
THE ARRANGEMENT OF UNKNOWN1 IS SAID NOT TO BE IN PACKETS AND
THE ARRANGEMENT OF UNKNOWN1 IS SAID NOT TO BE IN TETRADS AND
UNKNOWN1 IS PREPARED FOR DETERMINING GRAM REACTION AND
UNKNOWN1 IS GRAM POSITIVE AND
THE GRAM REACTION OF UNKNOWN1 IS SAID NOT TO BE GRAM NEGATIVE AND
UNKNOWN1 IS PREPARED FOR DETERMINING PIGMENTATION AND
UNKNOWN1 IS NON-PIGMENTED AND
THE PIGMENTATION OF UNKNOWN1 IS SAID NOT TO BE PIGMENTED AND
UNKNOWN1 IS PREPARED FOR DETERMINING OXYGEN REQUIREMENT AND
THE OXYGEN REQUIREMENT OF UNKNOWN1 IS SAID NOT TO BE AEROBIC AND
UNKNOWN1 IS FACULTATIVE AND
UNKNOWN1 IS STAPHYLOCOCCUS EPIDERMIDIS AND
UNKNOWN1 IS NOT GAFFKYA AND
UNKNOWN1 IS NOT SARCINA AND
UNKNOWN1 IS NOT STREPTOCOCCUS PYOGENES AND
UNKNOWN1 IS NOT STREPTOCOCCUS PNEUMONIAE AND
UNKNOWN1 IS NOT MICROCOCCUS AND
UNKNOWN1 IS NOT STAPHYLOCOCCUS AUREUS AND
UNKNOWN1 IS NOT MICROCOCCUS CITREUS AND
UNKNOWN1 IS NOT NEISSERIA
```

Figure 2
A sample session Conclusion

procedural attachment. When the inference system needs to determine the validity of a function node, it does so by executing the function associated with the atom at the end of the NAME: auxiliary arc. It is the presence of such an arc that declares the node to be a function node. M5 evaluates true when the specified test has been done on the organism. M6 evaluates true when the human assistant has reported that the indicated culture has the indicated property.

Building Patterns



(a) UNKNOWN1 IS YELLOW

(b) UNKNOWN1 IS GAFFKYA

(c) YELLOW IS A COLOR

(d) UNKNOWN IS PREPARED FOR DETERMINING PIGMENTATION

(e) A GRAM STAIN IS DONE ON UNKNOWN1

(f) THE COLOR OF UNKNOWN1 IS SAID TO BE YELLOW

Figure 3
The case frames used by COCCI

To enter the data into the SNePS semantic network, we first build some pattern nodes that will each be used in several rules. A pattern node for the open statement that some culture is gram positive is built by the SNePS User Language (SNePSUL) command:

```
** ((BUILD OBJ $X PROP GRAM/ POSITIVE) = GRMPOSPAT)
M7
```

The "**" is the system prompt. "/" indicates that the next character (in this case a blank) is an alphabetic character and can be included in an atom's print name. $X causes a new variable node to be created and made the value of the SNePSUL variable X. M7 is the node build by the BUILD command. This node has been retained as the value of the SNePSUL variable GRMPOSPAT. Figure 4 is a pictorial representation of the network built by this command. V1 is the variable node created to be the value of X. The system knows it to be a variable node, not because of the "V" in its identifier, but because of an auxiliary arc (not shown) hanging from it.

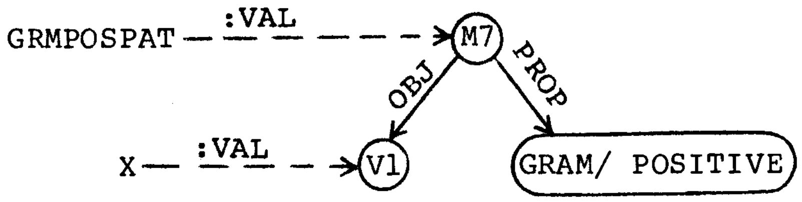


Figure 4
The network for the GRMPOSPAT pattern

For a more user-friendly output, we can pass the value of the BUILD command to the sentence generator with the SNePSUL function SURFACE. This is shown in the next two pattern building commands.

```
** (SURFACE (BUILD OBJ *X PROP IN/ CHAINS) = CHAINPAT)
V1 IS IN CHAINS
** (SURFACE (BUILD OBJ *X PROP IN/ PAIRS) = PAIRPAT)
V1 IS IN PAIRS
```

The "*" macro character causes the current value of the indicated SNePSUL variable to be used. One pattern like these is built for each of the twelve properties used in this domain.

Describing the Organisms

To describe the organisms to COCCI, we build SNePS deduction rules like these two:

```
**(SURFACE (BUILD AVB *X THRESH 1
              ARG (BUILD MIN 2 MAX 2
                         ARG *GRMPOSPAT ARG *CHAINPAT)
              ARG (BUILD OBJ *X CLASS STREPTOCOCCUS/ PYOGENES)
                  = STREPPY))
FOR EVERY V1, V1 IS STREPTOCOCCUS PYOGENES IF AND ONLY IF
   V1 IS IN CHAINS AND
   V1 IS GRAM POSITIVE

**(SURFACE (BUILD AVB *X THRESH 1
              ARG (BUILD MIN 2 MAX 2
                         ARG *GRMPOSPAT ARG *PAIRPAT)
              ARG (BUILD OBJ *X CLASS STREPTOCOCCUS/ PNEUMONIAE)
                  = STREPN))
FOR EVERY V1, V1 IS STREPTOCOCCUS PNEUMONIAE IF AND ONLY IF
   V] IS IN PAIRS AND
   V1 IS GRAM POSITIVE
```

AVB is an arc from a rule node to a variable universally bound in the rule. THRESH-1-ARG-...-ARG is a case frame asserting the mutual equivalence of all the ARGuments. MIN-n-MAX-n-ARG-...-ARG is a case frame asserting the conjunction of the n ARGuments. The representation of deduction rules is discussed more fully in [Shapiro 77] and [Shapiro 79a]. Notice that the pattern node GRMPOSPAT is used in both these rules. This is an example of structure sharing in semantic networks. Structure sharing not only saves space in the network, it also saves time during deduction since derived instances of a pattern node are immediately available to all rules which share the node. GRMPOSPAT is shared among eight rules in this domain.

One rule like these two is built for each of the organisms. SNePSUL variables were set to the organism patterns so that COCCI could be told that every culture is exactly one of the given organisms:

```
**(SURFACE (BUILD AVB *X MIN 1 MAX 1
            ARG *GAFFKYA ARG *SARCINA ARG *STREPPY ARG *STREPN
            ARG *MICROCOCCUS ARG *STAPH-AUREUS ARG *STAPH-EPI
            ARG *MICRO-CITREUS ARG *NEISSIRIA))
EITHER V1 IS NEISSERIA OR
       V1 IS MICROCOCCUS CITREUS OR
       V1 IS STAPHYLOCOCCUS EPIDERMIDIS OR
       V1 IS STAPHYLOCOCCUS AUREUS OR
       V1 IS MICROCOCCUS OR
       V1 IS STREPTOCOCCUS PNEUMONIAE OR
       V1 IS STREPTOCOCCUS PYOGENES OR
       V1 IS SARCINA OR
       V1 IS GAFFKYA
```

The MIN-1-MAX-1-ARG-...-ARG case frame asserts that exactly one of the ARGuments is _true_.

Categorizing Properties

We must tell COCCI what properties are together in what categories:

```
**(SURFACE (BUILD MEM (GRAM/ POSITIVE GRAM/ NEGATIVE)
                  CLASS GRAM/ REACTION))
GRAM NEGATIVE AND GRAM POSITIVE ARE GRAM REACTIONS
```

The node built here is not a rule, but a specific fact. However, just as in the rule nodes, this node has more than one descending arc with the same label (MEM). This node is equivalent to building two nodes, one for gram positive's being a gram reaction, the other for gram negative's being a gram reaction. What is significant is that a node exists with a MEM arc to the property and a CLASS arc to the property category. One node like this is built for each of the five property categories. These are the only specific facts needed for this problem.

An important fact about the property categories is when the culture is prepared for observing which property it has in the category. Pigmentation can be observed as soon as the unknown is assigned:

```
**(SURFACE (BUILD AVB *X
                  ANT (BUILD OBJ *X PROP ASSIGNED) = ASSIGNPAT
                  CQ  (BUILD OBJ *X DET PIGMENTATION)))
FOR EVERY V1, IF V1 IS ASSIGNED
   THEN V1 IS PREPARED FOR DETERMINING PIGMENTATION
```

COCCI is also allowed to ask for oxygen requirement immediately, but it doesn't make sense to ask for the color of an unknown unless it is known to be pigmented:

```
**(SURFACE (BUILD AVB *X ANT *ASSIGNPAT
                  CQ (BUILD ANT *PIGPAT
                            CQ (BUILD OBJ *X DET COLOR))))
FOR EVERY V1, IF V1 IS ASSIGNED
   THEN IF V1 IS PIGMENTED
           THEN V1 IS PREPARED FOR DETERMINING COLOR
```

COCCI will be started by asserting that an unknown is assigned and having this assertion trigger forward inference. The first rule shown above will allow the deduction that COCCI can ask about the unknown's pigmentation. We will shortly see how the request itself is triggered. The second rule, however, only allows the _deduction of another rule_, namely one that says that if the unknown is pigmented, COCCI can ask for its color. The forward deduction of this rule causes a backward deduction of its antecedent to be started. This is an example of SNePS' _bi-directional inference_ [Martins, et al. 81]. It may be that COCCI does not yet know whether or not the unknown is pigmented. In that case the backward deduction will be _suspended_, to be resumed if and when the unknown is determined to be pigmented. Notice the similarity of this to the creation of "demons". In Figure 2, since the unknown was said to be non-pigmented, this demon was never awakened, and COCCI never asked for the unknown's color.

Before an unknown is prepared for determining gram reaction, COCCI must ask its assistant to do a gram stain. This rule uses a function node:

```
**(SURFACE (BUILD AVB *X ANT *ASSIGNPAT
            CQ (BUILD ANT (BUILD NAME: REQUEST-TEST OBJ *X
                                 TEST GRAM/ STAIN) = GRAM-STAIN
                      CQ (BUILD OBJ *X DET GRAM/ REACTION)))
FOR EVERY V1, IF V1 IS ASSIGNED
   THEN IF A GRAM STAIN IS DONE ON V1
           THEN V1 IS PREPARED FOR DETERMINING GRAM REACTION
```

After an unknown is assigned, backward deduction is begun on the antecedent of the embedded rule. However, the SNePS inference system recognizes that this antecedent is a function node and executes it. We will discuss function node functions more fully when we discuss REQUEST-RESULT below. REQUEST-TEST prints the message shown in Figure 2, and after the assistant types a carriage return, REQUEST-TEST terminates in a way that signals the inference system that the function node evaluated to _true_. At that point the unknown is deduced to be prepared for determining gram reaction.

Arrangement also requires a gram stain. The rule is similar to the one for gram reaction and will not be shown here.

Triggering Requests

Some of the information we have discussed so far is specific to the particular organisms included in this example. Some of the information is specific to the property categories relevant to this example. The two rules discussed in this section are specific only to the general class of laboratory unknown problems.

The first rule says that if an unknown is prepared for making an observation, then COCCI can ask the assistant to make the observation, and should believe what the assistant reports:

```
**(SURFACE (BUILD AVB ($O $C)
             ANT (BUILD OBJ *O DET *C)
             CQ  (BUILD AVB $P
                   ANT (BUILD MEM *P CLASS *C)
                   CQ  (BUILD THRESH 1
                          ARG ((BUILD OBJ *O VALUE *P DIM *C)
                               (BUILD OBJ *O PROP *P))))))
FOR EVERY V2 AND V3, IF V2 IS PREPARED FOR DETERMINING V3
   THEN FOR EVERY V4, IF V4 IS A V3
           THEN V2 IS V4 IF AND ONLY IF
                THE V3 OF V2 IS SAID TO BE V4
```

This rule is easier to understand fully instantiated:

```
IF UNKNOWN1 IS PREPARED FOR DETERMINING COLOR
   THEN IF GOLD IS A COLOR
           THEN UNKNOWN1 IS GOLD IF AND ONLY IF
                THE COLOR OF UNKNOWN1 IS SAID TO BE GOLD
```

When an unknown is deduced to be ready for making an observation on a category by the rules we saw above, this rule collects the properties of the category, and deduces that the unknown has the property if the assistant says so, but doesn't if the assistant denies it. (The rule used in the run of Figure 2 did not use the THRESH, but just had "IF THE V3 OF V2 IS SAID TO BE V4 THEN V2 IS V4".) The node built by (BUILD OBJ *O VALUE *P DIM *C) is not a function node, but <u>matches</u> the REQUEST-RESULT function node in the rule we will see next. In backward inference, a node matches another even if that node has extra arcs.

The assistant only reports the properties that the unknown <u>has</u>. COCCI deduces that the assistant <u>means</u> that the unknown doesn't have the other properties in each category because of this rule:

```
**(SURFACE (BUILD AVB ($C $O)
            CQ (BUILD PEVB $P EMAX 1
                   &ANT (BUILD CLASS *C MEM *P)
                   CQ   (BUILD NAME: REQUEST-RESULT OBJ *O
                               DIM *C VALUE *P))))
FOR EVERY V5 AND V6, THERE IS AT MOST ONE V7 SUCH THAT
   V7 IS A V5 AND
   THE V5 OF V6 IS SAID TO BE V7
```

Again, this rule is more easily understood partially instantiated:

```
THERE IS AT MOST ONE V7 SUCH THAT
   V7 IS A COLOR AND
   THE COLOR OF UNKNOWN1 IS SAID TO BE V7
```

This rule uses a maximal numerical quantifier [Shapiro 79c]. When triggered by the previous rule, V5 and V6 will be instantiated, so let's consider the partially instantiated version. The inference system creates backward inference queries for all instances of V7 IS A COLOR and for all instances of THE COLOR OF UNKNOWN1 IS SAID TO BE V7. The former returns all colors. The latter, however, is recognized to be a function node. As we shall see, the REQUEST-RESULT function queries the assistant, and is told what color UNKNOWN1 is, say YELLOW. The function node then evaluates to _true_ in its substitution instance of V7 bound to YELLOW. This information is sent up to the previous rule which deduces that UNKNOWN1 IS YELLOW, and also triggers the maximal parameter of the numerical quantifier, causing the earlier rule to be told that the assistant reported UNKNOWN1 not to be the other colors. That rule will then deduce that UNKNOWN1 IS NOT the other colors.

SNePS contains facilities for making it easy to write function node functions. The code for REQUEST-RESULT is shown in Figure 5. Within the body of the function, the arc labels of the arcs emanating from the function node (OBJ, DIM and VALUE) are treated as lambda variables. These have as their values the constant nodes their arcs go to, or the nodes their variables are bound to in case of an arc going to a variable node. The function ISBOUND? returns T if its variable is bound to a constant node, and NIL if the arc goes to a free variable. BINDQ sets a variable to a constant in such a way that the inference system also binds the variable to the constant. (SUCCEED TRUE) causes the function node to evaluate to <u>true</u> in the current substitution, and (SUCCEED FALSE) causes it to evaluate to <u>false</u>. The function FAIL causes no value to be returned -- the function node is suspended like a normal antecedent that found no match in the network.

The form

```
(SNOC (STRIP (FIND (MEM- CLASS) (^ DIM)))
      'NOT/ APPLICABLE/ OR/ UNKNOWN)
```

uses the SNePSUL function FIND to retrieve from the network all the properties in the category which is the LISP value of DIM (Remember, that is the network node bound to the variable node V5 in the current instance of the function node.), and appends to this list the NONE-OF-THE-ABOVE catchall for presentation to the assistant.

When the assistant responds with a number, as was shown in Figure 2, REQUEST-RESULT succeeds in the proper binding of V7, and the inference system resumes.

```
(DP REQUEST-RESULT (OBJ DIM VALUE)
   (IF (AND (ISBOUND? OBJ) (ISBOUND? DIM)
            (NOT (ISBOUND? VALUE)))
       (PROG (N PROPS)
             (SETQ
              N (LENGTH
                 (SETQ
                  PROPS (SNOC
                         (STRIP (FIND (MEM- CLASS) ([illegible] DIM)))
                         'NOT/ APPLICABLE/ OR/ UNKNOWN))))
             (PRIN3 <> "WHAT IS THE" *DIM OF *OBJ ? <>
              "PLEASE TYPE THE APPROPRIATE NUMBER BETWEEN 1 AND"
              *N /-- <>)
             (FOR I FROM 1 TO N
                  (PRIN3 %5 *I /) %2 * (ARGN PROPS I) <>))
             (REPEAT (ANS)
                     (SETQ ANS (READ))
               UNTIL (IF (EQ ANS N) (FAIL 'UNKNOWN))
               UNTIL (IF (AND (FIXP ANS)
                              (GREATERP ANS -1)
                              (LESSP ANS N))
                         (BINDQ VALUE (ARGN PROPS ANS))
                         (SUCCEED TRUE)
                         T)
                     (PRIN3 "PLEASE JUST TYPE AN INTEGER BETWEEN
                            1 AND" *N)))))
```

Figure 5
The code for REQUEST-RESULT

Parsing, Generating and Tracing

Figure 2 begins with the input (: UNKNOWN1 IS ASSIGNED). The SNePSUL function ":" passes its argument to an ATN parser. In this case, an action on on ATN arc evaluates the SNePSUL form (ADD OBJ UNKNOWN1 PROP ASSIGNED). ADD builds a node, triggers forward inference and returns all nodes added to the network as a result. The grammar passes these to the generating sub-network which produces the last sentence shown in Figure 2.

The generation grammar consists of two parts. One part generates the framework of a sentence expressing a SNePS deduction rule. We have seen examples of these throughout this paper. This section of the grammar is designed for the quantifiers and connectives used in SNePS and has been used in several different domains.

The other part of the generation grammar was designed specifically for this domain and generates sentences for the atomic proposition case frames shown in Figure 3.

The inference tracing function is called whenever a rule successfully deduces one of its consequences. It is passed the antecedents that were found to hold, the consequents being deduced and a substitution giving the proper instance of the rule. A switch determines whether the tracing function will do nothing, print the antecedents and consequents in a SNePS internal format, or use SURFACE to print them. To produce Figure 2, SURFACE was used. The trace can also be conditional. To reduce the length of Figure 2, the trace function only printed when the consequent was an atomic proposition, the negation of an atomic proposition or the conjunction of such propositions.

Conclusions

The SNePS system consists of a fully indexed semantic network data base, a bi-directional inference system which operates according to deduction rules stored in the network using a set of non-standard quantifiers and propositional connectives, an interpreter for parsing-generating grammars, and a flexible tracing package.

SNePS is currently implemented in the ALISP dialect of LISP and runs on a CYBER 174 under the NOS operating system. Copies of the system are available from the author. Activity is currently planned or underway to translate SNePS into MACLISP and INTERLISP.

We have seen how SNePS can be used to impliment COCCI, a program for solving microbiology laboratory unknown problems. For such purposes, SNePS may be considered to be an AI programming system or a system for building expert systems.

References

1. Anderson, J. R. Language, Memory, and Thought. Lawrence Erlbaum, Hillsdale, NJ, 1976.

2. Deliyanni, A. and Kowalski, R. A. Logic and semantic networks. CACM 22, 3 (March 1979), 184-192.

3. Fahlman, S. E. NETL: A System for Representing and Using Real-World Knowledge. MIT Press, Cambridge, MA, 1979.

4. Hendrix, G. G. Encoding knowledge in partitioned networks. In Findler, N. V., ed. Associative Networks: The Representation and Use of Knowledge by Computers, Academic Press, New York, 1979, 51-92.

5. Martins, J. P., McKay, D. P., and Shapiro, S. C. Bi-directional inference. Department of Computer Science, SUNY at Buffalo, Amherst, NY, 1981.

6. Maida, A. S. and Shapiro, S. C. Intensional concepts in propositional semantic networks. Department of Computer Science, SUNY at Buffalo, Amherst, NY, 1981.

7. Quillian, M. R. Semantic memory. In Minsky, M., ed. Semantic Information Processing, MIT Press, Cambridge, MA, 1968, 227-270.

8. Schubert, L. K., Goebel, R. G., and Cercone, N. J. The structure and organization of a semantic net for comprehension and inference. In Findler, N. V., ed., Associative Networks: The Representation and Use of Knowledge by Computers, Academic Press, New York, 1979, 121-175.

9. Shapiro, S. C. A net structure for semantic information storage, deduction and retrieval. Proc. 2nd. IJCAI, The British Computer Society, London, England, 1971, 512-523.

10. Shapiro, S. C. Representing and locating deduction rules in a semantic network. Proc. Workshop on Pattern-Directed Inference Systems. SIGART Newsletter, 63 (June 1977), 14-18.

11. Shapiro, S. C. The SNePS semantic network processing system. In Findler, N. V., ed. Associative Networks: The Representation and Use of Knowledge by Computers, Academic Press, New York, 1979, 179-203.

12. Shapiro, S. C. Generalized augmented transition network grammars for generation from semantic networks. Proc. 17th Annual Meeting of the ACL. U. of California at San Diego, August, 1979, 25-29.

13. Shapiro S. C. Numerical quantifiers and their use in

reasoning with negative information. Proc. Sixth IJCAI, Computer Science Dept., Stanford U., 1979, 791-796.

14. Shortliffe, E. H. Computer-Based Medical Consultations: MYCIN. American Elsevier, New York, 1976.