Chapter 5

MYCIN

5.1 Introduction

MYCIN was developed at Stanford by Shortliffe (1970’s). It was an expert system for diagnosing blood diseases. It was a precursor to today’s expert systems and acts as an ideal case study.

5.2 Features

MYCIN is a rule based ES using backward chaining. A typical rule

IF stain of organism is gram negative
    AND
    morphology is rod
    AND
    is anaerobic
THEN
    suggestive that class is enterobacteriaceae (CF 0.8)

The rules were actual stored as LISP expressions.

Inexact reasoning was employed using certainty factors. This is a number on the scale -1 to 1. -1 being definitely false +1 definitely true. The calculation of the CFs for derived results is explained in a later chapter 9.

MYCIN used meta-rules to redirect search at stages. An example of such a meta rule is

IF infection is pelvic abscess
    AND
    rules mention in premise E
    AND
    rules mention in premise gram pos rods
THEN evidence should use rules for E before rules for gram pos rods

The above is guiding the search by saying which rules to try first.

To make the system acceptable to users (doctors) MYCIN incorporated

**Natural Language** Users entered information in English and were answered in English.

**Spelling** A spelling checker made intelligent corrections to the input.

**Explanation** Both WHY, WHY NOT and HOW explanation facilities were provided.

**Alternative Recommendations** Several suggestions were often offered listed with some priority, allowing the doctor some freedom of selection.

MYCIN has two phases in its approach, a *diagnosis* and a *prescription* phase. In the first phase the nature of the infection and the organism causing the infection are determined (hopefully). The prescription phase then indicates the drugs for the treatment taking into account any side effects they may have on the patient.

MYCIN consists of about 500 rules. It is a backward chaining rule based system using a depth-first search strategy.

The next few pages give a sample session with MYCIN.
5.2.1 Mycin-Physician Dialogue

Essentially from Durkin.

COMMENT: MYCIN begins the session by asking for general information about the patient.

STEPS

1 MYCIN: What is the patient’s name?
User: JILL EDWARDS

2 MYCIN: What is the patient’s age?
User: 45

3 MYCIN: What is the patient’s sex?
User: FEMALE

4 MYCIN: What is the patient’s race?
User: CAUCASIAN

COMMENT: After obtaining background information, MYCIN attempts to determine the nature of the infection. It first seeks help from the user to see if this information is known from test results that produced positive cultures.

5 MYCIN: Are there any cultures for JILL EDWARDS which may be related to the present illness, and from which organisms have been grown successfully in the microbiology laboratory?
User: YES

6 MYCIN: From what site was the specimen for the culture taken?
User: CSF

7 MYCIN: Please give the date and time when the pending CSF culture was obtained.
User: 7-JULY-2001-7:00

8 MYCIN: Was a smear examined at the time the specimen was obtained?
User: YES

9 MYCIN: Were any cultures seen on the smear of the specimen?
User: YES

10 MYCIN: Have you been able to obtain positive microbiological information about infection of JILL EDWARDS?
User: YES

11 MYCIN: What is the infection?
User: Endurteritis
COMMENT: To accommodate the user, MYCIN can recognize misspelled words and make a guess at the correct answer.

12 MYCIN: Did you mean to type ENDARTERITIS?
User: YES

COMMENT: In the absence of positive test data, MYCIN would ask for other information related to the patient’s symptoms to establish a general understanding of the problem leading to the following step. For example, MYCIN might ask questions related to the suspicion of meningitis, as follows:

MYCIN: Has JILL EDWARDS recently had symptoms of persistent headache or other abnormal neurological symptoms (dizziness, lethargy, etc.)?
User: YES
MYCIN: Has JILL EDWARDS recently had objective evidence of abnormal neurologic signs (rigidity, coma, seizures, etc.) documented by a physician?
User: YES
MYCIN: The CSF cultures will be considered to be associated with MENINGITIS.

COMMENT: In the absence of positive test information, MYCIN would now suspect a meningitis infection and display this result to the user. However, since another infection has been identified, MYCIN now attempts to determine the identity of the infecting organisms for this infection. A number of questions would be asked to help in this identification beginning with asking the physician if he can identify the organism from past tests.

13 MYCIN: Enter the identity of ORGANISM-1
User: UNKNOWN

COMMENT: The system wants to know if the user can identify the organism. However, in this case the organism is unknown to the user. We want the system to continue to reason even though some information may not be available. MYCIN will now attempt to identify the unknown organism from other available information.
Continuing

14 MYCIN: Is ORGANISM-1 a rod or coccus (etc.)?
User: ROD

15 MYCIN: The gram stain of ORGANISM-1?
User: GRAMNEG

16 MYCIN: What is the aerobicity of ORGANISM-1?
User: FACUL

17 MYCIN: Is the patient’s illness with ORGANISM-1 a hospital acquired infection?
User: WHY

COMMENT: User wants to know WHY this question is being asked.

MYCIN:
[2.0 ] This will aid in determining the identity of ORGANISM-1 that might be infecting JILL EDWARDS

[RULE 92 ] It has already been established that

[2.1 ] the morphology of ORGANISM-1 is rod, and
[2.2 ] the gram stain of ORGANISM-1 is gramneg, and
[2.3 ] the aerobicity of ORGANISM-1 is facul

Therefore, if

[2.4 ] the infection with ORGANISM-1 was acquired while the patient was hospitalized

Then

there is weakly suggestive evidence (.2) that the identity of ORGANISM-1 is pseudomonas

COMMENT: MYCIN displays the rule that it is pursuing in response to the WHY question, providing the physician with a view of the reasoning approach. The display shows what was proven and what is being sought. If the user would ask WHY again, the system would explain why it is pursuing the identity of this particular organism. MYCIN then asks the question again.
18 MYCIN: Is the patient’s illness with ORGANISM-1 a hospital acquired infection?
User: NO
19 MYCIN: Have there been POSITIVE cultures yielding organisms about which you will NOT be seeking advice?
User: YES
20 MYCIN: Enter the identity of ORGANISM-2
User: SALMONELLA (.4) E.COLI (.3)

COMMENT: The user qualifies the belief in this answer, i.e., I believe it is Salmonella with a degree of belief of 0.4 and I also believe it might be E.Coli with a degree of belief of 0.3.

The session would continue in this fashion, asking in total anywhere between 20 and 100 questions, and then finally arrive at the following listing of suspected organisms with corresponding levels of belief expressed numerically:

21 MYCIN: INFECTION-1 IS ENDARTERITIS with BACTEREMIA
   item1 E.COLI [ORGANISM-1] (.71)
   item2 SALMONELLA (species unknown) [ORGANISM-1] (.60)
   item3 KLEBSIELLA-PNEUMONIA [ORGANISM-1] (.25)

MYCIN concludes that the infection is endarteritis, and lists three possible bacteremia organisms that might be causing the infection. The listing is also rank ordered with corresponding beliefs represented numerically for each organism suspected. The therapy selection would then have to address all three suspected organisms.

After reviewing the results, the doctor may want to know how a given conclusion was derived.

22 User: HOW DID YOU DECIDE THAT ORGANISM-1 WAS AN E.COLI?
MYCIN: I used rule 084 to conclude that the identity of ORGANISM-1 is e.coli. This gave a cumulative c.f. of (.71)
RULE 084
   if
      1 The site of the culture is one of those sites that are normally sterile, and
      2 This organism has at least one of the likely pathogens associated with the portal of entry of the organism agree with respect to the following properties: gram morph air
then  There is strongly suggestive evidence (.8) that each of them is the identity of the organism.

COMMENT: MYCIN provides a list of rules that led to the conclusion. In most rule-based systems today, answering a query of HOW would provide a full trace of all the evidence that led to the final result.

The doctor can also ask the system why something was not found to be true.

23 User: WHY DIDN’T YOU THINK THAT ORGANISM-1 WAS NEISSERIA?
MYCIN: The following rules could have been used to determine that the identity of ORGANISM-1 is neisseria-RULE 9. However, none of these succeeded in the context of ORGANISM-1. If you would like an explanation for why any of these rules failed, please enter their numbers:
24 User: 9
MYCIN: Clause 2 of RULE 009 ["the morphology of the organism is coccus"] was already known to be false for ORGANISM-1, so the rule was never fired.

COMMENT: MYCIN demonstrates a simple capability of introspection by being able to explain WHY and WHY NOT something was found to be true. It is capable of performing this introspection using the set of rules as a model through which it can trace back along its reasoning path to provide an answer to the query.

COMMENT: MYCIN would now proceed in an attempt to form a recommendation for the best treatment to eliminate the suspected infection. This is accomplished through a plan-generate-test cycle of various candidate drug therapy recommendations. The system would also ask for personal information about the patient to assure a safe final recommendation.

25 MYCIN: Does JILL EDWARDS have a clinically significant allergic reaction to any antimicrobial agent?
User: NO

COMMENT: Other personal questions might be asked leading to the following recommendation that meets the specific needs of the patient.

26 MYCIN: [REC-1] My preferred therapy recommendation is as follows:
In order to cover the items < 123 >: Give the following combination:
1 AMPICILLIN Dose 2.5 g (30 ml)
2 **GENTAMICIN**

Dose 131 mg (3.2 ml, 80 mg/2 ml ampule) for 10 days.

**Comments:** Monitor serum concentrations. Since high concentrations of penicillins can interact aminoglycosides, do not mix these two antibiotics in the same IV bottle.

**COMMENT:** *MYCIN provides a recommendation for the treatment. It also provides information in the form of comments for administering the drugs.*

**COMMENT:** *The system then proceeds to see if the physician agrees with the recommendation or wants to consider some alternative. If the answer is YES, MYCIN asks which drugs should be eliminated from the present prescription and again goes through the plan-generate-test cycle.*

27 **MYCIN:** Do you wish to see the next choice therapy?

User: **YES**

28 **MYCIN:** Which, if any, of the above drugs do you wish eliminated from consideration?

User: **AMPICILLIN**

**MYCIN:** [REC-2]:: Next best therapy recommendation: In order to cover the items <23>: Give GENTAMICIN Dose 131 mg (3.2 ml, 80 mg/2 ml ampule) for 10 days

**Comments:** Monitor serum concentrations.

In order to cover the items <1>: PENICILLIN Dose 4 million units (7.8 ml) for 14 days

**RECOMMENDATION-2** is substantially inferior to **RECOMMENDATION-1**

29 **MYCIN:** Would you like to see details of this comparison?

User: **NO**

**COMMENT:** *MYCIN can provide further explanation that compares different recommendations.*
5.3 Evaluation

In order to assess the performance of MYCIN a set of ten case histories of meningitis were selected. The results of MYCIN were compared with medics of various degrees of expertise from Stanford. They were also compared with evaluations from experts outside of Stanford (the evaluators). They were to classify the prescriptions according to the categories:

Equivalent identical or equivalent recommendation.

Acceptable alternative different but also considered correct.

Unacceptable considered incorrect or inappropriate.

The evaluations were carried out in a double blind situation.

The outcome of the evaluation is summarised in the Table 5.1. We see that MYCIN

<table>
<thead>
<tr>
<th>Prescriber</th>
<th>% Acceptable</th>
<th>No. Wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>MYCIN</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>Prior Rx</td>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>Faculty 1</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>Faculty 2</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>Fellow</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>Faculty 4</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>Faculty 3</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>Resident</td>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>Faculty 5</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>Student</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5.1: Evaluation of MYCIN

performed as well as the best human expert in this experiment.

5.4 EMYCIN

At the end of the MYCIN project it was realised that the basic program could be used with the knowledge replaced by information from another domain. This lead to the development of EMYCIN (Empty MYCIN). As an example of its use, a knowledge base for the diagnosis of pulmonary infections, PUFF, was produced.

This really led to the development of expert system shells, allowing expert systems to be developed in a much shorter time. For example, PUFF was produced in about 5 man-years as compared with 20 for MYCIN.
Chapter 6

Rule-Based Expert Systems

Rule based systems were the predominant kind of expert systems produced up until the mid 1980’s. After that, there was a shift towards object-oriented systems.

Expert systems began proving their worth to industry, and the time for production of a workable system was dramatically reduced with better understanding of the technology. Two systems which were particularly successful were XCON and PROSPECTOR. Both systems saved their companies millions of dollars.

In Chapter 2 we showed the basic architecture of a rule-based expert system. A more complete picture can be seen in Figure 6.1.

![Expert System Architecture](image)

Figure 6.1: Expert System Architecture

We have added in the explanation facility. Also there is an interface for developers to
interact with the knowledge base directly and an interface mechanism to external programs. For example, there is frequently a way to call say C programs, or on PCs there may be facilities for DLL and ActiveX calling.

6.1 Advantages

We list several advantages for rule-based ESs:

Natural The use of IF-THEN rules is quite natural for humans.

Separation of Control & Knowledge This allows either module to be modified separately.

Expansion By the addition of extra rules it is easy to extend the system.

Use Relevant Rules The system can use sets of rules which are relevant to solving the problem.

Explanation HOW and WHY explanations can be derived from the rules.

Consistency Checking Because of the rigid form of the rules, it is possible to perform some simple consistency checking purely on the syntax of the rules.

6.2 Disadvantages

Here we list some disadvantages:

Exact Matching In the standard implementation, the premises of the rule must match exactly for the firing of the rule.

Speed In some cases a system can be slow as the whole knowledge base may have to be searched for a match, although on modern workstations this is probably not a problem.

Inappropriate For certain domains, the knowledge may not be represented easily using rules, in which case, a rule-based expert system might be inappropriate.
Chapter 7

Forward Chaining Systems

In this chapter we look at some forward chaining systems in a little more detail. Recall the basic mechanism is to have some facts in the WM and then to perform a recognize-resolve-act cycle — determine the rules which could fire, resolve using a conflict strategy and then add the conclusion to the WM. The simplest conflict resolution strategy is to take the rule which comes first. Another simple strategy is to have priorities attached to rules and to take the one with the highest priority.

7.1 Pumping Station Diagnostics

This is a system for finding faults in a pumping station. The station consist of a number of blocks containing a pump driven my a motor and connected to the next block by a pipeline as in Figure 7.1. Each block increases the pressure by 50 units.

The problem is solved by fault detection, isolation and diagnosis. The assumptions made are:

**detection** A fault is detected by a drop in the pressure in line 3 ( <150) (group S).

**isolation** The faulty block is where there is a normal input pressure but low output pressure (group I).

**diagnosis** The faulty component of the block is determined. The M rules for a faulty motor, the P rules for a faulty pump and L rules for faulty line.

So that the values which the variables can take are known, we have to assert some type information into the WM. These will consist of a number of is a facts. We also need to know the relationship of the components (motor1 is the motor of block1 etc.) The initial WM is then:

block1 is a block ... 
motor1 is a motor ... 
line1 is a line ...
Forward Chaining Systems

In this chapter we look at some forward chaining systems in a little more detail. Recall the basic mechanism is to have some facts in the WM and then to perform a recognize-resolve-act cycle — determine the rules which could fire, resolve using a conflict strategy and then add the conclusion to the WM. The simplest conflict resolution strategy is to take the rule which comes first. Another simple strategy is to have priorities attached to rules and to take the one with the highest priority.

7.1 Pumping Station Diagnostics

This is a system for finding faults in a pumping station. The station consists of a number of blocks containing a pump driven by a motor and connected to the next block by a pipeline as in Figure 7.1. Each block increases the pressure by 50 units.

Figure 7.1: Pumping Station

<table>
<thead>
<tr>
<th>block1</th>
<th>motor1</th>
<th>motor1</th>
<th>block1</th>
<th>pump1</th>
<th>pump1</th>
</tr>
</thead>
<tbody>
<tr>
<td>block1</td>
<td>input_line</td>
<td>line1</td>
<td>block1</td>
<td>output_line</td>
<td>line2</td>
</tr>
</tbody>
</table>

....

motor1 nominal_current 1 ....
line1 nominal_pressure 50 ....

So it is known when a component is out of range, the nominal values are also asserted.

The $KB$ is as follows, where identifiers starting with ? are variables:

; determine normal and low pressure and current

1S Line pressure low
IF ?Line is_a line
AND ?Line pressure ?X
AND ?Line nominal_pressure ?NP
AND ?X < ?NP
THEN ?Line pressure_status low
AND DISPLAY FAULT DETECTED

2S Line pressure normal
IF ?Line is_a line
AND ?Line pressure ?X
AND ?Line nominal_pressure ?NP
AND ?X >= ?NP
THEN ?Line pressure_status normal

3S Motor current low
IF ?Motor is_a motor
AND ?Motor current ?C
AND ?Motor nominal_current ?NC
AND ?C < ?NC
THEN ?Motor current_status low

4S Motor current normal
IF ?Motor is_a motor
AND ?Motor current ?C
AND ?Motor nominal_current ?NC
AND ?C >= ?NC
THEN ?Motor current_status normal

5S Pressure low in line3 - fault
IF line3 pressure_status low
THEN DISPLAY FAULT DETECTED

; fault isolation rules group I

1I block may be at fault
IF ?Block is_a block
AND ?Block input_line ?IL
AND ?Block output_line ?OL
AND ?IL pressure_status normal
AND ?OL pressure_status low
THEN ?Block status bad
AND DISPLAY FAULT ISOLATED

; diagnosis rules motor problem

1M motor may have problem
IF ?Block is_a block
AND ?Block status bad
AND ?Block motor ?Motor
AND ?Motor current_status low
THEN ?Motor status bad
AND DISPLAY FAULT FOUND
AND ASK Replace motor
; diagnosis rules pump problem

1P pump may have problem
IF ?Block is_a block
AND ?Block status bad
AND ?Block pump ?Pump
AND ?Block input_line ?IL
AND ?Block output_line ?OL
AND ?IL pressure ?IP
AND ?OL pressure ?OP
AND ?IP = ?OP
THEN ?Pump status bad
AND DISPLAY FAULT FOUND
AND ASK Replace pump

; diagnosis rules line problem

1L line may have problem
IF ?Block is_a block
AND ?Block status bad
AND ?Block motor ?Motor
AND ?Motor current_status normal
AND ?Block input_line ?IL
AND ?Block output_line ?OL
AND ?IL pressure ?IP
AND ?OL pressure ?OP
AND ?IP < ?OP
THEN ?OL status bad
AND DISPLAY FAULT FOUND
AND ASK Replace line

; Response rules

1R Replace motor
IF Replace motor
AND ?Block is_a block AND ?Block status bad
AND motor replacement ?NewMotor
THEN Remove ?Motor
AND ?Block motor ?NewMotor
AND DISPLAY FAULT FIXED
AND STOP
In this system, the user is ASKED for permission to make the recommended change in the diagnosis stage. The actual replacement is carried out by group R rules.

In order to make replacements, we assume that we have in the WM replacement parts indicated as follows:

```plaintext
motor replacement motorx
pump replacement pumpx
line replacement linex
```

Here we are assuming that all motors etc. are the same. If this were not the case, then we would have to include separate replacements for all motors, pumps and line (e.g. with facts of the form `motor1 replacement motorx1`).

A sample session follows. We assume in addition to the contents indicated above, the WM also has:

```plaintext
line1 pressure 50 motor1 current 1
line2 pressure 50 motor2 current 1
line3 pressure 100
```
The session starts by using rules in GROUP S to determine the status of the pipeline pressures and motor currents. The resulting WM contains

```
line1 pressure_status normal  motor1 current_status normal
line2 pressure_status low     motor2 current_status normal
line3 pressure_status low
```

When 5S fires, the system displays the message:

**SYSTEM:** Fault detected, will attempt to isolate source of problem.

Now RULE 1I fires asserting block1 status bad in the WM.

**SYSTEM:** Fault seems to be in block1.  
Will attempt to determine faulty component.

RULE 1P now fires with ?Pump instantiated to pump1. pump1 status bad is asserted into the WM.

**SYSTEM:** The fault seems to be pump1. 
Do you want to replace it?

**USER:** YES

The answer YES causes Replacement permission granted to be asserted in the WM causing RULE 2R to be fired. This instantiates ?NewPump to pumpx and replaces pump1 by pumpx. block1 pump pumpx is asserted in the WM and reference to pump1 removed.

**SYSTEM:** pump1 in block1 has been replaced by pumpx

In this example the data to start the session was asserted in the WM. This could have been achieved by a startup rule which asks the user for this information.

```
IF get info
THEN ASK....
AND ASK....
```

We only then need get info in the WM to start the session off.
7.2 Train Loading System

The purpose of this ES is to pack waiting passengers into a series of carriages. This is an essential constraint problem — minimise the number of carriages where each carriage has a maximum allowed weight. Also want to pack the heavier people near the front for stability purposes. This ES has some similarities with the XCON system for designing computer configurations.

In a design problem like this, it is usually important to keep track of the tasks to be completed and to solve them in a correct sequence. The rules will then often take the form

IF <step> AND <heuristic> THEN <action>

In this case the information in the heuristic is only applied at the correct step. The action part could include an indication as to what the next step should be.

The solution to this packing problem should:

1. pack persons by decreasing weight,
2. not exceed the maximum weight in any carriage,
3. maximize the number of people per carriage.

The KB appear below. The rules are split up into different tasks.

; BASIC CONTROL GROUP A

1A Display carriage
IF Task IS display present carriage
THEN display carriage contents
AND DISPLAY present carriage
AND TASK IS start new carriage

2A Packing complete
IF number of small people = 0
AND number of medium people = 0
AND number of large people = 0
THEN Task IS done
AND STOP
3A Start new carriage
IF Task IS start new carriage
THEN carriage number = carriage number +1
AND weight of carriage = 0
AND large people in carriage =0
AND medium people in carriage =0
AND small people in carriage = 0
THEN Task IS pack large person

; PACKING RULES GROUP B

1B Pack small person
IF Task IS pack small person
AND number of small people > 0
AND weight of small person + weight of carriage <= max weight of carriage
THEN small people in carriage = small people in carriage +1
AND weight of carriage = weight of carriage + weight of small person
AND number of small people = number of small people -1

2B Pack medium person
IF Task IS pack medium person
AND number of medium people > 0
AND weight of medium person + weight of carriage <= max weight of carriage
THEN medium people in carriage = medium people in carriage +1
AND weight of carriage = weight of carriage + weight of medium person
AND number of medium people = number of medium people -1

3B Pack large person
IF Task IS pack large person
AND number of large people > 0
AND weight of large person + weight of carriage <= max weight of carriage
THEN pack the large person
AND large people in carriage = large people in carriage +1
AND weight of carriage = weight of carriage + weight of large person
AND number of large people = number of large people -1
; SWITCH RULES GROUP C

1C Switch from packing small person to display carriage
IF Task IS pack small person
AND number of small people =0
OR weight of small person + weight of carriage >
    max weight of carriage
THEN Task IS display present carriage

2C Switch from packing medium person to small
IF Task IS pack medium person
AND number of medium people =0
OR weight of medium person + weight of carriage >
    max weight of carriage
THEN Task IS pack small person

3C Switch from packing large person to medium
IF Task IS pack large person
AND number of large people =0
OR weight of large person + weight of carriage >
    max weight of carriage
THEN Task IS pack medium person

Note the order of packing — large, medium and small — is controlled by the rules.

A session follows, where we assume the WM is

number of small people=4       weight of small person=100
number of medium people = 4   weight of medium person=200
number of large people = 4     weight of large person=300
weight of carriage=0           max weight of carriage=1000
carriage number=0              Task IS start new carriage

RULE 3A is first fired, initialising numbers of people and weight of carriage, as Task IS start new carriage is in the WM. The new task is pack large person. This fires RULE 3B. This continues to fire until the weight of a large person would overload the carriage. The number of large people is then 3 and the weight of the carriage is 900. This then causes the switch rule 3C to fire when the task becomes load medium person.

Now RULE 2B fails to fire but another switch rule, 2C, so that the task is pack small person. Now RULE 1B fires filling this carriage. This causes RULE 1C to change the task to display present carriage. Now RULE 1A fires displaying the contents and changing the task to start new carriage.
SYSTEM: CAR NUMBER 1
weight of car 1000
number of large people 3
number of medium people 0
number of small people 1

The session continues in a similar manner until eventually the numbers of the people have all been reduced to 0 and thus causing RULE 2A to fire stopping the execution. In the process of forward chaining the contents of all the carriages is printed out. The solution found appears in Table 7.1. This example has shown how a forward chaining ES can be used for a design problem. The control being achieved by keeping track of the task to be performed. This could also have been done by attaching priorities to rules but can cause problems in the maintenance of the system.

### 7.3 Design

The major steps in designing a backward chaining ES are:

1. define the problem;
2. define input data;
3. define data-driven structure;
4. write initial code;
5. design the interface;
6. expand/refine the system;
7. evaluate the system
Define the Problem

As for the backward chaining case you need to learn about the problem. In a diagnostic system it may be possible to have access to a trouble shooting manual which would then have the expertise already laid out for use. This would have been written by an expert anyway, and is more convenient than having to find a human expert.

Because of the problem of maintaining trouble shooting manuals, companies are now looking to build them as ESs. These are interactive and can incorporate text pages which would be in the original paper manuals or better still include video images.

A paper manual may well have a table or decision tree to aid in the isolation of a fault. These are ideal starting points for obtaining the rules. The table may be given in the format TEST-RESULT-PROCEED as in Table 7.2.

<table>
<thead>
<tr>
<th>Test</th>
<th>Results</th>
<th>Proceed to</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Turn on ignition</td>
<td>If engine turns slowly or not at all</td>
<td>2.1 cranking system</td>
</tr>
<tr>
<td></td>
<td>If engine runs normally</td>
<td>1.2</td>
</tr>
<tr>
<td>2.1 Put a screwdriver between ....</td>
<td>IF lights brighten or not on</td>
<td>Bad battery connection</td>
</tr>
<tr>
<td></td>
<td>If lights don’t brighten</td>
<td>2.2</td>
</tr>
<tr>
<td>2.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.2: Test Result Proceed Table

Define Input Data

Devise the startup rule to obtain the required data. For example

1 start up rule
IF task IS start
THEN ASK car problem

task IS start would be in the WM to commence the session and give a selection menu such as

What is problem?
  car won’t start
  car hesitates at high speeds
  engine rough when idling
...
Define Data-driven Structure

If we need to maintain control over the chaining then the rules will mainly be of the form:

IF task is ....
AND A
THEN ...

For example if we are extracting the rule from a table such as Table 7.2 then we might have:

IF task IS test battery connection TEST
AND lights don’t brighten RESULT
THEN task is test battery PROCEED

Initial Code

Start writing down rules from the decision table. Test this out on some input data to determine that form of KB is appropriate. Once you are confident this is the right form then adding new rules is straightforward.

Interface

The interface is extremely important if users are going to want to use the system. Many professional shells now provide toolkits for building good screen with pull down menus and buttons etc. The interface needs to be developed in parallel with the development of the KB. There should be screens for startup, question and intermediate findings and

<table>
<thead>
<tr>
<th>Screen</th>
<th>Issues</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>System Objective</td>
<td>Start</td>
</tr>
<tr>
<td></td>
<td>Problem Discussion</td>
<td>Exit</td>
</tr>
<tr>
<td></td>
<td>Session Needs</td>
<td></td>
</tr>
<tr>
<td>Intermediate Findings</td>
<td>What was found</td>
<td>Continue</td>
</tr>
<tr>
<td></td>
<td>Major Reason Why</td>
<td>Exit</td>
</tr>
<tr>
<td></td>
<td>What will be done</td>
<td>Restart</td>
</tr>
<tr>
<td>Conclusion</td>
<td>What was found</td>
<td>Exit</td>
</tr>
<tr>
<td></td>
<td>Major reasons why</td>
<td>restart</td>
</tr>
<tr>
<td>Question</td>
<td>Level of user</td>
<td>Exit</td>
</tr>
<tr>
<td></td>
<td>Why question</td>
<td>Restart</td>
</tr>
<tr>
<td></td>
<td>Menus/text</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.3: Screen Design Recommendations

conclusions stages. With a decent development tool, the screens can exhibit graphics with sliders etc. giving feedback to the user. It is important that the screens are consistent for
ease of use. For example similar buttons should be in the same place on different screens. Table 7.3 summarises some of the issues that should be taken into account.

**Expand/Refine**

Once the small prototype ES has been developed, then extra rules can gradually be added. This will also include the design of extra screens associated with these new rules.

**Evaluation**

The system now needs to be tested on some real data. This can be obtained from an expert or if we have used a manual for development then we can use this directly to check the results produced by the ES.
Chapter 8

Backward Chaining Systems

Here we look at some backward chaining systems in a little more detail.

8.1 Medical Diagnosis Example

The $KB$ consists of the following rules

GOAL 1 Infection meningitis

1 MENINGITIS INFECTION
IF doctor knows patient has meningitis
OR suspect meningitis
THEN
   infection is meningitis
AND DISPLAY infection ...
ELSE DISPLAY no infection found

2 SUSPECT MENINGITIS FROM TESTS/SYMPTOMS
IF suspect meningitis from tests
OR suspect meningitis from symptoms
THEN
   suspect meningitis

3 SUSPECT MENINGITIS FROM TESTS
IF tests run
AND cultures seen
AND cultures like meningitis
THEN
   suspect meningitis from tests
4 CULTURES LIKE MENINGITIS
IF culture appears coccus
AND stain grampos
THEN
cultures like meningitis

5 SUSPECT MENINGITIS FROM SYMPTOMS
IF patient has persistent headaches
AND patient has dizziness
AND patient lethargic
THEN
suspect meningitis from symptoms

From the goal an and/or proof tree is traversed in a depth first manner:

```
Figure 8.1: and/or tree
```

The tree itself isn't actually constructed, but we can use it to see the order in which the rules are considered. The value of the leaves have to be determined by asking the user or from the working memory. The session might have proceeded as follows:

The user is asked if the doctor says the patient has meningitis with answer NO, asserting `know meningitis` as false in the working memory. Next the other premise of RULE 1 is checked in the WM and found not to be there so RULE 2 used.
The first premise has not been proved, so now use RULE 3. The first two premises are primitive and so the user is asked if they are true, YES being the answer in both cases thus adding test run and cultures seen as true assertions in the WM. At this point the third premises is unknown and so RULE 4 is used. To determine the first premise of 4 the system asks:

SYSTEM: culture appears coccus?

If the user now asks WHY the system will respond:

To help determine if CULTURES LIKE MENINGITIS
RULE 4
  IF culture appears coccus [4.1]
  AND stain gramos [4.2]
  THEN cultures like meningitis [4.0]

The user may then wonder why the system wishes to determine the conclusion and ask WHY 4.0?. The system then proceeds to indicate it needs this to fire RULE 3:

To help determine SUSPECT MENINGITIS FROM TESTS
RULE 3 we know
  tests run [3.1]
  cultures seen [3.2]
So if cultures like meningitis [3.3]
THEN suspect meningitis from tests [3.0]

After this the user answers NO and so the current state of WM is

1. know meningitis — FALSE
2. tests run — TRUE
3. cultures seen — TRUE
4. culture appears coccus — FALSE

The system now backtracks and tries to fire RULE 2 with the second premise suspect meningitis from symptoms. It is not in the WM so RULE 5 is considered next. All the premises of RULE 5 are unknown primitives so the user has to be asked:

SYSTEM: patient has persistent headaches?
  YES
SYSTEM: patient has dizziness?
  YES
SYSTEM: patient lethargic?
  YES
This fires RULE 5, which fires RULE 2 and finally RULE 1 thus establishing the goal and cause the system to output some display to this effect:

From the information provided, I believe the infection is meningitis.

8.2 Prescription Example

In this simple example we illustrate how priorities can be attached to rules and see the system will answer a HOW query. The \( KB \) is as follows:

Goal 1: Prescription is X

1 MENINGITIS PRESCRIPTION 1 Priority 100
IF infection meningitis
AND patient is child
THEN
  prescription is no_1
AND drug is ampicillin
AND drug is gentamicin
AND DISPLAY prescription1
ELSE DISPLAY default

2 MENINGITIS PRESCRIPTION 2 Priority 80
IF infection meningitis
AND patient is adult
THEN
  prescription is no_2
AND drug is penicillin
AND DISPLAY prescription2

3 AGE CLASSIFICATION 1
IF age < 10
THEN
  patient is child

4 AGE CLASSIFICATION 2
IF age \( >= \) 10
THEN
  patient is adult

We assume that it has already been determined that infection is meningitis which is in the WM. Now from the goal RULE 1 or 2 could be considered. Since we have a priority
rating, the one with the higher priority is chosen first. This corresponds to traversing the and/or tree so that at OR nodes branches are selected on priority ordering. In this case it is RULE 1 that is selected first.

The subgoal patient is child is then launched and as it is not in the WM, RULE 3 is selected. The premise age < 10 is a primitive and so the user is asked:

SYSTEM: What is the age of the patient?
USER: 9

This fires RULE 3 which fires RULE 1 creating the WM

1. infection meningitis — TRUE
2. age — 9
3. patient is child — TRUE
4. prescription no_1 — TRUE (X is instantiated to no_1)
5. drug is ampicillin — TRUE
6. drug is gentamicin — TRUE

The system would then display the result:

The recommendation is:
AMPICILLIN: 3.8g
GENTAMICIN: 131mg

The user now wishes to see how the system arrived at the answer so asks HOW? The system prints a version of the proof of the goal. This is essentially a subtree of the and/or tree which proves the goal so that at OR nodes one branch is selected and at AND nodes all branches are selected and all leaves are true.

SYSTEM: goal: prescription no_1 was proved using
RULE 1
IF infection meningitis [1.1]
AND patient is child [1.2]
THEN
prescription is no_1 [1.0]
AND drug is ampicillin [1.0.1]
AND drug is gentamicin [1.0.2]
[1.1] was known

[1.2] used
RULE 3
IF age < 10 [3.1]
THEN
patient is child [3.0]
[3.1] was true from answer age = 8
So [3.0] = [1.2] is true,
[1.0], [1.0.1] and [1.0.2] follow from RULE 1

We note that the system used the priority or salience of rules to determine the order in which they should be tried.

8.3 Car Diagnosis System

In this example, we indicate how meta-rules can be used to select a module depending on the type of problem to be solved. This would be natural if the KBs had come from different experts. A module is loaded by the command CHAIN.

We also use a blackboard to exchange information between separate modules. This could be implemented as a commonly accessible file or area of memory.

This example also use object-attribute-values (o-a-v’s). This allows the system to ask other than YES/NO questions. For example if we represent the size of boxes as o-a-v (size Bx Sz) where Sz could be large, medium or small, then we can ask questions of the form:

Q: What is the size of the box? (large, medium, small)
A: small

The values that can be taken can be extracted from the KB or WM (or in some shells the values must be explicitly stated beforehand) and then used in query menus. In the example below, we use IS or = to designate o-a-v’s, e.g. we write the box’s size IS large or the box’s size = large for (size,box,large).

The Car Diagnosis System can be pictured as in Figure 8.2:
Here is the chief mechanic KB.

; Chief mechanic system
; Initialization
Fuel_cost = 20
AND ....

: GOALS
1 chief mechanic recommendation is X
1.1 cost of repair is known
1.1.1 display findings

;DETERMINE GENERAL FAULT

1 PROBLEM SEEMS TO BE WITH ELECTRICS
IF problem seems to be with electrics
THEN
   chief mechanic recommendation IS get electrical xs
AND problem area IS electrics
AND DISPLAY ....
AND CHAIN ELECTRIC

Figure 8.2: Car Diagnosis System Architecture
2 PROBLEM SEEMS TO BE FUEL SYSTEM
IF problem seems to be fuel system
THEN
  chief mechanic recommendation IS get fuel system xs
AND problem area IS fuel
AND DISPLAY ....
AND CHAIN FUEL

3 STARTER DOESN’T TURN
IF engine condition IS won’t start
AND starter condition IS doesn’t turn
THEN problem seems to be with electrics

4 STARTER TURNS
IF engine condition IS won’t start
AND starter condition IS turns
AND engine turns = very slowly
THEN problem seems to be with electrics

5 ENGINE RUNS ROUGH
IF engine condition IS will start
AND engine runs rough
AND engine turns = very slowly
THEN problem seems to be with electrics

6 ENGINE WON’T START BUT TURNS NORMALLY
IF engine condition IS won’t start
AND engine turns = normally but won’t start
THEN problem seems to be with fuel system

7 PROBLEMS ON ACCELERATION
IF car hesitates when accelerated
THEN problem seems to be with fuel system

; REPAIR COSTS

8 FUEL COST
IF solution IS add fuel
THEN cost of repair is known
AND repair_cost := fuel_cost
9 REBUILD CARBURETTOR
IF solution IS rebuild carburettor
THEN cost of repair is known
AND repair_cost := carb_rebuild_cost

10 REPLACE CARBURETTOR
......

11 FUEL FILTER
......

12 FUEL LINE
......

13 NO SOLUTION
IF solution IS unknown
THEN cost of repair is known
AND repair_cost := default_cost
AND DISPLAY ... 

14 SHOW RESULTS
IF cost of repair is known
THEN display findings
AND DISPLAY ...

The fuel system $KB$:

; fuel system xs

GOALS
1 determine fuel system fault
1.1 return findings to chief mechanic

1 OUT OF PETROL
IF engine condition IS won’t start
AND starter condition IS turns
AND fuel gauge = doesn’t move
THEN determine fuel system fault
AND fault IS out of petrol
AND solution IS add fuel
2 CARB REBUILD
IF engine condition IS won’t start
AND starter condition IS turns
AND fuel gauge = does move
AND car will = start occasionally
THEN determine fuel system fault
AND fault IS dirty carburettor
AND solution IS rebuild carburettor

3 CARB REPLACE
IF engine condition IS won’t start
AND starter condition IS turns
AND fuel gauge = does move
AND car will = never start
THEN determine fuel system fault
AND fault IS bad carburettor
AND solution IS replace carburettor

4 PETROL FILTER
IF engine condition IS will start
AND car hesitates when accelerated
AND fuel filter IS older than 1 year
THEN determine fuel system fault
AND fault IS bad fuel filter
AND solution IS replace fuel filter

5 PETROL LINE
IF engine condition IS will start
AND car hesitates when accelerated
AND fuel filter IS less than 1 year
THEN determine fuel system fault
AND fault IS blocked fuel line
AND solution IS replace fuel line

6 BACK TO CHIEF
IF CHAIN MECHANIC
THEN return findings to chief mechanic

The consultation might proceed as follows: RULE 1 is used first and the premise causing RULE 3 to be considered next.
**SYSTEM:** engine condition:
  won’t start
  will start ?
**USER:** won’t start

Notice how the use of o-a-v has allowed the system to ask the above query, the possible values for engine condition being extracted from the *KB*. Pursuing the second premise of RULE 3:

**SYSTEM:** starter condition:
  doesn’t turn
  turns?
**USER** turns

So RULE 3 doesn’t fire, and so we drop through to RULE 4, and the user is asked

**SYSTEM:** engine turns:
  very slowly
  normally but won’t start?
**USER** normally but won’t start

Now RULE 4 doesn’t fire so RULE 5 is checked and fails to fire since premise 1 is false. We now try RULE 2. To determine premise 1 we pursue RULE 6. The premises of RULE 6 are both true and this fires and so now 2 also fires. RULE 2 causes the system to load FUEL and display the message:

**SYSTEM:** the problem seems to be with the fuel system,
  this module will be loaded.

The system now tries to prove goal 1 of FUEL. RULE 1 is pursued and premises 1 and 2 known to be true from the blackboard. The third premise causes:

**SYSTEM:** fuel gauge:
  doesn't move
  does move?
**USER:** does move

RULE 1 thus fails and attention transferred to RULE 2. Premises 1,2 and 3 are known to be true and so to determine premise 4:

**SYSTEM:** car will:
  start occasionally
  never start?
**USER:** start occasionally
RULE 2 fires, so dirty carburettor and solution rebuild carburettor. Goal 1.1 is now tried causing RULE 6 to fire and transfer control to CHIEF MECHANIC. Goal 1.1 of CHAIN is pursued to obtain costs. From solution IS rebuild carburettor in the blackboard RULE 9 fires to calculate the cost. The final goal 1.1.1 then makes RULE 14 fire and the findings displayed.

SYSTEM: the problem was with the fuel system
the carburettor is dirty
recommendation is: rebuild carburettor
cost is: £50.

This example has displayed the use of o-a-v structures, chaining of separate modules via meta rules and using a blackboard for passing information between different modules.

### 8.4 Design

The major steps in designing a backward chaining ES are:

1. define the problem;
2. define the goals;
3. define the goal rules;
4. expand/refine the system;
5. design the interface;
6. evaluate the system

These tasks are iterated until a final system is produced.

**Define the problem**

You must learn about the problem from documents, reports etc. and discussions with the experts in the domain. At this stage an overview of the project is required, not the explicit details.

**Goals**

From discussion with the expert determine what are the goals to be solved by the system. Probably having just one main goal for the prototype.

**Goal Rules**

These are the top level rules which have (an instantiation of) the goal as a conclusion and must exist in the ES. From the expert the premises are derived possibly using a decision table.
Expand & Refine

Working back from rules added to the system, devise new rules with the help of the expert. The system should be tested as new rules are added. And/or proof trees can be drawn up to help understand the logic of the system.

See what refinements can be made to improve the system as regards both maintenance facilities and efficiency.

A default mechanism can be added for handling the case when the goal cannot be proved. This can be achieved by adding a goal which is pursued if the primary goal is unsatisfiable.

For example we could make the goal agenda:

1 advice IS X
2 display default

Goal 2 should only be followed if 1 fails. This could be achieved by having a rule

IF recommendation is unknown
THEN display default
AND DISPLAY DEFAULT TEXT

recommendation is unknown would have been asserted in the database at the start of the session. Then if 1 fails 2 will fire this rule and obtain the default message, which could include some partial information if necessary. Here we are assuming the system stops going through the goal agenda once if has found a successful goal. Otherwise, if the system is supposed to go through all goals in the agenda then we could have the system retract recommendation is unknown if goal 1 succeeds, thus prevent the above rule firing.

Interface

Many shells allow the developer to customise the interface (for example to use WINDOWS features). It is important that the interface is acceptable to users.

There should be start up screens introducing the user to the system and conclusion screens indicating the information that has been provided.

Where users are to enter data, then if the possible values to be entered are known then these should be exhibited. Better, if the system is WINDOW based then these items should be selected from a menu with a mouse.

It is also possible these days, to interface shells with other packages (especially if there are facilities for using DLL and OLE). For example, the user interface could be written using say VISUAL BASIC which then interacts with the shell.

Evaluation

Once the prototype has been built and all rules thought to be include, then it must be tested with real data. These should start with test cases from the expert for which answers
are known. Obviously these must be correct. It would then be best if the expert could try and break the system to see if there are any unsuspected/incorrect results. The use of WHY and HOW should help isolate problems.
Chapter 9

Uncertainty

In this chapter we consider two forms of inexact reasoning — Certainty Theory and Fuzzy Logic.

9.1 Certainty Theory

In this section, we outline the method used by MYCIN to calculate the certainty values for its results when rules have a given certainty factor. In MYCIN each rule has a certainty factor (CF) associated with it. This reflects to some extent the confidence one has in the rule and the CF values lie in the range -1 to +1, where +1 means absolutely certain it is true and -1 absolutely certain it is false.

With uncertain evidence we can attach a CF which can roughly be thought of giving a numerical value to the imprecise modifiers as interpreted in Table 9.1.

<table>
<thead>
<tr>
<th>Uncertain Modifier</th>
<th>CF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definitely not</td>
<td>-1</td>
</tr>
<tr>
<td>Almost certainly not</td>
<td>-0.8</td>
</tr>
<tr>
<td>Probably not</td>
<td>-0.6</td>
</tr>
<tr>
<td>Maybe not</td>
<td>-0.4</td>
</tr>
<tr>
<td>Unknown</td>
<td>-0.2 to 0.2</td>
</tr>
<tr>
<td>Maybe</td>
<td>0.4</td>
</tr>
<tr>
<td>Probably</td>
<td>0.6</td>
</tr>
<tr>
<td>Almost certainly</td>
<td>0.8</td>
</tr>
<tr>
<td>Definitely</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 9.1: CF Interpretation

So for example if we say

It is probably glandular fever
then we can think of this as

**It is glandular fever**  CF 0.6

and

**It is almost certainly not measles**

then we can think of this as

**It is measles**  CF -0.8

When obtaining inexact rules from an expert for incorporation into an ES then it is usually better if the expert expresses the rules in English and the knowledge engineer translates into the symbolic form with a numerical value for the CF provided in agreement with the values in Table 9.1.

We need to know how to combine CF’s under boolean operations. In particular how to obtain a certainty value when several rules have the same conclusion. Two properties that are required of the combination operation are that it is

**commutative** The value should not depend on the order in which the rules are taken.

**asymptotic** The more evidence we have for the belief in a conclusion the higher should be the CF but if its not absolutely certain, then it should remain below 1.

It should be noted that certainty theory is not the same as probabilistic reasoning as CF’s do not obey the laws of probability. For example it is not in general true that

\[ CF(H, E) + CF(\neg H, E) = 1 \]

whereas the corresponding equation for probability is. However, it has been found to work well in practice, and that the precise values of the CF’s of the rules is not terribly important (as long as the proofs are quite shallow).

### 9.1.1 Practical Certainty

Here we look at how the CF’s are calculated.

**Single Premise Rules**

Consider the following rule:

**IF** the sky is red at night  
**THEN** sunny next morning  CF 0.7
We want to establish the level of belief in the conclusion dependent on the certainty value of the premise. In this case the CF value is calculated according the equation:

\[ CF(H, E) = CF(E) \times CF(RULE) \]

So applying this to the above rule when then \( CF(\text{the sky is red at night}) = 0.8 \) we obtain \( CF(\text{sunny next morning}) = 0.8 \times 0.7 = 0.56 \), so we could infer maybe it will be sunny.

If we have a negative value for the premise, suppose \( CF(\text{the sky is red at night}) = -0.7 \), then \( CF(\text{sunny next morning}) = -0.7 \times 0.7 = -0.49 \) and so it will probably not be sunny.

**Multiple Premise Rules**

Now what if there is more than one premise? The reasonable rules to take are:

\[ CF(H, E_1 \land E_2 \land \ldots \land E_n) = \min\{\{CF(E_i)|i = 1, n\}\} \times CF(RULE) \]

for a conjunction and

\[ CF(H, E_1 \lor E_2 \lor \ldots \lor E_n) = \max\{\{CF(E_i)|i = 1, n\}\} \times CF(RULE) \]

for a disjunction of premises. As an example

**IF sky is dark**
**OR wind getting stronger**
**THEN it will rain** \( CF 0.9 \)

and suppose \( CF(\text{sky is dark}) = 0.7 \) and \( \text{wind getting stronger} = 0.8 \) then \( CF(\text{it will rain}) = \max\{0.7, 0.8\} \times 0.9 = 0.72 \), so it will probably rain.

**Similarly Conclude Rules**

If we have several rules with the same conclusion, they can combine to give confidence in the conclusion. For example suppose we have:

**IF weather forecast says rain**
**THEN it will rain** \( CF 0.8 \)

**IF seaweed wet**
**THEN it will rain** \( CF 0.7 \)

If we have evidence to support both premises then intuitively we feel we have more confidence in the common conclusion. The rule used by MYCIN for combining CF’s for rules with the same conclusion is as follows:

\[
CF(CF_1, CF_2) = \begin{cases} 
CF_1 + CF_2 \times (1 - CF_1) & \text{if both } > 0 \\
\frac{CF_1 + CF_2}{1 - \min\{|CF_1|, |CF_2|\}} & \text{if one } < 0 \\
CF_1 + CF_2 \times (1 + CF_1) & \text{if both } < 0 
\end{cases}
\]
In the above $CF_1$ and $CF_2$ are the certainty factors for the same conclusion from the different rules. Clearly this rule is commutative and it can be shown that it is also asymptotic and so satisfies the desired properties mentioned earlier.

Some examples using the above equation:

Suppose

\[
CF(\text{seaweed wet}) = 1 \text{ and } CF(\text{weather forecast says rain}) = 1 \text{ then } \\
CF(\text{it will rain}) = 0.8 + 0.7(1-0.8) = 0.8 + 0.14 = 0.94
\]

\[
CF(\text{seaweed wet}) = -1 \text{ and } CF(\text{weather forecast says rain}) = 1 \text{ then } \\
CF(\text{it will rain}) = (0.8 + (-0.7))/(1 - \min(0.8,0.7)) = 0.1/0.3 = .33
\]

\[
CF(\text{seaweed wet}) = -0.8 \text{ and } CF(\text{weather forecast says rain}) = -0.8 \text{ then } \\
CF(\text{it will rain}) = -0.64 +(-.56)*(1 - 0.64) = -.84
\]

**Worked Example**

We have the following $KB$ of inexact rules. We wish to determine the CF for *don’t play cricket*.

1. IF weather bad E1
   OR in bad mood E2
   THEN don’t play cricket 0.9 H1

2. IF believe going to rain E3
   THEN weather bad 0.8 E1

3. IF believe going to rain E3
   AND forecast says rain E4
   THEN in bad mood 0.9 E2

4. IF forecast says rain E4
   THEN weather bad 0.7 E1

5. IF weather bad E1
   THEN in bad mood 0.95 E2

Now suppose CF’s for primitives are:

\[
\text{CF(believe going to rain)} = .95 \\
\text{CF(forecast says rain)} = .85
\]

We use backward chaining with the goal *don’t play cricket* and the rules are exhaustively searched. From RULE 1 pursue premise E1. E1 matches 2 and 4, take one with higher CF first, i.e. RULE 2. From CF computation
CF(E1,E3) = 0.8*0.95 = 0.76

and for RULE 4
CF(E1,E4) = 0.7*0.85 = 0.6

From the two rules for E1 we have:

CF(E1) = CF(E1,E3) + CF(E1,E4)*(1-CF(E1,E3)) = .9

Now we pursue premise 2 of RULE 1, E2. There are two rule with conclusion E2. Pursuing RULE 5

CF(E2,E1) = CF(RULE 5)*CF(E1) = 0.95*0.9 = 0.86

and for RULE 3

CF(E2,E3 AND E4) = min(CF(E3),CF(E4))*CF(RULE 3) = min(.95,.85)*.9 = .77

There are two confirmations for E2 so the combined CF is given by

CF(E2) = CF(E2,E1) + CF(E2,E3 AND E4)*(1 - CF(E2,E1))
        = 0.86 + .77*(1-.086) = .97

Now from RULE 1

CF(H1,E1 OR E2) = max(CF(E1),CF(E2)) * CF(RULE 1)
                 = max(.9,.97)*.9 = .87
                 = CF(dont play cricket)

From the calculation we can say almost definitely dont play cricket.

9.1.2 Control

It is possible to use CF’s to control the search. For example, you could use CF’s in rules:

IF CF(problem is fuel system) < 0.4
THEN Goal = problem is electrics

So this changes the goal to electrics if the CF for fuel system dropped below 0.4. This is another instance of a meta-rule.

CF’s can also be used for pruning the search space. When pursuing a goal, if the CF value fell in the range say -.2 to .2 then pursuit of this goal could be abandoned. Different threshold could be set.
9.1.3 Difficulties

Deep Inference Chains

When we have a chain of inferences such as:

\[
\begin{align*}
\text{IF } A & \text{ THEN } B & \text{ CF } = 0.8 \\
\text{IF } B & \text{ THEN } C & \text{ CF } = 0.9
\end{align*}
\]

then because of the multiplication of the CF’s the resulting CF decreases. For example suppose \( \text{CF}(A) = 0.8 \), then \( \text{CF}(C) = 0.8 \times 0.8 \times 0.9 = 0.58 \). With a long chain of inferences the final CF may become very small.

In MYCIN much of the reasoning was shallow and so it was not a problem. However, in general deep inference chains should be avoided if using certainty factors for the inexact reasoning.

Many Rules with same Conclusion

The more rules with the same conclusion the higher the CF value. If there a many such rules, then this CF could become artificially high (say almost 1) and so distort the true picture. For this reason, the number of rules with the same conclusion should be limited.

Conjunctive rules

If a rule has a number of conjunctive premises, then since the minimum of the CF’s is taken this may reduce the overall CF too much. It may be better to split the rule up with separate premises but reduced rule CF.

\[
\begin{align*}
\text{IF sky dark AND temperature dropping} & \text{ THEN will rain } 0.9 \\
\text{IF sky dark} & \text{ THEN will rain } 0.7 \\
\text{IF temperature dropping} & \text{ THEN will rain } 0.5
\end{align*}
\]

\[
\begin{align*}
\text{CF(sky dark)} = 1 \text{, CF(temperature dropping)} = 0.1 \Rightarrow \text{CF(will rain)} = \min(1, 0.1) \times 0.9 = 0.09 \\
\text{CF1} = 1 \times 0.7 = 0.7, \text{ CF2} = 0.1 \times 0.5 = 0.05 \\
\text{CF(will rain)} = 0.7 + 0.05 \times (1 - 0.7) = 0.7 + 0.015 = 0.715
\end{align*}
\]

These above points emphasise that care must be taken when using certainty theory for inexact reasoning.
9.2 Fuzzy Logic

In everyday speech we use vague or imprecise terms to describe properties. Table 9.2 gives some examples. Fuzzy logic was developed by Zadeh to deal with these imprecise values in a mathematical way. It will allow us to deal with fuzzy rules for example:

IF the temperature is cold
THEN the motor speed stops

IF speed is slow
THEN make acceleration high

In the above, variables such as speed and temperature range over a domain of discourse e.g. 0 – 100 mph for speed and 0 – 100 C for temperature. The phrase speed is slow then represents a fuzzy set.

### 9.2.1 Fuzzy Sets

<table>
<thead>
<tr>
<th>Property</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>temperature</td>
<td>hot,cold,cool</td>
</tr>
<tr>
<td>height</td>
<td>short,medium,tall</td>
</tr>
<tr>
<td>weight</td>
<td>light,heavy</td>
</tr>
<tr>
<td>speed</td>
<td>slow,fast</td>
</tr>
<tr>
<td>age</td>
<td>young,old</td>
</tr>
</tbody>
</table>

Table 9.2: Imprecise Terms

In ordinary set theory, an element from the domain is either in a set or not in a set. In contrast, in fuzzy sets, a number in the range 0–1 is attached to an element — the degree

---

Figure 9.1: Small men

In ordinary set theory, an element from the domain is either in a set or not in a set. In contrast, in fuzzy sets, a number in the range 0–1 is attached to an element — the degree
to which the element belongs to the set. A value of 1 means the element is definitely in
the set, a value of 0 it definitely is not in the set. Other values are *grades* of membership.

Formally a fuzzy set $A$ from $X$ is given by its membership function which has type

$$\mu_A : X \rightarrow [0, 1]$$

In Figure 9.1 we have a representation of the fuzzy set of small men.

We often take a simpler version of the curve so that it is easy to represent in a computer.
These curves are just made of straight line segments. The representation of the small men
is then given in Figure 9.2.

![Figure 9.2: Object-attribute-value](image)

Figure 9.3 shows a representation of three fuzzy sets for small, medium and tall men.
We see that a man of height 4.8 feet is consider both small and medium to some degree

![Figure 9.3: Object-attribute-value](image)

(about .6 and .4 respectively).

**Hedges**

In English we have several adverbs for altering the vague terms, e.g. *very, slightly somewhat.*
We can achieve the effect of these modifiers by applying functions to the membership
function. For example:
Concentration — very
Here we want to decrease the values which are < 1. One possible function is to square the value

$$\mu_{\text{con}(A)}(x) = (\mu_A(x))^2$$

Dilation — somewhat
This time we want to increase the values < 1. A possible function is to take the square root

$$\mu_{\text{dil}(A)}(x) = (\mu_A(x))^{0.5}$$

Other functions can be given. Figure 9.4 illustrates the above two hedges.

![Figure 9.4: Hedges on small men set](image)

Boolean Operations
The boolean operations of union, intersection and complement can be defined in the straightforward manner.

Complement
The operation is

$$\mu_{\sim A}(x) = 1 - \mu_A(x)$$

and is illustrated in Figure 9.5.

Intersection
The intersection of two fuzzy sets A and B is given by

$$\mu_{A \land B}(x) = \min(\{\mu_A(x), \mu_B(x)\})$$
Union

The union of two fuzzy sets A and B is given by

\[ \mu_{A \cup B}(x) = \max(\{\mu_A(x), \mu_B(x)\}) \]

9.2.2 Fuzzy Reasoning

We now consider in this section, fuzzy rules and how inference is performed in a system of such rules.

This will be illustrated by a fuzzy system used to control an air conditioner. The variables to be used (which will have fuzzy values) are temperature (of the room) and speed (of the fan motor).

These fuzzy concepts are represented by the fuzzy sets illustrated in Figures 9.6 and 9.7.
The rules are given as follows:

1 IF temperature is cold
   THEN motor speed stops

2 IF temperature is cool
   THEN motor speed slows

3 IF temperature is just right
   THEN motor speed medium

4 IF temperature is warm
   THEN motor speed fast

5 IF temperature is hot
   THEN motor speed blasts

Now in a fuzzy system all the rules fire in parallel although many in the end will not contribute to the output. What we need to determine, in the above system, is given a particular value of the temperature how do we calculate the motor speed.

Now the temperature can be measured fairly accurately, but it will lie in several fuzzy sets (i.e. the ones where the value is not 0). For example, if the temperature were 17°C then from Figure 9.8 we see that it is about 25% cool and 80% just right.

This means that the rules 2 and 3 will contribute to the output speed of the motor. The fuzzy sets for the output can be illustrated in Figure 9.9 where we have multiplied the slow graph by .25 and the medium graph by .8 assuming the contribution is proportional to the fuzzy value of the input temperature.

Now one of the ways to amalgamate the two sets is to sum the values (with a maximum of 1 i.e. $\mu_{amalg(A,B)}(x) = \min\{1, \mu_A(x) + \mu_B(x)\}$), this leads to Figure 9.10 representing...
the output speed as a fuzzy set. It should be pointed out that there are other ways of amalgamation, for example taking the maximum, but this is now the most common one adopted in fuzzy systems and is quite natural.

Now it remains to determine the actual speed of the motor, because in practice we can’t set it to a fuzzy set — it must be a number. But this can be done by finding the average value of the curve — i.e. the position where the areas on either side of the perpendicular through this point are equal. Another common technique for defuzzification is to take the centroid (the vertical line along which the shape below the curve would balance). For the type of curves illustrated, this is quite easy to compute. This process of going from a fuzzy set to an explicit value from the domain is called defuzzification. In our example, we see from Figure 9.10 that the output is about 44.

The general method for interpreting additive fuzzy systems in general is expressed by the schema in Figure 9.11. Most fuzzy controllers available commercially are based on this technique.
9.2.3 Fuzzy Logic Applications

Password Authentication Example

We describe in this example how fuzzy logic can be used to authenticate passwords by using biometrics — analysis of a user’s unique keystrokes. For full details see: ‘Enhanced Password Authentication through Fuzzy Logic’, WG de Ru and JHP Eloff, *IEEE, Intelligent Systems*, vol 12 (6), 1997.

When a user has been using a password for some time, there is usually a standard pattern of entry which would be difficult for an impostor to duplicate even if the text of the password is known. The typing biometrics used here are the time intervals between consecutive keystrokes and the typing difficulty of successive characters in the password. Typing difficulty will depend on the number of keys separating the characters and whether key combinations are required (for example using the shift key). The input fuzzy variables are thus time and difficulty. The values for time are very short, short, moderately short and somewhat short with their fuzzy sets represented in Figure 9.12.

There is only one value used for difficulty of typing which is high and has the fuzzy set represented in Figure 9.13.
The output variable is categorization of the typing pattern and has values low, medium, high and very high with fuzzy set representation given in Figure 9.14.

Now if the input password were demO then we consider the pairs of characters (d,e), (e,m) and (m,O) and obtain the times say 30,050, 20,505, 50,550 and difficulties 1,5,7 where we have added 5 for the shift (m to O). These values are then used to categorize each character pair according to the following rules:

Rules stating relationships between inputs and outputs:

Rule 1 IF the time interval is somewhat short THEN the category is low

Rule 2 IF the time interval is moderately short THEN the category is medium

Rule 3 IF the time interval is short THEN the category is high
Rule 4 IF the time interval is very short THEN the category is very high

Rule 5 IF the typing difficulty is high THEN the category is high

Applying the usual inference rules for the fuzzy system will yield a typing template of categories \( t_1, t_2, t_3, \ldots, t_n \) (so for example we might have a set of values such as 151, 351, 597, 892. This is then compared against the saved template for the given password. As the values will not be exactly the same, we use the following fuzzy rule for matching:

IF \( (t_1 \text{ about } T_1) \text{ AND } (t_2 \text{ about } T_2) \text{ AND } \ldots \text{ (} t_n \text{ about } T_n ) \text{ THEN pattern is for user}X \)

where \( T_1, T_2, \ldots, T_n \) is the saved template for userX. The fuzzy sets for about \( T_n \) are all similar and illustrated for definite values in Figure 9.15. In this case the mean of the fuzzy values of the premises is taken to indicate the closeness of the match and accepted as matching the template if it is above a certain threshold (say 85%).

An assessment of the success of this technique is shown in Figure 9.16.
Table A. The performance of the typing-biometrics authentication system.

<table>
<thead>
<tr>
<th>USER ID</th>
<th>PASSWORD</th>
<th>SUCCESSFULLY IDENTIFIED LEGITIMATE USER (%)</th>
<th>FAILED TO DETECT IMPOSTOR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>aal</td>
<td>Pulling$String$</td>
<td>94</td>
<td>0</td>
</tr>
<tr>
<td>ajs</td>
<td>gihsinav</td>
<td>95</td>
<td>0</td>
</tr>
<tr>
<td>alla</td>
<td>Neural#Network</td>
<td>87</td>
<td>0</td>
</tr>
<tr>
<td>ben</td>
<td>Rich&amp;Chips</td>
<td>91</td>
<td>0</td>
</tr>
<tr>
<td>bvark</td>
<td>Millionaire$</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>deru</td>
<td>Encounter$</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>ema</td>
<td>Algorithm$</td>
<td>81</td>
<td>15</td>
</tr>
<tr>
<td>gavin</td>
<td>Maverick0</td>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td>gvdm</td>
<td>NaPoleon</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>jaco</td>
<td>TimeLord</td>
<td>96</td>
<td>4</td>
</tr>
<tr>
<td>jep</td>
<td>Foot&amp;Ball*</td>
<td>92</td>
<td>8</td>
</tr>
<tr>
<td>jjp</td>
<td>Commun1cations</td>
<td>72</td>
<td>0</td>
</tr>
<tr>
<td>jma</td>
<td>Hard+Ware</td>
<td>87</td>
<td>5</td>
</tr>
<tr>
<td>kkn</td>
<td>String:Array</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>lcjd</td>
<td>Allan Turing!</td>
<td>93</td>
<td>0</td>
</tr>
<tr>
<td>llr</td>
<td>Turbo+Reset</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>lvv</td>
<td>Not4Sale2U</td>
<td>91</td>
<td>0</td>
</tr>
<tr>
<td>mbes</td>
<td>#Digital#</td>
<td>87</td>
<td>5</td>
</tr>
<tr>
<td>mdr</td>
<td>DCDConverter</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>nan</td>
<td>MenWhoWin</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>nico</td>
<td>Spacem@n</td>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td>oaw</td>
<td>Insider?Info</td>
<td>85</td>
<td>5</td>
</tr>
<tr>
<td>pme</td>
<td>Cydoni@n</td>
<td>90</td>
<td>4</td>
</tr>
<tr>
<td>sarel</td>
<td>Tuyn7Huis</td>
<td>80</td>
<td>9</td>
</tr>
<tr>
<td>Space</td>
<td>Bubblemania</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>wcv</td>
<td>Person@lty</td>
<td>95</td>
<td>13</td>
</tr>
<tr>
<td>wil</td>
<td>Spitz&amp;Koppe</td>
<td>100</td>
<td>9</td>
</tr>
<tr>
<td>wim</td>
<td>FireWork$</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>wjlp</td>
<td>DoNetTell!</td>
<td>90</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 9.16: Assessment of Password Authentication
Commercial Applications

In the last few years, many companies have started making use of fuzzy controllers in their products. This has been especially prevalent in Japan.

The Canon H800 camera has a fuzzy controller with 13 rules to autofocus the lens. Small electrical sensors measure the image clarity and changes in clarity at six parts of the image. The rules then turn this information into a lens setting.

Panasonic has a fuzzy controller in its camcorder to stop the jitter due to hand shake. Sugeno’s fuzzy controller for a helicopter has broken new ground. The system is much simpler than any produced from a mathematical model.

Nissan has a fuzzy systems for anti-skid brakes that pumps the brakes in an optimal manner and on automatic cars changes gear based on road and car conditions as well the best human driver would do with a gear lever.

There are fuzzy systems in vacuum cleaners controlling the sucking power of the motor. Such rules might be

IF the floor is shag pile
THEN suck hard

IF the floor is tile
THEN suck little

IF the floor is very dirty
THEN suck very hard
In table 9.3 we list a number of products controlled by fuzzy systems manufactured by Japanese and Korean companies.

The are very many more products using fuzzy systems and the number is increasing each year. These applications have been successful and it is evident that Japanese and Korean (electronics) companies are investing heavily in this new software technology.