6.871 Problem Set 2 **Handed out: February 28, 2006**
Due: March 14, 2006

RULE-BASED SYSTEMS EXERCISES

INTRODUCTION

There are a few important things to keep in mind as you work through this first exercise. It is, first of all, a multi-part hand-simulation problem designed to teach you many things about how rule-based systems work. While hand-simulations can be a fair amount of work, they can also be quite instructive. They help to remove the mystery about exactly how certain things are accomplished inside the system. This problem set also has a fair amount of reading, as it is intended to teach you some things. Don't be surprise if the reading takes longer than the actual working of the problems.

There is also a computer-based version of the problem, which you will do as Problem Set 3,. This is less busywork (and more fun), but not as instructive in terms of understanding the internals.

The exercise is developed in several stages, in order to show you a number of different important points. In some cases it turns out that the best way to make a point is by seeing how *not* to do something. So you may find as you go along that there are things about the system and about the rule set you're given that are not quite correct. Have patience, because by the end of the exercise we will have fixed all of them.

Finally, there is a separate answer sheet, downloadable in both Word and PDF formats from the course web site. *Be sure to put all your answers on that sheet and hand in just that, in either hardcopy or electronic form, as you wish..*

THE BASICS OF THE INFERENCE ENGINE

GOAL
In this part of the exercise you will learn first hand the basics of the inference engine, seeing exactly how backward chaining works and how explanations are generated.

A SYSTEM FOR RECOMMENDING INVESTMENTS
With all the different kinds of mutual funds that are available these days, it's become difficult to decide how to invest your hard-earned money. Luckily we can call on knowledge-based systems for some help.

In this exercise we'll be using a simple system that can help you to decide among six of the most common categories of mutual funds. The system assumes that you have $2000 to invest and can help select among the following possibilities:

- a money-market fund
- an income fund (e.g., bonds)
- an aggressive growth fund
- a mixed growth and income fund (abbreviated “G&I”)
- a conservative growth fund
- a tax-free fund

- none (i.e., don't invest)

THE KNOWLEDGE BASE

In Table 1 you'll find the knowledge base, a set of rules written in a slightly abbreviated form. To help you read the rules, we have used some formatting conventions to help make their structure clear. Consider rule 12, for instance:

12]	*if*	Investment Goal = RETIREMENT and
		Number Of Years To Retirement < 10
	then	Category Of Fund = CONSERVATIVE GROWTH

Each rule is expressed in terms of an *if* part (the premise) and a *then* part (the conclusion). In rule 12 there are two clauses in the premise and (as in all of our rules) one in the conclusion. While there is no theoretical limit to the number of clauses in the premise, it is in general a very good idea to keep the premise small—say between one and six clauses. Rules with larger premises are generally an indication that the “fracturing” of knowledge into rules has not been very successful.

Each clause in a rule is expressed in terms of an attribute and its value. As a formatting convention, attributes are written as phrases with their first letters capitalized (e.g., Investment Goal, Number Of Years To Retirement, Category Of Fund); values are written in all capitals (RETIREMENT, 10, CONSERVATIVE GROWTH). The first clause thus asks whether the topic Invest Goal has the value RETIREMENT, or, in smoother English, “the goal for this investment is to fund your retirement.” (Keep in mind that this is strictly for presentation here; the system pays no attention to this.)

Putting the whole rule in somewhat better English, it says:

> if the goal for this investment is to fund your retirement, and
> the number of years until you retire is less than 10,
> then the category of fund to select is the conservative growth funds.

THE INFERENCE ENGINE IN ACTION

But how can we put all that knowledge to work? Our system will proceed by chaining backward. In this case the primary goal is to determine which Category of Fund to invest in. The system starts trying to determine this by retrieving all the rules that are relevant, i.e., all rules that conclude about this attribute.

Conveniently enough, the rules in Table 1 are organized according to what they conclude about, so it's clear that rules 10 through 22 are the relevant ones. Assume that the system tries them in the order shown there. It would thus begin with Rule 10, chaining backward:

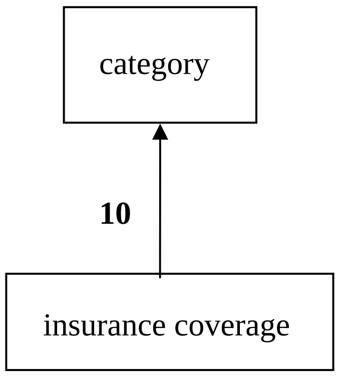


FIGURE 1

Rule 10 warns us not to invest if we don't have basic insurance coverage (i.e., use the money to buy insurance instead). But we don't know if the premise of Rule 10 is satisfied yet, that is, whether our Basic Insurance Coverage is INADEQUATE. How can the system determine that?

KNOWLEDGE BASED SYSTEMS HAVE A SIMPLE DATABASE

The system has a simple database in which it records any inferences it makes and any answers the user gives to questions. The first thing to do then, when trying to determine the value of any attribute, is to check the database to see if the answer is already there, either because the system deduced it earlier, or because the user told us earlier.

(Recording the answers also provides some common sense and efficiency. Some facts are used in more than one rule, so if the deductions and answers were not recorded, the system would deduce them all over again or ask the user all over again.)

Here's a more detailed description of what the system does to determine the value of an attribute. There are three steps to be taken:

1. Check the database to see if the answer has previously been deduced or supplied by the user. If so, that's the value to use.
2. If there's no value in the database, retrieve from the knowledge base all rules capable of deducing any value for the attribute. Try using all of those rules (i.e., for each rule, determine the truth of the premise; if the premise turns out to be true, make the conclusion.)
3. If, despite steps 1 and 2, there is still no value for the attribute, ask the user.

Note that in step 2 we might find some rules in the knowledge base, try them and discover that none of them is relevant. Thus we can get to step 3 and have to ask the user because there were no rules at all, or because there none of the rules were relevant to the current situation.

The System Retrieves All Rules About A Given Topic

In this case, since we have just begun, the database is empty so step 1 fails. To try step 2, the system looks in the knowledge base for *any* rule that can tell it *anything* about insurance coverage.

Note that even though the premise of Rule 10 specifies the condition “Basic Insurance Coverage = INADEQUATE”, the inference engine will retrieve *all* rules that conclude about insurance coverage, whether they conclude that it is ADEQUATE, INADEQUATE, or anything else.

This is done primarily to keep the system's questions focused on a specific topic until that topic has been fully explored. To see this, notice what would happen, if at this point the system retrieved only rules whose conclusions exactly matched the premise of Rule 10. In that case if it later encountered another rule whose premise said “Basic Insurance Coverage = ADEQUATE,” then it would have to take up the topic of insurance coverage all over again. The system's behavior is more focused, and more transparent, if it uses *all* the rules about a topic the first time that topic is mentioned.

The topic of insurance coverage is dealt with by rules 1, 2, and 3. After retrieving rule 1, the situation looks like this:

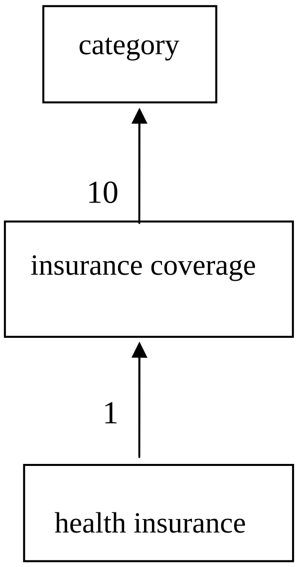


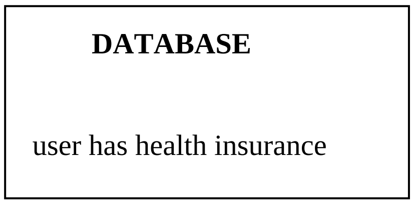
FIGURE 2

In order to determine whether it can fire Rule 1, the system needs to know whether the user Has Health Insurance. It looks in the database (step 1) and, once again, finds nothing. It looks in the knowledge base (step 2) and finds that there are no rules concluding about this topic. With no rules, it clearly can't infer the answer, so (step 3) it's time to ask.

The system will say something like:

1) Do you have health insurance?

Assume the user says “yes”; the system records this in the database for future reference.



This also means that Rule 1 *fails,* since its premise is not true. In Figure 3 we have drawn an XXX under the attribute “health insurance” to indicate that this line of reasoning failed.

The system then pushes on to try Rule 2 (Figure 3), the first clause of which asks whether the user has life insurance.

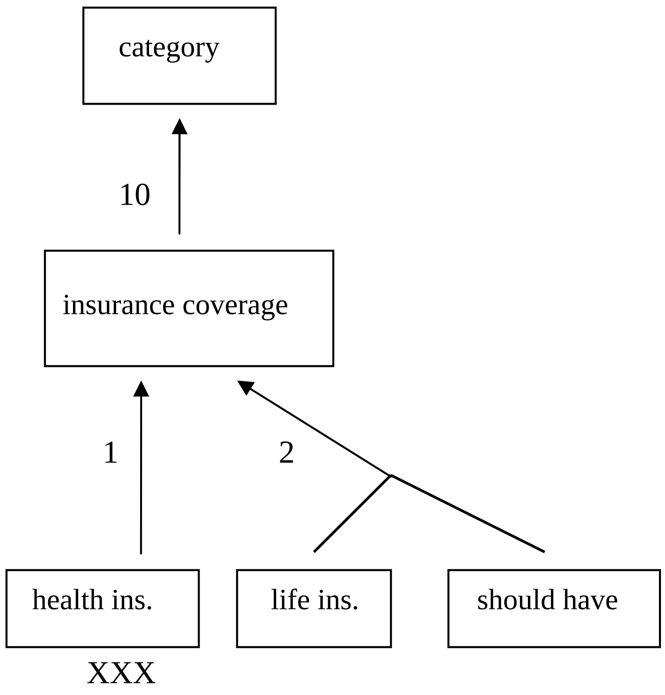


FIGURE 3

The system checks the database and finds no information there, checks the knowledge base and finds no rule that infers whether the user has life insurance, so as before it asks. Imagine that the answer is YES this time also. As a result, the first clause of Rule 2 has failed, since it says “Have Life Insurance = NO” (i.e., you do not have life insurance).

RULES MAY FAIL AFTER ONLY A SINGLE CLAUSE

Note that this has an interesting consequence: we know at this point that Rule 2 has failed. Since the premise has two clauses connected by *and*, it is clear that if the first of them is false the whole premise is false. As a result the system can ignore the rest of rule 2 and push on to the others. There is only a small saving in this case, but it is in general a useful idea to push on to the next rule as soon as we are sure the current one will fail. This avoids asking pointless questions: if, for example, the user has life insurance, then for our purposes there's no reason to ask whether he/she should have it. As a result the interaction will appear smarter. This can in some cases mean a sizable saving of effort, when there is a long chain of rules that is avoided. If, for instance, there had been a long chain of rules used to infer whether “you should have life insurance,” we would save a substantial amount of work by knowing to press on to the next rule whenever the first clause of Rule 2 is false.

Rule 3 is next and note what happens this time. The value of both attributes (Have Health Insurance; Have Life Insurance) are already available in the database, so the system doesn't have to ask (or try any additional chaining). Since both clauses turn out to be true, rule 3 succeeds, asserting (in the database) that Basic Insurance Coverage = ADEQUATE. That's the last rule concerning that attribute, so we now know its value.

Now the system can return its attention to rule 10 (which started all this). Its premise says Basic Insurance Coverage = INADEQUATE, hence it fails. Now what's next? The system was working on Category Of Fund and there are lots more rules about it, so the system continues with the next one, rule 11.

Note, by the way, what would have happened if rule 3 didn't exist. Rules 1 and 2 would have failed and we would have been left with no value for Basic Insurance Coverage. In that case the system would press on to step 3 and ask the user. If (as we might expect), the user also didn't know, and hence answered UNKNOWN, the final value for Basic Insurance Coverage would, by default, be unknown. Note that rule 10 would still fail in this case, since it is only applicable if Basic Insurance Coverage is known to be INADEQUATE.

In the first exercise, you'll have a chance to continue this simulation, seeing what the system does at each step and seeing how it comes up with its final conclusion.

TABLE 1: THE RULES

Commentary that follows the table explains some of the less obvious rules.

RULES ABOUT: adequacy of Basic Insurance Coverage

1]	*if*	Have Health Insurance = NO
	then	Basic Insurance Coverage = INADEQUATE
2]	*if*	Have Life Insurance = NO and
		Should Have Life Insurance = YES
	then	Basic Insurance Coverage = INADEQUATE
3]	*if*	Have Health Insurance = YES and
		Have Life Insurance = YES
	then	Basic Insurance Coverage = ADEQUATE

RULES ABOUT: whether you Should Have Life Insurance

4]	*if*	Married = YES or
		Have Children = YES
	then	Should Have Life Insurance = YES

RULES ABOUT: which Category Of Fund to choose

10]	*if*	Basic Insurance Coverage = INADEQUATE
	then	Category Of Fund = NONE
11]	*if*	Current Savings < 6 * Monthly Salary
	then	Category Of Fund = MONEY MARKET
12]	*if*	Investment Goal = RETIREMENT and
		Number Of Years To Retirement < 10
	then	Category Of Fund = CONSERVATIVE GROWTH
13]	*if*	Investment Goal = RETIREMENT and
		Number Of Years To Retirement > 10 and
		Number Of Years To Retirement < 20
	then	Category Of Fund = G&I
14]	*if*	Investment Goal = RETIREMENT and
		Number Of Years To Retirement > 20
	then	Category Of Fund = AGGRESSIVE
15]	*if*	Investment Goal = CHILDREN'S EDUCATION and
		Age Of Oldest Child < 7
	then	Category Of Fund = G&I

16]	*if*	Investment Goal = CHILDREN'S EDUCATION and
		Age Of Oldest Child > 7
	then	Category Of Fund = CONSERVATIVE GROWTH
17]	*if*	Investment Goal = HOME OWNERSHIP
	then	Category Of Fund = G&I
18]	*if*	Investment Goal = CURRENT INCOME
	then	Category Of Fund = INCOME
19]	*if*	Investment Goal = INVEST SPARE CASH and
		Risk Tolerance = LOW
	then	Category Of Fund = CONSERVATIVE GROWTH
20]	*if*	Investment Goal = INVEST SPARE CASH and
		Risk Tolerance = MEDIUM
	then	Category Of Fund = G&I
21]	*if*	Investment Goal = INVEST SPARE CASH and
		Risk Tolerance = HIGH
	then	Category Of Fund = AGGRESSIVE
22]	*if*	Investment Goal = INVEST SPARE CASH and
		Risk Tolerance = MEDIUM and
		Tax Bracket = HIGH
	then	Category Of Fund = TAX-FREE

RULES ABOUT: what your Life Stage is

23]	*if*	Your Age > 65
	then	Life Stage = RETIRED
24]	*if*	Your Age <= 65
	then	Life Stage = NOT-RETIRED

RULES ABOUT: what Investment Goal to select

31]	*if*	Pension = NO and
		Individual Retirement Account = NO
	then	Investment Goal = RETIREMENT
32]	*if*	Children Headed For College = YES and
		Children's Education Already Funded = NO
	then	Investment Goal = CHILDREN'S EDUCATION
33]	*if*	Own Home = NO and

Want Home = YES
then Investment Goal = HOME OWNERSHIP

34] *if* Life Stage = RETIRED
then Investment Goal = CURRENT INCOME

35] *if* (Own Home = YES or Want Home = NO) and
(Pension = YES or Individual Retirement Account = YES) and (Have Children = NO or Children's Education Already Funded = YES) and Life Stage = NOT-RETIRED
then Investment Goal = INVEST SPARE CASH

RULES ABOUT: what your Risk Tolerance is

41] *if* Enjoy Gambling = YES
then Risk Tolerance = HIGH

42] *if* Budgeting Very Important = YES
then Risk Tolerance = LOW

43] *if* Worry About Money At Night = YES
then Risk Tolerance = LOW

44] *if* Budget But Splurge Sometimes = YES
then Risk Tolerance = MEDIUM

RULES ABOUT: whether you have Children Headed For College

45] *if* Have Children = YES and
Age Of Youngest Child < 16
then Children Headed For College = YES

46] *if* Have Children = YES and
Age Of Youngest Child >= 16
then Children Headed For College = NO

47] *if* Have Children = NO
then Children Headed For College = NO.

RULES ABOUT: whether Children's Education Already Funded

51] *if* College Tuition Level = CHEAP
then Children's Education Already Funded = YES

52] *if* Children Have Scholarship = YES
then Children's Education Already Funded = YES

53] *if* Children Eligible For Loans = YES
then Children's Education Already Funded = YES

54] *if* Children Have Trust Fund = YES
then Children's Education Already Funded = YES

55] *if* Children Have Scholarship = NO and
Children Eligible For Loans = NO and
Children Have Trust Fund = NO
then Children's Education Already Funded = NO.

Commentary: Table 1.
In order to make this example something that could be done by hand simulation, the knowledge base is simple and the rules make a number of simplifying assumptions. A few examples will help make this clear.

The rationale behind rules 12-14, for example, may not be obvious.

```
12]   if     Investment Goal = RETIREMENT and
             Number Of Years To Retirement < 10
      then   Category Of Fund = CONSERVATIVE GROWTH

13]   if     Investment Goal = RETIREMENT and
             Number Of Years To Retirement > 10 and
             Number Of Years To Retirement < 20
      then   Category Of Fund =  G&I

14]   if     Investment Goal = RETIREMENT and
             Number Of Years To Retirement > 20
      then   Category Of Fund = AGGRESSIVE
```

Rule 14 indicates that we can be more aggressive if there's a long time to retirement, 13 indicates a more conservative strategy if there are fewer years to go, while 12 cautions for the most conservative in the shortest term. The rationale here is that, over the long term, aggressive funds have a higher rate of return but are also subject to wider swings in value. Hence the sooner you will need the funds, the more conservative you should be, since a market downswing at that time will hit the aggressive fund particularly hard. If, on the other hand, there's a long time to retirement, we can be aggressive and ride out the ups and downs, in the expectation that the higher payoff over the longer term will mean a higher payoff at retirement even if there is a market downswing at that point.

Note that we have captured only the end effect of this reasoning in rules 12-14. It would require a more sophisticated rule set, and perhaps a more sophisticated knowledge representation, to capture the reasoning outlined just above.

Rules 45 and 46 present a fairly simple picture of the world, indicating that all children under 16 are headed for college, anyone over 16 is not. Clearly this is a simplistic assumption, since not every youngster goes to college and occasionally someone over 16 does. You might want to think about what additional rules we might need to infer whether a child younger than 16 is in fact likely to be headed for college.

Rules 51—54 present a simple picture of college financing by assuming that it's an all-or-nothing affair: if you have any of the sources noted, then college is completely paid for (Children's Education Already Funded). This is good enough for our example, but once again a more sophisticated system would require additional knowledge.

Exercise 1: SIMULATING THE INFERENCE ENGINE

GOAL:
In this exercise you will discover how a backward-chaining rule-based system works. It will give you the opportunity to play the role of inference engine, making clear just what it does, and demonstrating how relatively straightforward the inference engine is.

A) Run the system by hand, following the procedure outlined above. Show what happens by drawing the tree of rules, by augmenting Worksheet 1 and indicating the dialog that would occur and the additions to the database by augmenting Worksheet 2

Use the following data:
The user

1. is 42 years old
2. has health and life insurance
3. has current savings balance of $20000
4. has a monthly salary of $3000
5. not covered by a pension plan, does have IRA
6. has one child, age 12, who does not have scholarship or a trust fund, is not eligible for a loan, and would like to attend a school with expensive tuition
7. does not currently own a house and would like to.

NOTE: For this and all other exercises, if you believe there are facts missing from this list that you need in order to do the problem, feel free to indicate what facts are missing and what you assumed true for the sake of the exercise.

IMPORTANT: Remember that all answers are to be written on the separate answer sheet.

B) How would the program respond if, in answer to the question "do you have an IRA", you had answered "WHY"?
(Hint: recall that the system displays the rule currently in use)

Do you have an IRA?
** WHY

C) How would the program respond to the question shown below?
(Hint: recall that the system displays the "audit trail" of the rules used to reach that conclusion, i.e., all the rules that lead to that specific answer.)

HOW DID YOU DECIDE THAT THE INVESTMENT SHOULD BE CONSERVATIVE GROWTH FUND?

D) How would the program respond to the question

HOW DID YOU DECIDE THAT THE INVESTMENT SHOULD BE GROWTH AND INCOME?

Notice that the system is performing as a decision support tool: it provides two answers (conservative growth, G&I) and is prepared to indicate the rationale for each (the line of reasoning that lead to each). The final decision about what to do is still up to the user.

Exercise 2: DEALING WITH UNCERTAIN RULES

GOAL:
In this exercise you will learn how the system deals with rules that are inexact, i.e., those with a degree of belief that is less than certain. The rule set will be augmented to include strength of belief and you will do several short exercises to explore one model of how uncertain reasoning can work.

We will now extend the language to allow each rule to have associated with it a degree of certainty, expressed as a number between zero and one. Consider rules 15 and 17, which now become

15] *if* Investment Goal = CHILDREN'S EDUCATION and
Age Of Oldest Child < 7
then Category Of Fund = G&I (.7)

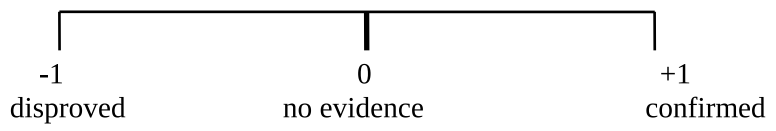
17] *if* Investment Goal = HOME OWNERSHIP
then Category Of Fund = G&I (.9)

Note that the only difference is that the conclusion now has a number indicating its strength.

What happens when both rules are applicable? We are essentially asking, What does “some evidence” (.7) plus “strong evidence” (.9) add up to? There are many schemes for doing this. One simple scheme (chosen for the Mycin system) works as follows.

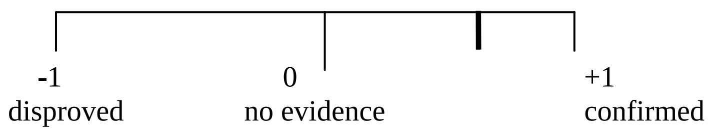
A fact like “the investment category should be G&I” is assigned a degree of belief that varies from -1 (definitely false, completely disproved) to +1 (definitely true); see below. All facts start out with a degree of belief of 0 (i.e., no evidence one way or another). Every rule that fires contributes to increasing (or decreasing) our strength of belief in a fact, by moving us some percentage of the way from where we are toward definiteness.

The investment category should be G&I



To see how this works let's consider what happens when rules 15 and 17 are both relevant. Suppose rule 15 fires first. Initially our degree of belief in the fact “the investment category should be G&I” is 0. Rule 15 tells us to move .7 of the distance from where we are (0) toward definiteness, leaving us with a degree of belief of .7

The investment category should be G&I

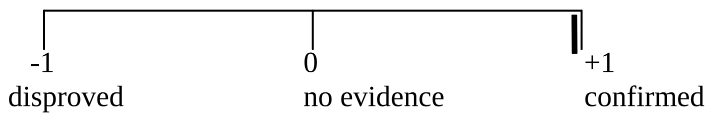


Later rule 17 will fire, telling us to move .9 of the way toward definiteness. Then since

.7 + (.9 * (1-.7)) = .97

we end up at .97. This approach gives us one way to add "evidence" and "strong evidence".

The investment category should be G&I

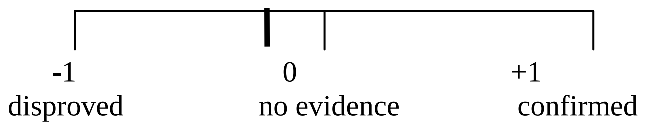


If we had additional rules recommending a G&I fund, our degree of belief would continue to increase.

At this point you might be tempted to say, "But wait, we're already at .97, within a hair's distance of being `confirmed'. Why bother to run additional rules? Aren't we sure enough?"

The answer is that we might also have rules indicating that "there is weak evidence that the investment category should be G&I (-.2)". Notice that the certainty is now negative; this is the way of saying "should *not* be G&I." The mixture of positive and negative evidence is handled by collecting evidence against the fact using the same formula as above. That is, we collect the evidence against the conclusion on a separate scale, this time moving toward the *left*.

The investment category should be G&I



When the system has tried all the relevant rules (i.e., all rules that conclude one way or the other about what category to invest in), the final degree of belief is the evidence in favor, less the evidence against. With a single rule of strength .2 arguing against this conclusion, we would end up with a degree of belief of .77, since .97 - .2 = .77.

We now need just one small additional piece of mechanism: the process described above is carried out independently for each value that the rules suggest. That is, all the rules that conclude that the category should be G&I combine as above, and we do the same for all the rules that conclude that category should be AGGRESSIVE, CONSERVATIVE, INCOME, etc. The end result is a collection of opinions indicating how strongly to believe in each of the possible categories.

A) Using the scheme outlined above, what would be the resulting degree of belief from the following sequence of rules (assume all are relevant, and note that some certainties are negative):

I) *If ... then* investment should be G&I .6
II) *If ... then* investment should be G&I –.2
III) *If ... then* investment should be G&I .8
IV) *If ... then* investment should be G&I –.45

(B) What happens if the rules fire in the order (IV) (II), (III), (I)? Does this seem intuitively reasonable? Why, or why not?

COMMENTARY
There is one other place that uncertainty can arise: in answering a question, the user may not be absolutely certain. So for instance in the dialog we might see:

```
Do you currently own your own home?
** NO

Are you interested in owning a home?
** YES .7
```

Here the user has qualified the answer to the second question with a .7, indicating less than complete certainty. By convention, all answers that are not qualified with a strength of belief are assumed to be certain, i.e., have a certainty of 1.0.

Let's see how this less than certainly in the user's answer affects the system's behavior.

These two questions are prompted by Rule 33:

```
33]     if      Own Home = NO and
                Want Home = YES
        then    Investment Goal = HOME OWNERSHIP (.8)
```

(Note that Rule 33 now has a degree of certainty associated with it.)

We can accommodate an uncertain answer by augmenting our rule evaluation scheme slightly. In evaluating the truth of the premise (the “if” part of the rule), we want to allow it to be only “partly” true.

In evaluating the first clause, since the user said “no”, it is (definitely) true that “the user does not currently own a home”. In evaluating the second clause, it is only “.7 true” that the user is interested in owning a home. So the first clause is definitely (1.0) true while the second is .7 true. What does the whole premise evaluate to?

A long-standing convention in inexact reasoning suggests that we should treat “and” as minimizing and “or” as maximizing. This comes from the common observation that if we need *all* elements of the premise (as with and), then it is safe to take the overall value to be the minimum, while if we can get by with *any* element (as with or), then we can take the value to be the maximum.

Hence the value of the premise of rule 33 comes out to be .7 in this case. Then, finally, we modify the strength of the conclusion by the strength of belief in the premise, so that the "effective strength" of rule 33 is .56 (i.e., .7 * .8). So in this case, the system would conclude that "the investment goal should be home-ownership" with strength .56.

C) What would the strength of conclusion be if the user had answered .9?

Note that with all of this machinery in place, we can now have degrees of belief that propagate along chains of reasoning:

If the user answers "NO .7" as above, rule 33 will conclude "goal = home-ownership" .56

Then when we come to rule 17, we will discover in the database "goal = home-ownership .56", evaluate the truth of the premise to .56, and conclude "investment = G&I .504" (since .9 * .56 = .504).

D) Work through the certainties that would be computed by the three rules below.

(Note that this example exercises all of the machinery we need for certainties. We have abbreviated the premises and conclusions to single letters, to simplify things and focus on the issue of modeling uncertainty.)

(I) *If* A and B *then* E .7
(II) *If* C or D *then* E –.45
(III) *If* E *then* X .8

Assume that the user answers the questions as follows:

A?
** YES .8
B?
** YES .9
C?
** YES .9
D?
** YES .5

How strongly do we now believe X?

F) Work through the certainties that result from the following four rules, assuming that the user's answers to F, G, H, J, and K are all YES with certainty 1.0.

(I) *If* F *then* Y .7
(II) *If* G *then* Y .8
(III) *If* H *then* Y .5
(IV) *If* J *then* Y .6

What is the strength of belief in Y at this point?

Imagine that we had the two additional rules (V) and (VI) shown below, and that J1 and J2 are true. So they are both fired. Would we be absolutely certain in our belief in Y?
(V) *If* J1 *then* Y .7
(VI) *If* J2 *then* Y .8

Now imagine that rule (VII) fires. Now how strong is our belief in Y?
(VII) *If* K *then* Y 1.0

What does this demonstrate about the strength of belief accumulating function?

COMMENTARY
This machinery for handling uncertainty sounds plausible and might make you think that the issue is settled. In fact uncertain reasoning has a long history of being a rather difficult problem. The scheme reviewed here is one plausible way to approach it, but is certainly not the only way to proceed. It has one important drawback and two virtues. The drawback is that we have no formal (i.e. mathematical) foundation for believing that the formulas shown above are a reasonable way to proceed. They were invented to try to capture some intuitions about this task, but have no solid foundation in theory. As a consequence, while they seem to give plausible results for the simple examples shown, we have no guarantee that some collection of rules in the future might result in a completely counter-intuitive result.

The virtues of this approach lie in its simplicity (it is easy to understand and fast to compute) and its intuitive appeal. Given the importance of transparency in these systems, the method of accumulating degrees of belief should be chosen carefully, to avoid clouding up the transparency of the rest of the system. This approach appears transparent and intuitive because it captures three basic characteristics.

- The order in which evidence is collected is unimportant, i.e., the formula is commutative. This seems to match our intuition about the strength of an argument being based on all the evidence, without respect to the order in which the evidence is encountered.
- As positive evidence is collected, we get closer and closer to certainty, but never actually reach certainty unless a rule is fired that does make a conclusion with certainty (1.0).
- It balances out positive and negative evidence, mimicking our natural tendency to “weigh” evidence.

These basic characteristics are probably the most important issue. *Any* formula that captured these would likely do as well as the one we have explored. There is likely very little to be gained by dwelling on the details of exactly what formula to use. Experience and some informal experimental evidence makes us believe that most of the power in this paradigm comes from having rules that make the inferences at all, i.e., knowing which facts lead to which conclusions. Much less of the power comes from the exact strength of the conclusion. One study was done with a set of rules that were initially expressed using a certainty measure with nine different values. When those values were “rounded off” so that every rule said either “possible” or “certain” the performance of the system actually got slightly better.

Keep in mind also that this general issue of uncertain reasoning is in fact a problem of significant mathematical and philosophical depth. The schemes we have explored here are simple engineering solutions that seem to work, but offer little insight into those deeper issues. For practical applications, though, they appear to be adequate.

Exercise 3: RULES ARE SUPPOSED TO CAPTURE THE LOGIC OF THE SITUATION

GOAL:
This exercise demonstrates what we mean by saying that rules are supposed to capture just the logic of the situation. You will use the augmented rule set from Exercise 2 to demonstrate some important properties of rules that have been written properly.

For reference, Table 2 has an updated listing of the rules, giving each a strength of belief.

A) Show the tree and dialog that would result from the following situation:

The user

1. is 70 years old
2. does not have health insurance
3. does have life insurance
4. has a current savings balance of $20000
5. has a monthly salary of $3000
6. is covered by a pension plan
7. has no children
8. currently owns a home

This time you should use the degree of certainty machinery we just reviewed. (Note that you have less work to do on this problem because we have given you the entire tree. You only have to traverse it, and show the dialog and conclusions.) The backward chaining machinery is by now familiar, but there is still work to do in computing strengths of belief.

What is the system's recommendation?

B) Now suppose rule 10 were moved down the page to the end of the list of rules concluding about fund category (i.e., after rule 22). Describe what difference this makes in the trace of the reasoning process and in the dialog. Also, does this change the final result?
(As in earlier exercises, this should not require a great deal of work if you think carefully and rely on what you did in the previous part of this exercise.)

COMMENTARY
One of the things we discovered in Exercise 2 was that the system's final answer was independent of the order in which the rules fired. This is an important property of the system for two reasons.

First, it matches our intuition that the final result should not depend on the order in which evidence is collected.

Second, it reinforces a very important (and often abused) property of rules:

- They are designed to capture the *logic* of the situation, and that's all.

All we are trying to tell the system is *how to reason about the task at hand.*

In the current situation, we might be tempted to think that asking about health insurance first was important, because if the user doesn't have it, then it's clear what to do: don't invest in the market, buy health insurance. We are tempted to tell the system: “*first* check for health and life insurance, then if the user has those, go on to ask the rest of the questions”.

But rules aren't designed to capture question order or conventions of data gathering. Neither is backward chaining.

- Rules are designed to capture the logic of the situation in single-step inferences.

- Backward chaining ensures that all questions are asked *when they are logically relevant,* i.e., when they contribute to a line of reasoning.

Notice that in the current example, the system's recommendation that we not invest (Category = NONE 1.0) arises eventually, even if the question ordering is not optimal.

This is very important. In fact, the first and most important goal of knowledge engineering is to *get the knowledge correct*.

Interestingly enough, the *second* most important goal of knowledge engineering is *get the knowledge correct*. This is often so difficult you need to put aside *all* other considerations for the time being. Eventually, of course you want to deal with issues of user interface, question ordering, and so forth. Those issues matter, but they must not deflect the knowledge engineering task.

It is very important to keep this in mind when writing rules. Unfortunately this is often difficult to do, primarily because traditional program design typically proceeds from a scenario of how the program should behave *from the user's point of view*. Here we want instead to focus on *the domain knowledge alone* and leave the question order to be handled by the system.

This may not produce the most familiar pattern of questions, but it will ensure that the rules capture just the underlying logic of the situation, without mixing in information about data gathering conventions (or anything else).

If we fail in this, and end up writing rules that attempt to force a certain question ordering or that depend on subtleties like the order in which they are retrieved from the knowledge base, we end up producing rules whose knowledge is considerably more obscure. The reason is simple enough: getting the rules to express just the logic is hard enough; forcing them to do double duty (logic plus question ordering) is overloading them. Such rules are not nearly as explicit, clear, or transparent. They are harder to understand, produce inferior explanations, and are much more difficult to debug.

One good test of whether we are following this approach is to see how many rules live up to the very important criterion that:

- Every rule should make sense when viewed in isolation.

That is, if we are careful in writing rules, each of them alone will be sensible. If a rule seems odd when viewed in isolation, it is likely that we have not been successful in sticking to the logic alone.

Note that this is largely true of the current set of rules: each of them does in general make sense. One consequence of this is that no matter how they are ordered, the result is correct.

ONE MORE ADDITION TO THE INFERENCE ENGINE

Our current system will in fact ask some useless questions: in example (A) above, the system will go on to ask about savings, pension, etc., when it's clear that the user shouldn't invest.

We can fix this with one simple augmentation to our system. To understand that augmentation, we need to consider whether an attribute should be labeled as “single-valued” or “multi-valued”. By single-valued we mean that it can have only one value that has strength of belief 1.0; multi-valued attributes can have multiple such values. Consider the attribute Investment Goal, for instance. If we label it multi-valued we are in effect saying that it is possible to be absolutely certain that we should invest for two different goals; e.g., absolutely sure that we should invest to fund retirement and absolutely sure that we should invest to fund education. This means in effect that the system can identify two different investment objectives and be sure about both. If we label it single-valued, it means we're telling the system that it can select only a single value to be sure about.

Note that neither of these the “right” one; they each say something different about the task. In one case we're telling the system it's ok to consider multiple investment goals with certainty. In the other case we're saying that if we identify one goal with certainty, that's it; all the other values are in effect ruled out.

We can change the inference engine so that it takes this information into account and stops attempting to deduce the value of any single-valued attribute if it ever deduces the value of that attribute with certainty. The intuition here is simple: if you are *absolutely certain* of an answer, and there can be only one answer, there's no point in continuing to try to additional rules.

Note that this reflects back a certain responsibility to the knowledge engineer: be very sure that you really mean *certain* when you write a rule with a certainty of 1.0 about a single-valued attribute.

With this modification the revised inference engine using the rules in Table 2 would halt as soon as it fired rule 10. If this happens first (i.e., if we start by retrieving rule 10), then the system will stop the consultation very quickly, with no pointless questions. (You will find that GRASS works this way.)

If we move rule 10 to the end of the list, as in exercise 3b, the revised inference engine will still go through a long list of questions, motivated by the other rules. However, note two things about this:

- it “couldn't have known not to,” in much the same sense as a human who didn't happen to get around to asking about health insurance until later. Until we ask about health insurance, it isn't clear that the other questions don't matter.

- as soon as it encounters and fires rule 10, it stops trying to infer what to invest in, because the answer is now known with certainty.

One final note: in this case the whole consultation ends whenever rule 10 fires, because it happens to determine the primary goal -- what to invest in. In general the revised system will stop attempting to determine the current attribute, but may then go on with its work in determining the value of other attributes in the tree.

TABLE 2: THE REVISED RULE SET

RULES ABOUT: adequacy of Basic Insurance Coverage

1]	*if*	Have Health Insurance = NO
	then	Basic Insurance Coverage = INADEQUATE 1.0
2]	*if*	Have Life Insurance = NO and
		Should Have Life Insurance = YES
	then	Basic Insurance Coverage = INADEQUATE 1.0
3]	*if*	Have Health Insurance = YES and
		Have Life Insurance = YES
	then	Basic Insurance Coverage = ADEQUATE 1.0

RULES ABOUT: whether you Should Have Life Insurance

4]	*if*	Married = YES or
		Have Children = YES
	then	Should Have Life Insurance = YES 1.0

RULES ABOUT: which Category Of Fund to choose

10]	*if*	Basic Insurance Coverage = INADEQUATE
	then	Category Of Fund = NONE 1.0
11]	*if*	Current Savings < 6 * Monthly Salary
	then	Category Of Fund = MONEY MARKET 1.0
12]	*if*	Investment Goal = RETIREMENT and
		Number Of Years To Retirement < 10
	then	Category Of Fund = CONSERVATIVE GROWTH .8
13]	*if*	Investment Goal = RETIREMENT and
		Number Of Years To Retirement > 10 and
		Number Of Years To Retirement < 20
	then	Category Of Fund = G&I .8
14]	*if*	Investment Goal = RETIREMENT and
		Number Of Years To Retirement > 20
	then	Category Of Fund = AGGRESSIVE .8
15]	*if*	Investment Goal = CHILDREN'S EDUCATION and
		Age Of Oldest Child < 7
	then	Category Of Fund = G&I .8

16] *if* Investment Goal = CHILDREN'S EDUCATION and
Age Of Oldest Child > 7
then Category Of Fund = CONSERVATIVE GROWTH .8

17] *if* Investment Goal = HOME OWNERSHIP
then Category Of Fund = G&I .9

18] *if* Investment Goal = CURRENT INCOME
then Category Of Fund = INCOME .9

19] *if* Investment Goal = INVEST SPARE CASH and
Risk Tolerance = LOW
then Category Of Fund = CONSERVATIVE GROWTH .9

20] *if* Investment Goal = INVEST SPARE CASH and
Risk Tolerance = MEDIUM
then Category Of Fund = G&I .8

21] *if* Investment Goal = INVEST SPARE CASH and
Risk Tolerance = HIGH
then Category Of Fund = AGGRESSIVE .8

22] *if* Investment Goal = INVEST SPARE CASH and
Risk Tolerance = MEDIUM and
Tax Bracket = HIGH
then Category Of Fund = TAX-FREE .9

RULES ABOUT: what your Life Stage is

23] *if* Your Age > 65
then Life Stage = RETIRED .8

24] *if* Your Age <= 65
then Life Stage = NOT-RETIRED .8

RULES ABOUT: what Investment Goal to select

31] *if* Pension = NO and
Individual Retirement Account = NO
then Investment Goal = RETIREMENT 1.0

32] *if* Children Headed For College = YES and
Children's Education Already Funded = NO
then Investment Goal = CHILDREN'S EDUCATION .8

33] *if* Own Home = NO and
Want Home = YES
then Investment Goal = HOME OWNERSHIP .8

34] *if* Life Stage = RETIRED
then Investment Goal = CURRENT INCOME .9

35] *if* Own Home = YES or Want Home = NO, and
Pension = YES or Individual Retirement Account = YES, and
Have Children = NO or Children's Education Already Funded = YES
, and Life Stage = NOT-RETIRED
then Investment Goal = INVEST SPARE CASH .8

RULES ABOUT: what your Risk Tolerance is

41] *if* Enjoy Gambling = YES
then Risk Tolerance = HIGH .8

42] *if* Budgeting Very Important = YES
then Risk Tolerance = LOW .8

43] *if* Worry About Money At Night = YES
then Risk Tolerance = LOW .8

44] *if* Budget But Splurge Sometimes = YES
then Risk Tolerance = MEDIUM .8

RULES ABOUT: whether you have Children Headed For College

45] *if* Have Children = YES and
Age Of Youngest Child < 16
then Children Headed For College = YES 1.0

46] *if* Have Children = YES and
Age Of Youngest Child >= 16
then Children Headed For College = NO 1.0

47] *if* Have Children = NO
then Children Headed For College = NO. 1.0

RULES ABOUT: whether Children's Education Already Funded

51] *if* College Tuition Level = CHEAP
then Children's Education Already Funded = YES 1.0

52] *if* Children Have Scholarship = YES
then Children's Education Already Funded = YES 1.0

```
53]    if      Children Eligible For Loans = YES
       then    Children's Education Already Funded = YES   1.0

54]    if      Children Have Trust Fund = YES
       then    Children's Education Already Funded = YES   1.0

55]    if      Children Have Scholarship = NO and
               Children Eligible For Loans = NO and
               Children Have Trust Fund = NO
       then    Children's Education Already Funded = NO. 1.0
```

Exercise 4: KNOWLEDGE ACQUISITION

GOAL:
One of the interesting benefits of organizing and representing knowledge as we have is the ability to add more of it in a way that is both relatively simple and clearly concerned with the knowledge itself, rather than with writing code. In this exercise we'll see how the current knowledge base can be augmented with several new rules that make it just a little bit more knowledgeable about one part of the problem.

Currently the system knows only one thing about tuition. Rule 51 in the current set says:

51] *if* College Tuition Level = CHEAP
then Children's Education Already Funded = YES 1.0

Since we have no way to infer whether the tuition is likely to be cheap or expensive, in all cases we simply ask. Wouldn't it be nice if we could make the system just a little smarter about this. Is there any knowledge we can bring into bear that will tell us whether the tuition is likely to be cheap or expensive?

A) Try writing two rules to help out. Do this by filling in the blanks below. Use anything you know that is a good plausible guess. Remember that we're trying to come up with good guesses here, not absolute, hard and fast answers.

If ____________________
then the tuition is likely to be EXPENSIVE.

If ____________________
then the tuition is likely to be CHEAP.

Two sample rules that deal with this issue are:

```
60]    if      College Category = IVY-LEAGUE
       then    College Tuition Level = EXPENSIVE  .9

61]    if      College Category = IN-STATE
       then    College Tuition Level = CHEAP  .5
```

Assume that these two new rules have been added to the knowledge base. In this exercise we will explore how the system would behave now. We will do this by running the same example as in Exercise 1, knowing the additional fact that the school is in an in-state school.

The following hints will make this easier:

Use the simpler inference engine we used there, i.e., the one that didn't have strengths of belief (and hence used all rules regardless of which ones succeeded). This reduces the unimportant detail for this example.

Note that the only change is that the system has rules to infer college tuition level, and will try these rules before asking about tuition. The only part of the worksheet from Exercise 1 that you need to examine is the tree of rules that starts with rule 12, moves back through rule 32, and ends at rule 51.

B) Why is it true that we need look at only this one part of the tree? We added rules 60 and 61 to the knowledge base, and gave no further guidance, yet without re-running the whole example we can say with assurance that the only thing that changes is the system's behavior when the issue of tuition level arises. Why is this so?

C) Using the hints above, show the revised sub-tree of rules that result and show the parts of the dialog that differ from the example of Exercise 1.

Notice a very important thing about what we've accomplished: we made the system a little smarter about one very simple part of the problem, and all it took was adding a new subset of rules about an existing topic. Nothing else had to change, no re-programming is necessary; the inference engine in particular was never touched. In this simple case at least, all we had to do was write down the relevant chunks of knowledge. The system itself retrieved them when they were needed, i.e., when we got to dealing with the topic of tuition level. This time when the inference engine looked at that topic, it found a set of relevant rules.

D) If the rules you made up in part (a) were different from those shown in part (b), add those rules to the knowledge base and show how the consultation will change. If they were the same as the ones shown in (b), write two new ones, add them to the knowledge base, and repeat exercise (c) using your revised knowledge base. (Show below the revised sub-tree and the parts of the dialog that change.)

Exercise 5: ASKING VS. INFERRING

GOAL:
This exercise will also bring out an interesting apparent conundrum: when we add more knowledge to the system, it seems in one sense to get “dumber.” As we'll see, the way out of this problem lies in the simple model of backward chaining we've been using so far. We’ve said that, “the system asks a question of the user when it runs out of rules to backchain”. In the introduction to the first exercise we augmented that slightly, saying that there is in fact a 3 step process: check the database, check the knowledge base, then ask.

As it turns out, this *still* isn't the complete answer. In this exercise we add one last simple but useful addition. We will see that this addition allows us to add more knowledge to the system and have that knowledge applied only when it is appropriate. What does “appropriate” mean?

KNOWING WHEN TO ASK AND WHEN TO INFER
Assume that the system has the two rules shown earlier for inferring the likely expense of tuition:

60] *if* College Category = IVY-LEAGUE
then College Tuition Level = EXPENSIVE

61] *if* College Category = IN-STATE
then College Tuition Level = CHEAP

Suppose we add the following rules to the knowledge base, simply as a way of adding a little more knowledge about this topic.

62] *if* College Setting = RURAL
then College Tuition Level = CHEAP

63] *if* College Setting = URBAN
then College Tuition Level = EXPENSIVE

64] *if* Very Competitive College = YES
then College Tuition Level = EXPENSIVE

65] *if* Famous College = YES
then College Tuition Level = EXPENSIVE

66] *if* College Faculty = PRESTIGIOUS
then College Tuition Level = EXPENSIVE

A) Run the same example as in Exercise 1, knowing the additional facts that:
- the College Category is OUT-OF-STATE (i.e., neither IVY-LEAGUE nor IN-STATE)
- the College Setting is SMALL-TOWN (i.e., neither RURAL nor URBAN)
- the college is not competitive, not famous, and doesn't have a prestigious faculty
- the College Tuition Level is CHEAP (i.e., you do in fact know the tuition level).

Use the same hints as in the previous example, namely, use the simple inference engine and note that once again we simply have a new subset of rules added at one part of the tree.

Show the part of the reasoning tree that has changed, and the new questions that result in the dialog.

Read over the dialog you've generated. It seems a bit labored, a bit silly to be asking all those questions, especially when the user already knows the answer that the system is working so hard to infer.

We seem to have encountered a paradox. We've added more knowledge, yet the second dialog seems to be more labored than the first. Why is this so? That is, why does it seem to be strange to be asking all the questions motivated by rules 60-66?

Simply put, the problem is that the system we have *always* tries to infer the answer, using all available rules, before it will ask the user. In general this makes good sense, since it means the system is doing most of the work and asking relatively few questions of the user, only requesting information that is strictly necessary.

This example shows, however, that this *isn't* always the right thing to do. Sometimes the information needed is already known to the user and yet difficult for the system to infer. It would be useful if the system could distinguish between these two situations.

We can make this distinction by knowing one more thing about attributes. We can indicate that the appropriate strategy for determining the value of an attribute is to “ask first”, that is, it makes sense to ask the user *first*, and only try rules if that fails. Hence for some attributes we wish to reverse the order of steps 2 and 3 given previously, so the system will:

1. check the database to see if the answer is already known

2. ask the user

3. if both of those fail, check the knowledge base to see if there are any rules capable of deducing the value of the attribute

B) Do the example above again, this time as it would appear if Tuition is marked as “ask first”. As before, all you need concentrate on is the part of the tree and dialog that changes. Show how the revised part of the reasoning tree would look and how the dialog would change.

C) How does the dialog change if the user responds “unknown” to the question “What is the college tuition level?” Show the sub-tree and dialog one last time.

Be sure to turn in your answers on the separate Answer Sheet.