Expert Systems For Beginners

Dr. Marc P. Lynn
John Carroll University
October, 2012

Mangers face variety of problems and make different types of decisions all day long, and sometimes they need to ask people with more expertise than themselves to help them make good decisions. If a particular problem/decision involves simply plugging know values into an equation, an expert may not be required as long as the data is available and the equation or clear statement of the appropriate policy is available. For example, if your job is to place restocking orders every week, and the policy is simply to compare the quantity on hand to the suggested minimum stocking quantity (which has already been determined), all you need to know is your current inventory, the minimum stocking quantity for each product, and how to subtract one from the other. You can use a calculator or spreadsheet program and be certain of obtaining the right answer. Of course, you could use an expert system, but why bother as long as you have access to the required data and second grade math skills? This does not imply that such decisions are unimportant to a business, but simply that they do not require significant expertise or an unusual amount of experience in a particular domain.

Let's modify this problem/decision a bit to demonstrate where an expert system approach may be of value. What if our general rule is to reorder to the suggested minimum stocking quantity, but we have an employee, Joe, who has been with our company for many years who is familiar with certain sales trends and indicators that have over time been associated with stockouts or overstock situations. Joe knows from experience that when a certain product (eg. Product A)shows sales spikes, sales of certain other products (eg. Products B and C) typically spike shortly thereafter. Joe also knows that if this happens around the winter holiday season, it is often difficult to get deliveries of Product B. He has developed a "rule of thumb" that when this happens, he orders to levels of 25% above the normal minimum restocking quantity for Product B unless there is a competitor offering special price discounts

on Product B at the time. However, if Joe sees that there is an overstocking of Product B, he will suggest discounting Product A just before the holiday season in order to move out the excess Product B.

These "rules of thumb," or heuristics, are things that Joe has learned from experience. As long as he is available, and the data he needs is available, he can make decisions that may not be consistent with the basic policy ("simply to compare the quantity on hand to the suggested minimum stocking quantity") but will keep the customers happy and improve the bottom line. But what if Joe is not available? He might be busy, retired, promoted or on vacation. How will the less experienced person make the best decision? Eventually they may learn from experience, or hope that Joe gave them a full briefing and that they remember everything he said.

If the company had created an **expert system** using Joe as the domain expert, the new person could consult the expert system when restocking decision must be made. The system would ask them a few questions and then make a suggestion that would be entirely consistent with the one(s) Joe would have made if he were there and had the same basic data. The company could have a policy that the system's suggestion(s) be followed to the letter (an example of a decision making system) or allow the new person to use the system's suggestion(s) as part of their decision making process, but assume responsibility for any final decision(s) themselves (an example of a decision support system).

In many large organizations, there are often several people around with plenty of experience to help in such cases, but in smaller companies or startups, resources may be limited. In fact, sometimes the expertise may not exist. Hiring a consultant every time decisions must be made outside of the realm of employees' expertise can be unreasonable, but developing an expert system that even inexperienced employees can consult may be a workable option.

Let's now take a closer look at just what an expert system is, and how expert systems are developed.

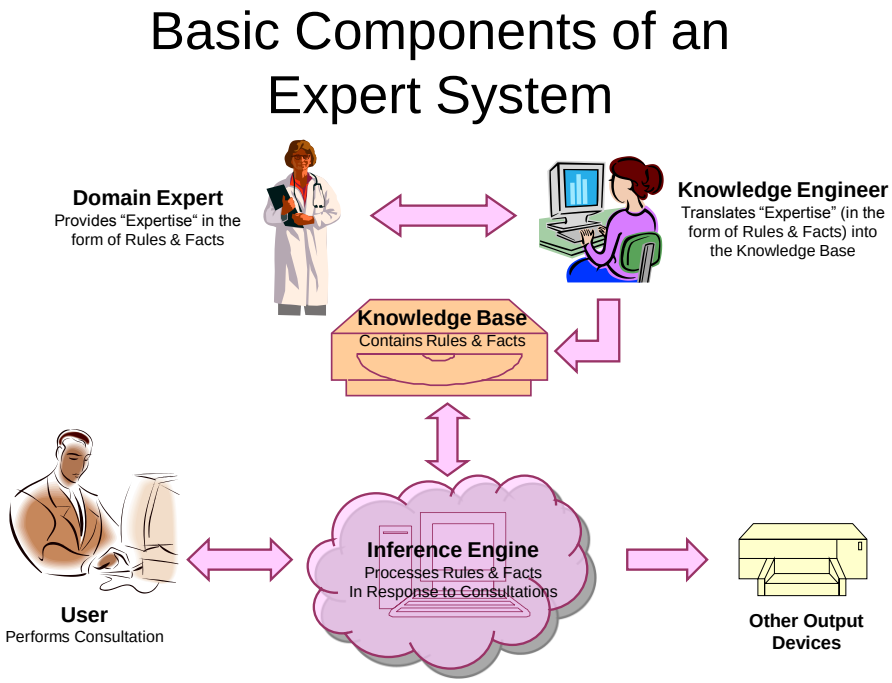
What is an EXPERT SYSTEM?

An expert system (ES) is an interactive computer-based decision tool that uses both facts and heuristics ("rules of thumb") based on knowledge acquired from an expert to provide advice or make decisions on problems falling within a specific domain.

Characteristics of expert systems typically include the following:

- **Goal oriented:** expert systems deliver answers to very specific questions that represent the *goals* of the interview: they aren't focused on abstract or theoretical information.
- **Efficient:** requests for new information consider earlier responses – minimizes requests for irrelevant input.
- **Adaptive:** may provide alternate paths allowing for deduction of sufficient facts to provide useful advice.
- **Able to deal with uncertainty:** by combining several pieces of uncertain information, may still be able to make strong recommendations.

Able to explain information requests and suggestions: justification for each question asked along with a detailed explanation of the reasoning that led to any recommendations is available.



Obtaining Rules & Facts

Domain Expert
Provides "Expertise" in the form of Rules & Facts

Knowledge Engineer
Translates "Expertise" (in the form of Rules & Facts) into the Knowledge Base

A **Knowledge Engineer (KE)** interviews a **Domain Expert (DE)** to gather rules and relevant facts the expert uses in making decisions. For example, if the domain involves selecting an appropriate drug for a patient with an infection, the expert might provide the following:

•if the infection is bacterial, I would consider antibiotic "X",
•If the patient is allergic to a particular drug, I would not prescribe it,
•If the patient has no fever, I would not prescribe an antibiotic.

The knowledge engineer must take this information and construct logical statements (that can be entered into a **Knowledge Base (KB)** – discussed on a later slide). That may result in something like this:

If infection = bacterial AND fever = Yes AND allergy-X = No Then Prescribe-X = Yes

This process can take quite a bit of time (on the part of both the DE and the KE. The KE must understand enough so that misrepresentations and ambiguities are prevented, and must be competent in the construction of logical statements.

Obtaining Rules,Facts & Attributes

Domain Expert
Provides "Expertise" in the form of Rules & Facts

Knowledge Engineer
Translates "Expertise" (in the form of Rules & Facts) into the Knowledge Base

Given the following information from the DE:

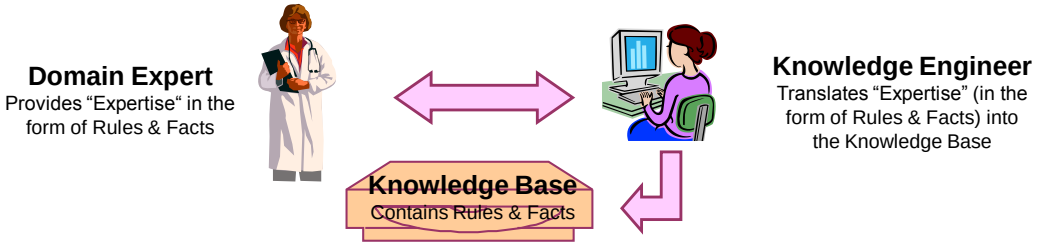
•if the infection is bacterial, I would consider antibiotic "X",
•If the patient is allergic to a particular drug, I would not prescribe it,
•If the patient has no fever, I would not prescribe an antibiotic,

And the following logic statement from the KE:

If infection = bacterial AND fever = Yes AND allergy-X = No Then Prescribe-X = Yes,

The KE now must realize that information about a patient's fever and allergies are required to determine if this Rule can be applied in a specific case. The KE has to ask the DE how one may obtain this information. For example, can we simply ask the patient if they have a fever or an allergy? Will they know? Who will know, and how will we get this information? You might now begin to see how difficult this process can become. Our **Goal** might be to determine whether to prescribe antibiotic "X", but we now have **Sub-goals** which include "determine if patient has a fever" and "determine if patient is allergic to X." ("Fever" and "Allergic to X" may be considered **attributes** here.) If the ES is to be used by the examining physician, they may be able to provide this information. If it is to be used by the patient directly, this information may not be readily available. The KE must understand the context in which the Expert System **(ES)** will be used.

Developing the Knowledge Base

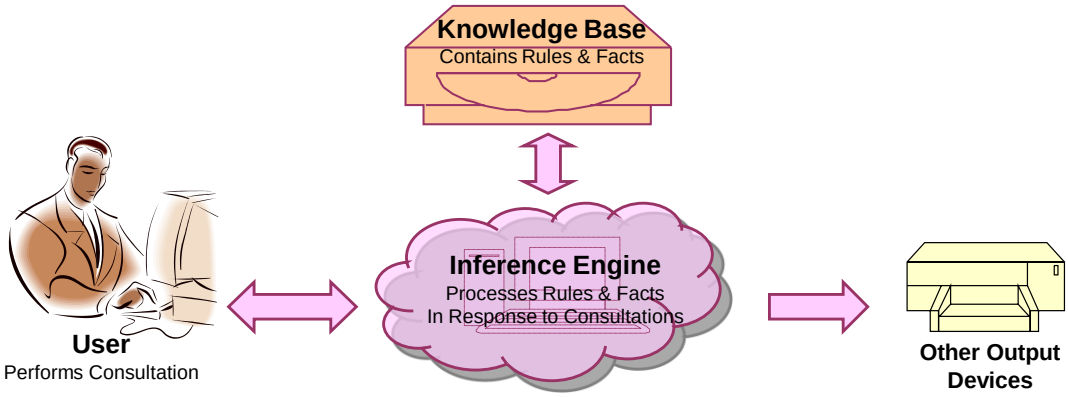


As the KE obtains the information from the DE, it is necessary to load this information into the **Knowledge Base (KB)**. The KB is not just a database. It must contain logical statements or rules along with facts/data. Most expert systems are currently designed with modules that help the KE develop the KB by providing user interfaces to facilitate the required tasks. Rules can be entered as **"If-Then-Else" statements**, and goals & sub-goals can be identified. Most systems also provide one or more ways of dealing with **uncertainty**, and this is very important in developing many KBs. For example, if we are required to "know" if the patient has a specific allergy, and the patient has never exhibited any indication of the allergy, do we know for sure that they will not exhibit it in the future? The answer is NO, but does this mean we should assume the allergy is present? One way ESs deal with these situations is to provide the ability to enter a **"confidence factor"** in association with a data attribute, fact or rule. Instead of simply having Allergy-X = Yes or No, we may provide the capability of entering Allergy-X = No, Confidence = 90%. Then our rule might look something like this:

If infection = bacterial AND fever = Yes AND allergy-X = No CF>.9 Then Prescribe-X = Yes.

As you can see, developing the KB can be a very challenging task, and as the repository for all rules, facts, and other required information associated with the ES, it is essential that the KE be competent. This is not simply a data entry job!

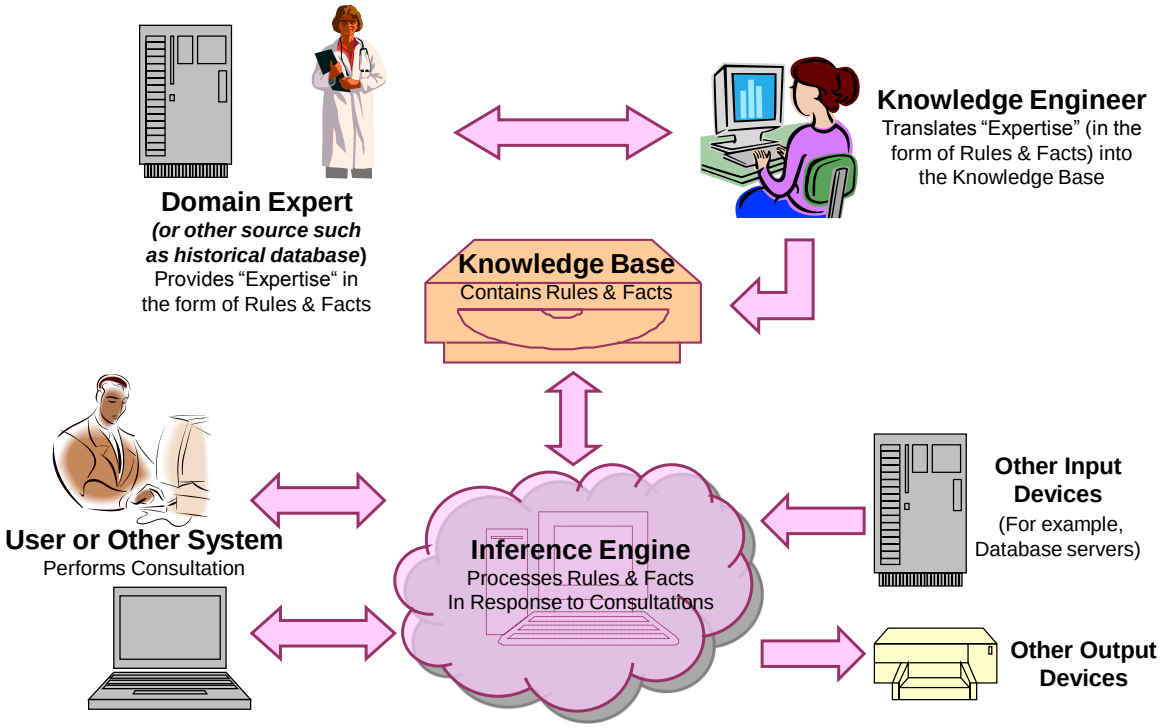
The Inference Engine



Now that we have our KB, we must provide an a way to process the information in the KB in response to requests. We call such requests for processing of the KB **"Consultations"**. For example, if a doctor wants to interact with the ES in order to determine whether to prescribe antibiotic "X", they would "consult" with the system as a user. The ES would have to determine what questions to ask and how to process the responses from the user against the contents of the KB in order to provide advice. This latter task is the job of the "I**nference Engine" (IE)**. The IE is a set of software modules that handle data flow and processing between the user interface and the KB. (Don't confuse this user interface with that used by the KE when building the KB. They are very different.) IEs may use several methods in processing the information in the KB during consultations. These include "Forward Chaining" and "Backwards Chaining." The topic of how IEs work is well beyond the scope of this presentation, but additional information may be found through the links provided on the last slide.

Components of an Expert System

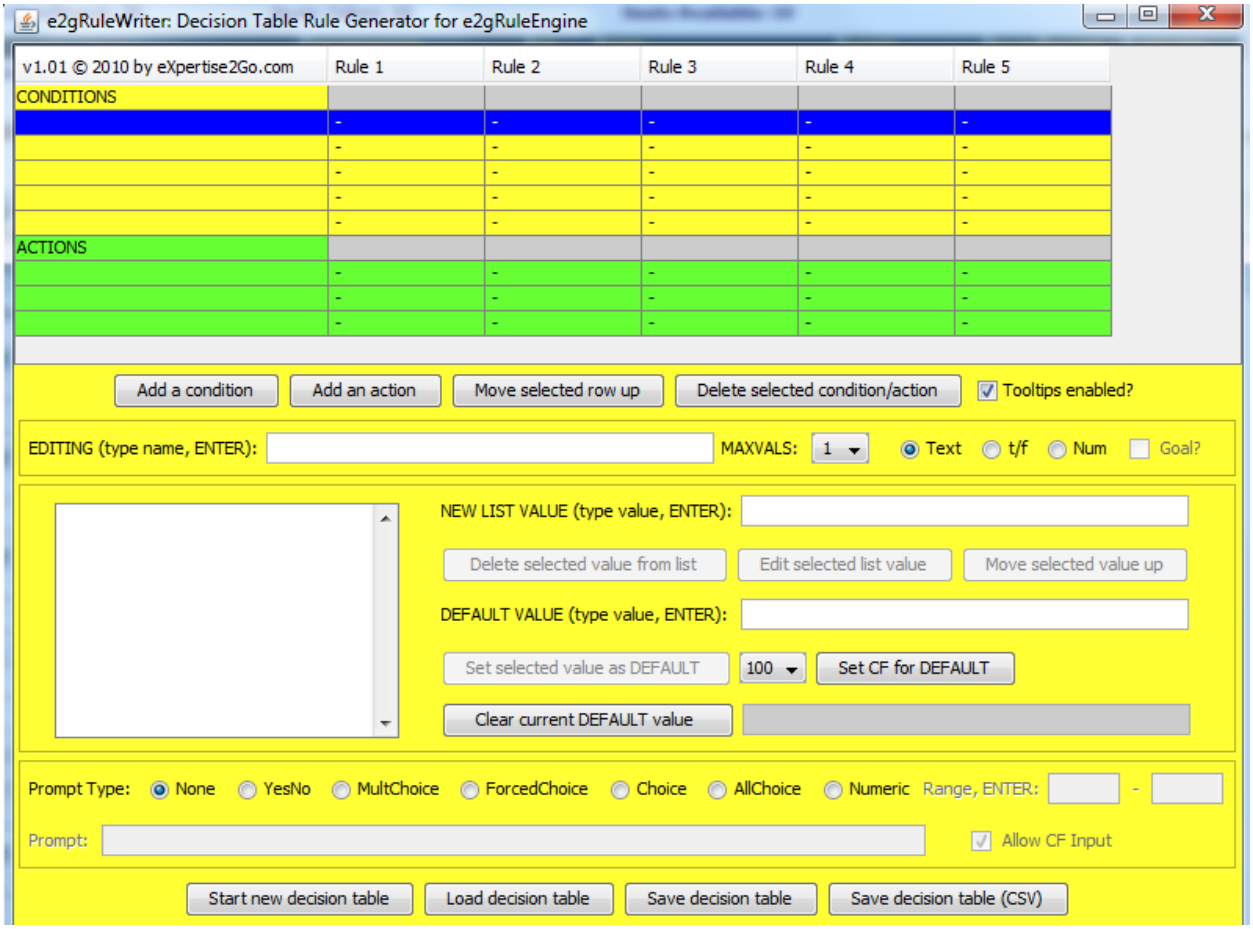
A More General View



Now let's look at an example of how to create a very simple expert system using the e2g3 system. First, we need to define the purpose of the system. Our system will decide whether you should eat something now. The expert has some rules that will allow him/her to tell you if you should "eat now," "don't eat," or "eat something light" (referred to in the e2g3 system as "ACTIONS") based on certain information that must be obtained. The expert needs to know if you are on a diet, if you are hungry, and whether or not you have already eaten (referred to in the e2g3 system as "CONDITIONS").

The "GOAL" of the system is to determine which "ACTION(s)" is/are appropriate based on the status of the "CONDITIONS." We will create a set of "RULES" that are used to consider the status of the "CONDITIONS" and determine an appropriate "ACTION."

We start off by creating a "Decision Table" where we specify conditions, action(s) and rules. Below we see the blank user interface e2g3 provides for this purpose.



We begin by entering information provided by the expert about conditions and actions. This is done by the knowledge engineer.

The expert has told us that if someone is not hungry, they shouldn't eat. (Remember, this is just a simple example. If we were creating a "real" system, we would be considering many other conditions.)

The expert has also said that if they just woke up, we should assume they are hungry and have not yet eaten.

The expert has also said that if they are dieting, and hungry, and have eaten, they shouldn't eat.
If they are hungry and have eaten, but are not on a diet, they should eat something light.

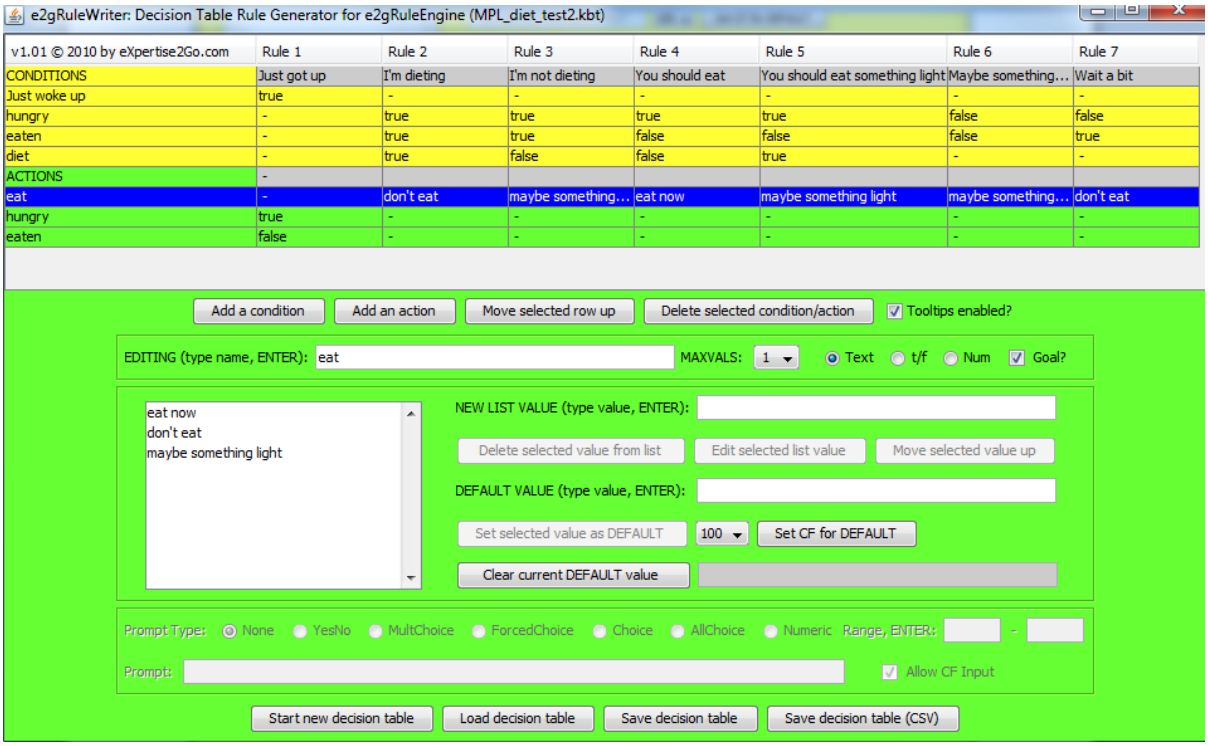
The expert also said that if you are hungry, have not eaten and are not on a diet, they should eat now.

The expert also said that if you are not hungry and have not eaten, but didn't just wake up, they should eat something light.

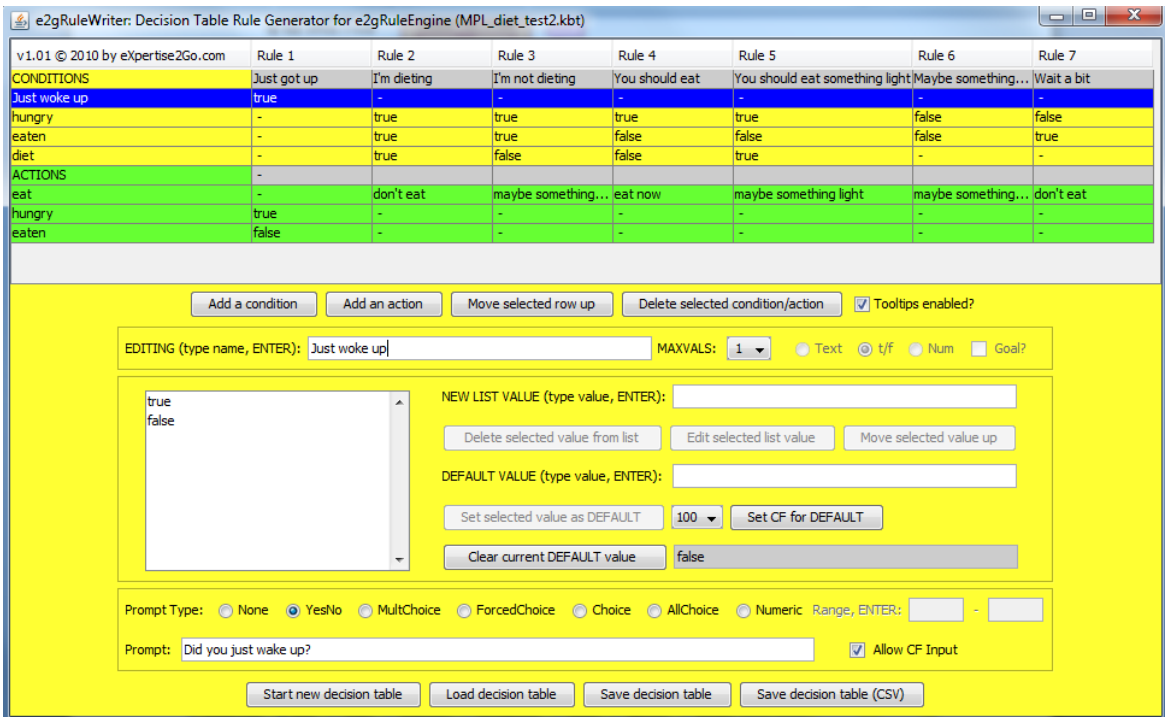
Now we will put this information in the decision table interface and create the rules and then the knowledge base.

Below we see the main Action identified as "eat," along with the three possible action choices the system will make. They are "eat now," "don't eat," and "maybe something light." (The information

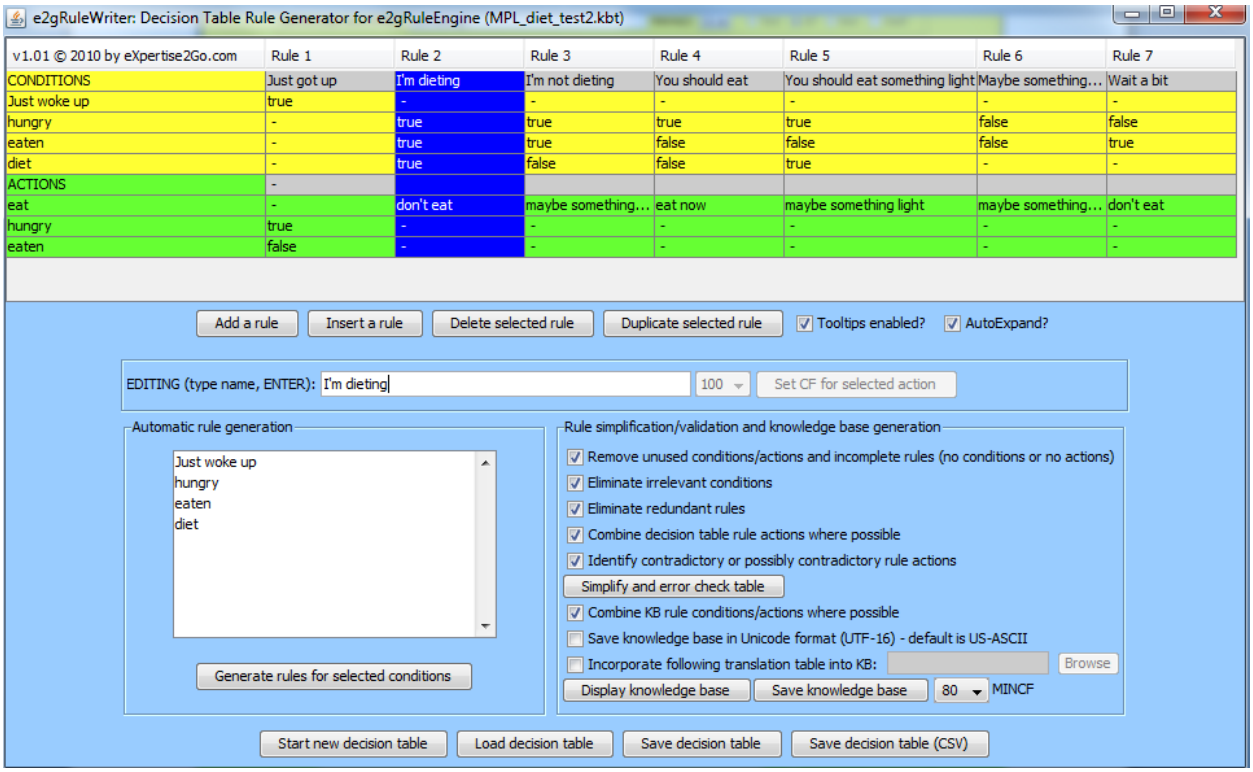
about actions is entered in the green areas on the interface.) Determining the appropriate action choice is the goal of the system. Also, notice that there are two other actions identified in the interface, neither of which are actually goals of the system. We have indicated that the condition "hungry" is set to true and the condition "eaten" is set to false if rule 1 ("just got up") is true.



In the figure below, we see the conditions identified as "just woke up," "hungry," "eaten," and "diet.' In this simple example, we have set the choices for these conditions as simply true or false. (The information about actions is entered in the green areas on the interface.)



The following figure shows how we create the RULES for the system running in vertical columns (Rule 2 seen in blue). There are seven rules that have been created in this example. Each rule is developed by selecting a set of condition values and then selecting the option value for each action based on what the expert has told the knowledge engineer. If a condition value does not affect an action choice, the value may be left blank. In a similar fashion, if a set of condition values does not impact an action choice, it may be left blank. We see examples of the latter with the "hungry" and "eaten" action selection at the bottom of the rule table in green.



Once we have created the rules, we click the "Save knowledge base" button in the light blue area of the screen. This generates a plain text file that is the actual knowledge base file that the inference engine processes during a "CONSULTATION." The file generated in our example is shown below.

REM Generated by v1.01 of e2gRuleWriter 04/23/2012 13:44 from: MPL_diet_test2.kbt

RULE [Just got up]
If [Just woke up] = true
Then [hungry] = "true" and
[eaten] = "false"

RULE [I'm dieting]
If [hungry] = true and
[eaten] = true and
[diet] = true
Then [eat] = "don't eat"

RULE [I'm not dieting]
If [hungry] = true and
[eaten] = true and
[diet] = false
Then [eat] = "maybe something light"

RULE [You should eat]
If [hungry] = true and
[eaten] = false and
[diet] = false
Then [eat] = "eat now"

RULE [You should eat something light]
If [hungry] = true and
[eaten] = false and
[diet] = true
Then [eat] = "maybe something light"

RULE [Maybe something light]
If [hungry] = false and
[eaten] = false
Then [eat] = "maybe something light"

RULE [Wait a bit]
If [hungry] = false and
[eaten] = true
Then [eat] = "don't eat"

PROMPT [Just woke up] YesNo CF
"Did you just wake up?"

PROMPT [hungry] YesNo CF
"Are you hungry?"

PROMPT [eaten] YesNo CF
"Have you eaten?"

PROMPT [diet] YesNo CF
"Are you on a diet?"

DEFAULT [Just woke up] = false
DEFAULT [hungry] = true
DEFAULT [eaten] = false
DEFAULT [diet] = false

GOAL [eat]

MINCF 80

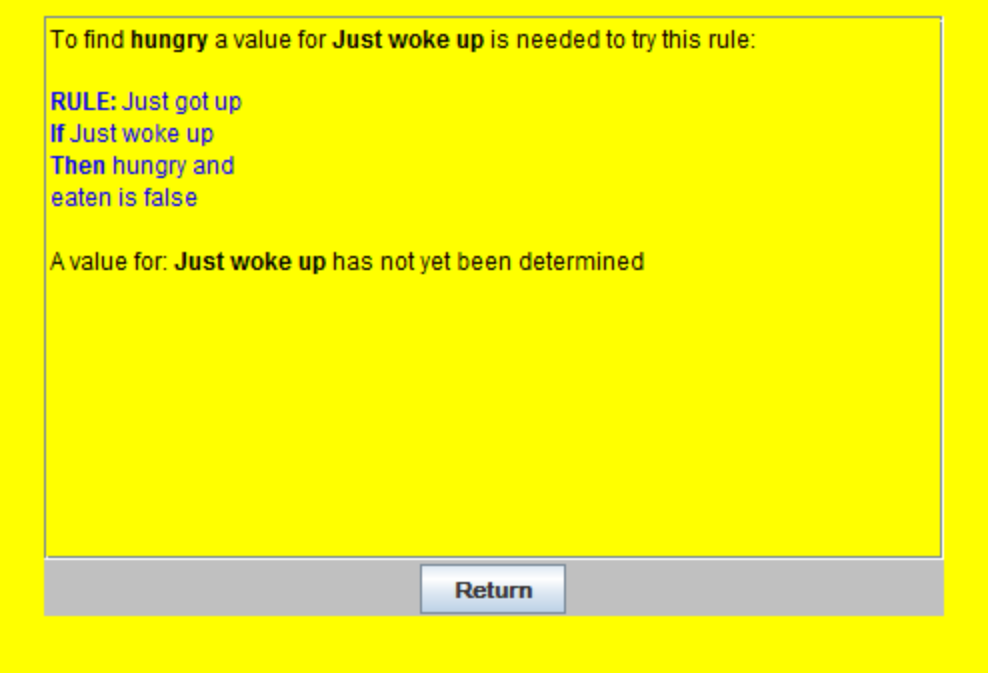
Now we create a small HTML file to “run” the system.

```
<html>
<head><title>mpl_diet_test</title></head>
<body bgcolor="#ffff00">
<center>
<applet code="e2gRuleEngine" archive="e2gRuleEngine.jar" width=450 height=300>
<param name="KBURL" value="mpl_diet_test.kb">
<param name="APPTITLE" value="Should I Eat?">
<param name="APPSUBTITLE" value="mpl_diet_test">
<param name="BGCOLOR" value="#ffff00">
<param name="TITLECOLOR" value="#0000ff">
<param name="PROMPTCOLOR" value="#000000">
<param name="ENCODING" value="US-ASCII">
<param name="DEBUG" value="false">
Java-enabled browser required
</applet>
</center>
</body>
</html>
```

The above code contains two key lines. First, you must identify the INFERENCE ENGINE file, which in this case is done in the fifth line containing …="e2gRuleEngine" archive="e2gRuleEngine.jar"… Next, you must identify the KNOWLEDGE BASE file, which in this case is done in the next line containing … ="KBURL" value="mpl_diet_test.kb"…

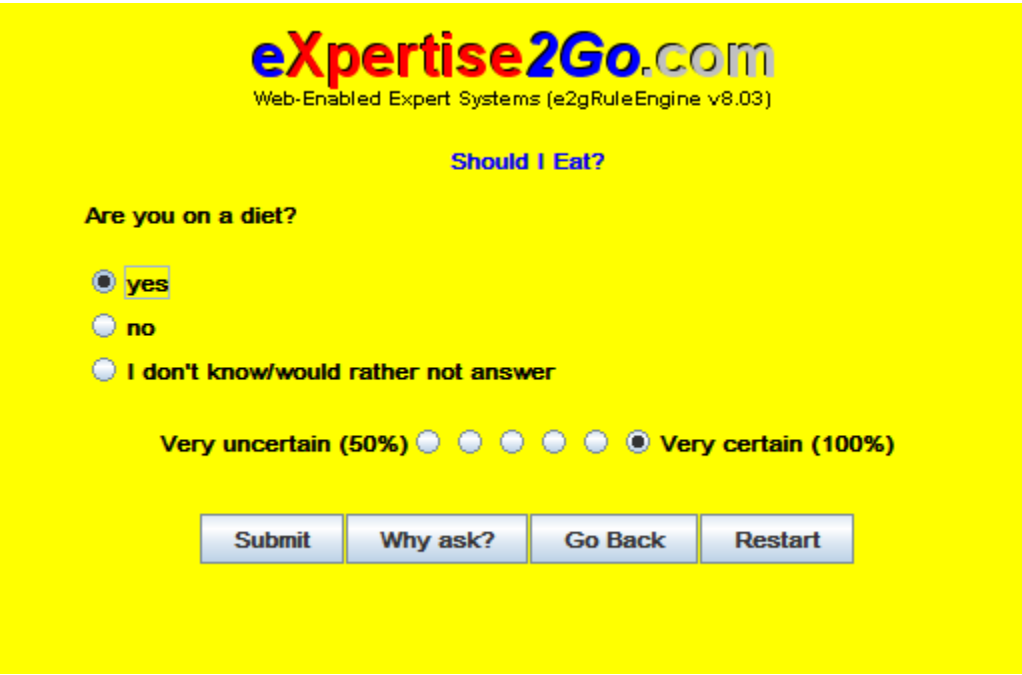
Now let’s execute/run the system.

If you click the “Why ask?” button, the system will explain why it is asking the question (See Below):

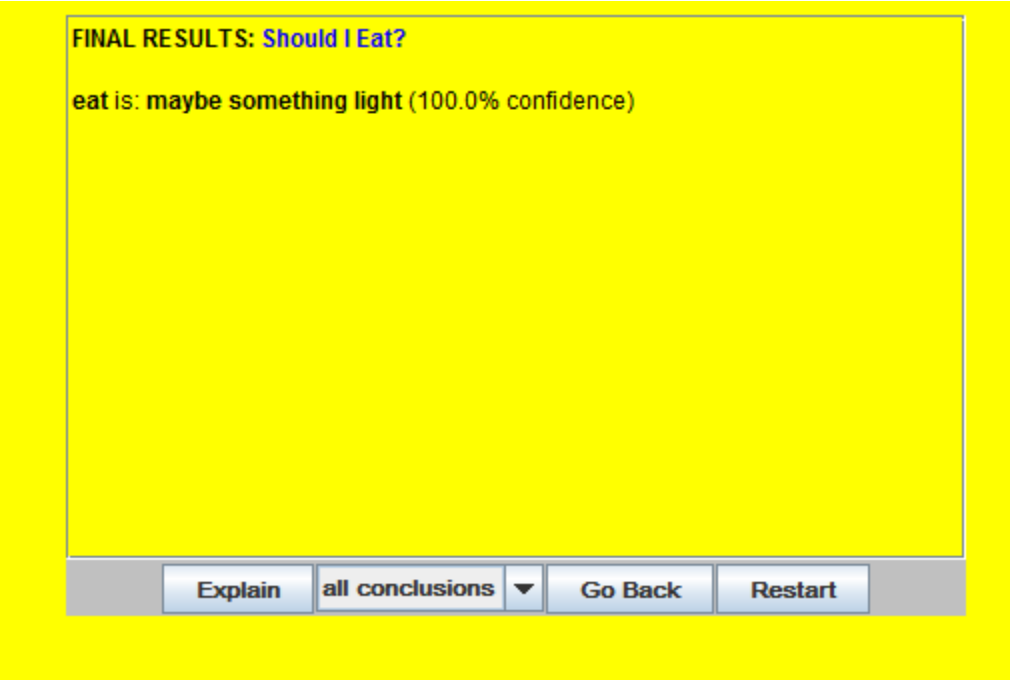


This question is asked first because the inference engine has determined that it can resolve the most uncertainty and move toward determining the goal most quickly by finding out about multiple conditions with one question.

Assuming we answered yes to the "Did you just wake up?" question, the system next asks the following:

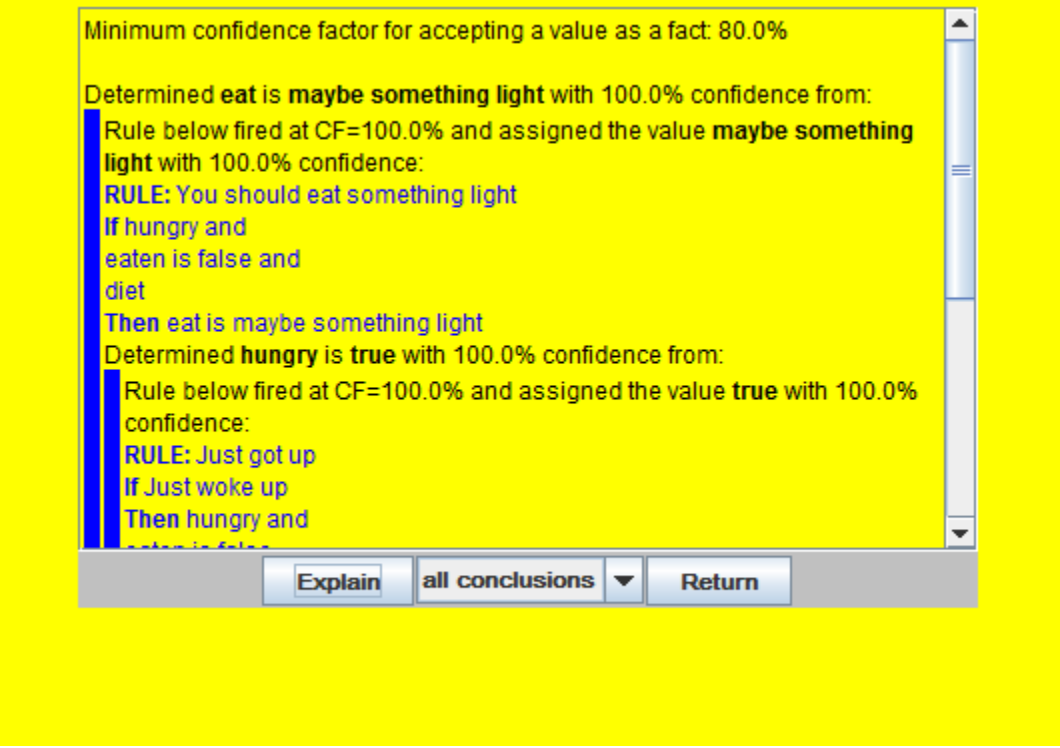


Let's answer "yes" again and see what happens.



Our system has made a decision based on the rules and our answers to the questions.

We can click the "Explain" button and it will tell us how it reached the decision:



We could scroll through the explanation if we wish.

Now you should be ready to try some expert systems development for yourself. Check out the resources provided on the next few pages.

BELOW IS A LIST OF GREAT RESOURCES FOR THE e2g3 SYSTEM:

FOLLOWING MATERIAL FROM:

http://expertise2go.com/e2g3g/e2g3gdoc/

The **e2gRuleEngine/e2gDroid** expert system building tool or "shell" and **e2gRuleWriter** decision table software are free for your private or commercial use subject to the conditions you agree to by downloading the software. Here's a suggested step-by-step approach that will get you started building your own expert systems:

- **Module 1:** **Is this the right technology for your problem?** A discussion of the features of the client-based e2gRuleEngine shell along with some demonstrations of e2gRuleEngine knowledge bases.
- **Module 2:** **Acquiring and installing the software.** Creating a development and delivery environment using the demonstration knowledge bases.

  Note: Modules 3, 5 and 6 are generic eXpertise2Go.com tutorials that introduce basic concepts you need to understand to build your own expert systems.
- **Module 3:** **Introduction to expert systems.** Overview of expert systems technology that includes representing knowledge as if-then rules and implementing expert systems with shells.
- **Module 4:** **Creating your first knowledge base.** Build a simple knowledge base and run it from a stand-alone computer or Web site.
- **Module 5:** **Inference methods and uncertainty.** Interactive demonstrations of how an expert system reasons with rules and explanations of how uncertain facts are represented and processed by these systems.
- **Module 6:** **Introduction to knowledge engineering.** Strategies and techniques for capturing knowledge and representing it in a rule-based expert system knowledge base.
- **Module 7:** **Designing and implementing e2gRuleEngine/e2gDroid knowledge bases that deliver *your* knowledge.** Suggestions for building and debugging more complex applications. Also introduces the use of numerical expressions in rule consequents,the inclusion of hyperlinks in a consultation's results and a variety of v8.0 enhancements including the ability to deliver expert systems on Android devices.
- **Module 8:** **Building internationalized e2gRuleEngine/e2gDroid expert systems.** The process of translating e2gRuleEngine knowledge bases to allow delivery of rule-based expert systems in languages other than English.
- **Module 9:** **Advanced applications: Using the e2gRuleEngine/JavaScript interface to dynamically control inferencing.** Techniques for building advanced applications that allow the e2gRuleEngine inference engine to generate HTML output, load and position Web pages and transfer between knowledge bases to support linked rule sets.
- **Module 10:** **e2gRuleWriter Tutorial and Reference Documentation.** Building, validating and maintaining e2gRuleEngine knowledge bases in a decision table format with e2gRuleWriter.
- **Reference:** **e2gRuleEngine/e2gDroid commands and error messages.** You may want to print these for reference while developing your own knowledge bases.

eXpertise2Go
Expert System Glossary

The index below provides alphabetical access to topics.

A B C D E F G H I J K L M N
O P Q R S T U V W X Y Z
[Home: Demos, Tutorials and Software]

The [Top] link after each glossary definition returns to the index. The [Tutorial] link (when included) accesses relevant tutorial material.

Antecedent. See premise. [Top]

Attribute. A variable that takes on values that might be numeric, text, or logical (true/false). Attributes store the factual knowledge in a knowledge base. [Top]

Backward chaining. The process of determining the value of a goal by looking for rules that can conclude the goal. Attributes in the premise of such rules may be made subgoals for further search if necessary. [Top|Tutorial]

Breadth first search. A search strategy that examines all rules that could determine the value of the current goal or subgoal before backtracking through other rules to determine the value of an unknown attribute in the current rule. [Top]

Certainty processing. Allowing confidence levels obtained from user input and rule conclusions to be combined to increase the overall confidence in the value assigned to an attribute. [Top|Tutorial]

Certainty factor. A measure of the confidence assigned to the value of an attribute. Often expressed as a percentage (0 to 100%) or probability (0 to 1.0). 100% or 1.0 implies that the attribute's value is known with certainty. [Top|Tutorial]

Clause. One expression in the **If** (premise) or **Then**(consequent) part of a rule. Often consists of an attribute name followed by a relational operator and anattribute value. [Top]

Conclusion. See consequent. [Top]

Confidence factor. See certainty factor. [Top]

Consequent. The **Then** part of a rule, or one clauseor expression in this part of the rule. [Top]

Control information. Elements of a knowledge baseother than the attributes and rules that control the user interface, operation of the inference engine and general strategies employed in implementing a consultation with an expert system. [Top]

Depth first search. A search strategy that backtracks through all of the rules in a knowledge base that could lead to determining the value of theattribute that is the current goal or subgoal. [Top]

Domain. A specific problem environment for which knowledge is captured in a knowledge base. [Top]

Expert system. A domain specific knowledge basecombined with an inference engine that processes knowledge encoded in the knowledge base to respond to a user's request for advice. [Top]

Expert system building tool. See expert system shell. [Top]

Expert system shell. A suite of software that allows construction of a knowledge base and interaction with this knowledge base through use of an inference engine. [Top|Tutorial]

Expertise. Specialized domain knowledge, skills, tricks, shortcuts and rules-of-thumb that provide an ability to rapidly and effectively solve problems in the problem domain. [Top]

Firing a rule. A rule fires when the **if** part (premise) is proven to be true. If the rule incorporates an **else**component, the rule also fires when the **if** part is proven to be false. [Top]

Forward chaining. Applying a set of previously determined facts to the rules in a knowledge base to see if any of them will fire. [Top|Tutorial]

Fuzzy variables and fuzzy logic. Variables that take on multiple values with various levels of certaintyand the techniques for reasoning with such variables. [Top|Tutorial]

Goal. A designated attribute: determining the values of one or more goal attributes is the objective of interacting with a rule based expert system. [Top]

Inference. New knowledge inferred from existing facts. [Top|Tutorial]

Inference engine. Software that provides the reasoning mechanism in an expert system. In a rulebased expert system, typically implements forward chaining and backward chaining. strategies. [Top|Tutorial]

Knowledge base. The encoded knowledge for anexpert system. In a rule-based expert system, a knowledge base typically incorporates definitions ofattributes and rules along with control information. Knowledge base format is specific to the implementing expert system shell or other software. [Top]

Knowledge engineering. The process of codifying an expert's knowledge in a form that can be accessed through an expert system. [Top|Tutorial]

Premise. The **if** part of a rule: represents an hypothesis. [Top]

Primary goal. A goal for which a value is sought during an expert system consultation. Primary goals are terminal objectives for the consultation: once their value are found, or a determination is made that they cannot be found, the consultation ends. [Top]

Production rule. See rule. Rules are called production rules because new information is produced when the rule fires. [Top]

Relational operator. Conditions such as **is equal to**or **is less than** that link an attribute name with an attribute value in a rule's premise to form logical expressions that can be evaluated as **true** or **false**. In the example logical expression:

credit rating is less than good

credit rating is the name of an attribute, **is less than**is a relational operator and **good** is an attribute value. [Top]

Rule. A statement of the form: **if <x> then <y> else <z>**. The **if** part is the rule premise, and the **then** part is the consequent. The **else** component of the consequent is optional. The rule fires when the **if** part is

determined to be true or false. Here is an example rule:
If the credit rating is good and the amount of the sale is less than 10000
Then the decision is accept the sale
Else the decision is reject the sale
[Top|Tutorial]

Shell. See expert system shell. [Top]

Subgoal. An attribute the becomes a temporary intermediate goal for the inference engine. Subgoal values need to be determined because they are used in the premise of rules that can determine primary goals. [Top]

[Home: expert system demos and tutorials]