Many knowledge-based software development tools focus on rule-based knowledge representation technology, among the more comprehensive of which are EMYCIN, OPS5, and S.1™. In each of these tool systems, rules are conceptually represented as IF/THEN statements with the logical form:

IF <predicate> THEN <consequent>

Using such statements, knowledge crafters formulate the knowledge they obtain from the experts into sets of such rules. The inference engine then analyzes and processes these IF/THEN rules in one of two ways: backward or forward. In backward-chaining, the inference engine works backward from hypothesized consequents to locate known predicates that would provide support. In forward-chaining, the inference engine works forward from known predicates to derive as many consequents as possible.

Backward-Chaining

In backward-chaining, the inference engine identifies a set of one or more hypotheses by finding those consequents that do not appear as predicate elements in other rules. Similarly, predicates that do not appear as the consequents of any other rule are marked as terminal predicates. The analysis process begins with the user specifying one or more hypotheses (either selected from the inference engine's generated list or selected from the set of rule consequents).

The reasoning process begins with the inference engine taking the first hypothesis and locating all rules having the hypothesis as a consequent. It then moves backward from the consequent to the predicate in each of those rules and tests the truth of the predicate. If any predicate tests TRUE, the consequent is established. It is added to (inserted into) the knowledge base, and the process either terminates or proceeds to examine the next user-specified hypothesis.

If the truth of no predicate can be determined, then each such unknown predicate is established as a new hypothesis and the process continues. The inference engine selects one of the new hypotheses, examines each rule having that hypothesis as a consequent, tests the predicate of each such rule, and so forth. This process forms a chain, linking rule predicates backward to the consequents of other rules—hence the term *backward-chaining.*

When one of the predicates being tested evaluates to TRUE, not only is the consequent of that rule inserted into the knowledge base, but the consequent of every other rule in the chain linking the predicate to the user-specified hypothesis is similarly inserted into the knowledge base and the user notified of the truth of the hypothesis. Again, the reasoning process either terminates or proceeds to examine the next user-specified hypothesis.

Should the backward-chaining process reach a terminal predicate and should the value of that predicate be unknown, either the user (or a

data base) will be asked to supply the value, or the chaining along that path will be abandoned. Should the terminal consequents of all chains from the hypothesis test negatively, the testing of that hypothesis will be abandoned, and another one selected. This process continues until the appropriate number of user-supplied hypotheses has been proven true or until all hypotheses have been considered.

The system keeps track of the values of the predicates and only asks that a value be supplied when the predicate applies to a hypothesis being investigated and when the value of that predicate cannot otherwise be determined. These two properties permit the inference engine to behave similarly to a person questioning the user.

Backward-chaining is often used as the cornerstone of selection applications, in which one item is to be selected from a fixed set of items, for example, in diagnosing a particular failure to be one from a fixed set of known failure types.

Forward-Chaining

Forward-chaining also operates with sets of rules that look syntactically similar to backward-chaining rules. Each rule is examined, and each predicate is evaluated to determine its truth. The consequents of those rules with predicates evaluating to TRUE are then added to (inserted into) the knowledge base and the entire process repeated. The process continues until the truth of no additional consequents can be determined. Note that failure to derive the truth of a consequent does not necessarily mean that the consequent is FALSE; it only means that truth has not yet been established.

This process of reasoning from the consequent of one rule to the predicate of another forms a chain in the forward direction—hence the term *forward-chaining.* Conceptually, the reasoning process must evaluate the predicate of each rule whenever a new fact is inserted into the knowledge base. This, however, is very inefficient. Many inference engines maintain an elaborate set of pointers, enabling only those rules containing the new fact as a predicate clause to be reevaluated.

Forward-chaining essentially permits the knowledge crafter to use rules to develop as much information as possible from a limited set of initial data or to judge changes efficiently when a new data item is added. This type of reasoning would be appropriate in a monitoring situation, for example, in which it was desirable to learn as much as possible about the state of the monitored system based upon the available data.

A Simple Example

To illustrate the manner in which an inference engine might process a set of rules, consider a simple example of a set of rules that relates the

type of day and my location to what I eat for lunch. These rules will be used first in a backward and then in a forward direction.

The Rules

The following seven rules constitute the knowledge base for the example. Assume that no other information is available at the time processing of the rules is initiated.

Rule 1:	IF	IT IS A WORKDAY
	AND	I AM IN THE OFFICE
	THEN	I EAT IN THE CAFETERIA
Rule 2:	IF	I EAT IN THE CAFETERIA
	THEN	I EAT SOUP AND A SANDWICH
Rule 3:	IF	IT IS A WORKDAY
	AND	I AM OUT OF THE OFFICE
	THEN	I EAT OUT
Rule 4:	IF	I EAT OUT
	THEN	I EAT CHINESE FOOD
Rule 5:	IF	IT IS A WEEKEND DAY
	AND	I AM AT HOME
	THEN	I EAT AT HOME
Rule 6:	IF	I EAT AT HOME
	THEN	I EAT SWEDISH PANCAKES
Rule 7:	IF	IT IS A WEEKEND DAY
	AND	I AM OUT SHOPPING
	THEN	I EAT OUT

The Hypotheses

The set of all basic hypotheses derived from this rule-set (i.e., the consequents that do not appear as predicates in any other rule in this rule-set) are:

1. I EAT SOUP AND A SANDWICH
2. I EAT CHINESE FOOD
3. I EAT SWEDISH PANCAKES.

The Terminal Predicates

Similarly, the terminal predicates (i.e., the predicates that do not appear as consequents in any other rule in this rule-set) are:

1. IT IS A WORKDAY
2. IT IS A WEEKEND DAY
3. I AM IN THE OFFICE
4. I AM OUT OF THE OFFICE
5. I AM AT HOME
6. I AM OUT SHOPPING.

How a Backward-Chaining Inference Engine Might Operate

The typical use of a backward-chaining rule-set is to prove a hypothesis. If you treat the above rule-set as the knowledge base for a backward-chaining inference engine, you would then select one or more hypotheses for testing from the list previously given. If you select all hypotheses for testing, the inference engine might operate as follows:

1. Select a possible hypothesis—

   1. I EAT SOUP AND A SANDWICH

2. Locate the hypothesis in the rule-set—

Rule 2:	IF	I EAT IN THE CAFETERIA
	THEN	I EAT SOUP AND A SANDWICH

3. Examine the predicates of the rule and determine whether they have been evaluated and, if not, evaluate them (In this case there is only a single predicate clause to evaluate.)—

   They evaluate to UNKNOWN

   (If the predicate had evaluated to TRUE, the hypothesis would be true and the process would terminate; if the predicate had evaluated to FALSE, then the inference engine would search for another rule having the same consequent as the hypothesis, i.e., the same consequent as the consequent of this rule.)

4. Locate a rule whose consequent contains one of the predicate clauses—

Rule 1:	IF	IT IS A WORKDAY
	AND	I AM IN THE OFFICE
	THEN	I EAT IN THE CAFETERIA

5. Evaluate the predicates of this rule if they have not already been evaluated—

   They evaluate to UNKNOWN

6. Determine whether these are terminal predicates—

   They are:

   1. IT IS A WORKDAY
   3. I AM IN THE OFFICE

7. Ask the user for the value of each unknown predicate; if the value disproves the rule, select another path to the hypothesis, and, if no other path exists, try another hypothesis.

How a Simple Forward-Chaining Inference Engine Might Operate

The same rule-set can be used as the knowledge base for a forward-chaining inference engine. In this case, however, no hypotheses are provided because the rules are not used to try to derive the truth of any particular consequent. Rather, they are used to derive all possible consequents that can be derived from a set of predicates (actually from a set of values that cause one or more predicates to evaluate to TRUE).

If you provided the knowledge base with values such that the predicates

2. IT IS A WEEKEND DAY

5. I AM AT HOME

would evaluate to TRUE, a forward-chaining inference engine might operate as follows:

1. The rules containing either of the provided predicates are located

Rule 5:	IF	IT IS A WEEKEND DAY
	AND	I AM AT HOME
	THEN	I EAT AT HOME
Rule 7:	IF	IT IS A WEEKEND DAY
	AND	I AM OUT SHOPPING
	THEN	I EAT OUT

   and one is selected—

   Rule 5 is selected.

2. This rule is interpreted (*fired*)—

Rule 5:	IF	IT IS A WEEKEND DAY
	AND	I AM AT HOME
	THEN	I EAT AT HOME

   and the consequent I EAT AT HOME is given the value TRUE.

3. A search is made for a rule containing I EAT AT HOME as a predicate—

Rule 6:	IF	I EAT AT HOME
	THEN	I EAT SWEDISH PANCAKES

4. The rule is interpreted (*fired*)—

   I EAT SWEDISH CAKES is given the value TRUE.

5. A search is made for a rule containing I EAT SWEDISH PANCAKES as a predicate—

   No such rule exists.

6. The process terminates.

Attributes of Rule-Based Applications Development Tools

Inference engines with rule-oriented processing capabilities have several attributes, as described below.

Rule-Sets

As indicated, each of the rule-based reasoning techniques simultaneously works on a group of related rules—the rule-set—to obtain information by a parallel matching of predicates or consequents. The order in which the rules are listed in the knowledge base should be unrelated to the order in which they will be considered or fired.