The system must be able to justify that its answer is correct, particularly when it is giving advice to a human.

The same features can be used for explanation and for debugging the knowledge base.

There are three main mechanisms:

- Ask **HOW** a goal was derived.
- Ask **WHYNOT** a goal wasn’t derived.
- Ask **WHY** a subgoal is being proved.
If $g$ is derived, there must be a rule instance

$$g \iff a_1 \land \ldots \land a_k.$$ 

where each $a_i$ is derived.

If the user asks HOW $g$ was derived, the system can display this rule. The user can then ask HOW $i$.

The system can display the rule that was used to prove $a_i$.

The HOW command moves down the proof tree.
Why Did the System Ask a Question?

It is useful to find out why a question was asked.

- Knowing why a question was asked will increase the user’s confidence that the system is working sensibly.
- It helps the knowledge engineer optimize questions asked of the user.
- An irrelevant question can be a symptom of a deeper problem.
- The user may learn something from the system by knowing why the system is doing something.
WHY question

- When the system asks the user a question \( g \), the user can reply with

  WHY

- This gives the instance of the rule

  \[ h \leftarrow \cdots \land g \land \cdots \]

  that is being tried to prove \( h \).

- When the user asks WHY again, it explains why \( h \) was proved.
There are four types of nonsyntactic errors that can arise in rule-based systems:

- An incorrect answer is produced; that is, some atom that is false in the intended interpretation was derived.
- Some answer wasn’t produced; that is, the proof failed when it should have succeeded, or some particular true atom wasn’t derived.
- The program gets into an infinite loop.
- The system asks irrelevant questions.
Debugging Incorrect Answers

- An **incorrect answer** is a derived answer which is false in the intended interpretation.
- An incorrect answer means a clause in the KB is false in the intended interpretation.
- If $g$ is false in the intended interpretation, there is a proof for $g$ using $g \Leftarrow a_1 \& \ldots \& a_k$. Either:
  - Some $a_i$ is false: debug it.
  - All $a_i$ are true. This rule is buggy.
Debugging Missing Answers

- **WHYNOT g.** g fails when it should have succeeded.

  Either:
  - There is an atom in a rule that succeeded with the wrong answer, use HOW to debug it.
  - There is an atom in a body that failed when it should have succeeded, debug it using WHYNOT.
  - There is a rule missing for g.
There is no automatic way to debug all such errors: \textit{halting problem.}

There are many errors that can be detected:
- If a subgoal is identical to an ancestor in the proof tree, the program is looping.
- Define a well-founded ordering that is reduced each time through a loop.