MIT EECS
6.034: Artificial Intelligence
(Fall 2005)
Instructor: Luis E. Ortiz

Recitation 1: Rule-Based Systems
September 16, 2005

- **Forward Chaining**

ASSERTIONS ⇒ CONCLUSIONS

Below is pseudocode for a naive implementation of forward chaining.

```
FORWARD-CHAINING
repeat
  for every rule do
    if antecedents match assertions in the working memory and consequents would change
    the working memory then
      Create triggered rule instance
    end if
  end for
  Pick one triggered rule instance, using conflict resolution strategy if needed, and fire it
  (throw away other instances)
until no change in working memory, or no STOP signal
```

Note that this is a general implementation that applies to both deduction and production/reaction systems. Be aware that many variations are possible. For instance, in deduction systems, forward chaining *usually* fires a rule (instance) as soon as it is triggered. Note also that even in deduction systems forward chaining might use a conflict resolution strategy to decide the order in which it considers the rules for matching (in the for-loop). In class examinations, it will be made clear which implementation you should use.

- **Backward Chaining**

CONCLUSIONS (HYPOTHESES) ⇒ ASSERTIONS

Below is a naive implementation of backward chaining.

```
BACKWARD-CHAINING(H)
if H matches an assertion in working memory then
  return true
end if
if there is no rule with a consequent that matches H then
  ASK USER or ASSUME false
end if
for every rule R with a consequent that matches H do
  if for all antecedents A of rule R, we have BACKWARD-CHAINING(A) = true then
    return true
  end if
end for
return false
```

Note the recursive nature of backward chaining. Note also that the for-loop creates the *or-nodes* of the goal (and/or) tree, while the if-statement inside that for-loop creates the *and-nodes*.

Other more efficient implementations are possible. For instance, we might add assertion to the working memory for later reuse as we find them to be true during the backward chaining process. Also, many variants resulting from different implementation decisions are possible. For example, the system can ask the user even after all rules with a consequent that match the hypothesis fail. Another option is to use conflict resolution to decide the order in which the system will consider the rules with consequents that match the hypothesis (in the for-loop). Once again, in class examinations, it will be made clear which implementation you should use.