Forward chaining

Suppose we have query

1? – B
{!B}

In forward chaining we start with facts, and see if we can derive B.
Initial have one fact {A} <- List of facts is called agenda

For Each Rule we check how many variables in its tail are not in the agenda

Example:

B :- C, E	2
C :- A	0
B :- A	0

{A, B, C} <- New Agenda

We check does agenda have B in it?
Yes -> return true (we are done)
Has agenda changed by adding these heads?
No –> return false
Otherwise we loop

<u>1-st Order Logic</u>

Point: Want to be able to reason about sets of objects rather than true/false values.

Where used: Parsers, Prolog, relational databases, planning

Syntax:

Variables x, y, z, … -> range over set Example: x might be an element in a set of colors.

Constants: a, b, c, … Examples: Fixed values from a set 0, 'Bob'

Functions: f, g, h, …

Example: x + y is a function

Formulas in 1st Order Logic

Predicates: P, Q, R … take inputs and output true/false

An atomic formula is a predicate where each of the predicates slots has been filled with a term.
Example: IsPrime? (X*X+3)

A formula is either an atomic formula or built out of simpler formulas. F1 and F2 by one of the following operations.

1. NOT(F1)
2. (F1 AND F2)
3. (F1 OR F2)