

MODULE-5

COURSE NAME:ARTIFICIAL
INTELLIGENCE-BCS515B



FORWARD & BACKWARD CHAINING IN AI

Two Smart Ways AI Reaches a Conclusion





Inference in First Order Logic:

Backward Chaining Logical inference algorithms uses the backward chaining approach.

These algorithms work backward from the goal, chaining through rules to find known facts that support the proof. We describe the basic algorithm, and then we describe how it is used in logic programming, which is the most widely used form of automated reasoning.

```
function FOL-BC-ASK( $KB, query$ ) returns a generator of substitutions
return FOL-BC-OR( $KB, query, \{ \}$ )
```

```
generator FOL-BC-OR( $KB, goal, \theta$ ) yields a substitution
for each rule ( $lhs \Rightarrow rhs$ ) in FETCH-RULES-FOR-GOAL( $KB, goal$ ) do
    ( $lhs, rhs$ )  $\leftarrow$  STANDARDIZE-VARIABLES( $((lhs, rhs))$ )
    for each  $\theta'$  in FOL-BC-AND( $KB, lhs, UNIFY(rhs, goal, \theta)$ ) do
        yield  $\theta'$ 
```

```
generator FOL-BC-AND( $KB, goals, \theta$ ) yields a substitution
if  $\theta = failure$  then return
else if LENGTH( $goals$ ) = 0 then yield  $\theta$ 
else do
     $first, rest \leftarrow$  FIRST( $goals$ ), REST( $goals$ )
    for each  $\theta'$  in FOL-BC-OR( $KB, SUBST(\theta, first), \theta$ ) do
        for each  $\theta''$  in FOL-BC-AND( $KB, rest, \theta'$ ) do
            yield  $\theta''$ 
```

Figure 5.1: A simple backward-chaining algorithm for first-order knowledge bases.

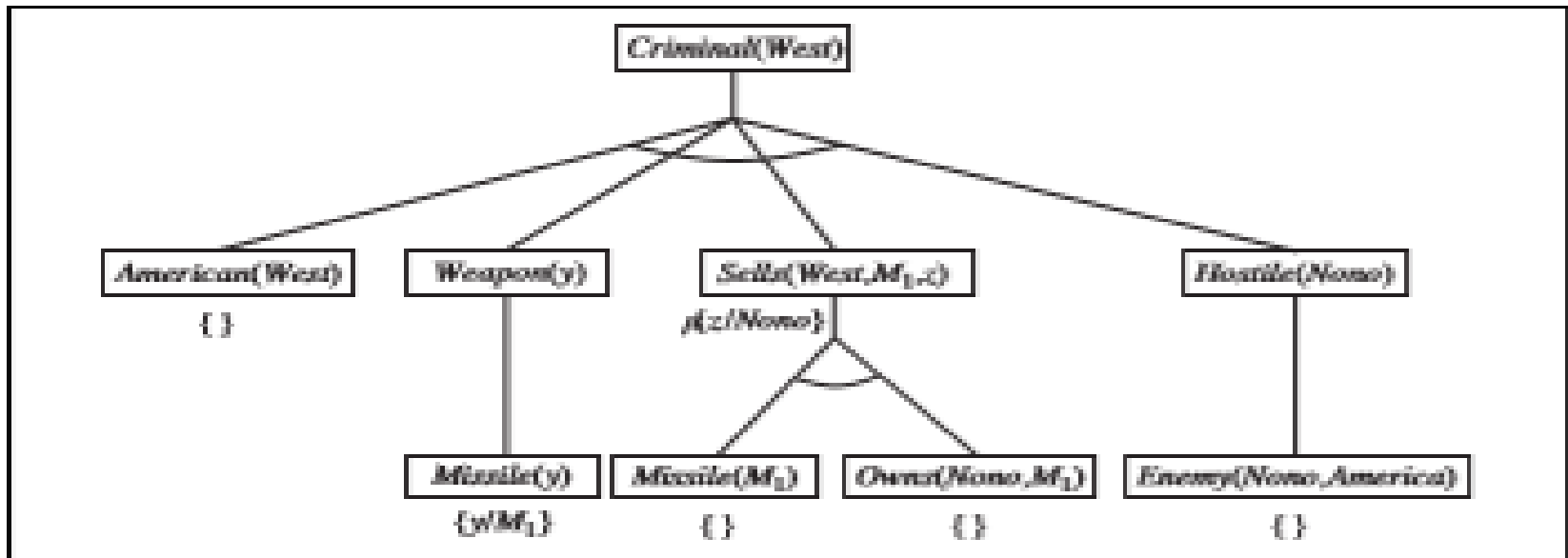


Figure 9.7 Proof tree constructed by backward chaining to prove that West is a criminal. The tree should be read depth first, left to right. To prove *Criminal(West)*, we have to prove the four conjuncts below it. Some of these are in the knowledge base, and others require further backward chaining. Bindings for each successful unification are shown next to the corresponding subgoal. Note that once one subgoal in a conjunction succeeds, its substitution is applied to subsequent subgoals. Thus, by the time FOL-BC-ASK gets to the last conjunct, originally *Hostile(z)*, *z* is already bound to *Nono*.

What is the rule?

The rule given is:

**If a person is American AND
he sells a weapon to someone AND
that “someone” is hostile,
→ then that person is a criminal.**

In logical form:

**$\text{American}(x) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \rightarrow$
 $\text{Criminal}(x)$**

Meaning:

If x is American,
and x sells y to z,
and z is hostile,
then x is a criminal.

We want to prove: **Criminal(West)**

How backward chaining works (simple)

Backward chaining works **from the goal backwards**.

STEP 1 — Goal

We want to prove: Criminal(West)

To prove this, we must prove all conditions of the rule:

STEP 2 — Break goal into sub-goals

To prove Criminal(West), we must check:

Is West American?

Did West sell a weapon?

Is the buyer hostile?

These become sub-questions:

American(West)?

Sells(West, Weapon, ?)

Hostile(?)



To prove **Criminal(West)**

we must prove all three things under it.

Then it goes deeper:

Weapon(M1)

Missiles(M1)

Hostile(Nono)

Enemy(Nono, America)

These are facts given in the database.

Backward chaining starts with the question:

“Is West a criminal?”

Then it moves backward to check all necessary conditions.

Since all required facts exist in the knowledge base, the algorithm successfully proves:

West is a criminal.

From the knowledge base:

West is American → condition 1 OK

West sold missiles (a weapon) to Nono →
condition 2 OK

Nono is hostile → condition 3 OK

Since **all three conditions are TRUE**,
we conclude:

West is a criminal



LOGIC PROGRAMMING

logic programming, mainly using the language **Prolog**, and how it uses **backward chaining** to solve problems.

In Prolog:

Facts → statements that are always true

Example:

american(john).

Rules → say when something is true based on other facts

Example:

criminal(X) :- american(X), sells(X,Y,Z), hostile(Z).

This means:

X is a criminal if X is American **and** X sells Y to Z **and** Z is hostile.

Example goal:

criminal(john)?

Prolog checks:

Is John American?

Did John sell something?

Is the buyer hostile?

If **all are true**, then the goal is true.

Prolog does:

Depth-first search

Backtracking

Works backwards from the goal and Uses built-in operations for arithmetic

Uses "unification" (match variables with values)

Logic programming is a way of programming where:

You write **facts** and **rules** (knowledge).

The computer uses **logic** to find answers.

You don't tell "how to solve" the problem step-by-step.

Instead, you tell **what is true**, and the computer figures out the rest.

This is called **declarative programming**.

Robert Kowalski described it with a formula:

Algorithm = Logic + Control

Logic = facts and rules

Control = how the system searches for solutions

Prolog has its own style:

Variables → start with uppercase (X, Y, Z)

Constants → start with lowercase (john, india, apple)

Commas (,) mean **AND**

Rules are written **backwards**

Standard logic:

$A \wedge B \Rightarrow C$

(If A and B are true, then C is true)

Prolog:

$C :- A, B.$

(Read: C is true if A and B are true)

4. Skolemization (most important part)

This removes **existential quantifiers** ($\exists$).

Case 1: No universal quantifier around

If we have:

$\exists y \text{ Loves}(y, x)$

Replace y with a **new constant or function**.

If no outer variable controls it $\rightarrow$ use **constant**

If some outer variable controls it $\rightarrow$ use **function of that variable**

Final CNF Meaning (Simplified)

After all steps, the final CNF represents:

“Each person either

- does **not** love some animal,

OR

- is loved by someone (possibly different for each person).”

This preserves the original meaning.

Resolution

Resolution is a method used in logic to **prove something by contradiction**.

To use resolution, we must convert sentences into a special format called **CNF** (Conjunctive Normal Form).

To *prove something is true*, we try to show that the **negation** of that thing leads to a contradiction.

Example Sentence

“Everyone who loves all animals is loved by someone.”

Symbolically:

$$\forall x [(\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(x,y)) \Rightarrow (\exists y \text{ Loves}(y,x))]$$

We convert this into CNF step by step.

1. Remove $\Rightarrow$ (implication)

$A \Rightarrow B$ becomes $\neg A \vee B$.

2. Move negation inward (push $\neg$ inside)

When we push NOT inside quantifiers:

$\neg \forall x P$ becomes $\exists x \neg P$

$\neg \exists x P$ becomes $\forall x \neg P$

This converts everything into expressions without $\Rightarrow$ and without complicated negations.

The meaning stays the same.

3. Standardize variables

If two different $\exists$ use the same variable name, rename one.

Example:

$(\exists x P(x)) \vee (\exists x Q(x))$

becomes

$(\exists x P(x)) \vee (\exists z Q(z))$

This avoids confusion later.



A T M E
College of Engineering

GOLD

QS I-GAUGE
INDIAN COLLEGE RATINGS



Thank you