Inference In FOL

Universal instantiation (UI)

- Every instantiation of a universally quantified sentence is entailed by it:
- $$\frac{\forall v\, \alpha}{\text{Subst}(\{v/g\}, \alpha)}$$

  for any variable v and ground term g

- E.g., $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ yields:
- $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
  $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
  $King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$
  .
  .

Existential instantiation (EI)

- For any sentence α, variable *v*, and constant symbol *k* that does not appear elsewhere in the knowledge base:
-

$$\frac{\exists v\, \alpha}{\text{Subst}(\{v/k\}, \alpha)}$$

- E.g., $\exists x\ Crown(x) \wedge OnHead(x, John)$ yields:

$$Crown(C_1) \wedge OnHead(C_1, John)$$

provided C_1 is a new constant symbol, called a Skolem constant

Unification

- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

- Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x,OJ)	{fail}

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- To unify *Knows(John,x)* and *Knows(y,z)*,
- θ = {y/John, x/z } or θ = {y/John, x/John, z/John}
- The first unifier is more general than the second.
- There is a single most general unifier (MGU) that is unique up to renaming of variables.
- MGU = { y/John, x/z }

The unification algorithm

function UNIFY(L1, L2*)* **returns** substitution to make *L1 = L2*

If L1 or L2 are both variables or constants then:

- If L1 and L2 are identical, then return NIL
- Else if L1 is a variable, then return (L2/L1)
- Else if L2 is a variable, then return (L1/L2)
- Else return {FAIL}

if the predicate in L1 and L2 not identical, then return {FAIL}
if L1 and L2 have different number of arguments, return {FAIL}
for I = 1 to number of arguments of L1:

- Unify ith argument of L1 and ith argument of L2, result S
- if S contains FAIL then return {FAIL}
- SUBST := COMPOSE(S, SUBST)

return SUBST.

Generalized Modus Ponens (GMP)

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta}$$

where $p_i'\theta = p_i\theta$ for all i

p_1' is *King*(*John*)
p_2' is *Greedy*(*y*)
θ is {x/John,y/John}
q θ is *Evil*(*John*)

p_1 is *King*(*x*)
p_2 is *Greedy*(*x*)
q is *Evil*(*x*)

- GMP used with KB of definite clauses (exactly one positive literal)
- All variables assumed universally quantified
-

Example knowledge base

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.
-
- Prove that Col. West is a criminal
-

Example knowledge base

... it is a crime for an American to sell weapons to hostile nations:

$American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \Rightarrow Criminal(x)$

Nono … has some missiles, i.e., $\exists x\ Owns(Nono,x) \land Missile(x)$:

$Owns(Nono,M_1)$ and $Missile(M_1)$

… all of its missiles were sold to it by Colonel West

$Missile(x) \land Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$

Missiles are weapons:

$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile“:

$Enemy(x,America) \Rightarrow Hostile(x)$

West, who is American …

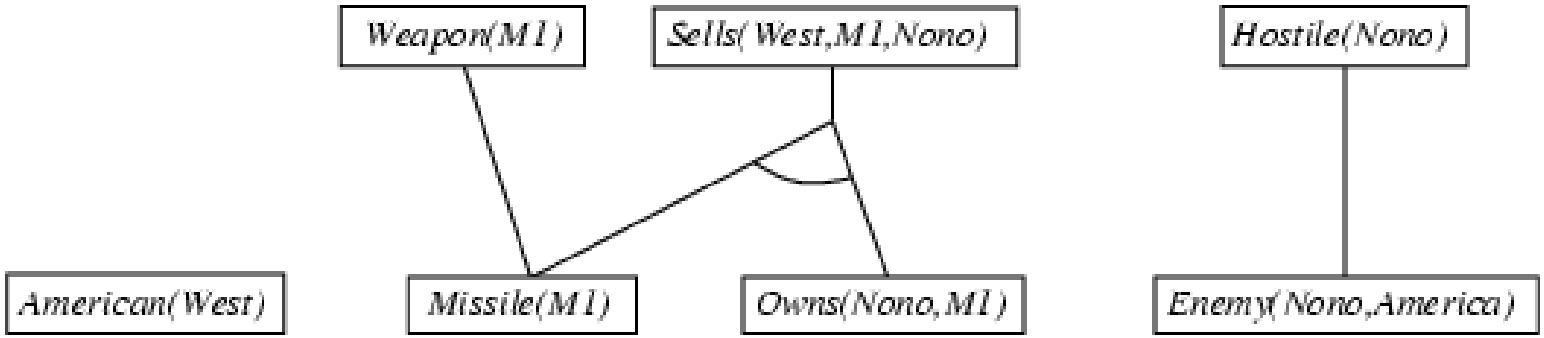
$American(West)$

The country Nono, an enemy of America …

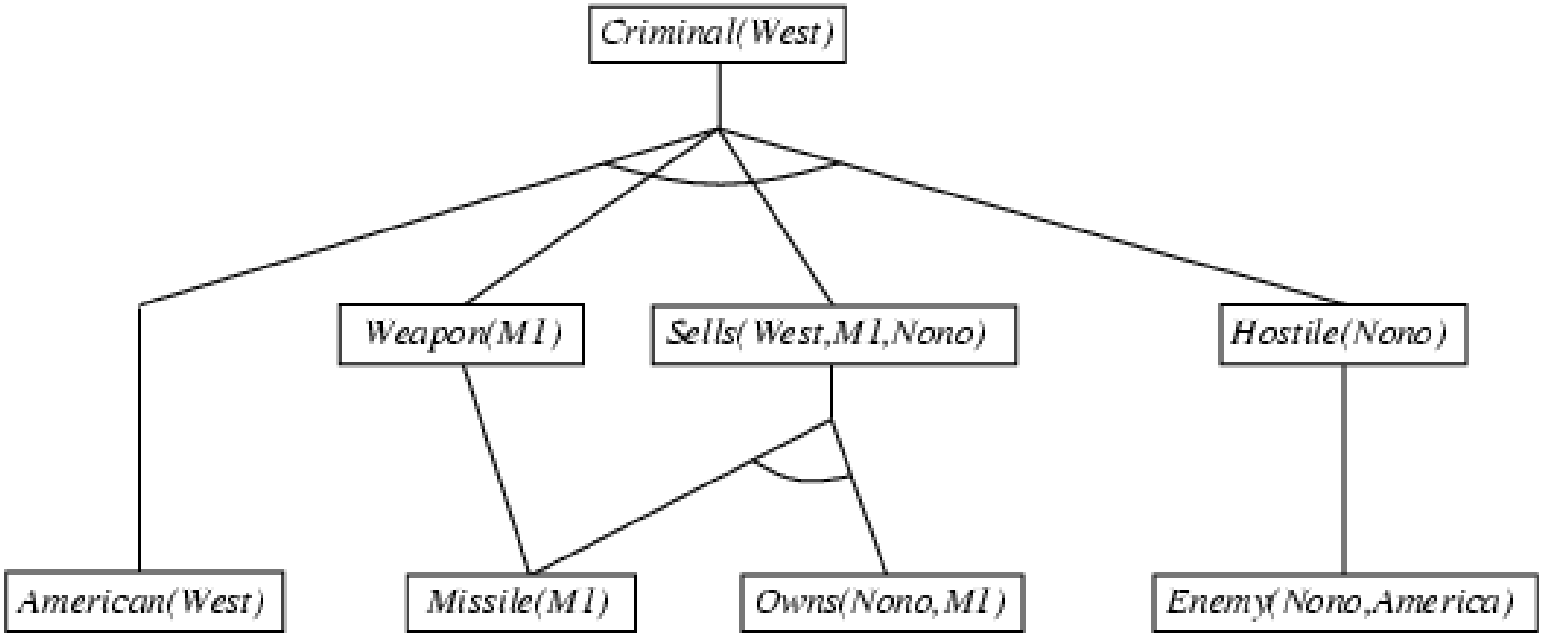
$Enemy(Nono,America)$

Forward chaining proof

Forward chaining proof



Forward chaining proof

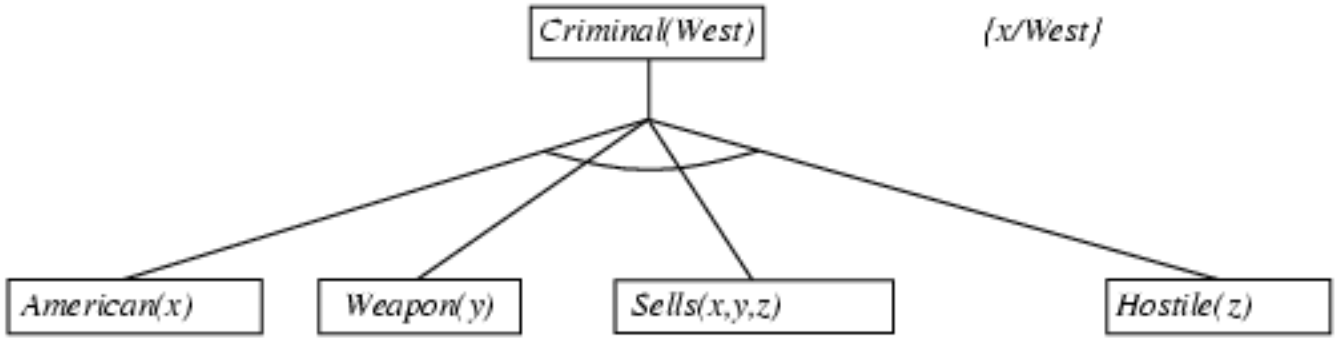


Properties of forward chaining

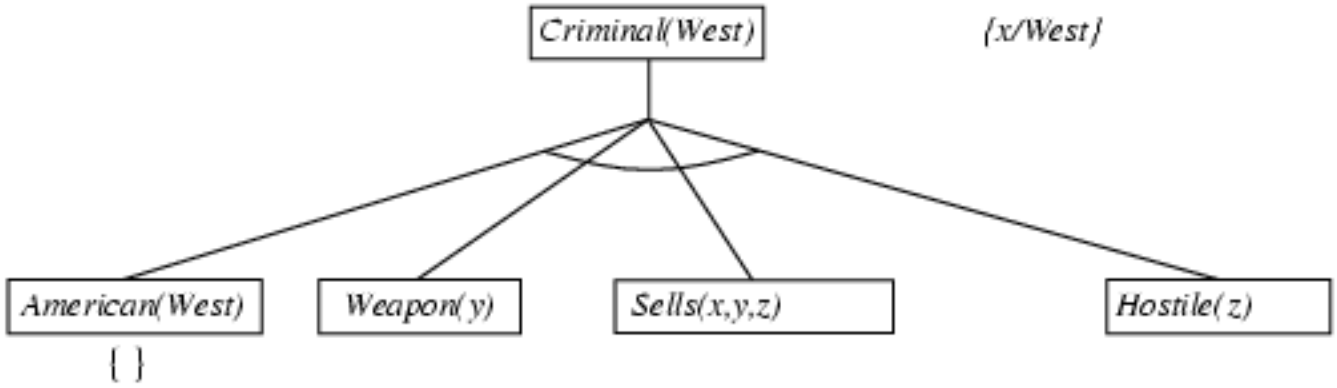
- Sound and complete for first-order definite clauses
-
- Datalog = first-order definite clauses + no functions
- FC terminates for Datalog in finite number of iterations
-
- May not terminate in general if α is not entailed
-
- This is unavoidable: entailment with definite clauses is semidecidable
-

Backward chaining example

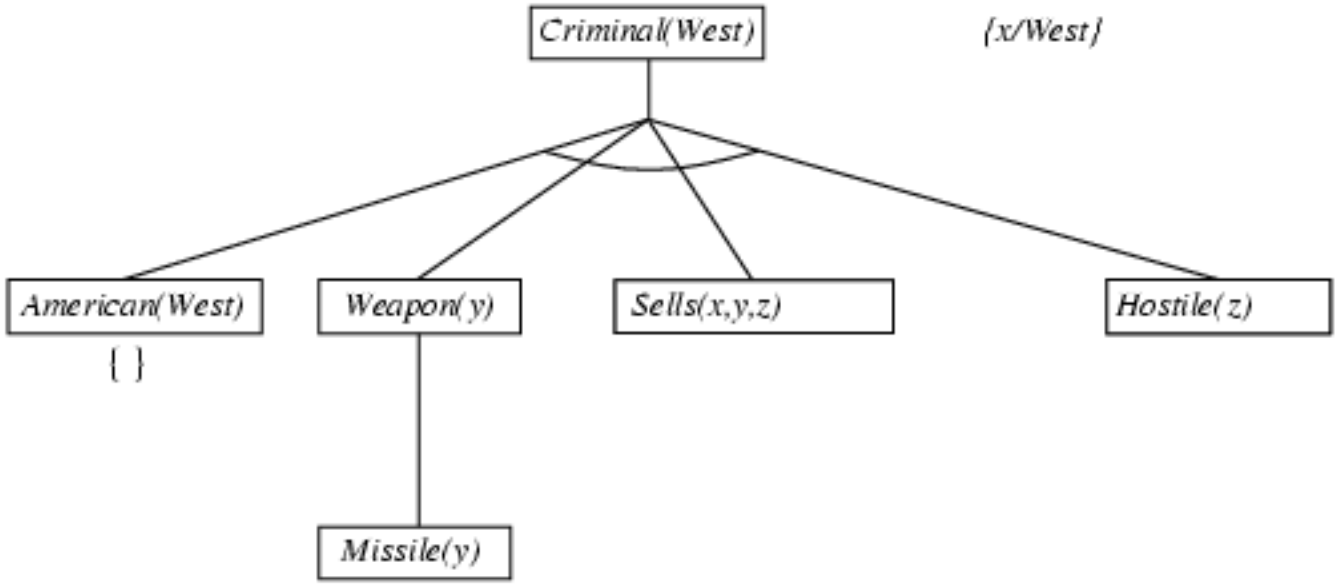
Backward chaining example



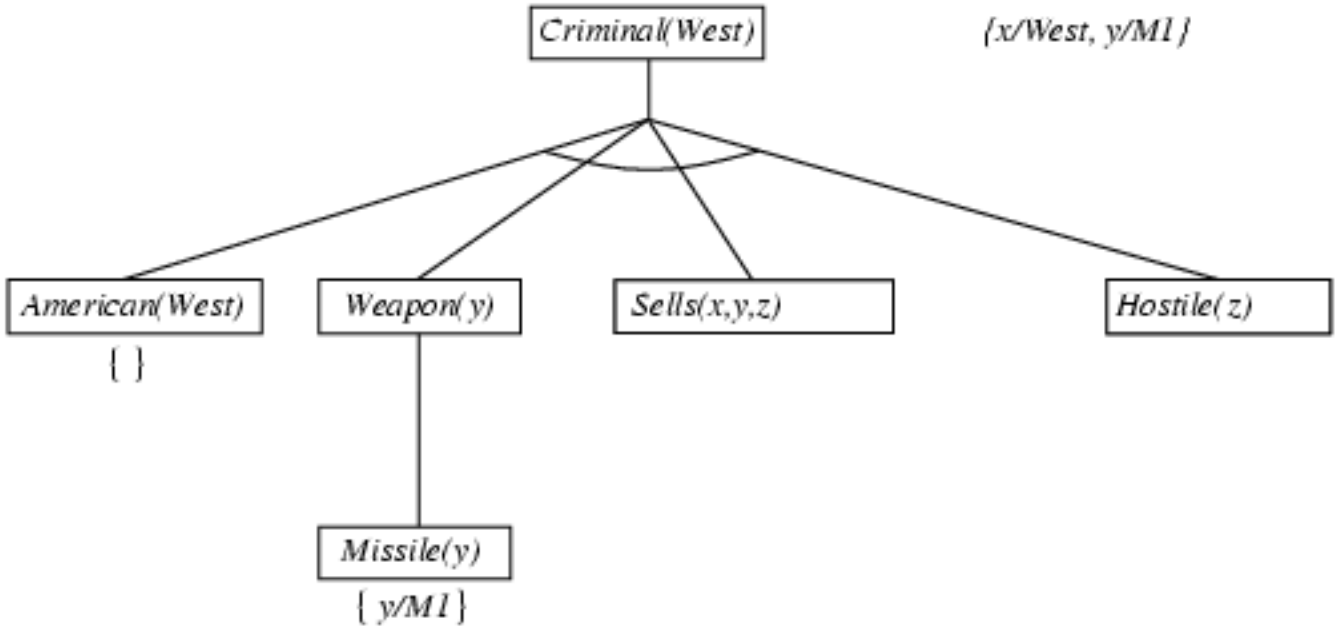
Backward chaining example



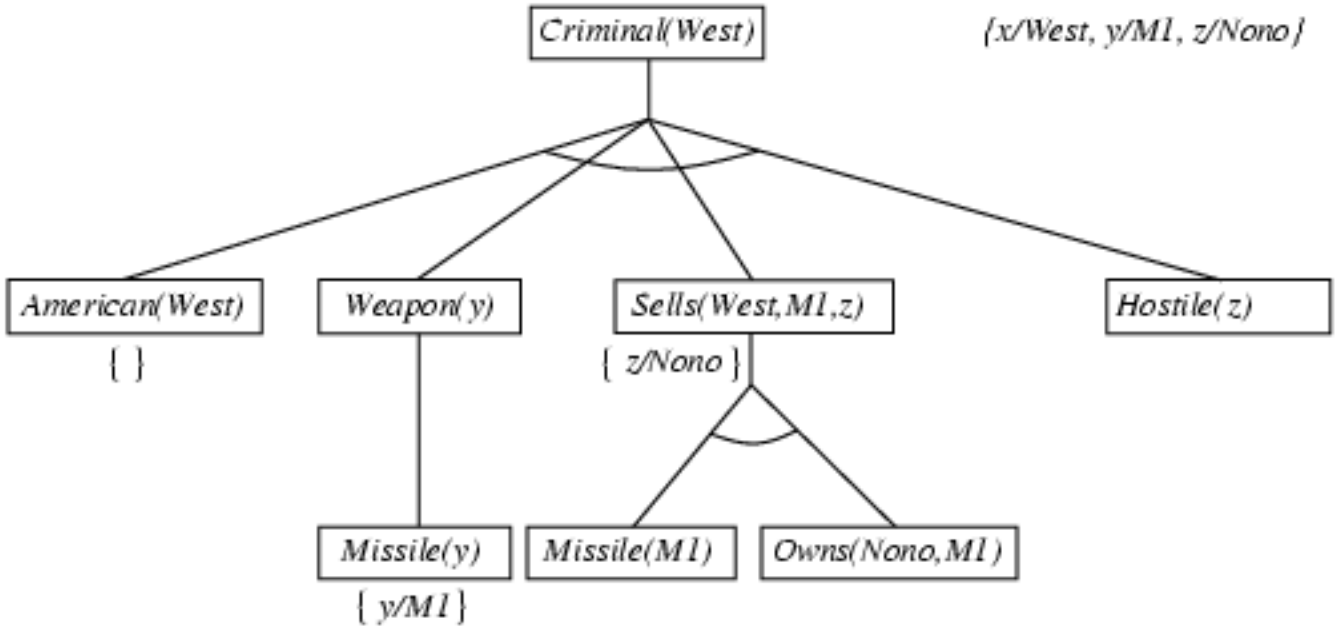
Backward chaining example



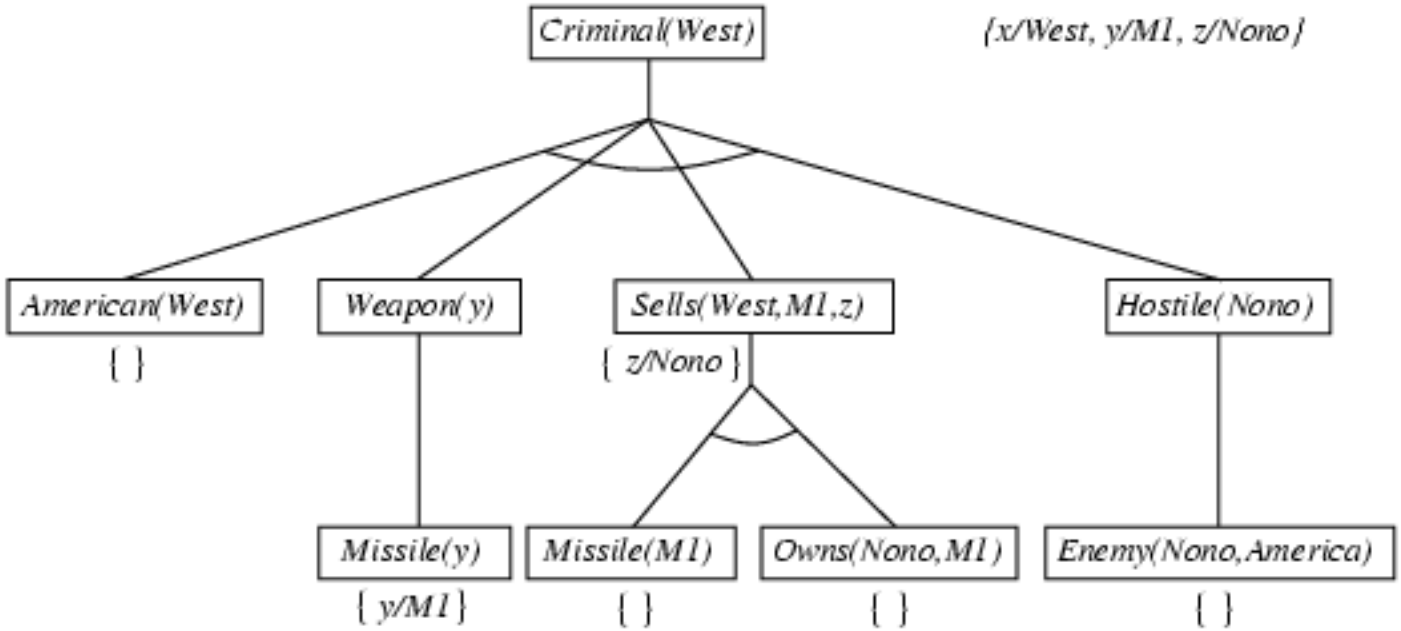
Backward chaining example



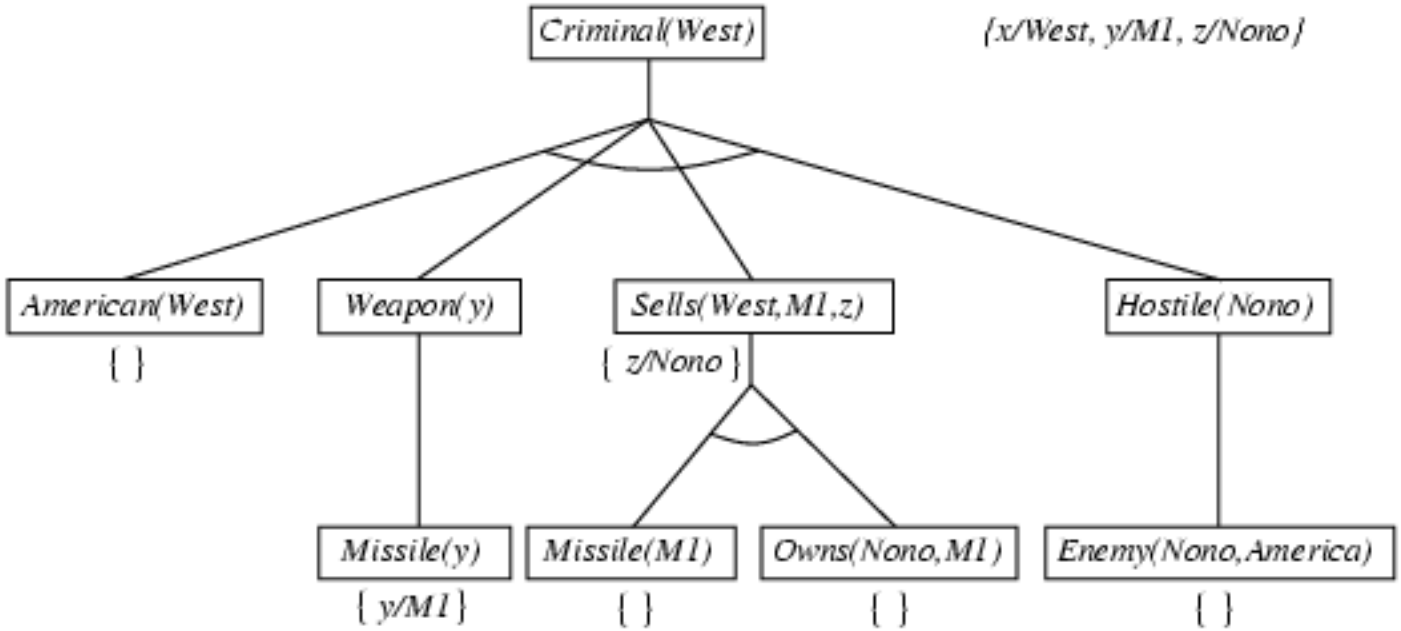
Backward chaining example



Backward chaining example



Backward chaining example



The same figure as forward chaining

Properties of backward chaining

- Depth-first recursive proof search: space is linear in size of proof
- Incomplete due to infinite loops
- $\Rightarrow$ fix by checking current goal against every goal on stack
-
- Inefficient due to repeated subgoals (both success and failure)

  $\Rightarrow$ fix using caching of previous results (extra space)

SUBST(COMPOSE(θ_1, θ_2), p) = SUBST(θ_2, SUBST(θ_1, p))

- Widely used for logic programming

Logic programming: Prolog

- Algorithm = Logic + Control
- Basis: backward chaining with Horn clauses + bells & whistles
  Widely used in Europe, Japan (basis of 5th Generation project)
  Compilation techniques ⇒ 60 million LIPS

- Program = set of clauses = `head :- literal`$_1$`, … literal`$_n$.
-

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

- Depth-first, left-to-right backward chaining
- Built-in predicates for arithmetic etc., e.g., `X is Y*Z+3`
- Built-in predicates that have side effects (e.g., input and output
-
- predicates, assert/retract predicates)
- Closed-world assumption ("negation as failure")
  - e.g., given `alive(X) :- not dead(X).`
  - `alive(joe)` succeeds if `dead(joe)` fails
  -

Resolution: brief summary

- Full first-order version:
-

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where $\texttt{Unify}(\ell_i, \neg m_j) = \theta$.

- The two clauses are assumed to be standardized apart so that they share no variables.
-
- For example,
-

$$\frac{\neg \mathit{Rich}(x) \vee \mathit{Unhappy}(x) \qquad \mathit{Rich}(\mathit{Ken})}{\mathit{Unhappy}(\mathit{Ken})}$$

Conversion to CNF

- Everyone who loves all animals is loved by someone:
  $\forall x\ [\forall y\ \mathit{Animal}(y) \Rightarrow \mathit{Loves}(x,y)] \Rightarrow [\exists y\ \mathit{Loves}(y,x)]$

- 1. Eliminate biconditionals and implications
- $\forall x\ [\neg\forall y\ \neg\mathit{Animal}(y) \vee \mathit{Loves}(x,y)] \vee [\exists y\ \mathit{Loves}(y,x)]$
- 2. Move $\neg$ inwards: $\neg\forall x\ p \equiv \exists x\ \neg p,\ \neg\exists x\ p \equiv \forall x\ \neg p$
- $\forall x\ [\exists y\ \neg(\neg\mathit{Animal}(y) \vee \mathit{Loves}(x,y))] \vee [\exists y\ \mathit{Loves}(y,x)]$
  $\forall x\ [\exists y\ \neg\neg\mathit{Animal}(y) \wedge \neg\mathit{Loves}(x,y)] \vee [\exists y\ \mathit{Loves}(y,x)]$
  $\forall x\ [\exists y\ \mathit{Animal}(y) \wedge \neg\mathit{Loves}(x,y)] \vee [\exists y\ \mathit{Loves}(y,x)]$

Conversion to CNF

3. Standardize variables: each quantifier should use a different on

$\forall x\ [\exists y\ Animal(y) \land \neg Loves(x,y)] \lor [\exists z\ Loves(z,x)]$

4. Skolemize: a more general form of existential instantiation.

Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

$\forall x\ [Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$

5. Drop universal quantifiers

$[Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$

6. Distribute $\lor$ over $\land$:

$[Animal(F(x)) \lor Loves(G(x),x)] \land [\neg Loves(x,F(x)) \lor Loves(G(x),x)]$

Resolution proof