# Artificial Intelligence

CSE 440/EEE 333/ETE333

Chapter 9
Fall 2017

**Mirza Mohammad Lutfe Elahi**
Department of Electrical and Computer Engineering
North South University

# Creating a Knowledge Base

- Identify the Task
- Assemble the relevant knowledge
- Decide on a vocabulary of predicates, functions and constants
- Encode general knowledge about the domain (rules)
- Encode a description of the problem
- Pose queries to the inference procedure and get answers
- Debug the KB

# Inference in First Order Logic

Given $\forall x King(x) \wedge Greedy(x) \rightarrow Evil(x)$

One can infer

- $King(John) \wedge Greedy(John) \rightarrow Evil(John)$
- $King(Richard) \wedge Greedy(Richard) \rightarrow Evil(Richard)$
- $King(Father(John)) \wedge Greedy(Father(John)) \rightarrow$
  $Evil(Father(John))$

# Inference in First Order Logic

- Universal Instantiation (in a $\forall$ rule, substitute all symbols)
- Existential Instantiation (in a $\exists$ rule, substitute one symbol, use the rule and discard)
- Skolem Constants is a new variable that represents a new inference.

# Inference in First Order Logic

Suppose KB:

- $\forall x King(x) \wedge Greedy(x) \rightarrow Evil(x)$
- $King(John)$
- $Greedy(John)$
- $Brother(Richard, John)$

Apply UI using $\{x/John\}$ and $\{x/Richard\}$

- $King(John) \wedge Greedy(John) \rightarrow Evil(John)$
- $King(Richard) \wedge Greedy(Richard) \rightarrow Evil(Richard)$

And discard the Universally quantified sentence. We can get the KB to be propositions.

# Inference in First Order Logic

Suppose KB:

- $\forall x King(x) \land Greedy(x) \rightarrow Evil(x)$
- $King(John)$
- $\exists y Greedy(y)$

Apply UI using $\{x/John\}$ and $\{x/Richard\}$

# Inference Generalized Modus Ponens

for atomic sentences $p_i, p_i'$ and $q$, where there is a substitution $\theta$ such that $SUBST(\theta, p_i') = SUBST(\theta, p_i)$, for all $i$

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

$$\begin{array}{ll}
p_1' = King(John) & p_1 = King(x) \\
p_2' = Greedy(y) & p_2 = Greedy(x) \\
\theta = \{x/John, y/John\} & q = Evil(x) \\
SUBST(\theta, q) & .
\end{array}$$

# Inference Unification

$UNIFY(p, q) = \theta$ Where $SUBST(\theta, p) = SUBST(\theta, q)$

For example:

- We ask $ASKVARS(Knows(John, x))$ (Whom does John know?)
- $UNIFY(Knows(John, x), Knows(John, Jane)) = \{x/Jane\}$
- $UNIFY(Knows(John, x), Knows(y, Bill)) = \{y/John, x/Bill\}$
- $UNIFY(Knows(John, x), Knows(y, Mother(y))) = \{y/John, x/Mother(John)\}$

# Unification / Algorithm

**function** UNIFY($x, y, \theta$) **returns** a substitution to make $x$ and $y$ identical
  **inputs**: $x$, a variable, constant, list, or compound expression
    $y$, a variable, constant, list, or compound expression
    $\theta$, the substitution built up so far (optional, defaults to empty)

  **if** $\theta$ = failure **then return** failure
  **else if** $x = y$ **then return** $\theta$
  **else if** VARIABLE?($x$) **then return** UNIFY-VAR($x, y, \theta$)
  **else if** VARIABLE?($y$) **then return** UNIFY-VAR($y, x, \theta$)
  **else if** COMPOUND?($x$) **and** COMPOUND?($y$) **then**
    **return** UNIFY($x$.ARGS, $y$.ARGS, UNIFY($x$.OP, $y$.OP, $\theta$))
  **else if** LIST?($x$) **and** LIST?($y$) **then**
    **return** UNIFY($x$.REST, $y$.REST, UNIFY($x$.FIRST, $y$.FIRST, $\theta$))
  **else return** failure

---

**function** UNIFY-VAR($var, x, \theta$) **returns** a substitution

  **if** $\{var/val\} \in \theta$ **then return** UNIFY($val, x, \theta$)
  **else if** $\{x/val\} \in \theta$ **then return** UNIFY($var, val, \theta$)
  **else if** OCCUR-CHECK?($var, x$) **then return** failure
  **else return** add $\{var/x\}$ to $\theta$

# Inference Putting it all together

$UNIFY(Knows(John, x), Knows(y, Mother(y)), \varphi)$

$= UNIFY((John, x), (y, Mother(y)), UNIFY(Knows, Knows, \varphi))$

$= UNIFY((John, x), (y, Mother(y)), \varphi)$

$= UNIFY((x), (Mother(y)), UNIFY(John, y, \varphi))$

$= UNIFY((x), (Mother(y)), \{y/John\})$

$= UNIFYVAR(x, Mother(y), \{y/John\})$

$= \{y/John, x/Mother(y)\}$

# Inference Putting it all together

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

Prove that Colonel West is a Criminal

# Inference Putting it all together

The law says that it is a crime for an American to sell weapons to hostile nations.

$\forall x\ American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \rightarrow Criminal(x)$

An enemy of America,

$Enemy(Nono, America)$

has some missiles,

$\exists y\ Missile(x) \wedge Owns(Nono, x)$

and all of its missiles were sold to it by Colonel West,

$\forall x\ Missile(x) \wedge Owns(Nono, x) \rightarrow Sells(West, x, Nono)$

who is American.

$American(West)$

# Inference Putting it all together

Additional background knowledge:

Missiles are weapons.

$\forall x\ Missile(x) \rightarrow Weapon(x)$

Enemies of America are hostile.

$\forall x\ Enemy(x, America) \rightarrow Hostile(x)$

Prove that Col. West is a criminal

$Criminal(West)$?

# Inference Putting it all together

The knowledge base can be simplified by existential instantiation and omitting universal quantifiers (as all free variables are universally quantified anyway)

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

- R1: $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \rightarrow Criminal(x)$
- R2: $Owns(Nono, M_1)$ Nono has some missiles
- R3: $Missile(M_1)$
- R4: $Missile(x) \rightarrow Weapon(x)$ A missile is a weapon
- R5: $Missile(x) \wedge Owns(Nono, x) \rightarrow Sells(West, x, Nono)$ All missiles sold by west
- R6: $Enemy(x, America) \rightarrow Hostile(x)$ Enemies of America are hostile
- R7: $American(West)$ West is american
- R8: $Enemy(Nono, America)$

# Forward Chaining Algorithm

**function** FOL-FC-ASK($KB, \alpha$) **returns** a substitution or *false*
  **inputs**: $KB$, the knowledge base, a set of first-order definite clauses
    $\alpha$, the query, an atomic sentence
  **local variables**: $new$, the new sentences inferred on each iteration

  **repeat until** $new$ is empty
    $new \leftarrow \{ \}$
    **for each** $rule$ in $KB$ **do**
      $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-VARIABLES($rule$)
      **for each** $\theta$ such that SUBST($\theta, p_1 \wedge \ldots \wedge p_n$) = SUBST($\theta, p'_1 \wedge \ldots \wedge p'_n$)
          for some $p'_1, \ldots, p'_n$ in $KB$
        $q' \leftarrow$ SUBST($\theta, q$)
        **if** $q'$ does not unify with some sentence already in $KB$ or $new$ **then**
          add $q'$ to $new$
          $\phi \leftarrow$ UNIFY($q', \alpha$)
          **if** $\phi$ is not *fail* **then return** $\phi$
    add $new$ to $KB$
  **return** *false*

# Inference Putting it all together

Iteration 1:

- R5 satisfied with $\{x/M_1\}$ and R9: *Sells*(*West*, $M_1$, *Nono*) is added
- R4 satisfied with $\{x/M_1\}$ and R10: *Weapon*($M_1$) is added
- R6 satisfied with $\{x/Nono\}$ and R11: *Hostile*(*Nono*) is added

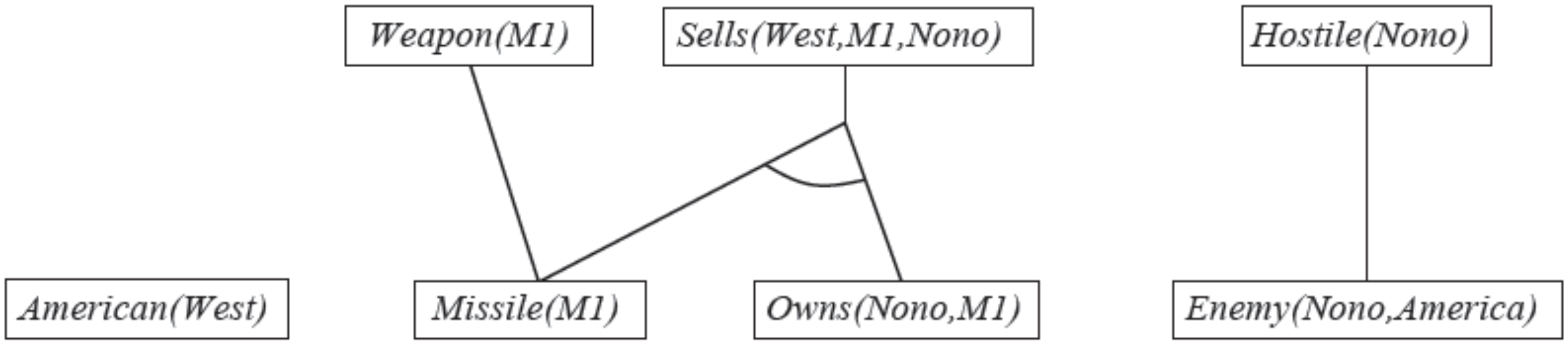
Iteration 2:

- R1 is satisfied with $\{x/West, y/M1, z/Nono\}$ and *Criminal*(*West*) is added.

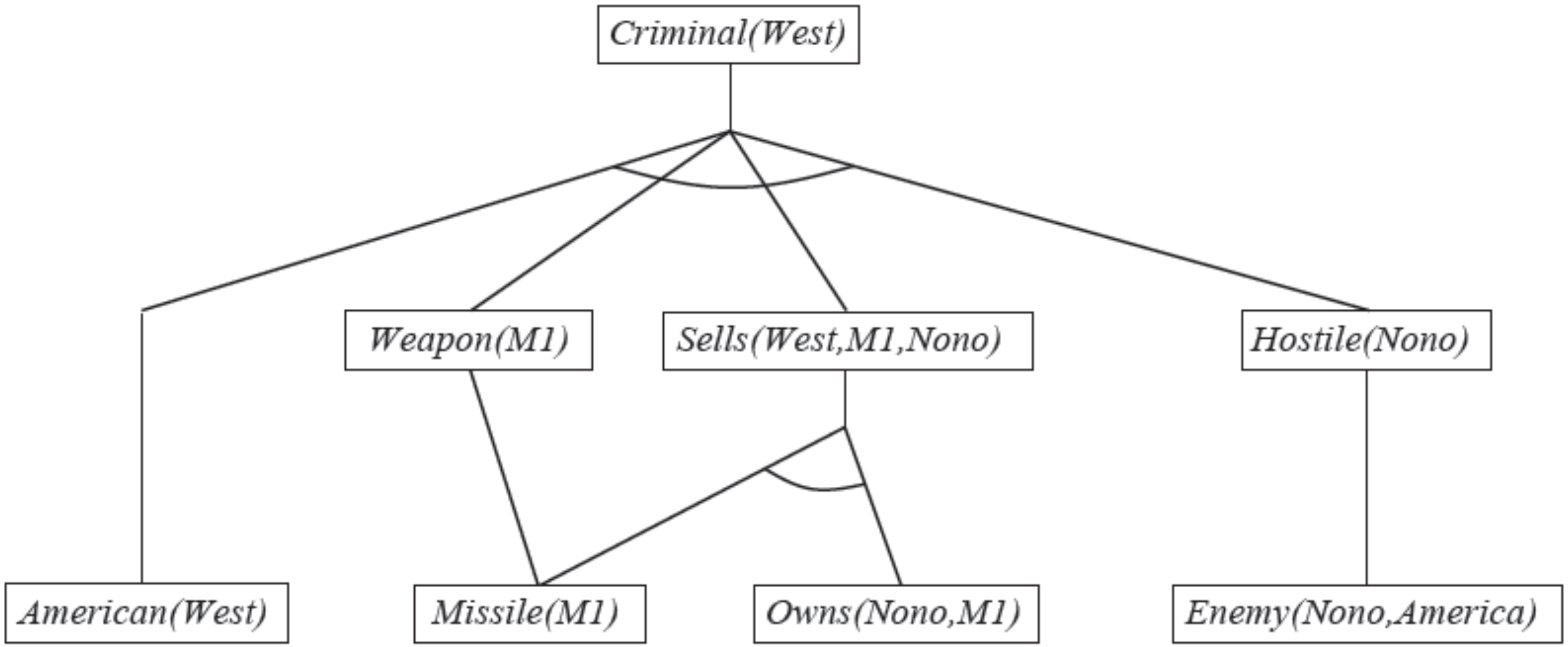
# Forward Chaining / Example

$\boxed{American(West)}$ $\boxed{Missile(M1)}$ $\boxed{Owns(Nono,M1)}$ $\boxed{Enemy(Nono,America)}$

# Forward Chaining / Example



# Forward Chaining / Example



# Backward Chaining

Backward chaining works the other way around:

- keep a list of yet unsatisfied atoms Q
  - starting with the query atom.

- try to find rules which head match atoms in Q (after unification) and replace the atom from Q by the atoms of the body of the matching rule.
- proceed recursively until no more atoms have to be satisfied.

Backward chaining keeps track of the substitution needed during the proof.

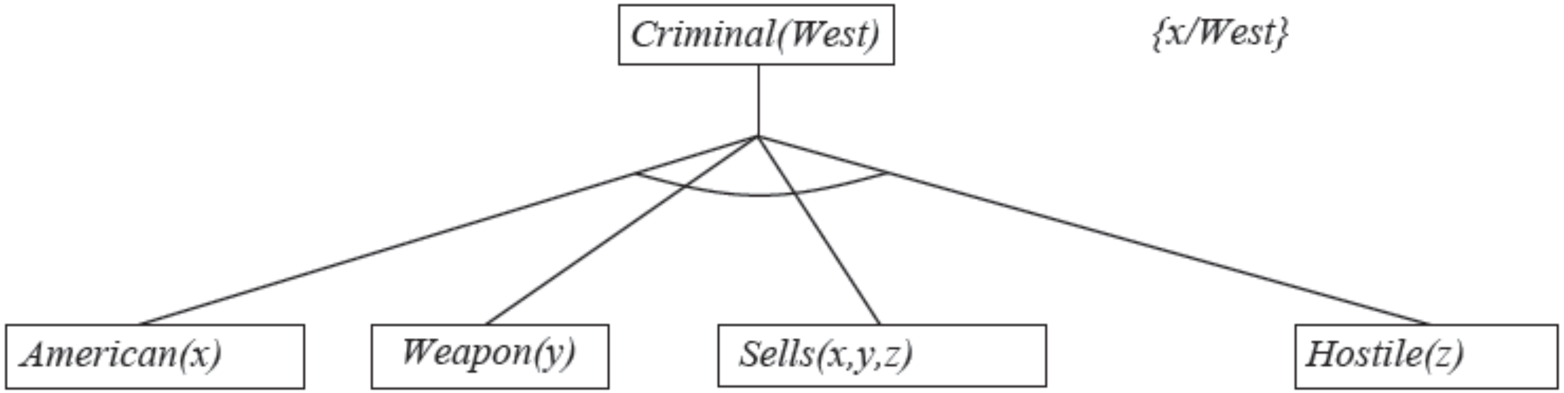
# Backward Chaining Algorithm

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

generator FOL-BC-OR(KB, goal, θ) yields a substitution
  for each rule (lhs ⇒ rhs) in FETCH-RULES-FOR-GOAL(KB, goal) do
    (lhs, rhs) ← STANDARDIZE-VARIABLES((lhs, rhs))
    for each θ′ in FOL-BC-AND(KB, lhs, UNIFY(rhs, goal, θ)) do
      yield θ′

generator FOL-BC-AND(KB, goals, θ) yields a substitution
  if θ = failure then return
  else if LENGTH(goals) = 0 then yield θ
  else do
    first,rest ← FIRST(goals), REST(goals)
    for each θ′ in FOL-BC-OR(KB, SUBST(θ, first), θ) do
      for each θ″ in FOL-BC-AND(KB, rest, θ′) do
        yield θ″
```

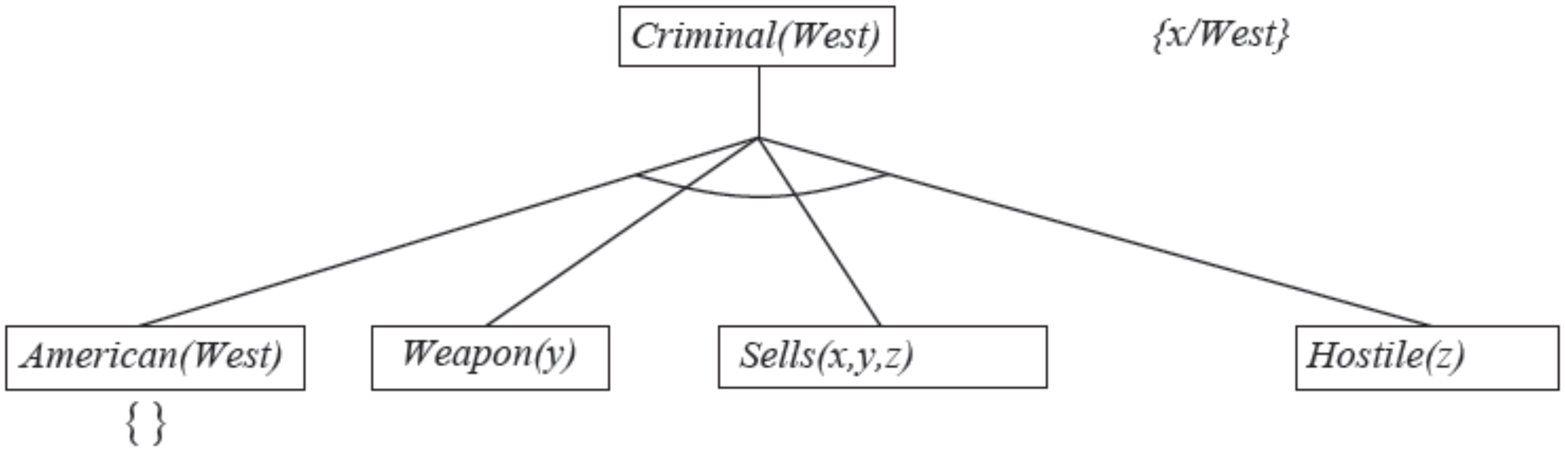
# Backward Chaining Example

$\boxed{Criminal(West)}$

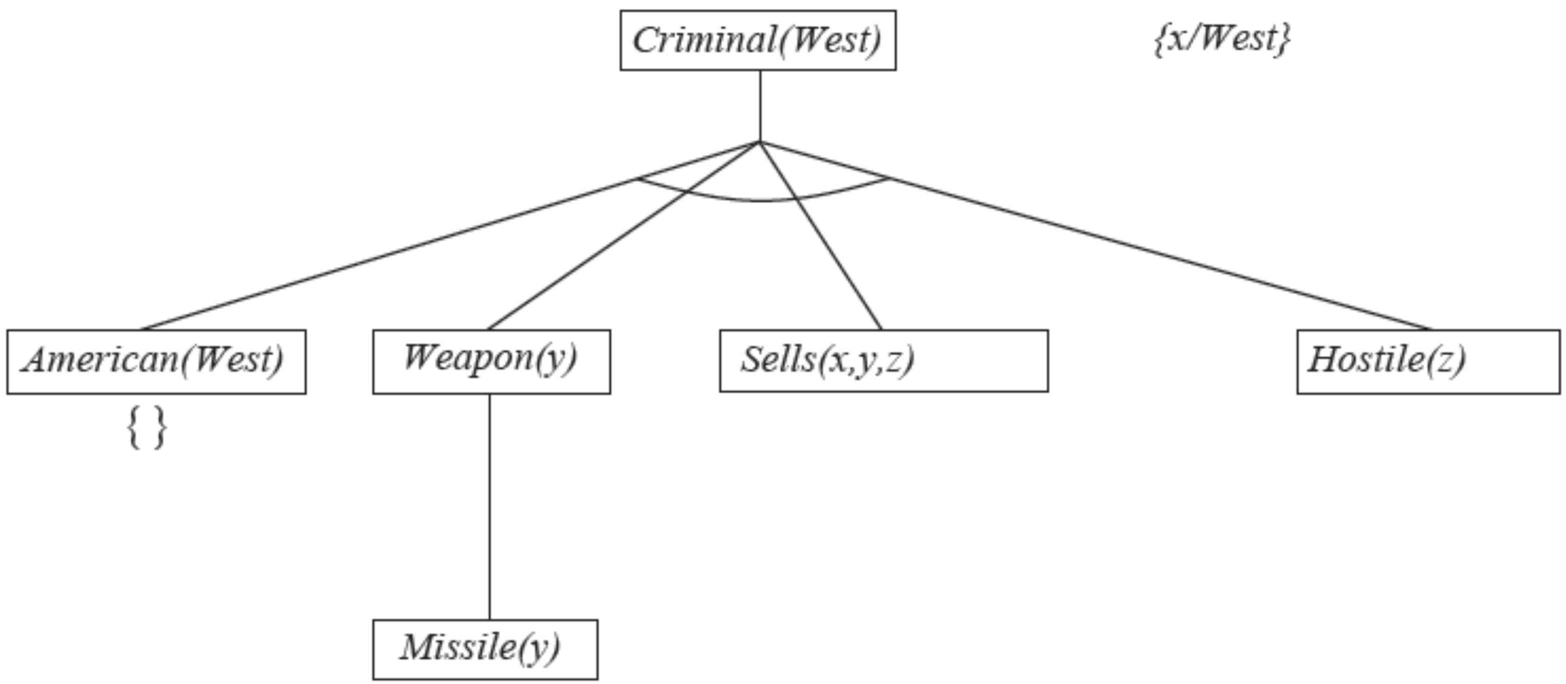
# Backward Chaining Example



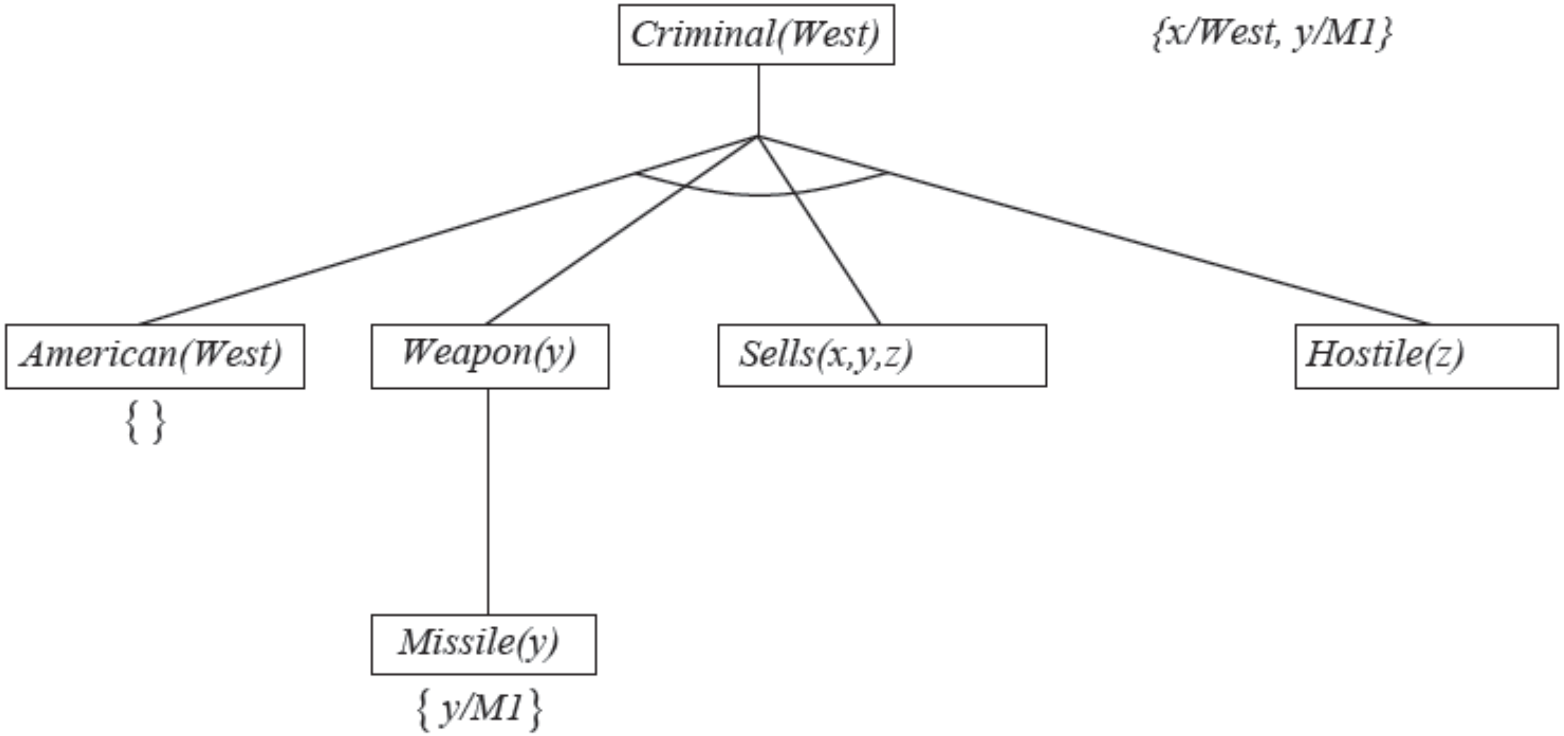
# Backward Chaining Example



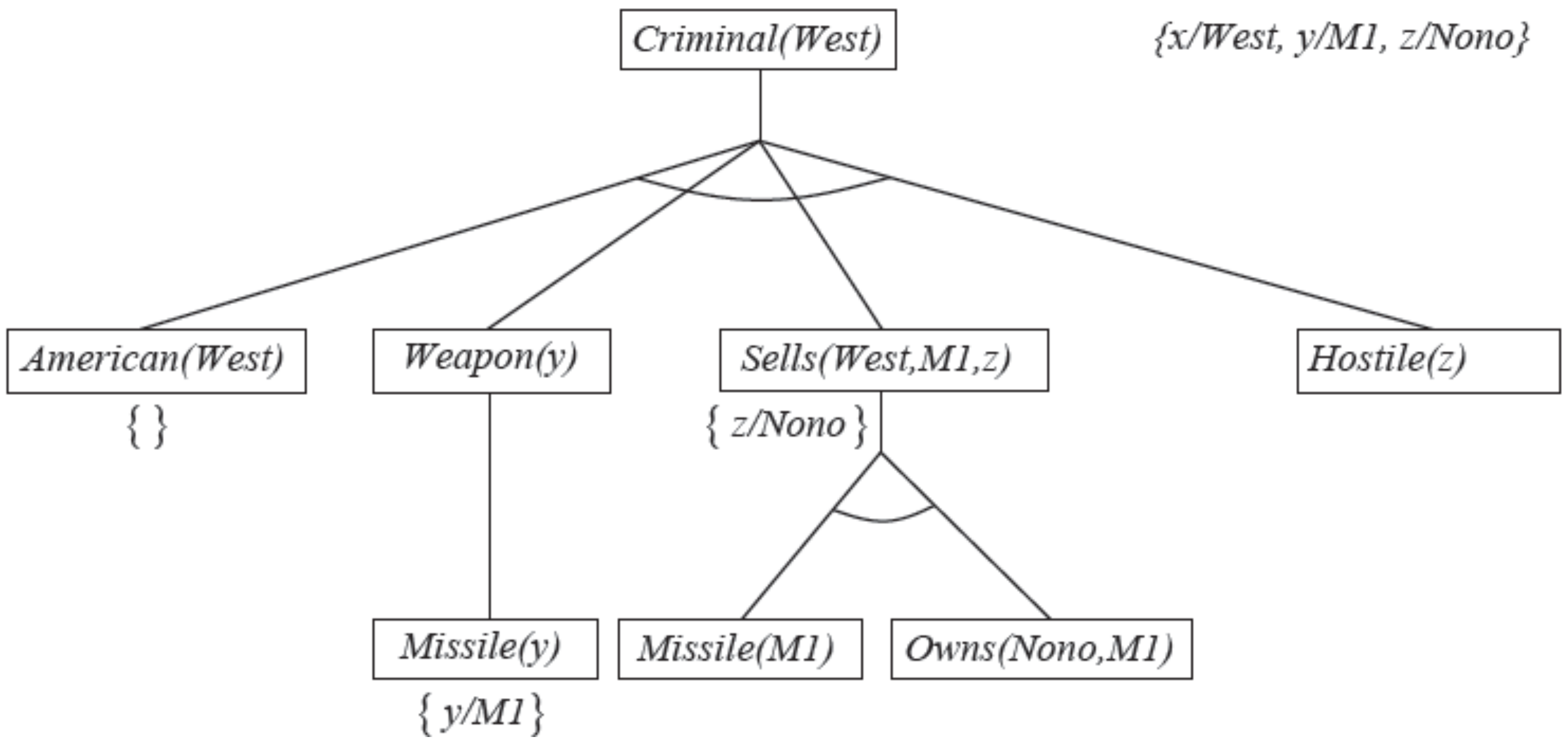
# Backward Chaining Example



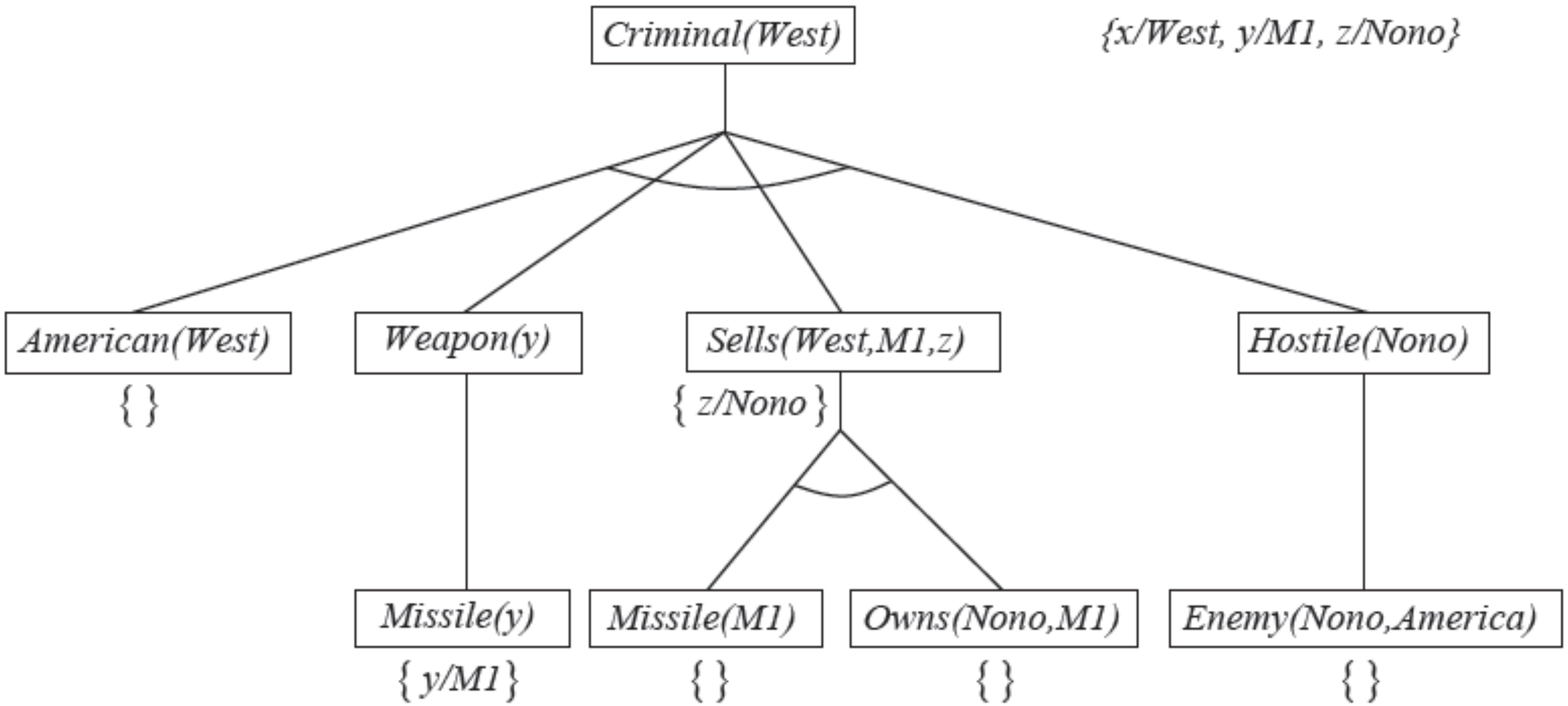
# Backward Chaining Example



# Backward Chaining Example



# Backward Chaining Example



# FOL Resolvents

- 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 Unify$(\ell_i, \neg m_j) = \theta$.

For example,

$$\frac{\neg \text{Rich}(x) \vee \text{Unhappy}(x), \quad \text{Rich}(\text{Ken})}{\text{Unhappy}(\text{Ken})}$$

with $\ell_i = \neg$Rich$(x)$, $m_j =$ Rich(Ken) and $\theta = \{x/\text{Ken}\}$

Apply resolution steps to CNF(KB $\wedge$ $\neg$query); complete for FOL.

# Conversion to CNF

Everyone who loves all animals is loved by someone:

$\forall x\ [\forall y\ Animal(y) \rightarrow Loves(x, y)] \rightarrow [\exists y Loves(y, x)]$

1. Eliminate biconditionals and implications

$\forall x\ [\neg \forall \mathrm{y}\ \neg Animal(y) \vee Loves(x, y)] \vee [\exists y\ Loves(y, x)]$

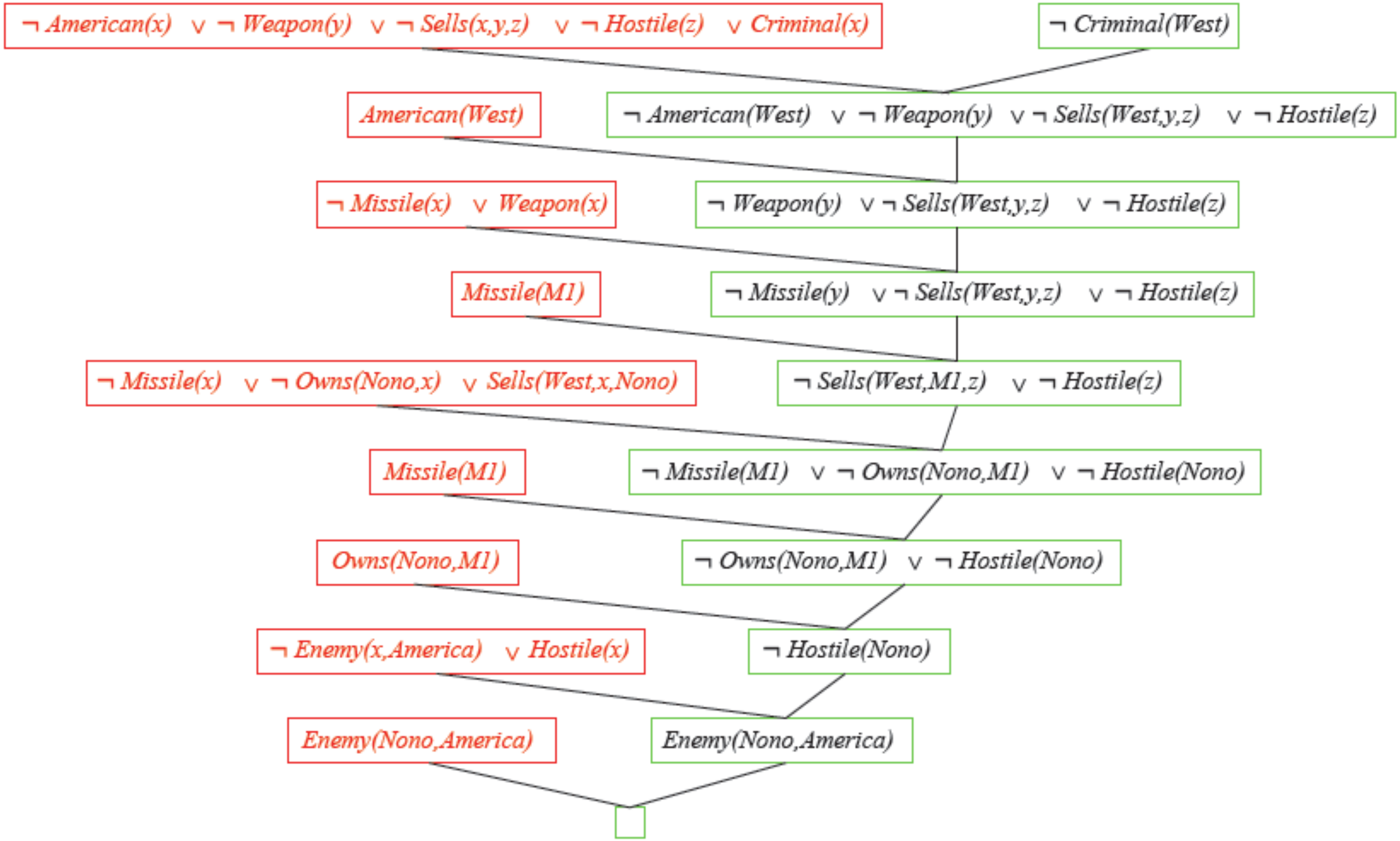
2. Move : inwards: $\neg \forall x, p \equiv \exists x \neg p, \neg \exists x, p \equiv \forall x \neg p$:

$\forall x[\exists y \neg(\neg Animal(y) \vee Loves(x, y))] \vee [\exists y\ Loves(y, x)]$

$\forall x[\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x, y)] \vee [\exists y\ Loves(y, x)]$

$\forall x[\exists y\ \ Animal(y) \wedge \neg Loves(x, y)] \vee [\exists y\ Loves(y, x)]$

# Resolution Example



# Inference Discussion

- Once we have facts that evaluate to T or F
- We can apply Forward Chaining, Backwards Chaining and Resolution
- The key is to understand Unification
- Very similar to Logical agents.