Artificial Intelligence

Arjang Fahim Ph.D.
Department of Computer Engineering and Science
California State University Long Beach

CECS 451, Summer 2020

Outline

Inference in FOL - Chapter 9

Inference in FOL – The Idea

- All rules of inference for propositional logic apply to first-order logic
- We just need to reduce FOL sentences to PL sentences by instantiating variables and removing quantifiers

Reduction of FOL to PL – The Idea

- Suppose the KB contains the following:

  ∀x King(x) ^ Greedy(x) ⇒ Evil(x)

  King(John) Greedy(John) Brother(Richard,John)

- How can we reduce this to PL?
- Let's instantiate the universal sentence in all possible ways:

  King(John) ^ Greedy(John) ⇒ Evil(John)

  King(Richard) ^ Greedy(Richard) ⇒ Evil(Richard)

  King(John) Greedy(John) Brother(Richard,John)

- The KB is *propositionalized*
  - Proposition symbols are King(John), Greedy(John), Evil(John), King(Richard), etc.

Reduction of FOL to PL – The Idea

- What about existential quantification, e.g.,

  $\exists x\ Crown(x) \wedge OnHead(x, John)$?

- Let's instantiate the sentence with a new constant that doesn't appear anywhere in the KB:

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

Propositionalization

- Every FOL KB can be *propositionalized* so as to preserve entailment
  - A ground sentence is entailed by the new KB iff it is entailed by the original KB

- **Idea:** propositionalize KB and query, apply resolution, return result

- **Problem:** with function symbols, there are infinitely many ground terms
  - For example, Father(X) yields Father(John), Father(Father(John)), Father(Father(Father(John))), etc.

Propositionalization

- **Theorem** (Herbrand 1930):
  - If a sentence α is entailed by an FOL KB, it is entailed by a *finite* subset of the propositionalized KB

- **Idea:** For n = 0 to Infinity do
  - Create a propositional KB by instantiating with depth-n terms
  - See if α is entailed by this KB

- **Problem:** works if α is entailed, loops if α is not entailed

- **Theorem** (Turing 1936, Church 1936):
  - Entailment for FOL is **semidecidable**: algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence

Inference in FOL

- *“All men are mortal. Socrates is a man; therefore, Socrates is mortal.”*
- Can we prove this without full propositionalization as an intermediate step?

Substitution

- **Substitution** of variables by *ground terms*:

SUBST({v/g},P)

- Result of SUBST({x/Harry, y/Sally}, Loves(x,y)):
  Loves(Harry,Sally)

- Result of SUBST({x/John}, King(x) ^ Greedy(x) ⇒ Evil(x)):
  King(John) ^ Greedy(John) ⇒ Evil(John)

Universal instantiation (UI)

- A universally quantified sentence entails every instantiation of it:
- $\forall v\ P(v)$

  $$\frac{\forall v\ P(v)}{\text{SUBST}(\{v/g\}, P(v))}$$

  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)

- An existentially quantified sentence entails the instantiation of that sentence with a new constant:

$$\frac{\exists v\ P(v)}{\mathrm{SUBST}(\{v/C\}, P(v))}$$

for any sentence P, variable v, and constant C that does not appear elsewhere in the knowledge base

- 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*

Generalized Modus Ponens (GMP)

Generalized Modus Ponens (GMP)

$$\frac{(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q),\ p_1',\ p_2',\ \ldots,\ p_n' \quad \text{such that } \mathrm{SUBST}(\theta, p_i) = \mathrm{SUBST}(\theta, p_i') \text{ for all } i}{\mathrm{SUBST}(\theta, q)}$$

- All variables assumed universally quantified

- **Example:**

  $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

  King(John) Greedy(John)   Brother(Richard,John)

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

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

  SUBST(θ,q) is Evil(John)

UNIFY(α,β) = θ means that **SUBST(θ, α) = SUBST(θ, β)**

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,Mary)	{x/Mary,
Knows(John,x)	Knows(y,Mother(y))	{y/John,
Knows(John,x)	Knows(x,Mary)	{x_1/John,
Knows(John,x)	Knows(y,z)	{y/John, x/z}

- Standardizing apart eliminates overlap of variables
- Most general unifier

Inference with GMP

$$(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q),\ p_1',\ p_2',\ \ldots,\ p_n'$$

such that **SUBST(θ, p_i)= SUBST(θ, p_i')** for all i

$$\overline{}$$

SUBST(θ,q)

- **Forward chaining**
  - Like search: keep proving new things and adding them to the KB until we can prove q
- **Backward chaining**
  - Find $p_1, \ldots, p_n$ such that knowing them would prove q
  - Recursively try to prove $p_1, \ldots, p_n$

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:

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

Nono has some missiles

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

$Owns(Nono,M_1) \wedge Missile(M_1)$

All of its missiles were sold to it by Colonel West

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

Missiles are weapons:

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

An enemy of America counts as “hostile”:

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

West is American

$American(West)$

The country Nono is an enemy of America

$Enemy(Nono,America)$

- 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

Forward chaining proof

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

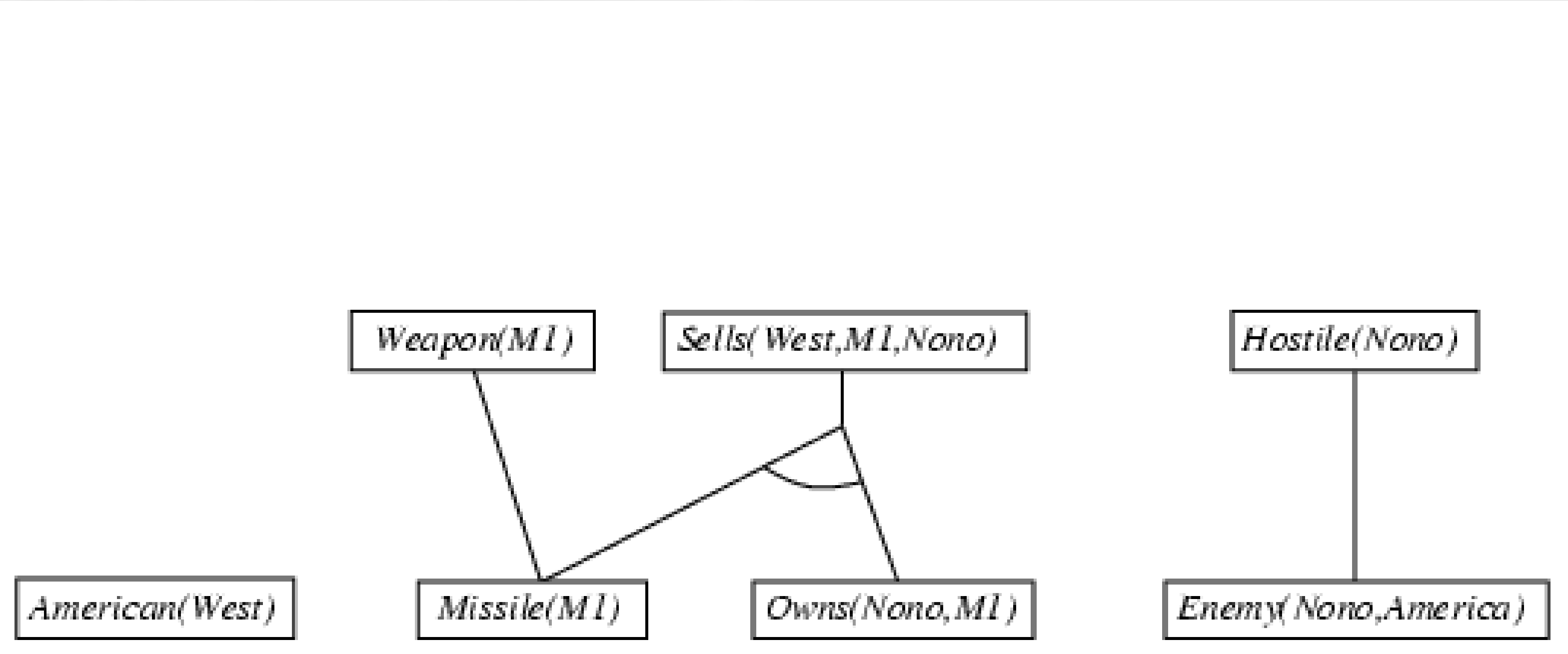
$Owns(Nono,M_1) \wedge Missile(M_1)$

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

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

$American(West)$ $Enemy(Nono,America)$

Forward chaining proof



American(x) ^ Weapon(y) ^ Sells(x,y,z) ^ Hostile(z) ⇒ Criminal(x)

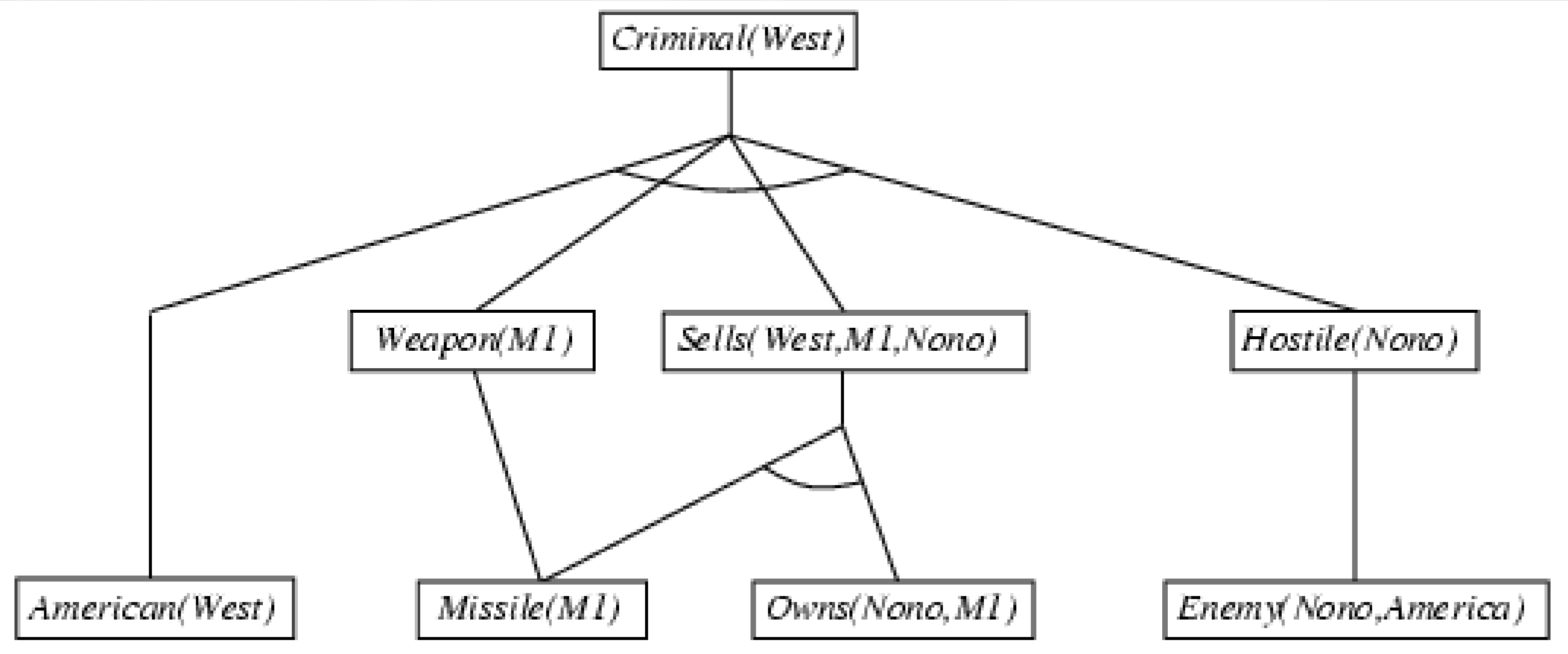
Owns(Nono,M_1) ^ Missile(M_1)

Missile(x) ^ Owns(Nono,x) ⇒ Sells(West,x,Nono)

Missile(x) ⇒ Weapon(x) Enemy(x,America) ⇒ Hostile(x)

American(West) Enemy(Nono,America)

Forward chaining proof



American(x) ^ Weapon(y) ^ Sells(x,y,z) ^ Hostile(z) ⇒ Criminal(x)

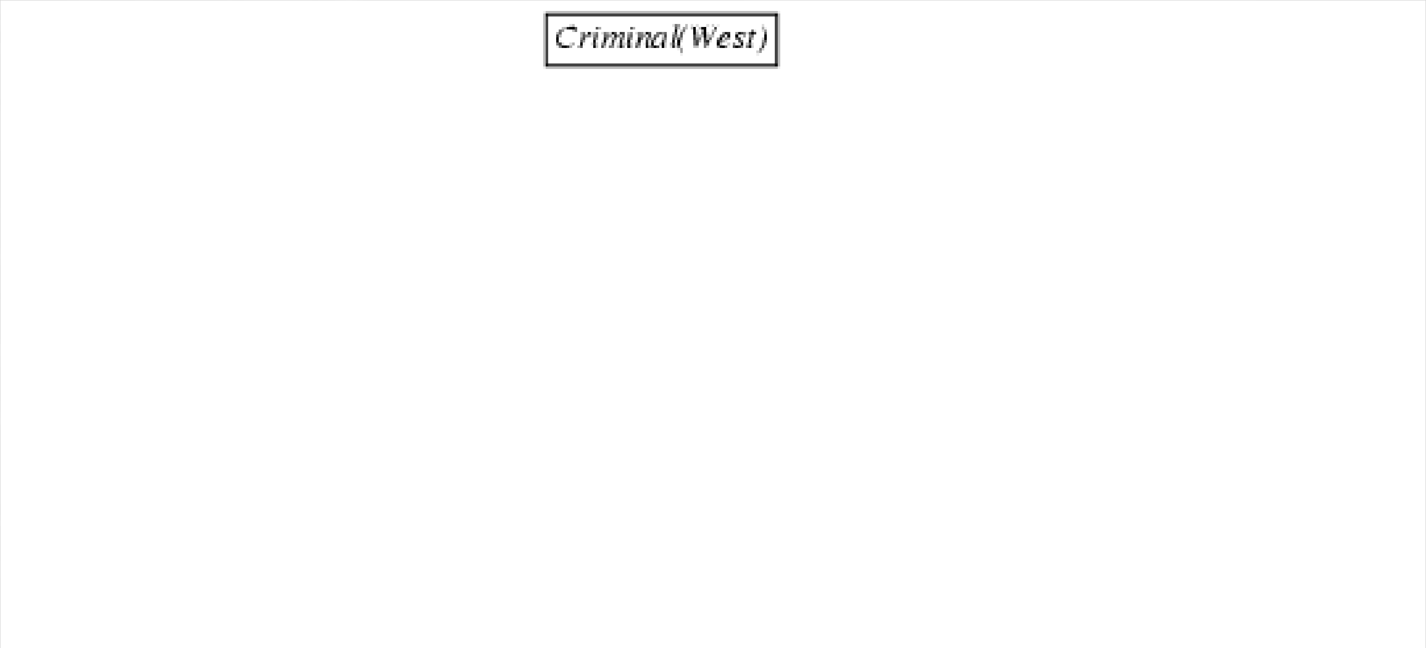
Owns(Nono,M_1) ^ Missile(M_1)

Missile(x) ^ Owns(Nono,x) ⇒ Sells(West,x,Nono)

Missile(x) ⇒ Weapon(x)   Enemy(x,America) ⇒ Hostile(x)

American(West)   Enemy(Nono,America)

Backward chaining example



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

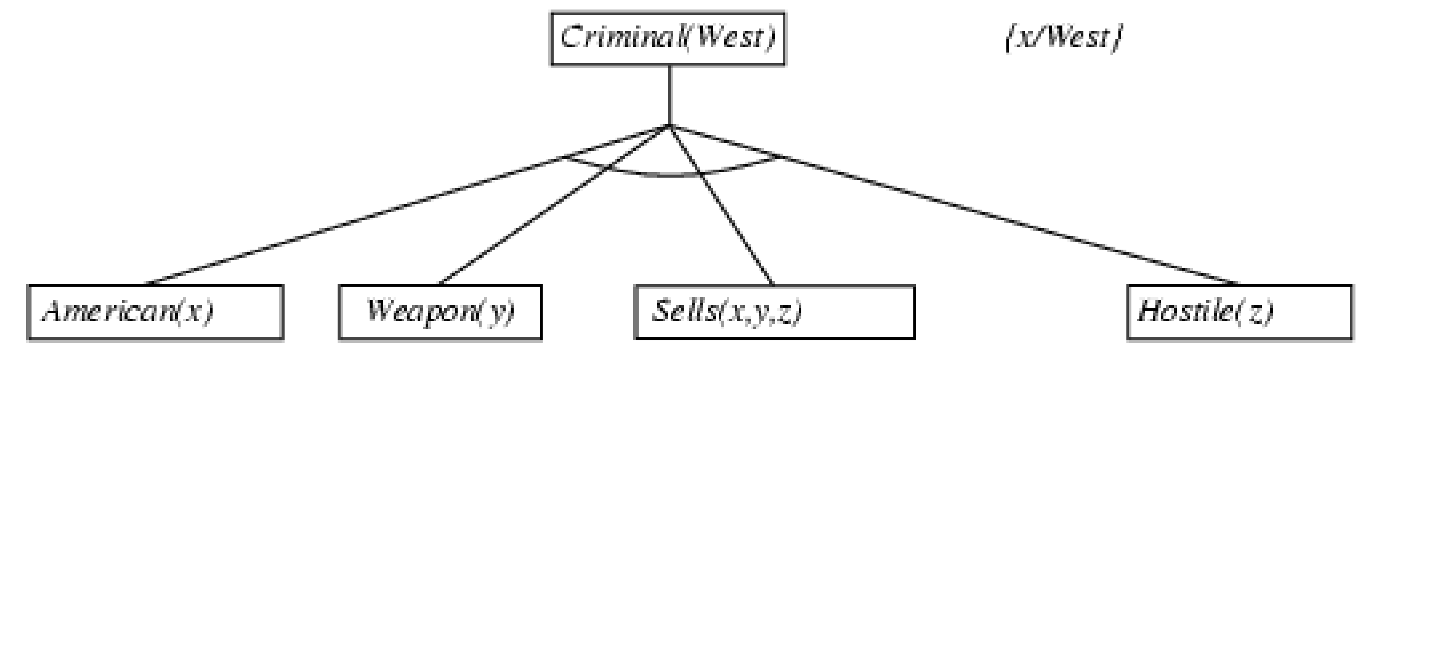
Owns(Nono,M_1) $\land$ Missile(M_1)

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

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

American(West)   Enemy(Nono,America)

Backward chaining example



American(x) ^ Weapon(y) ^ Sells(x,y,z) ^ Hostile(z) ⇒ Criminal(x)

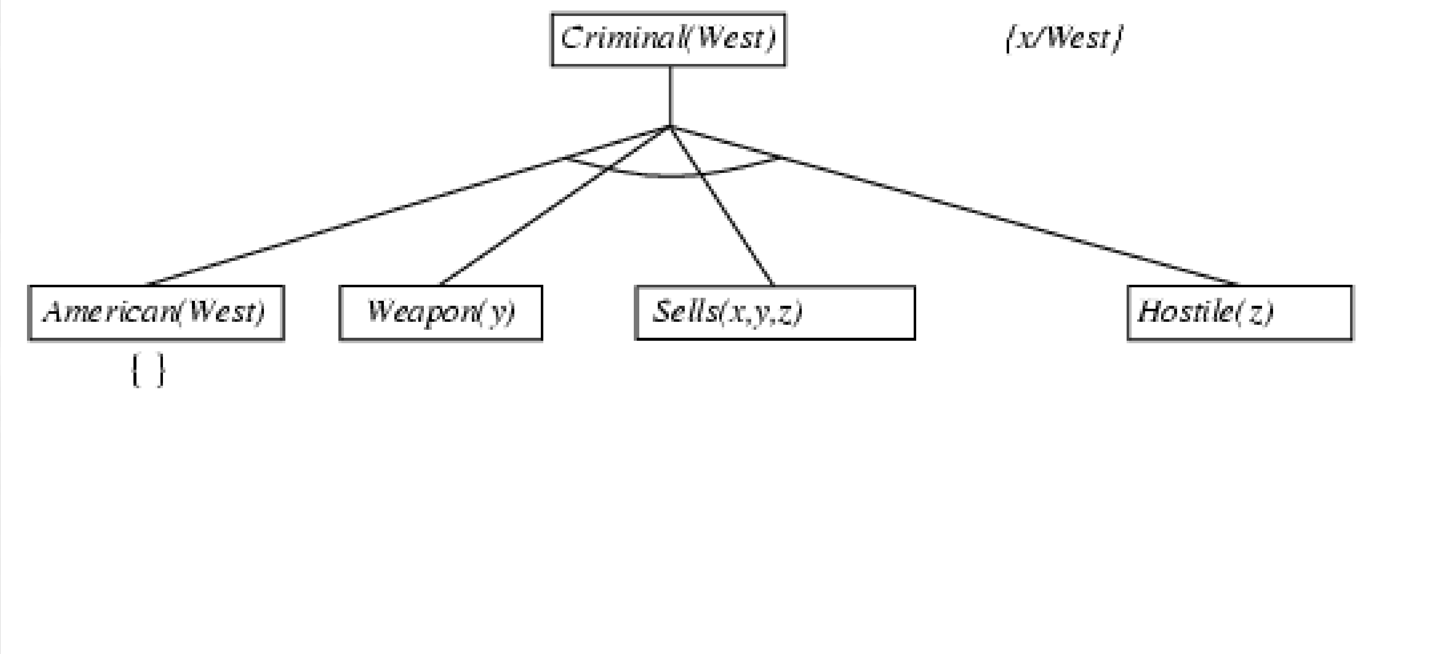
Owns(Nono,M_1) ^ Missile(M_1)

Missile(x) ^ Owns(Nono,x) ⇒ Sells(West,x,Nono)

Missile(x) ⇒ Weapon(x)   Enemy(x,America) ⇒ Hostile(x)

American(West)   Enemy(Nono,America)

Backward chaining example



American(x) ^ Weapon(y) ^ Sells(x,y,z) ^ Hostile(z) ⇒ Criminal(x)

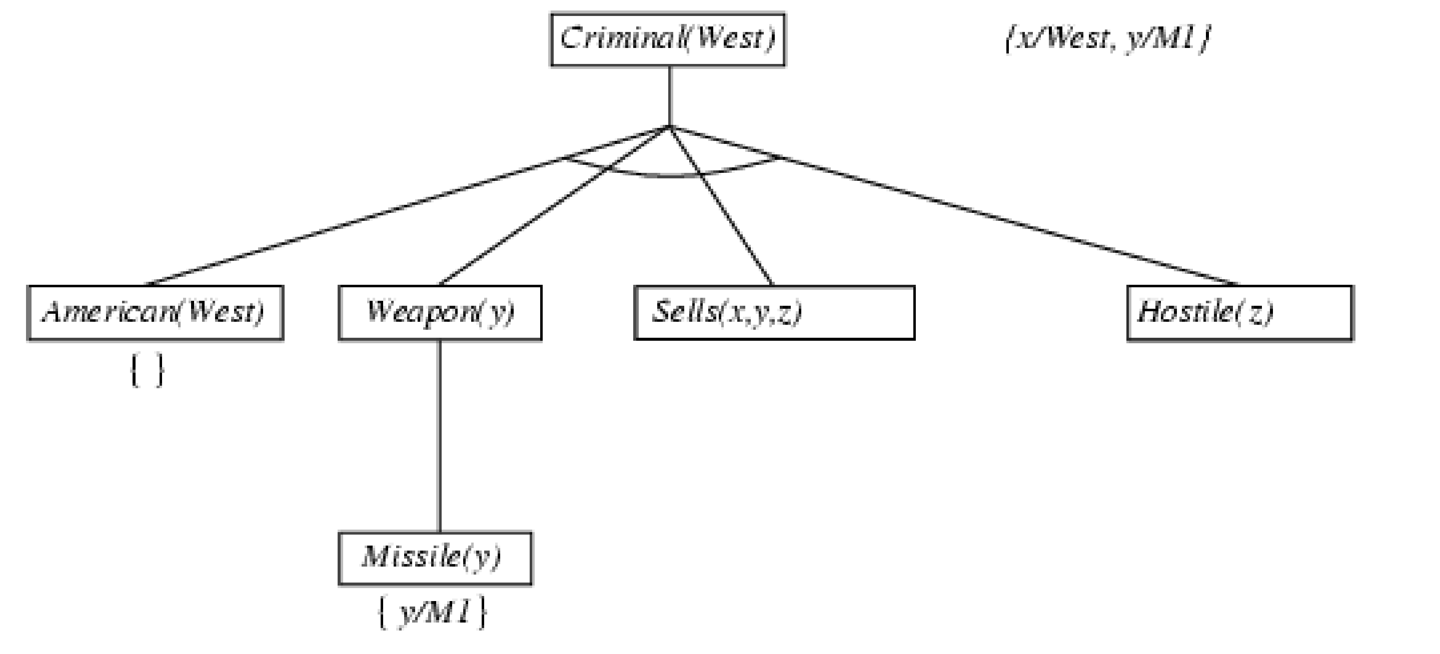
Owns(Nono,M_1) ^ Missile(M_1)

Missile(x) ^ Owns(Nono,x) ⇒ Sells(West,x,Nono)

Missile(x) ⇒ Weapon(x)   Enemy(x,America) ⇒ Hostile(x)

American(West)   Enemy(Nono,America)

Backward chaining example



American(x) ^ Weapon(y) ^ Sells(x,y,z) ^ Hostile(z) ⇒ Criminal(x)

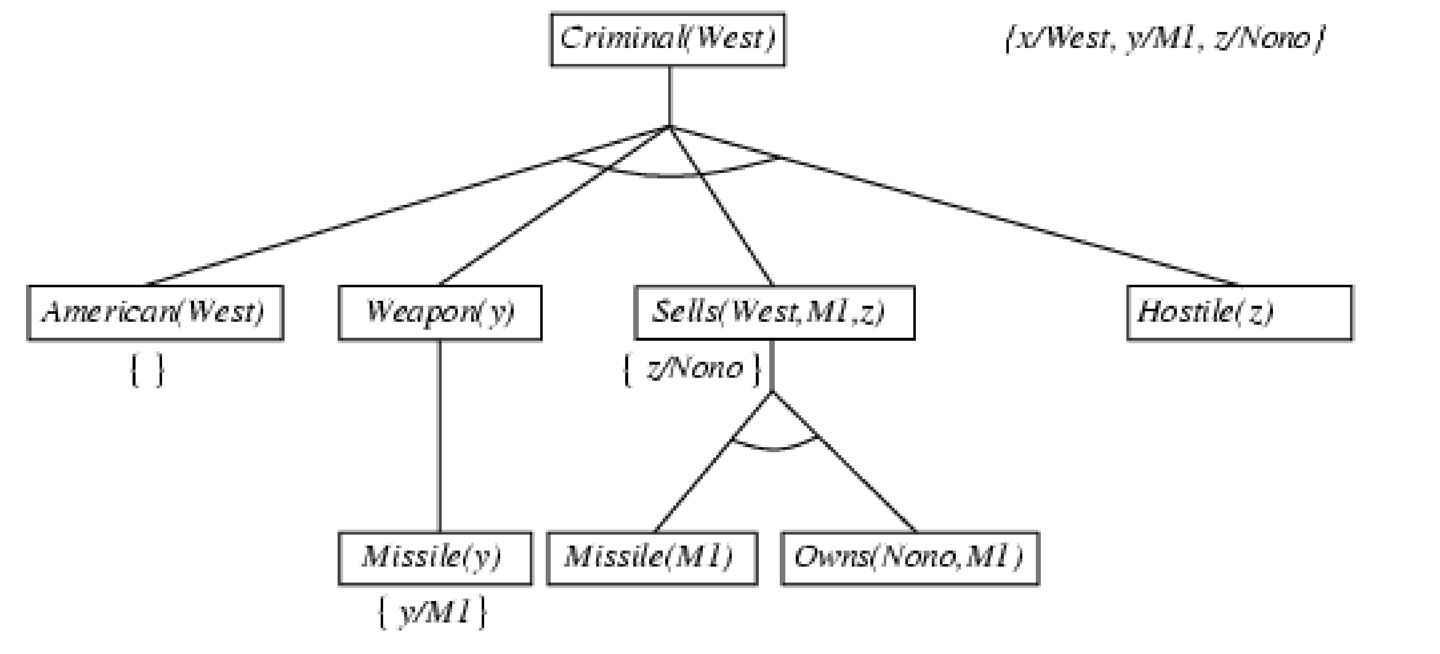
Owns(Nono,M_1) ^ Missile(M_1)

Missile(x) ^ Owns(Nono,x) ⇒ Sells(West,x,Nono)

Missile(x) ⇒ Weapon(x)   Enemy(x,America) ⇒ Hostile(x)

American(West)   Enemy(Nono,America)

Backward chaining example



American(x) ^ Weapon(y) ^ Sells(x,y,z) ^ Hostile(z) ⇒ Criminal(x)

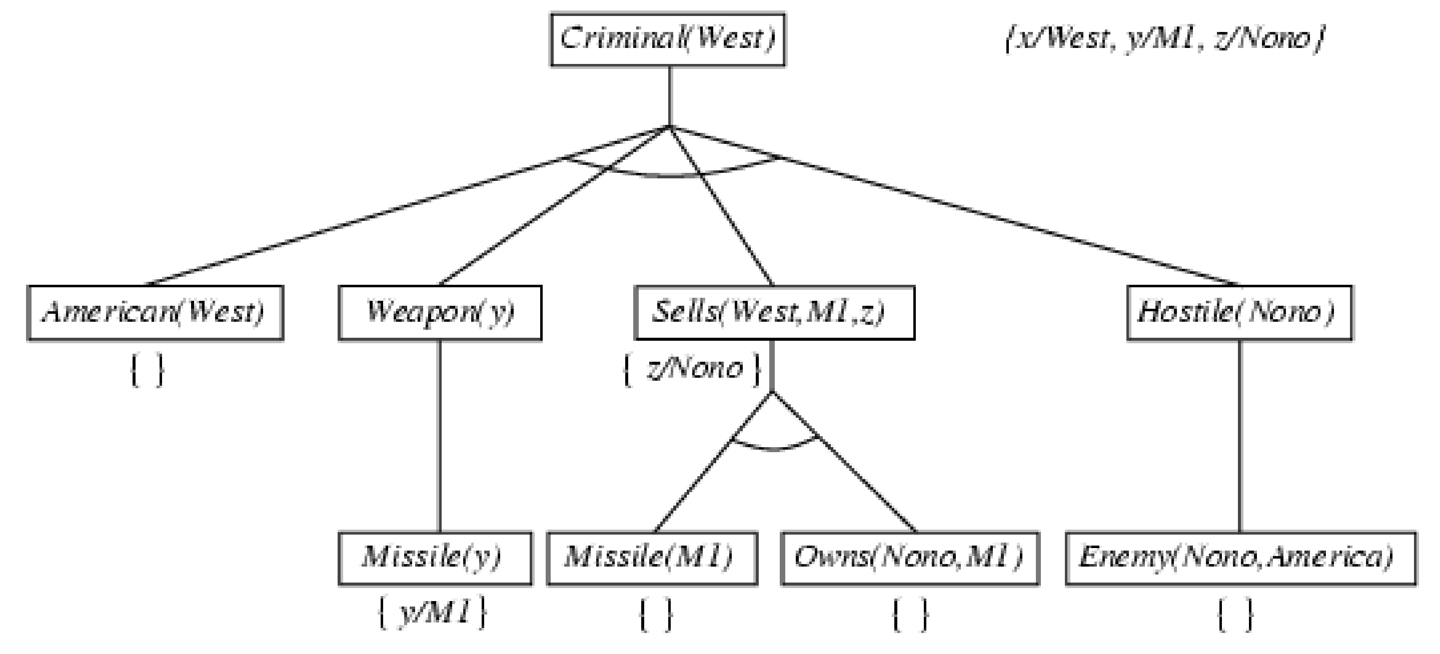
Owns(Nono,M_1) ^ Missile(M_1)

Missile(x) ^ Owns(Nono,x) ⇒ Sells(West,x,Nono)

Missile(x) ⇒ Weapon(x)　　Enemy(x,America) ⇒ Hostile(x)

American(West)　　Enemy(Nono,America)

Backward chaining example



American(x) ^ Weapon(y) ^ Sells(x,y,z) ^ Hostile(z) ⇒ Criminal(x)

Owns(Nono,M_1) ^ Missile(M_1)

Missile(x) ^ Owns(Nono,x) ⇒ Sells(West,x,Nono)

Missile(x) ⇒ Weapon(x)   Enemy(x,America) ⇒ Hostile(x)

American(West)   Enemy(Nono,America)

Backward chaining algorithm

function FOL-BC-ASK(KB, $goals$, θ) **returns** a set of substitutions
  inputs: KB, a knowledge base
    $goals$, a list of conjuncts forming a query (θ already applied)
    θ, the current substitution, initially the empty substitution { }
  local variables: $answers$, a set of substitutions, initially empty

  if $goals$ is empty **then return** $\{\theta\}$
  $q' \leftarrow$ SUBST(θ, FIRST($goals$))
  for each sentence r **in** KB
    where STANDARDIZE-APART(r) = ($p_1 \land \ldots \land p_n \Rightarrow q$)
    and $\theta' \leftarrow$ UNIFY(q, q') succeeds
   $new\_goals \leftarrow [p_1, \ldots, p_n | \text{REST}(goals)]$
   $answers \leftarrow$ FOL-BC-ASK(KB, new_goals, COMPOSE(θ', θ)) $\cup$ $answers$
  return $answers$

Resolution: FOL version

$$\frac{p_1 \vee \dots \vee p_k, \qquad q_1 \vee \dots \vee q_n \quad \text{such that } \mathrm{UNIFY}(p_i, \neg q_j) = \theta}{\mathrm{SUBST}(\theta, p_1 \vee \dots \vee p_{i-1} \vee p_{i+1} \vee \dots \vee p_k \vee q_1 \vee \dots \vee q_{j-1} \vee q_{j+1} \vee \dots \vee q_n)}$$

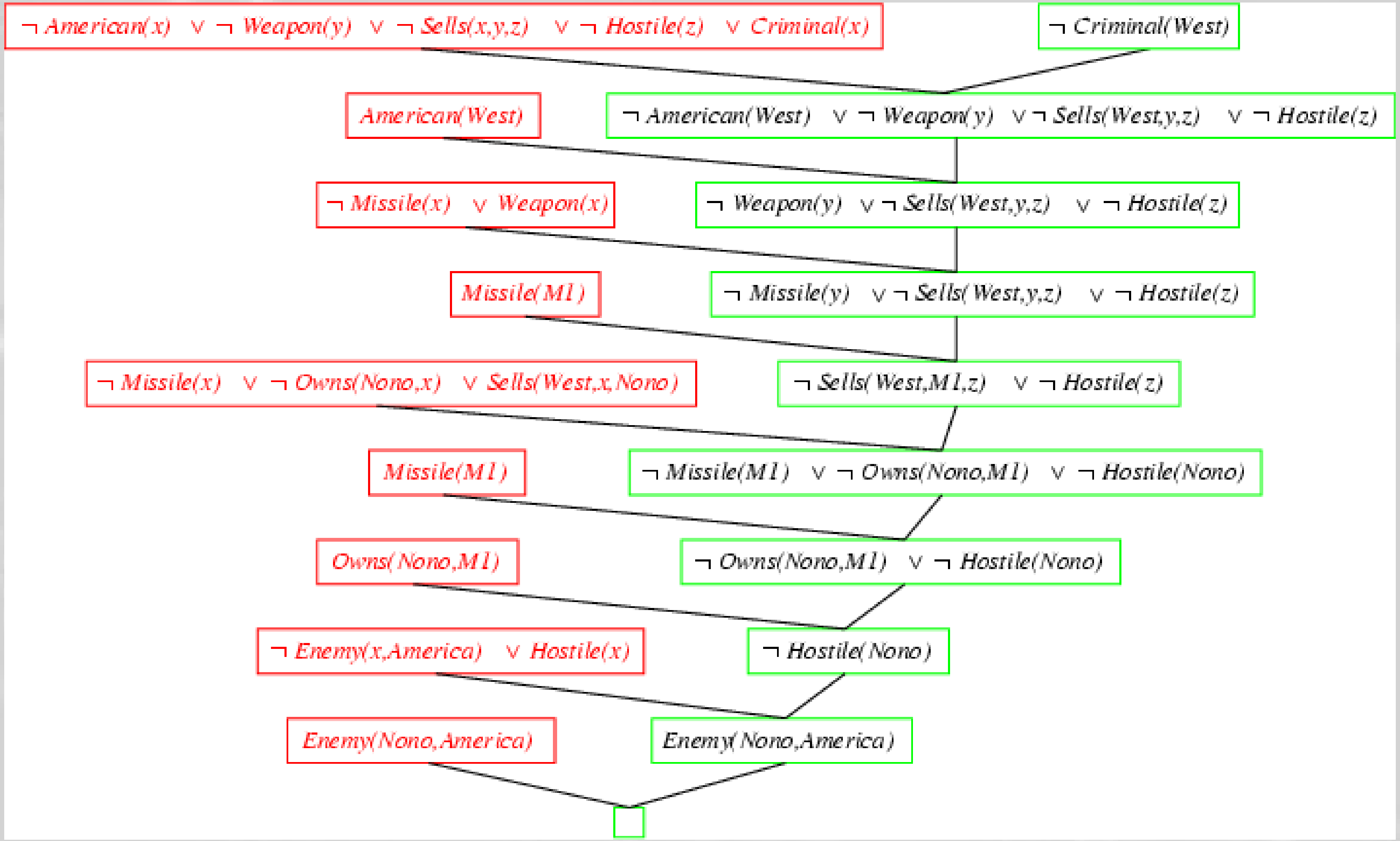
- For example,

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

  with $\theta = \{x/Ken\}$

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

Resolution proof: definite clauses



Logic programming: Prolog

- **FOL:**

  King(x) ^ Greedy(x) ⇒ Evil(x)

  Greedy(y)

  King(John)

- **Prolog:**

```
evil(X) :- king(X), greedy(X).
greedy(Y).
king(john).
```

- Closed-world assumption:
  - Every constant refers to a unique object
  - Atomic sentences not in the database are assumed to be false
- Inference by backward chaining, clauses are tried in the order in which they are listed in the program, and literals (predicates) are tried from left to right

Prolog example

```
parent(abraham,ishmael).
parent(abraham,isaac).
parent(isaac,esau).
parent(isaac,jacob).

grandparent(X,Y) :- parent(X,Z), parent(Z,Y).
descendant(X,Y) :- parent(Y,X).
descendant(X,Y) :- parent(Z,X), descendant(Z,Y).


? parent(david,solomon).
? parent(abraham,X).
? grandparent(X,Y).
? descendant(X,abraham).
```

Prolog example

```
parent(abraham,ishmael).
parent(abraham,isaac).
parent(isaac,esau).
parent(isaac,jacob).
```

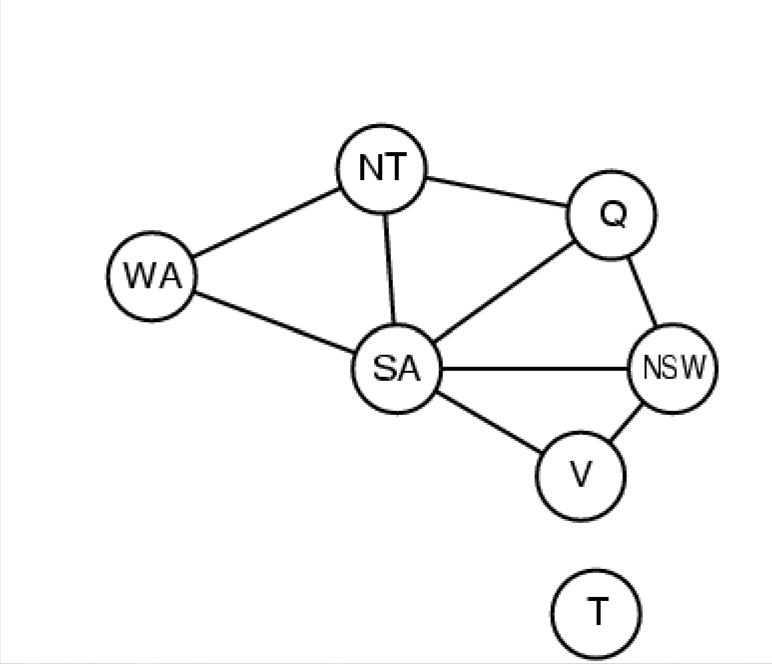
- What if we wrote the definition of **descendant** like this:

```
descendant(X,Y) :- descendant(Z,Y), parent(Z,X).
descendant(X,Y) :- parent(Y,X).

? descendant(W,abraham).
```

- Backward chaining would go into an infinite loop!
  - Prolog inference is **not complete**, so the ordering of the clauses and the literals is really important

Graph coloring




```
colorable(Wa,Nt,Sa,Q,Nsw,V) :-
diff(Wa,Nt), diff(Wa,Sa), diff(Nt,Q), diff(Nt,Sa), diff(Q,Nsw),
diff(Q,Sa), diff(Nsw,V), diff(Nsw,Sa), diff(V,Sa).

diff(red,blue).    diff(red,green).   diff(green,red).
diff(green,blue).  diff(blue,red).    diff(blue,green).
```

Prolog lists

- Appending two lists to produce a third:

```
append([],Y,Y).
append([X|L],Y,[X|Z]) :- append(L,Y,Z).
```

- query: `append(A,B,[1,2])`

- answers:

```
A=[]     B=[1,2]
A=[1]    B=[2]
A=[1,2]  B=[]
```