Inference

Chapter 8 *(some of the materials are from Welling at UCI)*

How do you do Inference in FOL?

- In Propositional logic:
  - Resolution
  - Forward Chaining
  - Backward chaining
- FOL
  - Reduce to Propositional
  - Adapt Propositional logic algorithms to FOL

Universal instantiation (UI)

- Notation: Subst({v/g}, α) substitute g for variable v in sentence α
- 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:

$\{x/John\}\quad King(John) \wedge Greedy(John) \Rightarrow Evil(John)$

$\{x/Richard\}\ King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$

Existential instantiation (EI)

- $\exists x\ Crown(x) \wedge OnHead(x, John)$ yields.
- We need to add a constant that doesn't exist anywhere in KB
- E.g., C_1 is a new constant symbol, called a Skolem constant

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

Reduction to propositional form

Suppose the KB contains the following:

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

King(John)

Greedy(John)

Brother(Richard, John)

- Instantiating the universal sentence in all possible ways, we have:

(there are only two ground terms: John and Richard)

King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)

King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ Evil(Richard)

King(John)

Greedy(John)

Brother(Richard, John)

Reduction continued

- Every FOL KB can be propositionalized so as to preserve entailment: A ground sentence is entailed by new KB iff entailed by original KB
- Problems?
  - function symbols, there are infinitely many ground terms, e.g., *Father*(*Father*(*Father*(*John*))), etc
  - Propositionalization generates lots of irrelevant sentences, why is this a problem?

How do you do Inference in FOL?

- In Propositional logic:
  - Resolution
  - Forward Chaining
  - Backward chaining
- FOL
  - Reduce to Propositional
  - Adapt Propositional logic algorithms to FOL

    UNIFICATION

Unification

- Subst(θ, p) = result of substituting θ into sentence p
- Unification algorithm:
- takes 2 sentences p and q and returns a unifier if one exists

  Unify(p,q) = θ where Subst(θ, p) = Subst(θ, q)
- Example:

  p = Knows(John,x)

  q = Knows(John, Jane)

  Unification(p, q) = {x/Jane}

Unification examples

- simple example: query = Knows(John,x), i.e., who does John know?

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}

- Last unification fails: only because x can't take values John and OJ at the same time
- Simple solution: Standardizing apart eliminates overlap of variables, e.g., Knows(z,OJ)

Generalized Modus Ponens (GMP)

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

where we can unify p_i' and p_i for all *i*

Example:

p_1' is *King*(*John*) $\quad$ p_1 is *King*(*x*)

p_2' is *Greedy*(*y*) $\quad$ p_2 is *Greedy*(*x*)

θ is {x/John,y/John} $\quad$ q is *Evil*(*x*)

Subst(θ,q) is *Evil*(*John*)

$\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
King(John)
$\forall y$ Greedy(y)
Brother(Richard, John)

How do you do Inference in FOL?

- In Propositional logic:
  - Resolution
  - Forward Chaining
  - Backward chaining
- FOL
  - Reduce to Propositional
  - Adapt Propositional logic algorithms to FOL

Inference appoaches in FOL

- **Forward-chaining**
  - Uses GMP to add new atomic sentences
  - Useful for systems that make inferences as information streams in
  - Requires KB to be in form of first-order definite clauses
- **Backward-chaining**
  - Works backwards from a query to try to construct a proof
  - Can suffer from repeated states and incompleteness
  - Useful for query-driven inference
- **Resolution-based inference (FOL)**
  - Can be used to confirm or refute a sentence p
  - Requires FOL KB to be reduced to CNF

Knowledge Base in FOL

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.

Knowledge Base in FOL

… 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, i.e., $\exists x\ Owns(Nono,x) \wedge Missile(x)$:

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

… 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, who is American …

$American(West)$

The country Nono, an enemy of America …

$Enemy(Nono,America)$

Forward chaining proof

American(West) | Missile(M1) | Owns(Nono,M1) | Enemy(Nono,America)

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

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

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

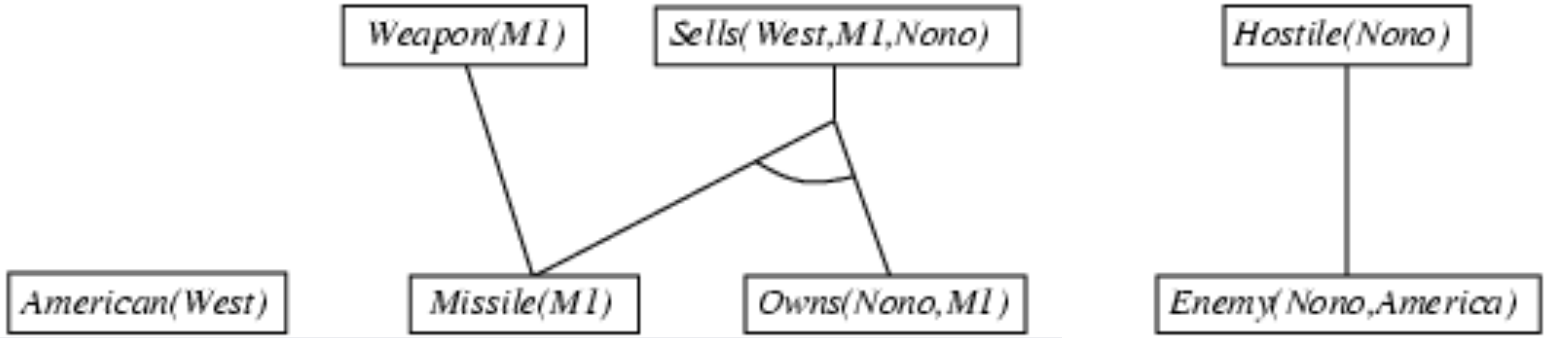
$Missile(x) \Rightarrow Weapon(x)$

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

$American(West)$

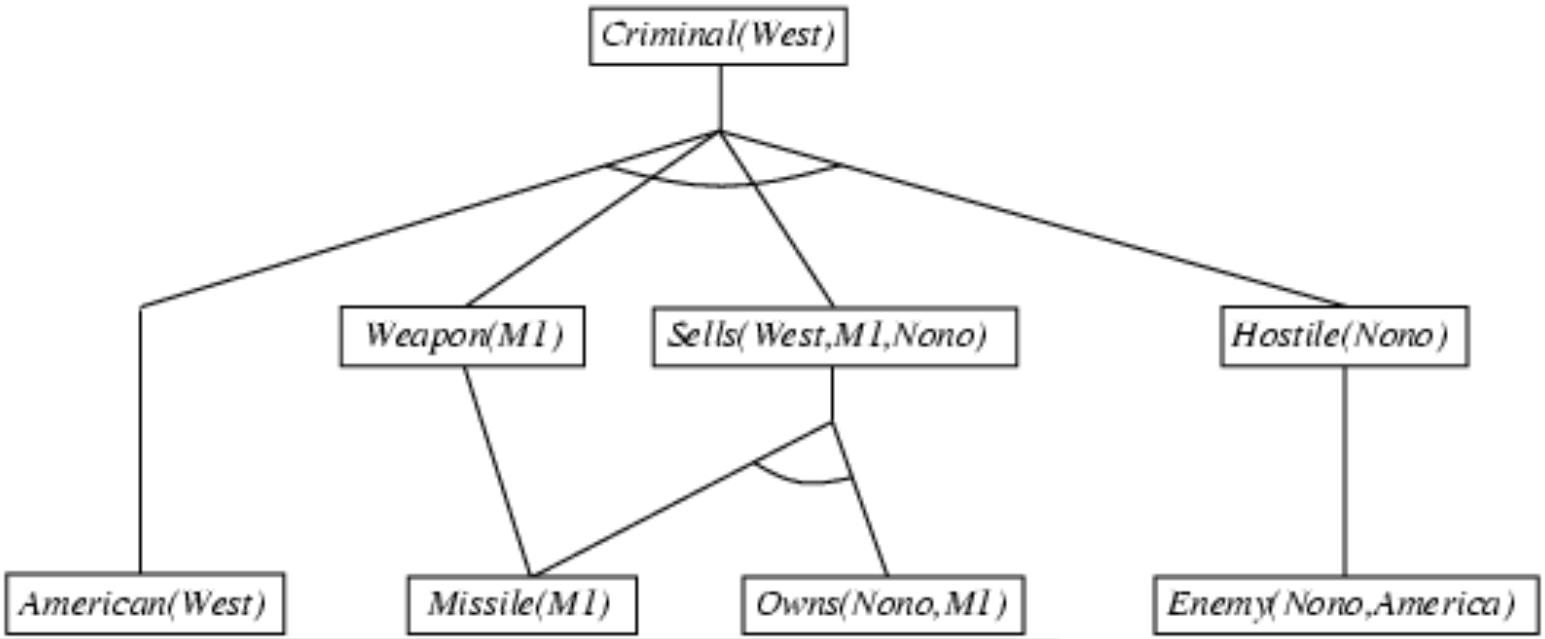
$Enemy(Nono,America)$

Forward chaining proof



American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono,M_1) and 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) and 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)

Properties of forward chaining

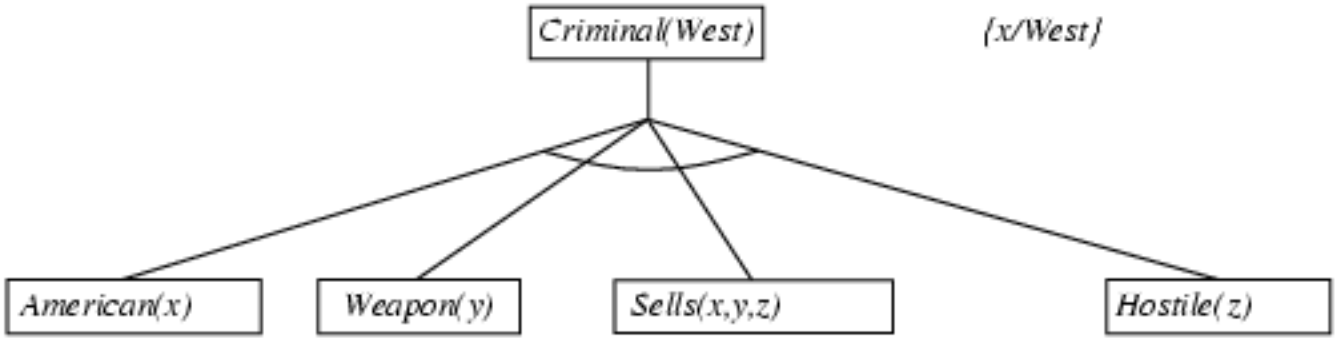
- Sound and complete for first-order definite clauses
- Datalog = first-order definite clauses + no functions
- FC terminates for Datalog in finite number of iterations
- May not terminate in general if α is not entailed

Backward chaining example

$\boxed{Criminal(West)}$

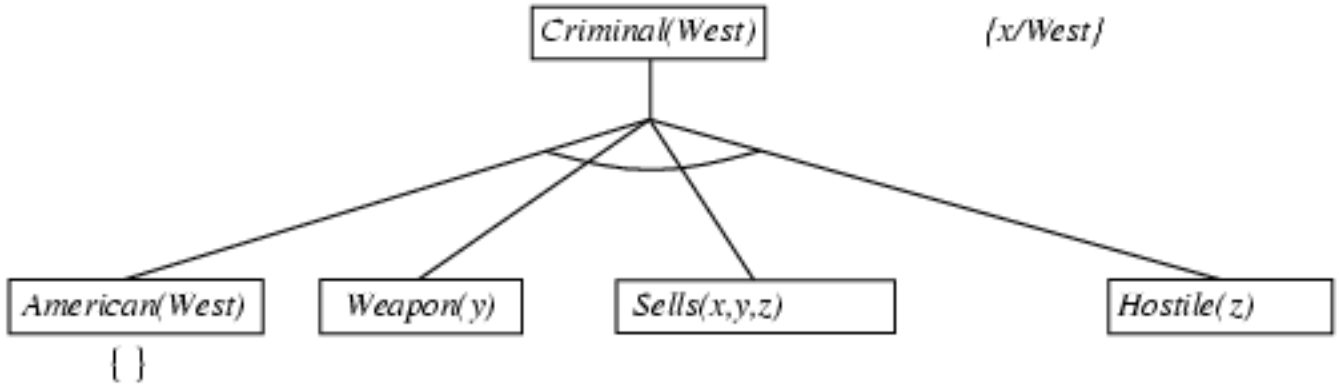
$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$
$Owns(Nono,M_1)$ and $Missile(M_1)$
$Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$
$Missile(x) \Rightarrow Weapon(x)$
$Enemy(x,America) \Rightarrow Hostile(x)$
$American(West)$
$Enemy(Nono,America)$

Backward chaining example



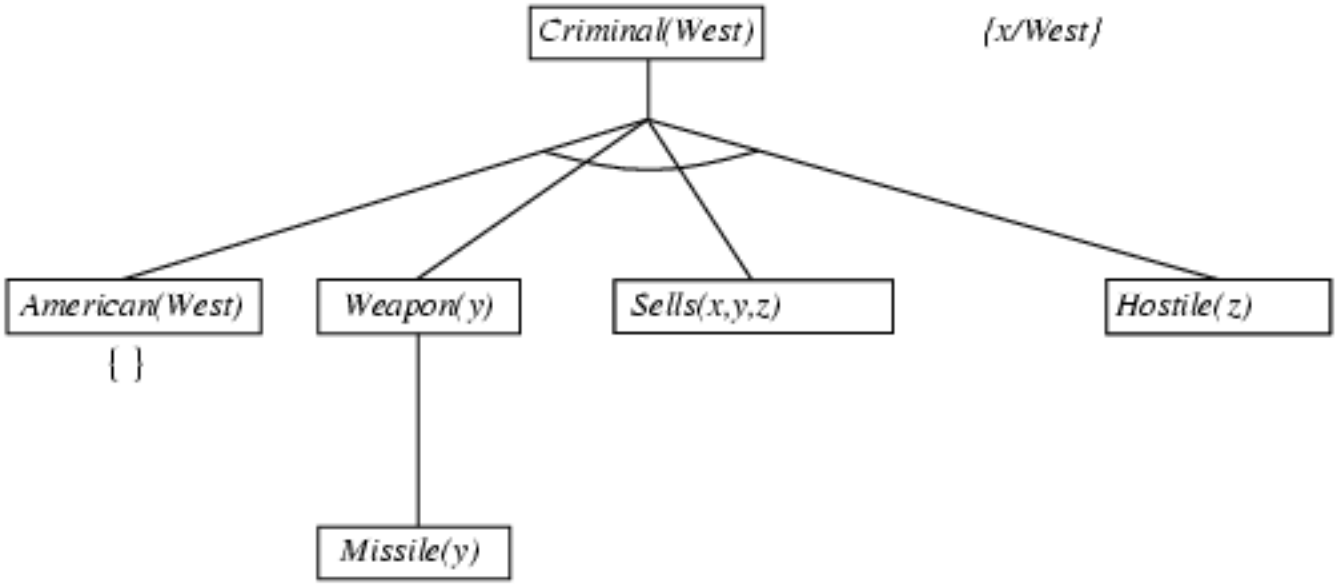
American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono,M_1) and Missile(M_1)
Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x,America) ⇒ Hostile(x)
American(West)
Enemy(Nono,America)

Backward chaining example



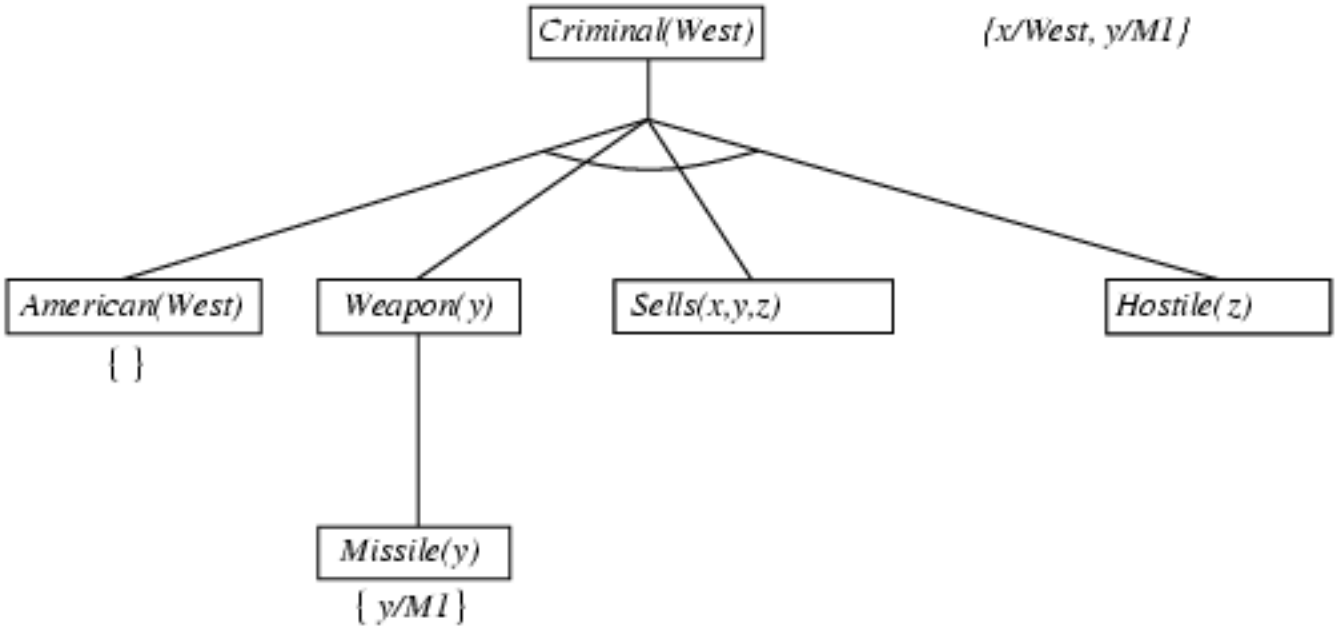
American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono,M_1) and Missile(M_1)
Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x,America) ⇒ Hostile(x)
American(West)
Enemy(Nono,America)

Backward chaining example



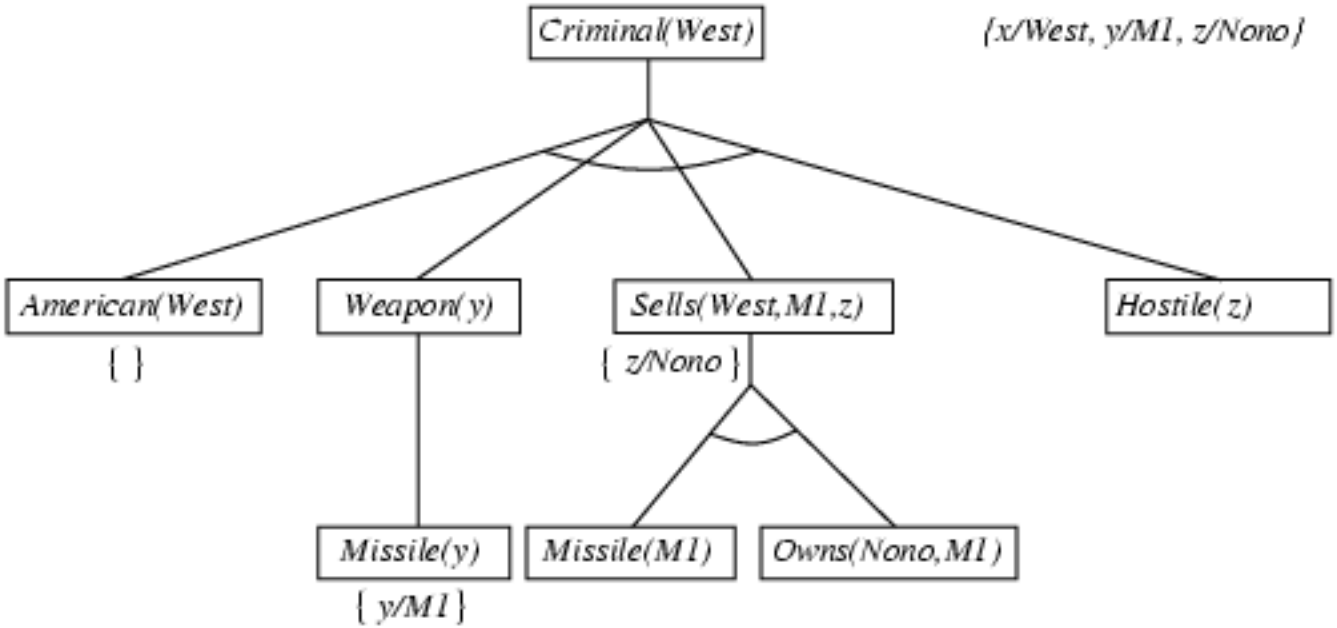
American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono,M_1) and Missile(M_1)
Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x,America) ⇒ Hostile(x)
American(West)
Enemy(Nono,America)

Backward chaining example



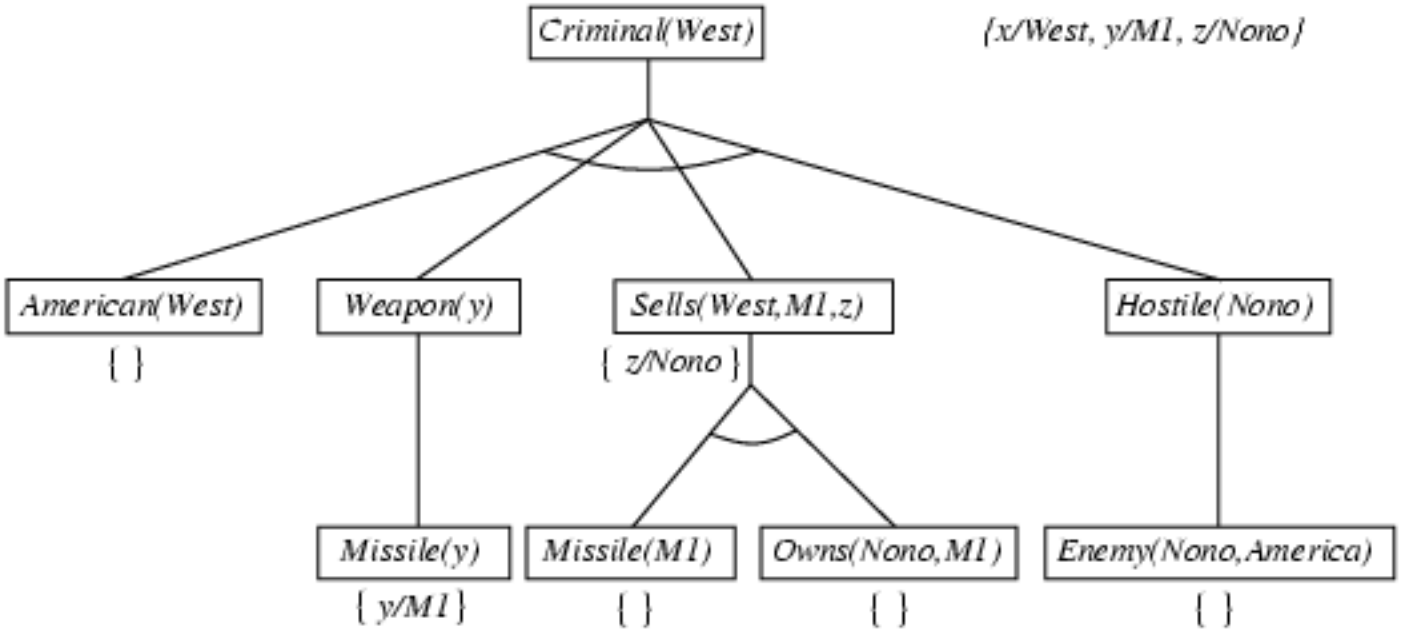
American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono,M_1) and Missile(M_1)
Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)
Missile(x) ⇒ Weapon(x)
Enemy(x,America) ⇒ Hostile(x)
American(West)
Enemy(Nono,America)

Backward chaining example



American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono,M_1) and 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) and 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)

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 (memorization)

Resolution in FOL

- Full first-order version:

$$\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n$$

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

where `Unification`$(\ell_i, \neg m_j) = \theta$.

- For example,

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

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

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

Converting FOL sentences to CNF

Original sentence:

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

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

2. Move ¬ inwards:

Recall: $\neg \forall x\ p \equiv \exists x\ \neg p$, $\neg \exists x\ p \equiv \forall x\ \neg p$

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

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

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

Conversion to CNF contd.

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

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

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

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

$\forall x\ [Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$

(reason: animal y could be a different animal for each x.)

Conversion to CNF contd.

5. Drop universal quantifiers:

$[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$

(all remaining variables assumed to be universally quantified)

6. Distribute $\vee$ over $\wedge$:

$[Animal(F(x)) \vee Loves(G(x),x)] \wedge [\neg Loves(x,F(x)) \vee Loves(G(x),x)]$

- Need to include negated query
- Then use resolution to attempt to derive the empty clause

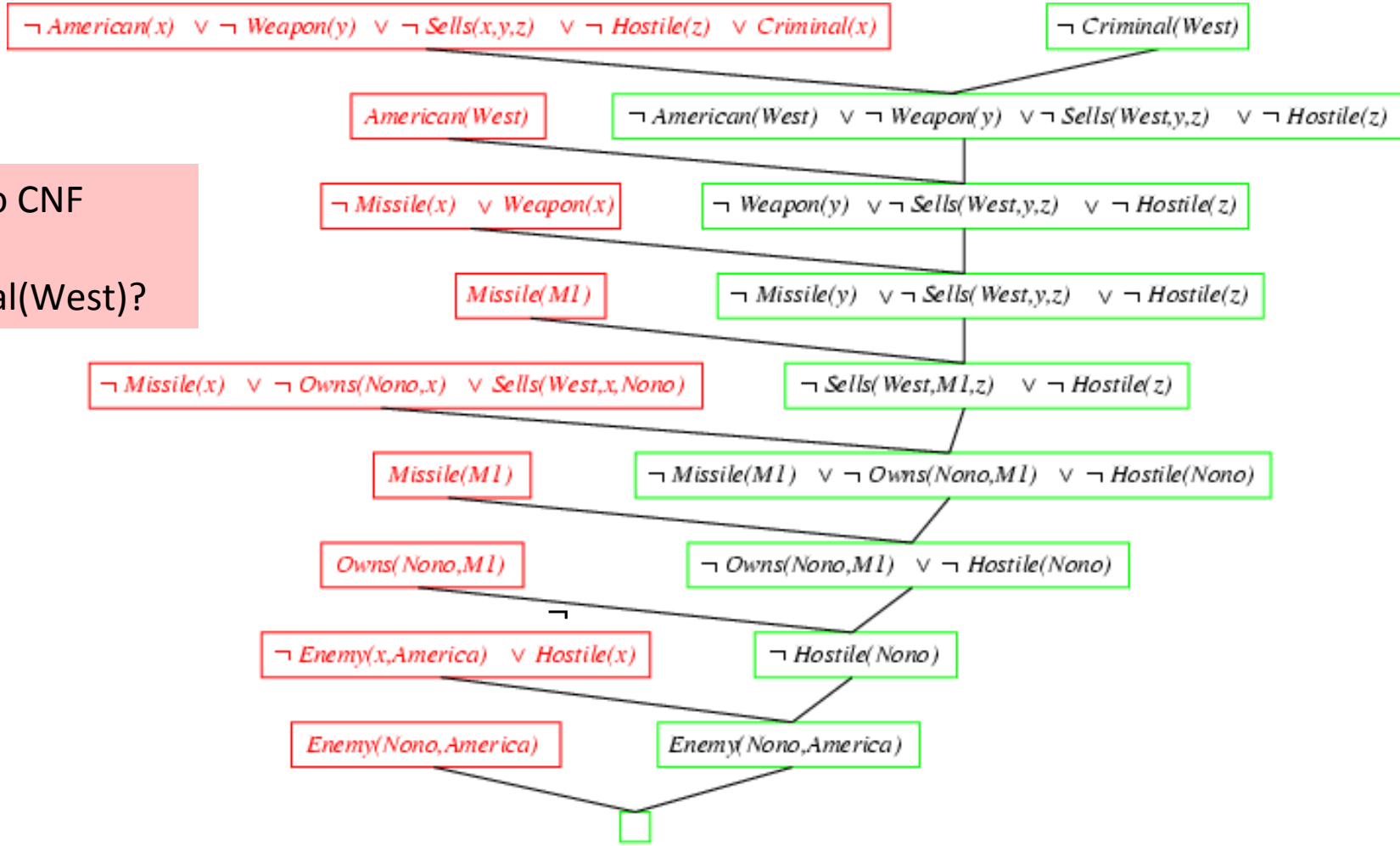
  which show that the query is entailed by the KB

American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
Owns(Nono,M_1) and 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)

Resolution proof

Convert to CNF

Q: Criminal(West)?



Class Exercise

KB:

Everyone who loves all animals is loved by someone

Anyone who kills animals is loved by no-one

Jack loves all animals

Either Curiosity or Jack killed the cat, who is named Tuna

Query: *Did Curiousity kill the cat?*

Inference Procedure:

Express sentences in FOL

Convert to CNF form and negated query