Inference in first-order logic

CE417: Introduction to Artificial Intelligence
Sharif University of Technology
Spring 2016

Soleymani

"Artificial Intelligence: A Modern Approach", 3rd Edition, Chapter 9

Outline

- Reducing first-order inference to propositional inference
  - Potentially great expense

- Lifting inference (direct inference in FOL)
  - Unification
  - Generalized Modus Ponens
  - KB of Horn clauses
    - Forward chaining
    - Backward chaining
  - General KB
    - Resolution

FOL to PL

- FOL to PL conversion
  - First order inference by converting the knowledge base to PL and using propositional inference.
- How to remove universal and existential quantifiers?
  - Universel Instantiation
  - Existential Instantiation

Universal instantiation (UI)

- Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall v \quad \alpha}{Subst(\{v/g\}, \alpha)}$$

- Example, α: $\forall x\, King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  - $Subst(\{x/John\}, \alpha)$
    - $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
  - $Subst(\{x/Richard\}, \alpha)$
    - $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
  - $Subst(\{x/Father(John)\}, \alpha)$
    - $King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$

Existential instantiation (EI)

- For any sentence α, variable v, and a new constant symbol k:

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

- Example, α: $\exists x\ Crown(x) \land OnHead(x, John)$
  - $Subst(\{x/C_1\}, \alpha)$
    - $Crown(C_1) \land OnHead(C_1,\ John)$
    - C_1 is a new constant symbol, called a Skolem constant

- EI can be applied one time (existentially quantified sentence can be discarded after it)

Reduction to propositional inference: Example

- KB of FOL sentences:
  - $\forall x \; King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  - $King(John)$
  - $Greedy(John)$
  - $Brother(Richard, John)$
  - $\exists x \; Crown(x) \; \wedge \; OnHead(x, John)$

- A universally quantified sentence is replaced by all possible instantiations & an existentially quantified sentence by one instantiation:
  - $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
  - $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
  - $King(John)$
  - $Greedy(John)$
  - $Brother(Richard, John)$
  - $Crown(C_1) \wedge OnHead(C_1, John)$

Propositionalization

- Every FOL KB and query can be propositionalized
  - Algorithms for deciding PL entailment can be used

- Problem: infinitely large set of sentences
  - Infinite set of possible ground-term substitution due to function symbols
    - e.g., $Father(\dots Father(Father(John)))$

- Solution:
  - Theorem (Herbrand, 1930): If a sentence α is entailed by an FOL KB, it can be entailed by a finite subset of its propositionalized KB

> **for** $n = 0$ **to** ∞ **do**
>
> Generate all instantiations with depth n nested symbols
>
> **if** α is entailed by this KB **then return** $true$

Propositionalization (Cont.)

- Problem: When procedure go on and on, we will not know whether it is stuck in a loop or the proof is just about to pop out

- Theorem (Turing-1936, Church-1936): Deciding entailment for FOL is semidecidable
  - Algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence.

Propositionalization is inefficient

- Generates lots of irrelevant sentences

- Example:
  - $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$ is irrelevant.

KB

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

Query

$$Evil(x)$$

- p k-ary predicates and n constants
  - $p \times n^k$ instantiations

Lifting & Unification

- Lifting: raising inference rules or algorithms from ground (variable-free) PL to FOL.
  - E.g., generalized Modus Ponens

- Unification: Lifted inference rules require finding substitutions that make different expressions looks identical.
  - Instantiating variables only as far as necessary for the proof.

- All variables are assumed universally quantified.

Unification: Simple example

- Example: Infer immediately when finding a substitution θ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

KB

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

Query

$Evil(x)$

Unification: examples

- $UNIFY(p, q) = \theta$ where $Subst(\theta, p) = Subst(\theta, q)$

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	
$Knows(John, x)$	$Knows(y, Bill)$	
$Knows(John, x)$	$Knows(y, Mother(y))$	
$Knows(John, x)$	$Knows(x, Elizabeth)$	

Unification: examples

- $UNIFY(p, q) = \theta$ where $Subst(\theta, p) = Subst(\theta, q)$

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, Bill)$	$\{x/Bill,\ y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{x/Mother(John),\ y/John\}$
$Knows(John, x)$	$Knows(x, Elizabeth)$	$fail$

Unification: complications

- Complications
  - Two sentences using the same variable name
    - Standardizing apart: renaming the variables causing name clashes in one of sentences
  - More than one unifier
    - Most General Unifier (MGU)

p	q	θ
$Knows(John, x)$	$Knows(x, Elizabeth)$	$fail$
$Knows(John, x)$	$Knows(x_2, Elizabeth)$	$\{x/Elizabeth,\ x_2/John\}$

Unification (MGU)

- $UNIFY(\ Knows(John, x), Knows(y, z))$
  - $\theta_1 = \{y/John,\ x/z\ \}$
  - $\theta_2 = \{y/John,\ x/John,\ z/John\}$
  - θ_1 is more general

- There is a single most general unifier (MGU) that is unique up to renaming of variables.

Generalized Modus Ponens (GMP)

- For atomic sentences p_i , p_i', q & $Subst(\theta, p_i') = Subst(\theta, p_i)$ for all i

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

KB

$King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
$King(John)$
$Greedy(y)$
...

p_1': $King(John)$ $\quad$ p_1 : $King(x)$

p_2': $Greedy(y)$ $\quad$ p_2 : $Greedy(x)$

q: $Evil(x)$

$\theta = \{x/John, y/John\}$

$Subst(\theta, q)$ is $Evil(John)$

Query

$Evil(x)$

GMP is sound

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

- Universal instantiation: $p \vDash Subst(\theta, p)$

1) $$\frac{p_1' \wedge \ldots \wedge p_n'}{Subst(\theta, p_1') \wedge \ldots \wedge Subst(\theta, p_n')}$$

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

3) $Subst(\theta, q)$ follows by Modus Ponens since we have $Subst(\theta, p_i') = Subst(\theta, p_i)$.

Unification algorithm

function UNIFY(x, y, θ) **returns** a substitution to make x and y identical

inputs: x, a variable, a constant, list, or compound expression

y, a variable, a constant, list, or compound expression

θ, the substitution built up so far (optional, defaults to empty)

if $\theta = failure$ **then return** $failure$

else if $x = y$ **then return** θ

else if VARIABLE?(x) **then return** UNIFY_VAR(x, y, θ)

else if VARIABLE?(y) **then return** UNIFY_VAR(y, x, θ)

else if COMPOUND?(x) **and** COMPOUND?(y) **then**

return UNIFY$(x.ARGS,\ y.ARGS,\ \text{UNIFY}(x.OP,\ y.OP,\ \theta))$

else if LIST?(x) **and** LIST?(y) **then**

return UNIFY$(x.REST,\ y.REST,\ \text{UNIFY}(x.FIRST,\ y.FIRST,\ \theta))$

else return $failure$

Unification algorithm

function UNIFY_VAR(var, x, θ) **returns** a

if $\{var/val\} \in \theta$ **then return** UNIFY(val, x, θ)

else if $\{x/val\} \in \theta$ **then return** UNIFY(var, val, θ)

else if OCCUR_CHECK? (var, x) **then return** $failure$

else return add $\{var/x\}$ to θ

KB of definite clauses

- Definite clause: disjunctions of literals of which exactly one is positive.
  - Atomic: $P_1, P_2, \dots$
  - Implication: $(P_1 \wedge P_2 \wedge \dots \wedge P_k) \Rightarrow P_{k+1}$

- First order definite clause examples:
  - $King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  - $King(John)$
  - $Brother(John, Richard)$

KB of first order definite clauses: Example

- "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."

- Question: <u>Prove that Colonel West is a criminal</u>

KB of first order definite clauses: Example

- "<u>The law says that it is a crime for an American to sell weapons to hostile nations</u>. 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."

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

KB of first order definite clauses: Example

- "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."

$$Enemy(Nono, America)$$

KB of first order definite clauses: Example

- "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."

$$\exists x \; Owns(Nono, x) \land Missile(x)$$

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

KB of first order definite clauses: Example

- "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 <u>all of its missiles were sold to it by Colonel West</u>, who is American."

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

KB of first order definite clauses: Example

- "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 <u>Colonel West, who is American</u>."

$$American(West)$$

KB of first order definite clauses: Example

- "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."

- We need to know missiles are weapons:

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

- We need to know an enemy of America counts as "hostile":

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

KB of first order definite clauses: Example

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

$$Enemy(Nono, America)$$

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

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

$$American(West)$$

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

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

FC algorithm

function FOL_FC_ASK(KB, α) **returns** a substitution or $false$

inputs: KB, a set of first-order definite clauses

α, the query, an atomic sentence

repeat until $new = \{\}$

$new = \{\}$

for each $rule$ in KB **do**

$(p_1 \wedge \cdots \wedge p_n \Rightarrow q) \leftarrow \text{STANDARDIZE\_VARS}(rule)$

for each θ such that $\text{SUBST}(\theta, p_1 \wedge \cdots \wedge p_n) = \text{SUBST}(\theta, p'_1 \wedge \cdots \wedge p'_n)$

for some $p'_1, \ \ldots \ , \ p'_n$ in KB

$q' \leftarrow \text{SUBST}(\theta, q)$

if q' does not unify with some sentence already in KB or new **then**

add q' to new

$\varphi \leftarrow \text{UNIFY}(q', \alpha)$

if φ is not $fail$ **then return** φ

add new to KB

return $false$

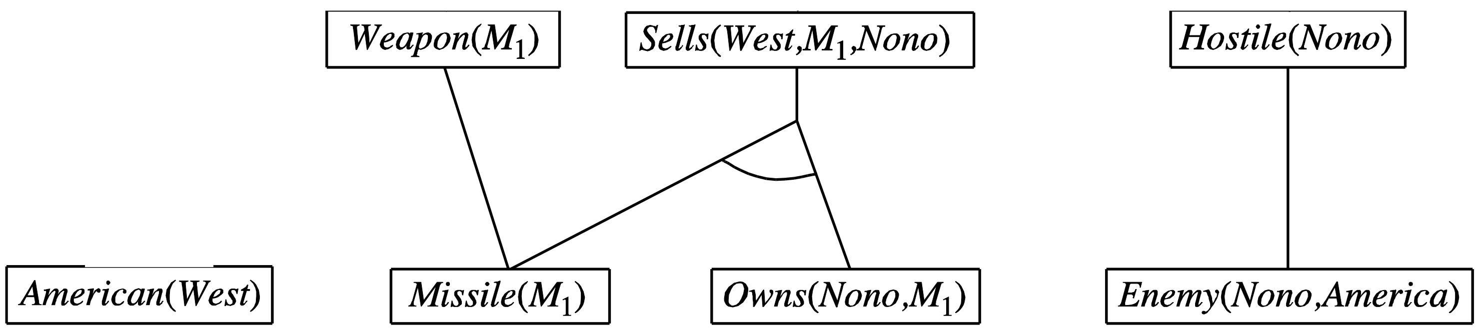
FC: example

Known facts

$\boxed{American(West)}$ $\boxed{Missile(M_1)}$ $\boxed{Owns(Nono, M_1)}$ $\boxed{Enemy(Nono, America)}$

1) $American(West)$
2) $Owns(Nono, M_1) \land Missile(M_1)$
3) $Enemy(Nono, America)$
4) $Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
5) $American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$
6) $Missile(x) \Rightarrow Weapon(x)$
7) $Enemy(x, America) \Rightarrow Hostile(x)$

FC: example



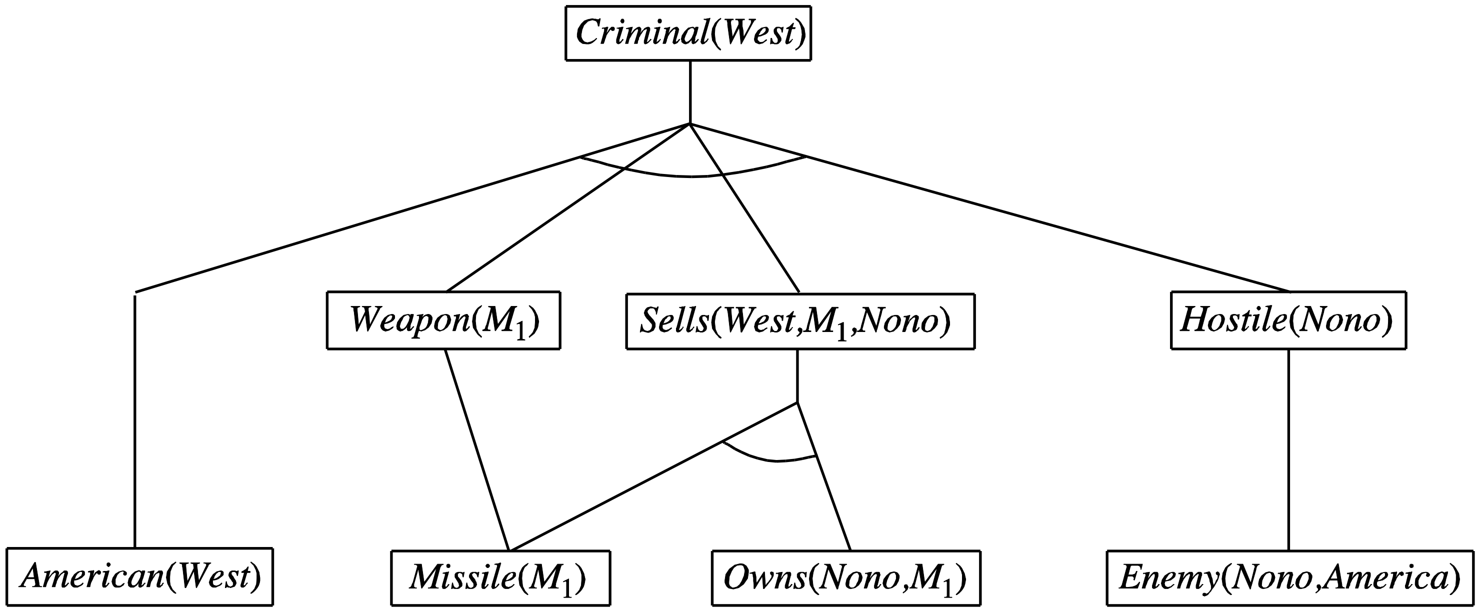
1) $Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$ $\{x/M_1\}$

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

3) $Missile(x) \Rightarrow Weapon(x)$ $\{x/M_1\}$

4) $Enemy(x, America) \Rightarrow Hostile(x)$ $\{x/Nono\}$

FC: example



1) $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$ $\{x/M_1\}$

2) $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$ $\{x/West, y/M_1, z/Nono\}$

3) $Missile(x) \Rightarrow Weapon(x)$ $\{x/M_1\}$

4) $Enemy(x, America) \Rightarrow Hostile(x)$ $\{x/Nono\}$

Properties of FC

- Sound

- Complete (for KBs of first-order definite clauses)
  - For Datalog KBs, proof is fairly easy.
    - Datalog KBs contain first-order definite clauses with no function symbols
    - FC reaches a fix point for Datalog KBs after finite number of iterations
  - It may not terminate in general if α is not entailed (query has no answer)
    - entailment with definite clauses is also semi-decidable

More efficient FC

- Heuristics for matching rules against known facts
  - E.g., conjunct ordering

- Incremental forward chaining to avoid redundant rule matching
  - Every new fact inferred on iteration t must be derived from at least one new fact inferred on iteration $t-1$.
    - Check a rule only if its premise includes a newly inferred fact (at iteration t $-$ 1)

- Avoid drawing irrelevant facts (to the goal)

Backward Chaining (BC)

- Works backward from the goal, chaining through rules to find known facts supporting the proof

- AND-OR search
  - OR: any rule $lhs \Rightarrow goal$ in KB can be used to prove the goal
  - AND: all conjuncts in lhs must be proved.

- It is used in logic programming (Prolog)

BC algorithm (depth-first recursive proof search)

function FOL_BC_ASK($KB, query$) **returns** a generator of substitutions
  return FOL_BC_OR($KB, query, \{\}$)

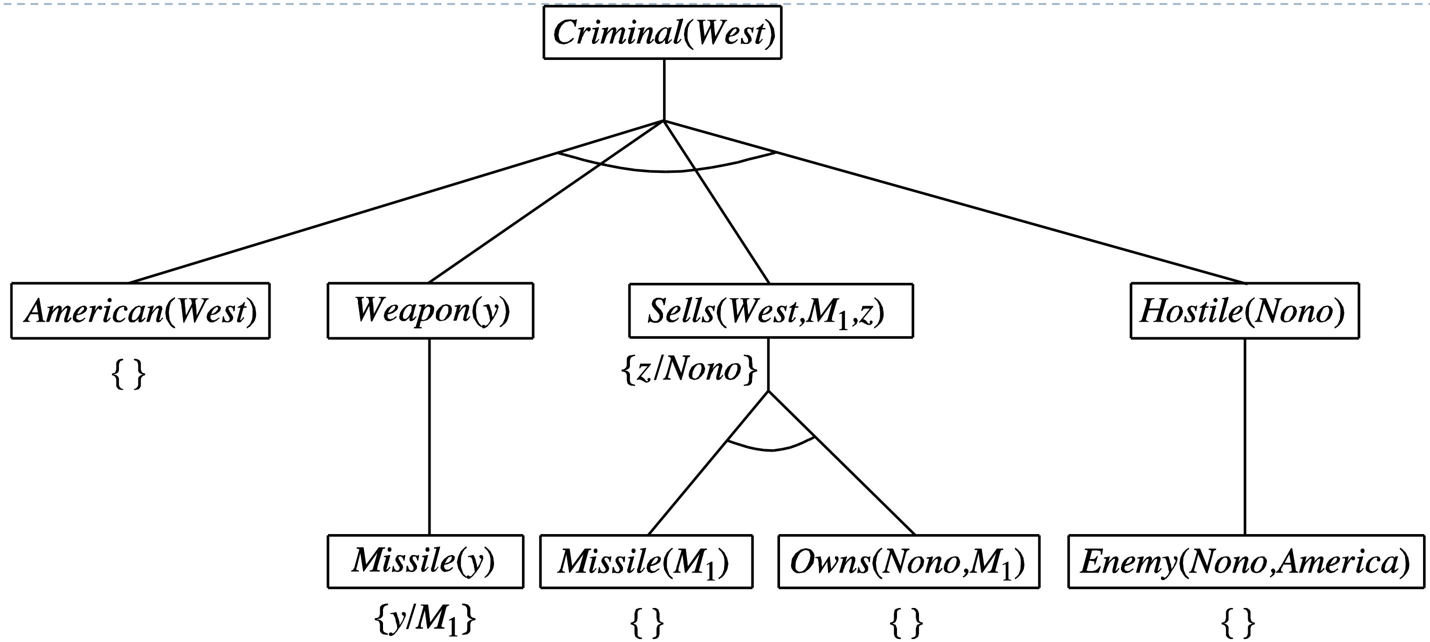
---

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_VARS(lhs, rhs)
    for each θ' **in** FOL_BC_AND($KB, lhs,$ UNIFY($rhs, goal, \theta$) **do**
      yield θ'

---

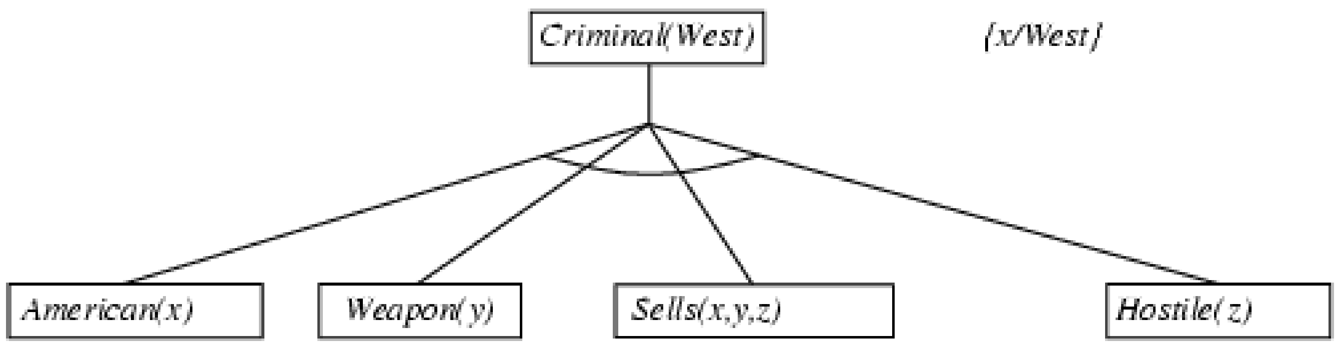
generator FOL_BC_AND($KB, goals, \theta$) **yields** a substitution
  if $\theta = failure$ **then return**
  else if LENGTH($goals$) $= 0$ **then yield** θ
  else do
    $first \leftarrow$ FIRST($goals$)
    $rest \leftarrow$ REST($goals$)
    for each θ' **in** FOL_BC_OR($KB,$ SUBST($\theta, first$)$, \theta$) **do**
      for each θ'' **in** FOL_BC_AND($KB, rest, \theta'$) **do**
        yield θ''

Backward chaining example



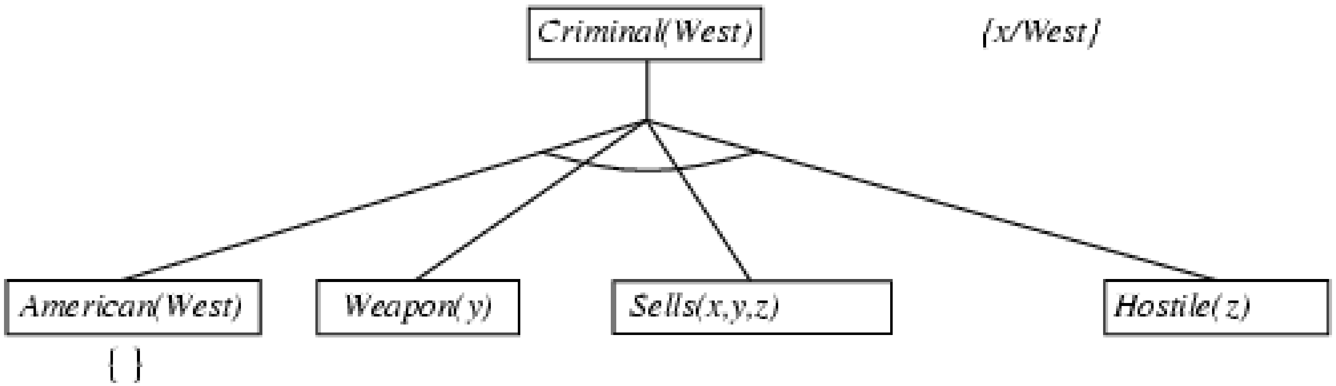
1) $Enemy(Nono, America)$
2) $Owns(Nono, M_1) \wedge Missile(M_1)$
3) $American(West)$
4) $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
5) $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
6) $Missile(x) \Rightarrow Weapon(x)$
7) $Enemy(x, America) \Rightarrow Hostile(x)$

Backward chaining example



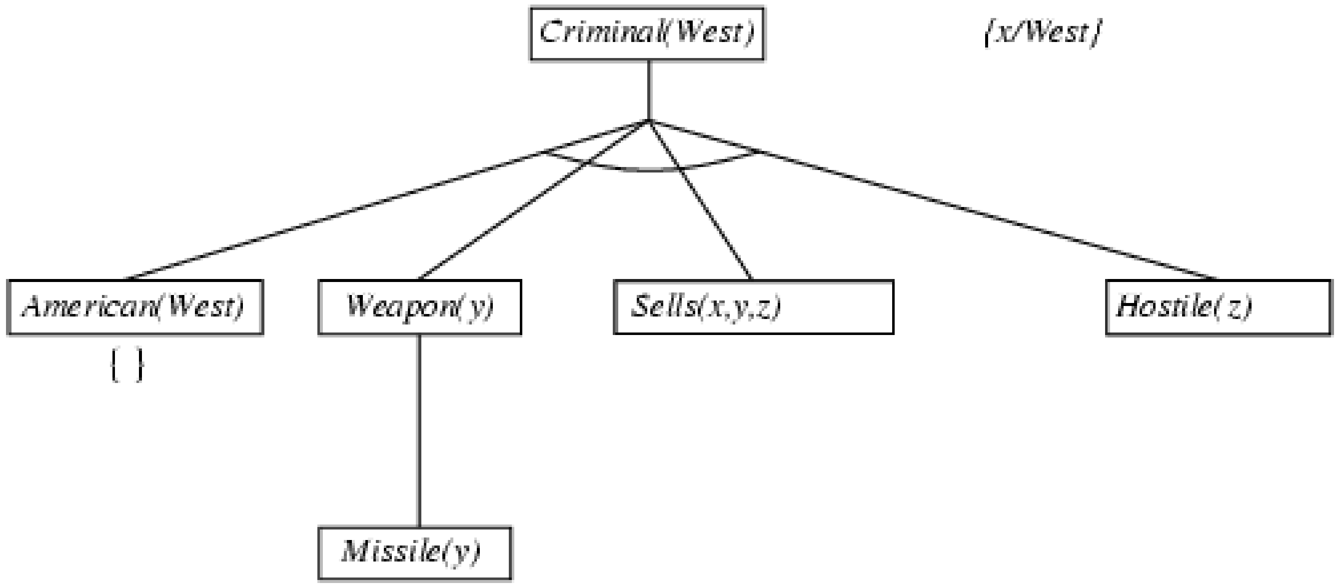
1) $Enemy(Nono, America)$
2) $Owns(Nono, M_1) \land Missile(M_1)$
3) $American(West)$
4) $Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
5) $American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$
6) $Missile(x) \Rightarrow Weapon(x)$
7) $Enemy(x, America) \Rightarrow Hostile(x)$

Backward chaining example



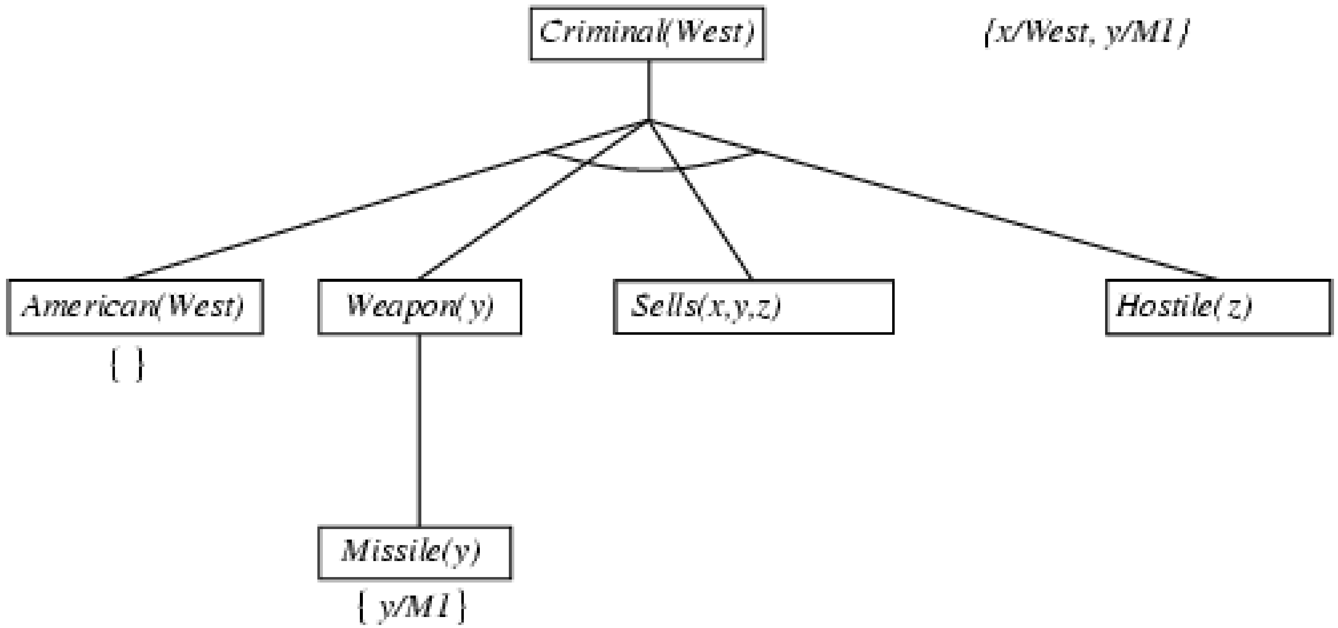
1) $Enemy(Nono, America)$
2) $Owns(Nono, M_1) \wedge Missile(M_1)$
3) $American(West)$
4) $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
5) $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
6) $Missile(x) \Rightarrow Weapon(x)$
7) $Enemy(x, America) \Rightarrow Hostile(x)$

Backward chaining example



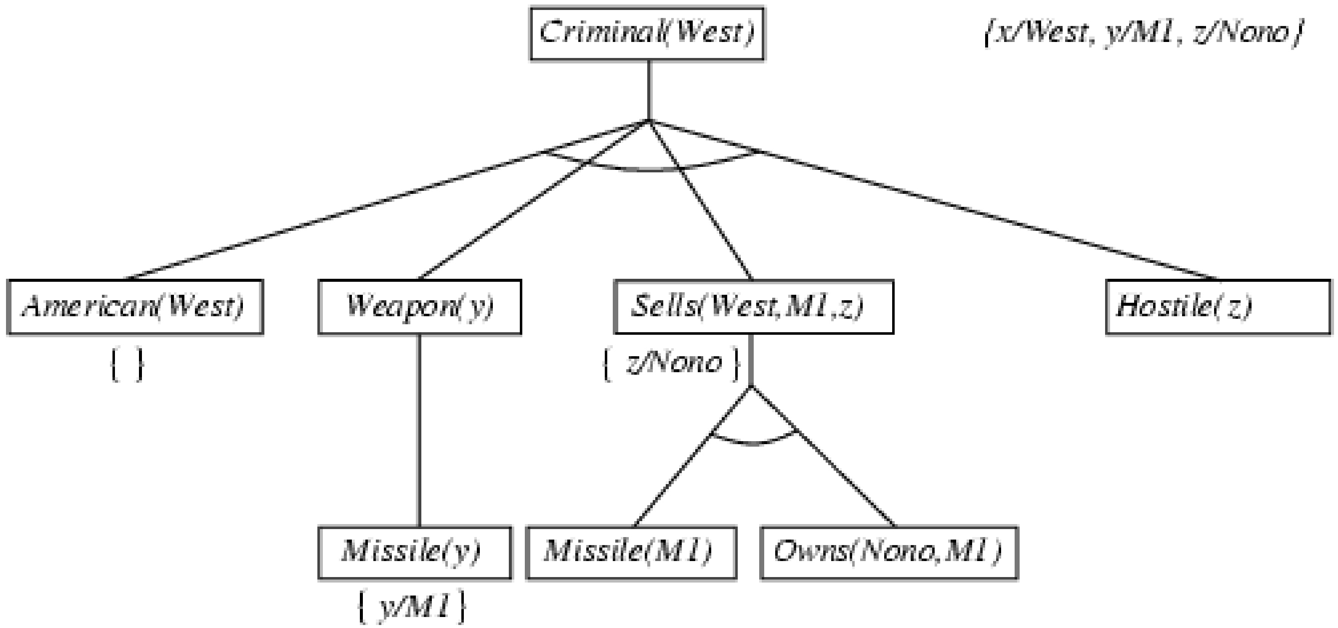
1) $Enemy(Nono, America)$
2) $Owns(Nono, M_1) \wedge Missile(M_1)$
3) $American(West)$
4) $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
5) $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
6) $Missile(x) \Rightarrow Weapon(x)$
7) $Enemy(x, America) \Rightarrow Hostile(x)$

Backward chaining example



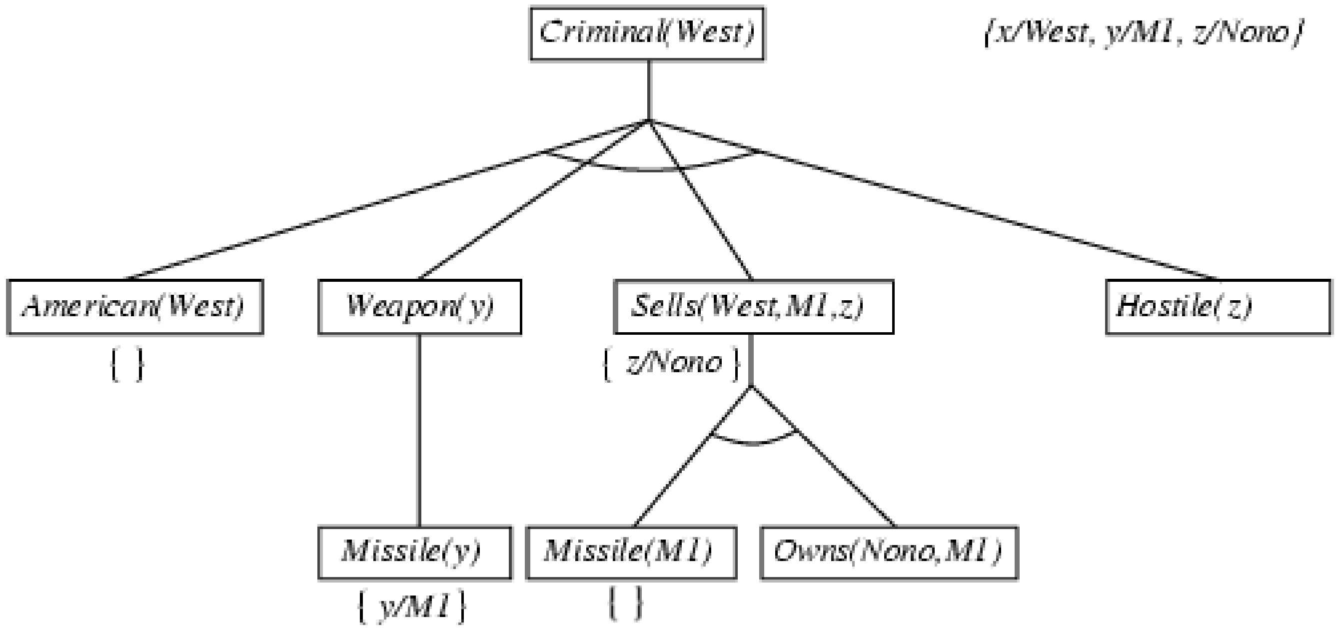
1) $Enemy(Nono, America)$
2) $Owns(Nono, M_1) \wedge Missile(M_1)$
3) $American(West)$
4) $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
5) $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
6) $Missile(x) \Rightarrow Weapon(x)$
7) $Enemy(x, America) \Rightarrow Hostile(x)$

Backward chaining example



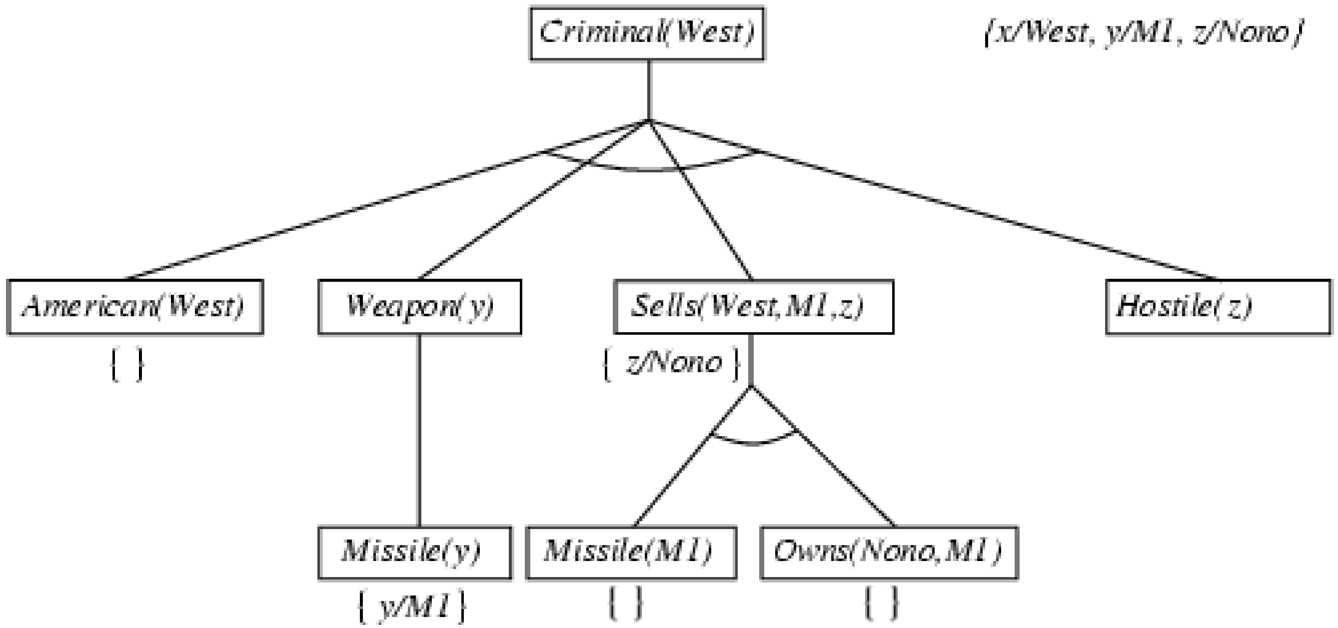
1) $Enemy(Nono, America)$
2) $Owns(Nono, M_1) \wedge Missile(M_1)$
3) $American(West)$
4) $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
5) $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
6) $Missile(x) \Rightarrow Weapon(x)$
7) $Enemy(x, America) \Rightarrow Hostile(x)$

Backward chaining example



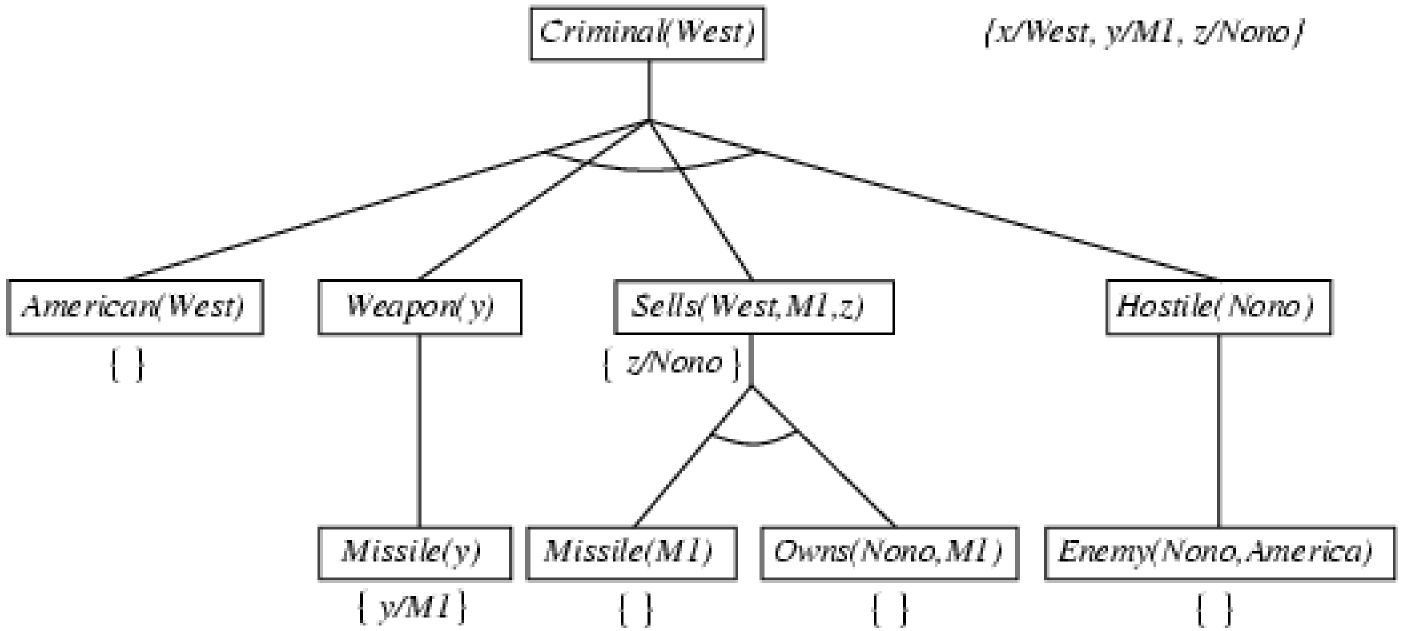
1) $Enemy(Nono, America)$
2) $Owns(Nono, M_1) \wedge Missile(M_1)$
3) $American(West)$
4) $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
5) $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
6) $Missile(x) \Rightarrow Weapon(x)$
7) $Enemy(x, America) \Rightarrow Hostile(x)$

Backward chaining example



1) $Enemy(Nono, America)$
2) $Owns(Nono, M_1) \wedge Missile(M_1)$
3) $American(West)$
4) $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
5) $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
6) $Missile(x) \Rightarrow Weapon(x)$
7) $Enemy(x, America) \Rightarrow Hostile(x)$

Backward chaining example



1) $Enemy(Nono, America)$
2) $Owns(Nono, M_1) \wedge Missile(M_1)$
3) $American(West)$
4) $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
5) $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
6) $Missile(x) \Rightarrow Weapon(x)$
7) $Enemy(x, America) \Rightarrow Hostile(x)$

Properties of BC

- Space is linear in the size of the proof

- Suffers from repeated states and incompleteness
  - Repeated subgoals (both success and failure)
    - Solution: caching of previous results

  - Incompleteness due to infinite loops
    - Solution: checking current goal against every goal on stack

Resolution algorithm

- General theorem proving

- Apply resolution rule to $CNF(KB \land \neg\alpha)$
  - Refutation-complete for FOL

Resolution

- PL resolution: l_i and m_j are complement

$$\frac{l_1 \vee l_2 \vee \cdots \vee l_k, \qquad m_1 \vee m_2 \vee \cdots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

- First-order resolution $\mathrm{UNIFY}(l_i, \neg m_j) = \theta$
  - Clauses are assumed to be standardized apart (sharing no variables).

$$\frac{l_1 \vee l_2 \vee \cdots \vee l_k, \qquad m_1 \vee m_2 \vee \cdots \vee m_n}{\mathrm{SUBST}(\theta,\ l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)}$$

- Example:

$$\frac{\neg Rich(x) \vee Unhappy(x), \qquad Rich(Ken)}{Unhappy(Ken)} \qquad \theta = \{x/Ken\}$$

Converting FOL sentence to CNF

- Example: "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 implications ($P \Rightarrow Q \equiv \neg P \vee Q$)
   - $\forall x\ [\neg\forall y\ \ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

2) Move $\neg$ inwards: $\neg\forall$x p to $\exists$x $\neg$p, $\neg\ \exists$x p to $\forall$x $\neg$p
   - $\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\ Loves(y,x)]$
   - $\forall x\ [\exists y\ \neg\neg Animal(y)\ \wedge\ \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$
   - $\forall x\ [\exists y\ Animal(y)\ \wedge\ \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

3) Standardize variables (avoiding same names for variables)
   - $\forall x\ [\exists y\ Animal(y)\ \wedge\ \neg Loves(x,y)] \vee [\exists z\ Loves(z,x)]$

Converting FOL sentence to CNF

4) Skolemize: a more general form of existential instantiation.
   - Each existentially quantified variable is replaced by a Skolem function of the enclosing universally quantified variables.

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

5) Drop universal quantifiers:
   - $[Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(x), x)$

6) Distribute $\vee$ over $\wedge$:
   - $[Animal(F(x)) \vee Loves(G(x), x)] \wedge [\neg Loves(x, F(x)) \vee Loves(G(x), x)]$

Properties of resolution

- Resolution
  - Complete when used in combination with factoring

- Extend factoring to FOL
  - Factoring in PL: reduces two literals to one if they are identical
  - Factoring in FOL: reduces two literals to one if they are unifiable. The unifier must be applied to the entire clause

Resolution: definite clauses example

1) $Enemy(Nono, America)$
2) $Owns(Nono, M_1) \wedge Missile(M_1)$
3) $American(West)$
4) $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$
5) $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$
6) $Missile(x) \Rightarrow Weapon(x)$
7) $Enemy(x, America) \Rightarrow Hostile(x)$

Convert to CNF

1) $Enemy(Nono, America)$
2) $Owns(Nono, M_1) \wedge Missile(M_1)$
3) $American(West)$
4) $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$
5) $\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$
6) $\neg Missile(x) \vee Weapon(x)$
7) $\neg Enemy(x, America) \vee Hostile(x)$

Resolution: definite clauses example

KB	$\neg\alpha$
$\neg American(x) \lor \neg Weapon(y) \lor \neg Sells(x,y,z) \lor \neg Hostile(z) \lor \mathbf{Criminal(x)}$	$\mathbf{\neg Criminal(West)}$
$\mathbf{American(West)}$	$\mathbf{\neg American(West)} \lor \neg Weapon(y) \lor \neg Sells(West,y,z) \lor \neg Hostile(z)$
$\neg Missile(x) \lor \mathbf{Weapon(x)}$	$\mathbf{\neg Weapon(y)} \lor \neg Sells(West,y,z) \lor \neg Hostile(z)$
$\mathbf{Missile(M_1)}$	$\mathbf{\neg Missile(y)} \lor \neg Sells(West,y,z) \lor \neg Hostile(z)$
$\neg Missile(x) \lor \neg Owns(Nono,x) \lor \mathbf{Sells(West,x,Nono)}$	$\mathbf{\neg Sells(West,M_1,z)} \lor \neg Hostile(z)$
$\mathbf{Missile(M_1)}$	$\mathbf{\neg Missile(M_1)} \lor \neg Owns(Nono,M_1) \lor \neg Hostile(Nono)$
$\mathbf{Owns(Nono,M_1)}$	$\mathbf{\neg Owns(Nono,M_1)} \lor \neg Hostile(Nono)$
$\neg Enemy(x,America) \lor \mathbf{Hostile(x)}$	$\mathbf{\neg Hostile(Nono)}$
$\mathbf{Enemy(Nono,America)}$	$\mathbf{\neg Enemy(Nono,America)}$
	$\square$

Resolution: general example

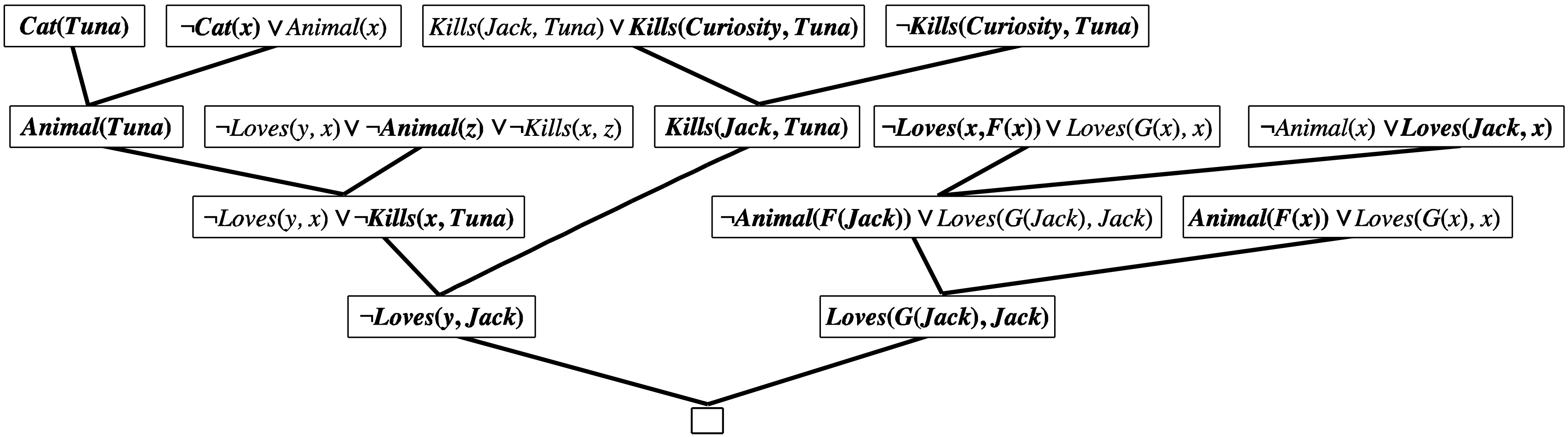
- Every one who loves all animals is loved by someone.
  - $\forall x\ [\forall y\ Animal(y)\ \Rightarrow\ Loves(x,y)] \Rightarrow [\exists y\ Loves(y,x)]$
- Anyone who kills an animal is loved by no one.
  - $\forall x\ [\exists z\ Animal(z)\ \wedge Kills(x,z)] \Rightarrow [\forall y\ \neg Loves(y,x)]$
- Jack loves all animals.
  - $\forall x\ Animal(x) \Rightarrow Loves(Jack,x)$
- Either Jack or Curiosity killed the cat, who is named Tuna.
  - $Kills(Jack,Tuna)\ \vee\ Kills(Curiosity,Tuna)$
  - $Cat(Tuna)$
- <u>Query</u>: Did Curiosity kill the cat?
  - $Kills(Curiosity,Tuna)$

Resolution: general example

- CNF ($KB \wedge \neg\alpha$):
  - $Animal(F(x)) \vee Loves(G(x), x)$
  - $\neg Loves(x, F(x)) \vee Loves(G(x), x)$
  - $\neg Loves(y, x) \vee \neg Animal(z) \vee \neg Kills(x, z)$
  - $\neg Animal(x) \vee Loves(Jack, x)$
  - $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$
  - $Cat(Tuna)$
  - $\neg Kills(Curiosity, Tuna)$

- Query: "Who killed the cat"?
  - α: $\exists w\, Kills(w, Tuna)$
  - The binding $\{w/Curiosity\}$ in one of steps

Resolution: general example



- Query: "Who killed the cat"?
  - α: $\exists w\ Kills(w, Tuna)$
  - The binding $\{w/Curiosity\}$ in one of steps

Prolog (Programming in logic)

- Basis: backward chaining with Horn clauses
  - Depth-first, left-to-right BC
- Program = set of clauses
  - Fact
    - e.g., `american(west).`
  - Rule: `head :- literal`$_1$`, … ,literal`$_n$`.`
    - e.g., `criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).`

Prolog: Example

- 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=[]
```

Summary

- Lifting and unification
- Inference on KB of definite clauses
  - Forward chaining
  - Backward chaining
- Inference on general KB of FOL
  - Resolution