Artificial Intelligence
Inference in First-Order Logic

S. Komal Kaur

Assistant Professor

CSE Dept

Contents

- Reduction to propositional inference
- Unification and Lifting
- Generalized Modus Ponens
- Forward and Backward Chaining
- Resolution

Inference with Quantifiers

- Universal Instantiation:
  - Given $\forall$ x, person(x) $\Rightarrow$ likes(x, McDonalds)
  - Infer person(John) $\Rightarrow$ likes(John, McDonalds)

- Existential Instantiation:
  - Given $\exists$ x, likes(x, McDonalds)
  - Infer $\Rightarrow$ likes(S1, McDonalds)
  - S1 is a "Skolem Constant" that is not found anywhere else in the KB and refers to (one of) the indviduals that likes McDonalds.

Universal Instantiation

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

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

  – for any variable v and ground term g
    - ground term…a term with out variables

- Example:
  – $\forall$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

- For any sentence α, variable *v*, and constant k that does not appear in the KB:

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

- Example:
  - $\exists$ x Crown(x) $\wedge$ OnHead(x, John) yields:
    - Crown(C_1) $\wedge$ OnHead(C_1, John)
    - provided C_1 is a new constant (Skolem)

Existential Instantiation

- UI can be applied several times to add new sentences
  - The KB is logically equivalent to the old
- EI can be applied once to replace the existential sentence
  - The new KB is not equivalent to the old but is satisfiable iff the old KB was satisfiable

Reduction to Propositional Inference

- Use instantiation rules to create relevant propositional facts in the KB, then use propositional reasoning.

Reduction to Propositional Inference

- Suppose the KB had the following sentence

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

  King(John)

  Greedy(John)

  Brother(Richard, John)

Reduction to Propositional Inference

- Instantiating the universal sentence in all possible ways…

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

- The new KB is propositionalized: propositional symbols are…

  King(John), Greedy(John), Evil(John), King(Richard), etc…

Problems with Propositionalization

- Propositionalization tends to generate lots of irrelevant sentences
- Example

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

  King(John)

  $\forall y \;\; Greedy(y)$

  Brother(Richard, John)

  – Obvious that Evil(John) is true, but the fact Greedy(Richard) is irrelevant.

Unification

- **Unification**: The process of finding all legal substitutions that make logical expressions look identical

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)
- $\theta = \{x/John, y/John\}$ works
- $\text{Unify}(\alpha,\beta) = \theta$ if $\alpha\theta = \beta\theta$

Unification

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

Generalized Modus Ponens

- This is a general inference rule for FOL that does not require instantiation
- Given:

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

  p1’, p2’ … pn’ (p1 $\wedge$ … pn) $\Rightarrow$ q

  Subst(θ, pi’) = subst(θ, pi) for all p

- Conclude:
  - Subst(θ, q)

GMP in “CS terms”

- Given a rule containing variables

- If there is a consistent set of bindings for all of the variables of the left side of the rule (before the arrow)

- Then you can derive the result of substituting all of the same variable bindings into the right side of the rule

GMP Example

- $\forall$ x, Parent(x,y) $\wedge$ Parent(y,z) $\Rightarrow$ GrandParent(x,z)
- Parent(James, John), Parent(James, Richard), Parent(Harry, James)
- We can derive:
  - GrandParent(Harry, John), bindings: ((x Harry) (y James) (z John)
  - GrandParent(Harry, Richard), bindings: ((x Harry) (y James) (z Richard)

Base Cases for Unification

- If two expressions are identical, the result is (NIL) (succeed with empty unifier set)

- If two expressions are different constants, the result is NIL (fail)

- If one expression is a variable *and is not contained in the other,* the result is ((x other-exp))

Storage and retrieval

- Most systems don’t use variables on predicates
- Therefore, hash statements by predicate for quick retrieval (predicate indexing)
- Subsumption lattice for efficiency

Forward Chaining

- Forward Chaining
  - Start with atomic sentences in the KB and apply Modus Ponens in the forward direction, adding new atomic sentences, until no further inferences can be made.

Forward Chaining

- Given a new fact, generate all consequences
- Assumes all rules are of the form
  - C1 and C2 and C3 and…. --> Result
- Each rule & binding generates a new fact
- This new fact will “trigger” other rules
- Keep going until the desired fact is generated
- (Semi-decidable as is FOL in general)

FC: 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 America, has some missiles, and all of its missiles were sold to it by Col. West, who is an American.

- Prove that Col. West is a criminal.

FC: 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) $\wedge$ Missiles(x)

  $Owns(Nono, M_1)$ *and* $Missle(M_1)$

- …all of its missiles were sold to it by Col. West
  $\forall x\ Missle(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

- Missiles are weapons
  $Missle(x) \Rightarrow Weapon(x)$

FC: Example Knowledge Base

- An enemy of America counts as "hostile"

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

- Col. West who is an American

  American(Col. West)

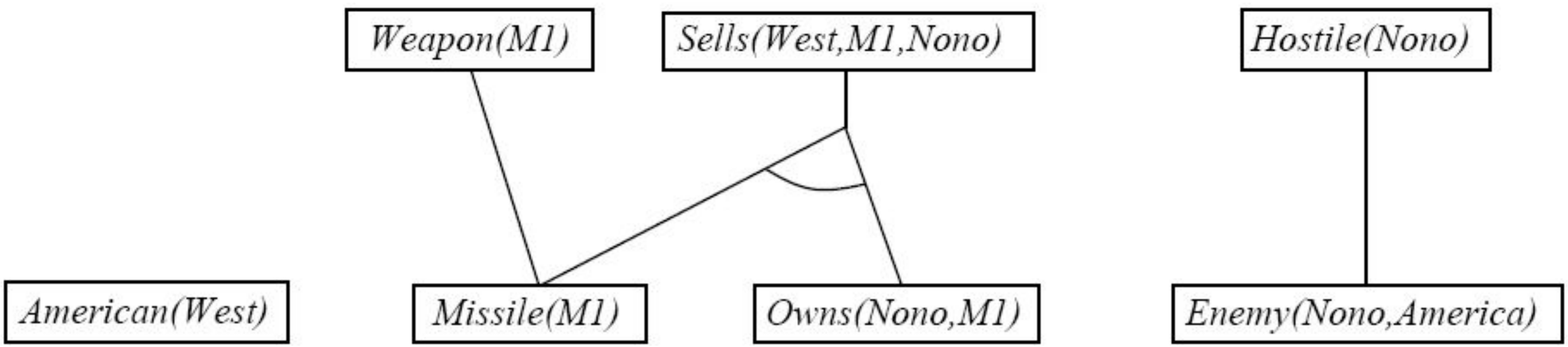
- The country Nono, an enemy of America

  Enemy(Nono, America)

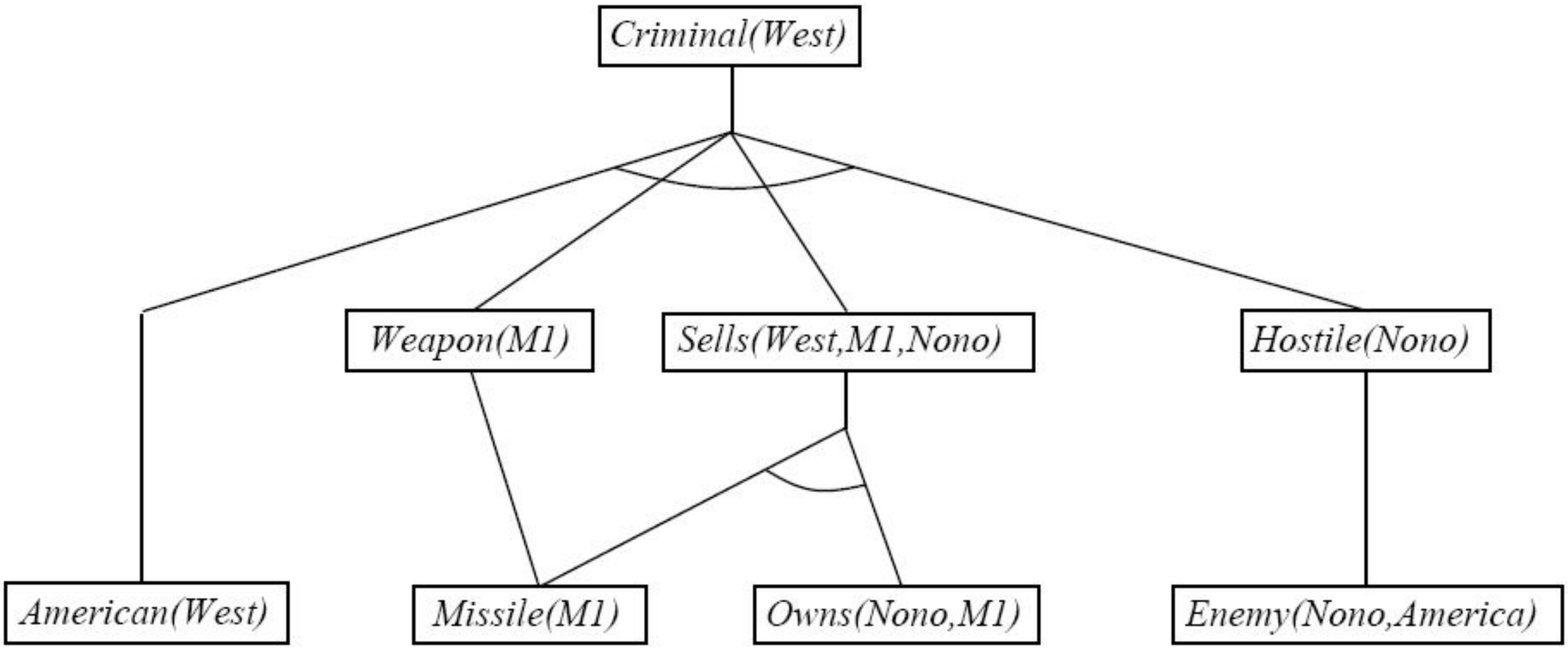
FC: Example Knowledge Base

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

FC: Example Knowledge Base



FC: Example Knowledge Base



Efficient Forward Chaining

- Order conjuncts appropriately
  - E.g. most constrained variable
- Don't generate redundant facts; each new fact should depend on at least one newly generated fact.
  - Production systems
  - RETE matching
  - CLIPS

Forward Chaining Algorithm

```
function FOL-FC-ASK(KB, α) returns a substitution or false

   repeat until new is empty
        new ← { }
        for each sentence r in KB do
             ( p1 ∧ . . . ∧ pn ⇒ q) ← STANDARDIZE-APART(r)
             for each θ such that (p1 ∧ . . . ∧ pn)θ = (p'1 ∧ . . . ∧ p'n)θ
                            for some p'1, . . . , p'n in KB
                 q' ← SUBST(θ, q)
                 if q' is not a renaming of a sentence already in KB or new then do
                      add q' to new
                      φ ← UNIFY(q', α)
                      if φ is not fail then return φ
        add new to KB
   return false
```

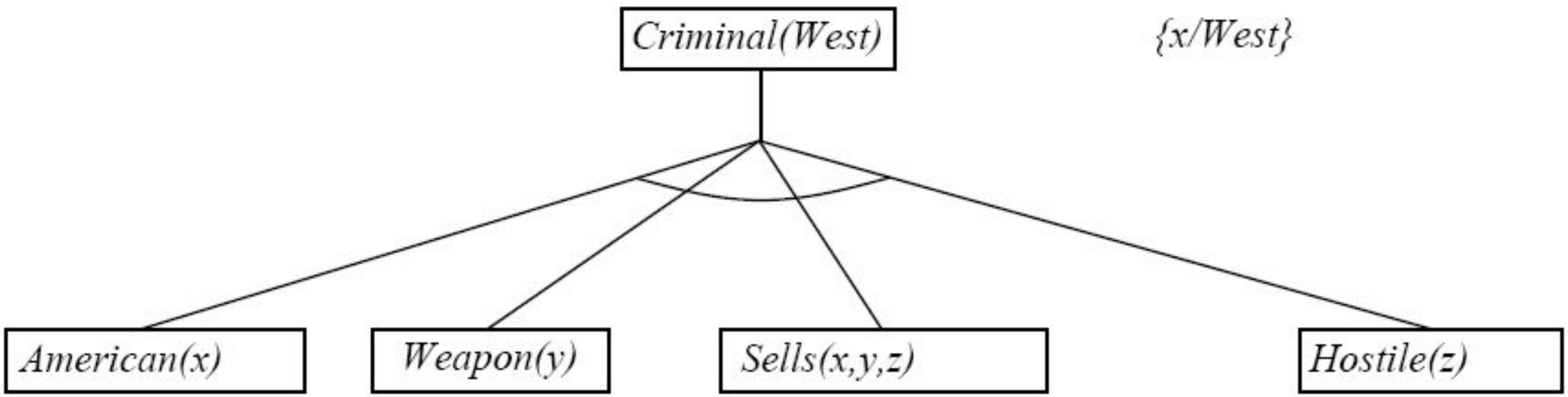
Backward Chaining

- Consider the item to be proven a goal
- Find a rule whose head is the goal (and bindings)
- Apply bindings to the body, and prove these (subgoals) in turn
- If you prove all the subgoals, increasing the binding set as you go, you will prove the item.
- Logic Programming (cprolog, on CS)

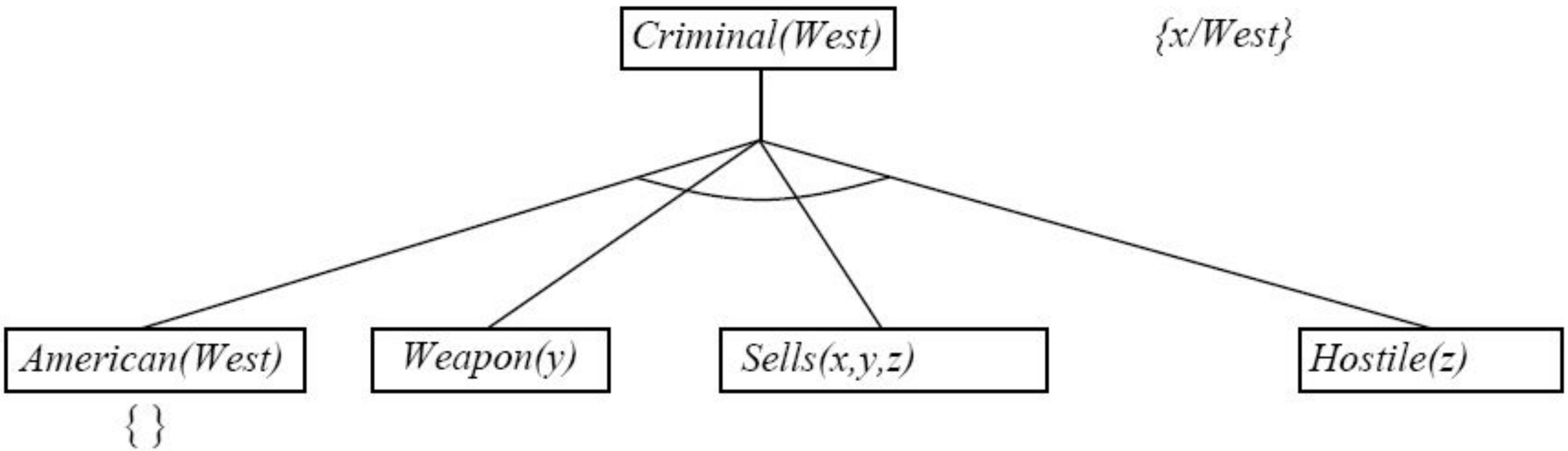
Backward Chaining Example

$\boxed{Criminal(West)}$

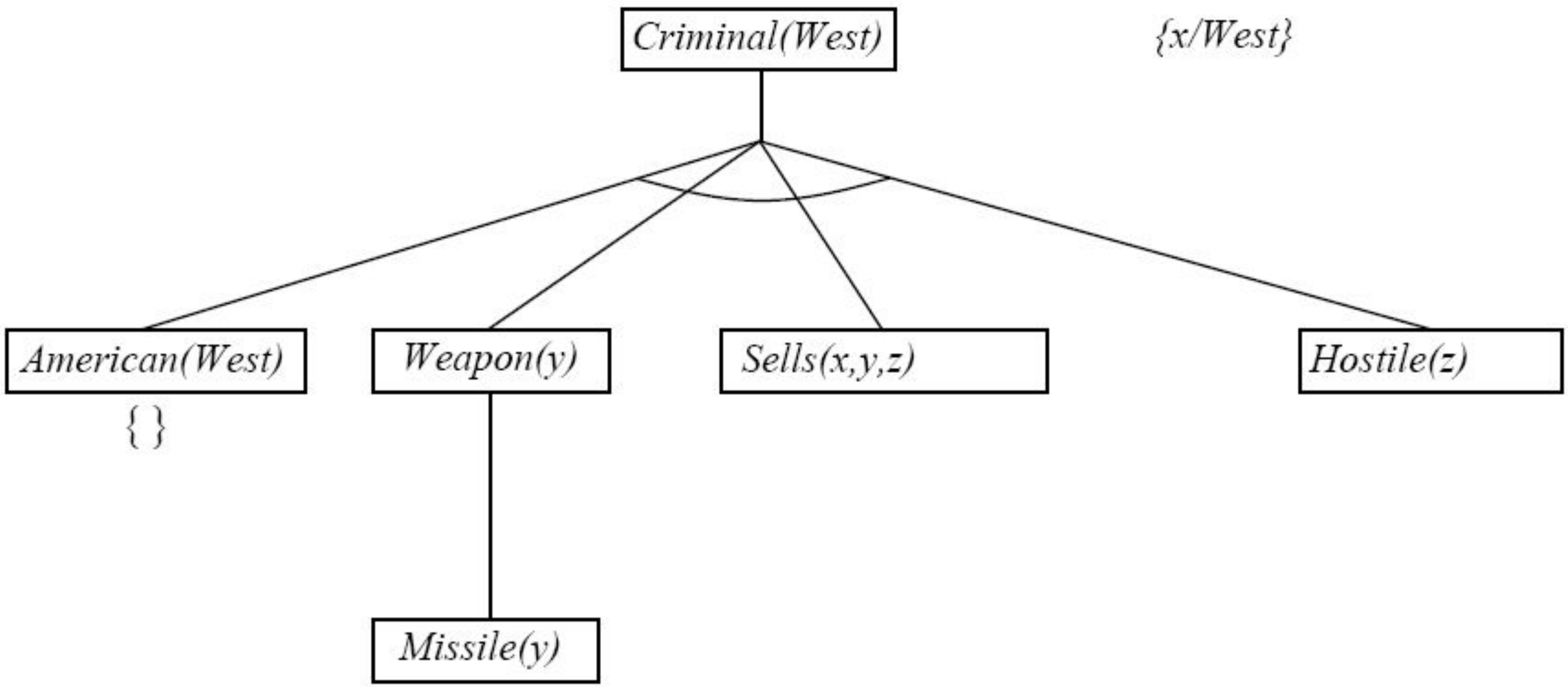
Backward Chaining Example



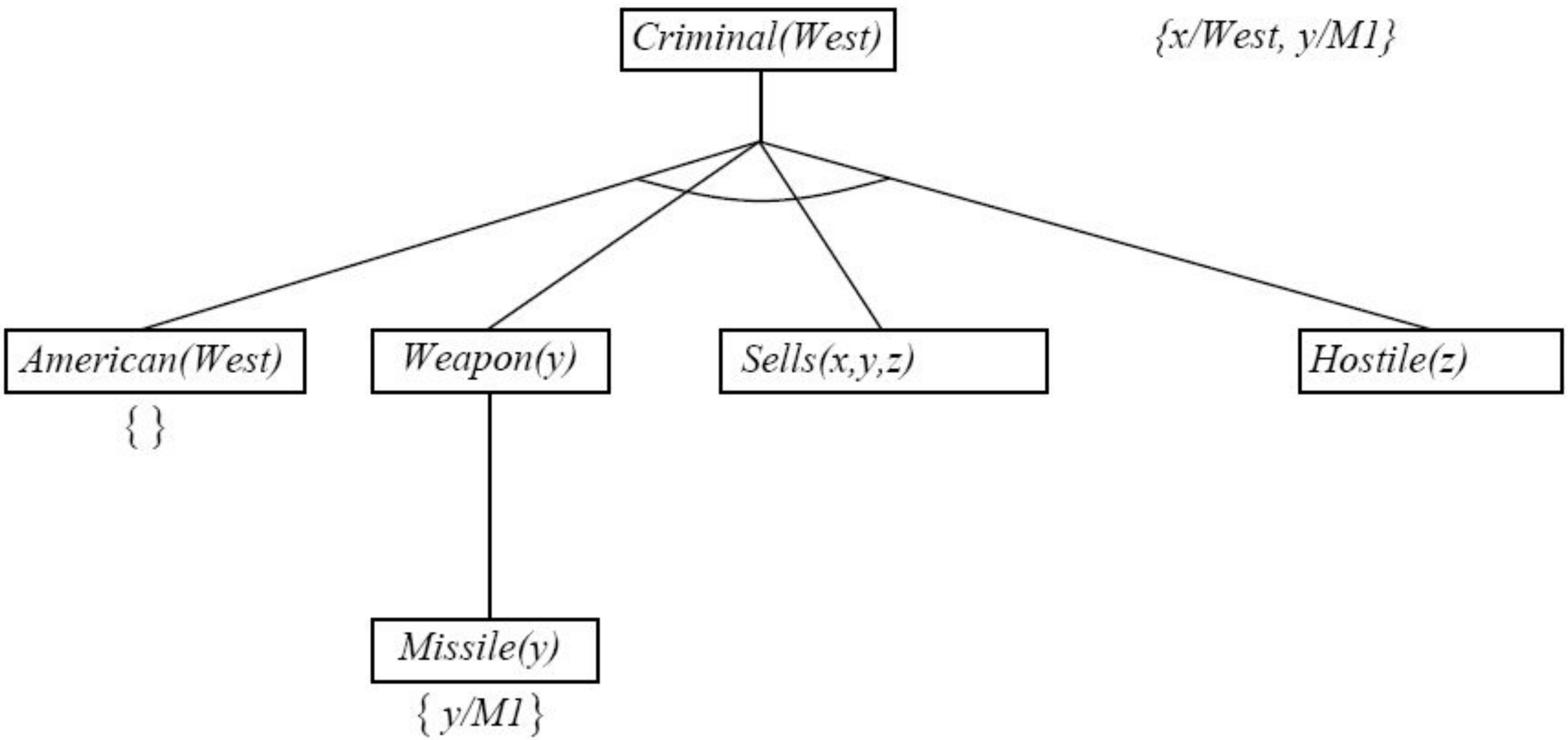
Backward Chaining Example



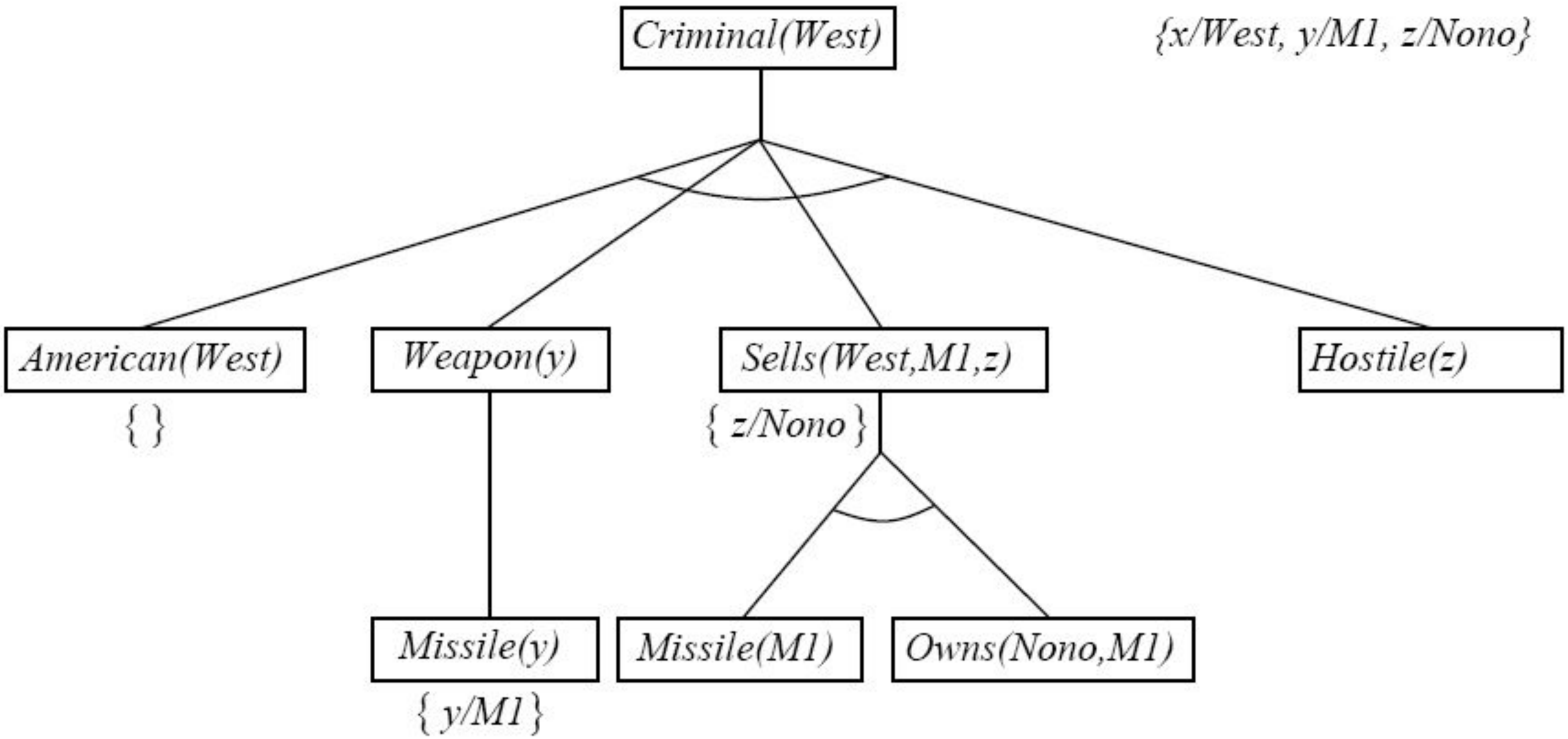
Backward Chaining Example



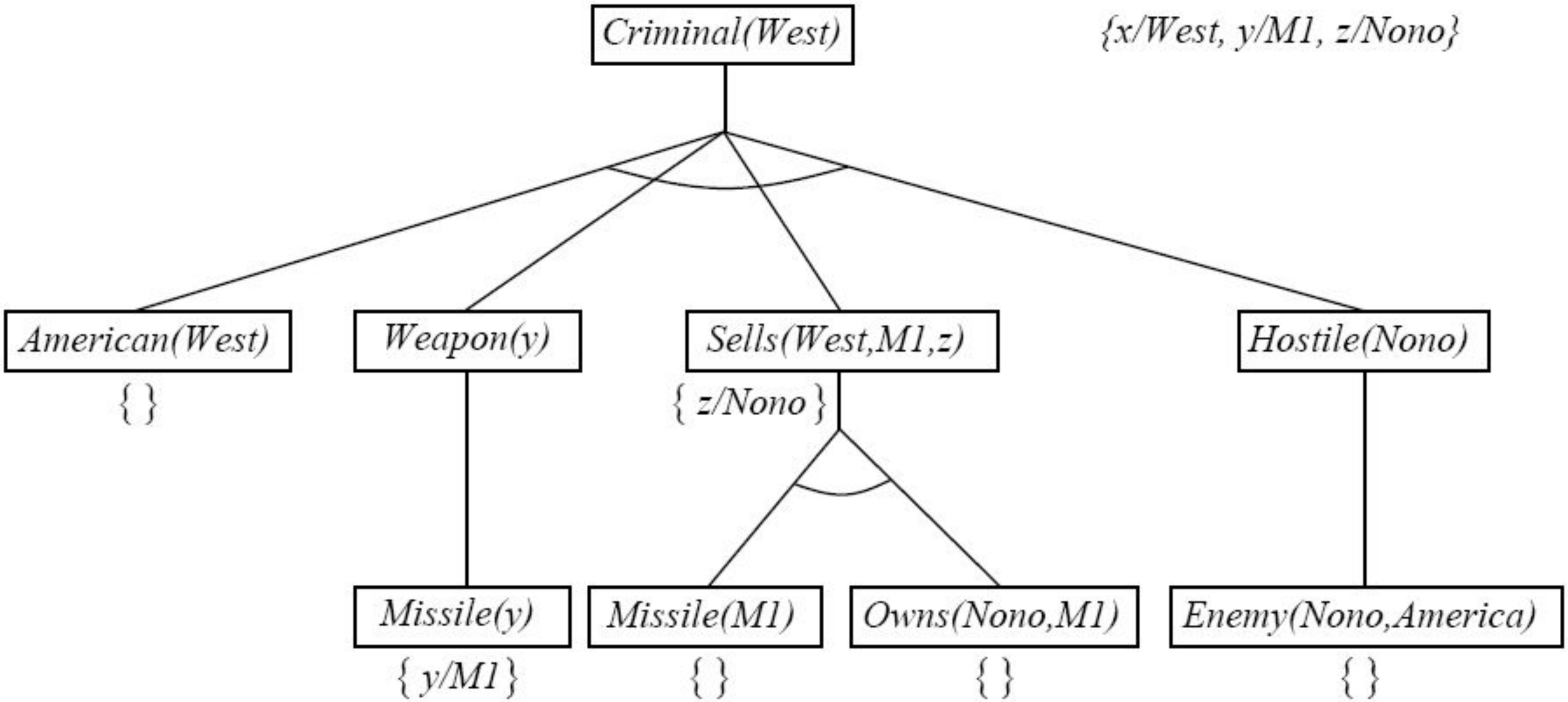
Backward Chaining Example



Backward Chaining Example



Backward Chaining Example



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, the current substitution, initially the empty substitution $\{\ \}$
  local variables: ans, a set of substitutions, initially empty

  if $goals$ is empty **then return** $\{\theta\}$
  $q' \leftarrow \textsc{Subst}(\theta, \textsc{First}(goals))$
  for each r **in** KB where $\textsc{Standardize-Apart}(r) = (\, p_1 \wedge \ldots \wedge p_n \Rightarrow q)$
    and $\theta' \leftarrow \textsc{Unify}(q, q')$ succeeds
   $ans \leftarrow \text{FOL-BC-ASK}(KB, [p_1, \ldots, p_n | \textsc{Rest}(goals)], \textsc{Compose}(\theta', \theta)) \cup ans$
  return ans

Properties of Backward Chaining

- Depth-first recursive proof search: space is linear in size of proof
- Incomplete due to infinite loops
  - Fix by checking current goal with every subgoal on the stack
- Inefficient due to repeated subgoals (both success and failure)
  - Fix using caching of previous results (extra space)
- Widely used without improvements for logic programming

Inference Methods

- Unification (prerequisite)
- Forward Chaining
  - Production Systems
- Backward Chaining
  - Logic Programming (Prolog)
- Resolution
  - Transform to CNF
  - Generalization of Prop. Logic resolution

Resolution

- Convert everything to CNF
- Resolve, with unification
- If resolution is successful, proof succeeds
- If there was a variable in the item to prove, return variable's value from unification bindings

Resolution

- Resolution allows a complete inference mechanism (search-based) using only one rule of inference
- Resolution rule:
  - Given: $P_1 \lor P_2 \lor P_3 \ldots \lor P_n$, and $\neg P_1 \lor Q_1 \ldots \lor Q_m$
  - Conclude: $P_2 \lor P_3 \ldots \lor P_n \lor Q_1 \ldots \lor Q_m$

    Complementary literals P_1 and $\neg P_1$ "cancel out"
- To prove a proposition F by resolution,
  - Start with ¬F
  - Resolve with a rule from the knowledge base (that contains F)
  - Repeat until all propositions have been eliminated
  - If this can be done, a contradiction has been derived and the original proposition F must be true.

Propositional Resolution Example

- Rules
  - Cold and precipitation -> snow
    $\neg$cold $\vee$ $\neg$precipitation $\vee$ snow
  - January -> cold
    $\neg$January $\vee$ cold
  - Clouds -> precipitation
    $\neg$clouds $\vee$ precipitation
- Facts
  - January, clouds
- Prove
  - snow

Propositional Resolution Example

$\neg snow \qquad \neg cold \lor \neg precipitation \lor snow$

$\neg cold \lor \neg precipitation \qquad \neg January \lor cold$

$\neg January \lor \neg precipitation \qquad \neg clouds \lor precipitation$

$January \qquad \neg January \lor \neg clouds$

$\neg clouds \qquad clouds$

$\square$

Resolution Theorem Proving (FOL)

- Convert everything to CNF
- Resolve, with unification
  - Save bindings as you go!
- If resolution is successful, proof succeeds
- If there was a variable in the item to prove, return variable’s value from unification bindings

Converting to CNF

1. Replace implication ($A \Rightarrow B$) by $\neg A \vee B$
2. Move $\neg$ "inwards"
   - $\neg \forall x\, P(x)$ is equivalent to $\exists x\, \neg P(x)$ & vice versa
3. Standardize variables
   - $\forall x\, P(x) \vee \forall x\, Q(x)$ becomes $\forall x\, P(x) \vee \forall y\, Q(y)$
4. Skolemize
   - $\exists x\, P(x)$ becomes $P(A)$
5. Drop universal quantifiers
   - Since all quantifiers are now $\forall$, we don't need them
6. Distributive Law

Another Resolution Example