CSE 473
Lecture 13
Chapter 9

Reasoning with First-Order Logic

Chaining

Resolution

Compilation to SAT

FOL Reasoning: Motivation

- What if we want to use modus ponens?

  Propositional Logic:

  $$\frac{a \wedge b, \quad a \wedge b \Rightarrow c}{c}$$

- In First-Order Logic?

  $\forall x$ Monkey(x) $\Rightarrow$ Curious(x)

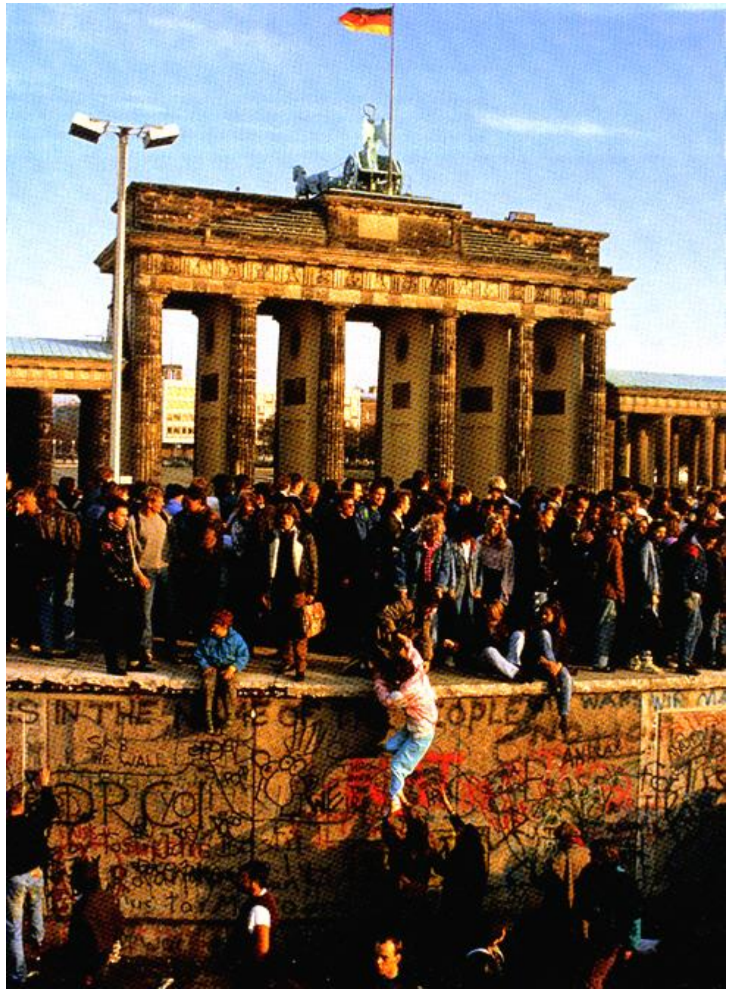
  Monkey(George)

  ????

- Must "*unify*" x with George:

  Need to substitute {x/George} in Monkey(x) $\Rightarrow$ Curious(x) to infer Curious(George)

What is Unification?



CITY PRICES

London Herald

LATE EDITION

SATURDAY 11th NOVEMBER 1989

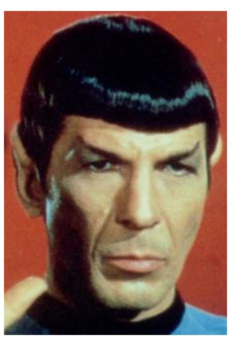
BERLIN WALL TUMBLES

'Beginning of the End' for Communism

Germany Re-unified

Breaching the Wall

OTHER NEWS



Not this kind of unification…

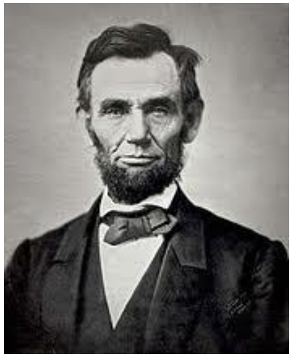
What is Unification?

- Match up expressions by *finding variable values that make the expressions identical*

Unify city(x) and city(seattle) using {x/seattle}

- Unify(a, b) returns most general unifier (MGU)

Most General Unifier



- Unify(a, b) returns most general unifier (MGU)
- MGU places fewest restrictions on values of variables
- Examples:

  Unify(city(x), city(seattle)) returns {x/seattle}

  Unify(PokesInTheEyes(Moe,x), PokesInTheEyes(y,z)) returns {y/Moe,z/x}

  {y/Moe,x/Moe,z/Moe} also possible but not MGU

Unification and Substitution

- Unification produces a mapping from variables to values (e.g., {x/seattle,y/tacoma})
- Substitution: Subst(mapping,sentence) returns new sentence with variables replaced by values

  Subst({x/seattle,y/tacoma}),connected(x, y)),

  returns connected(seattle, tacoma)

Unification Examples I

- Unify(road(x, kent), road(seattle, y))

  Returns {x / seattle,   y / kent}

  When substituted in both expressions, the resulting expressions match:

  Each is   **(road(seattle, kent))**

- Unify(road(x, x), road(seattle, kent))

  Not possible – Fails!

  x can't be seattle and kent at the same time!

Unification Examples II

- Unify(f(g(x, dog), y)), f(g(cat, y), dog)

  {x / cat, y / dog}

- Unify(f(g(x)), f(x))

  Fails: no substitution makes them identical.
  E.g. {x / g(x) } yields f(g(g(x))) and f(g(x))
  which are not identical!

- Thus: A variable may not ***contain*** itself in a substitution

  Directly or indirectly

Unification Examples III

- Unify(f(g(cat, y), y), f(x, dog))
  - { x / g(cat, dog), y / dog}
- Unify(f(g(y)), f(x))
  - {x / g(y)}
- Back to curious monkeys:

$$\frac{\begin{array}{l}\text{Monkey}(x) \rightarrow \text{Curious}(x) \\ \text{Monkey}(\text{George})\end{array}}{\text{Curious}(\text{George})} \qquad \{x \,/\, \text{George}\}$$

Unify and then use modus ponens =

generalized modus ponens (GMP)

("Lifted" version of modus ponens)

Inference I: Forward Chaining

- The algorithm:

  Start with the KB

  Add any fact you can generate with GMP (i.e., unify expressions and use modus ponens)

  Repeat until: goal reached or generation halts

Example

- It is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles. All of its missiles were sold to it by Colonel West, who is American.

- Is Col. West a criminal?

- KB of *definite clauses* (exactly 1 positive literal):

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

$Owns(Nono,M_1)$ {Skolem constant}

$Missile(M_1)$

$Enemy(Nono,America)$

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

$American(West)$

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

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

Forward chaining example

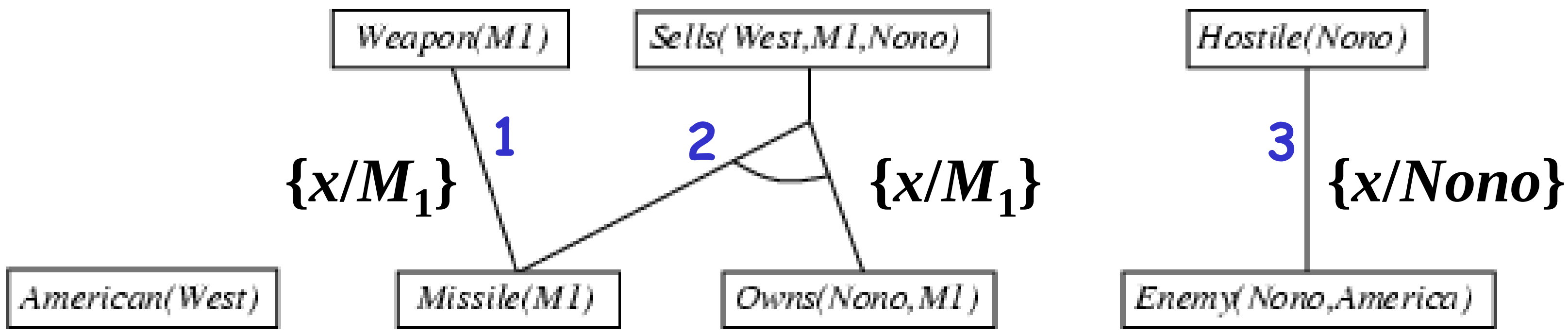
Missile(x) $\Rightarrow$ *Weapon(x)*
Missile(x) $\wedge$ *Owns(Nono,x)* $\Rightarrow$ *Sells(West,x,Nono)*
Enemy(x,America) $\Rightarrow$ *Hostile(x)*

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

Initial facts in KB

Forward chaining example

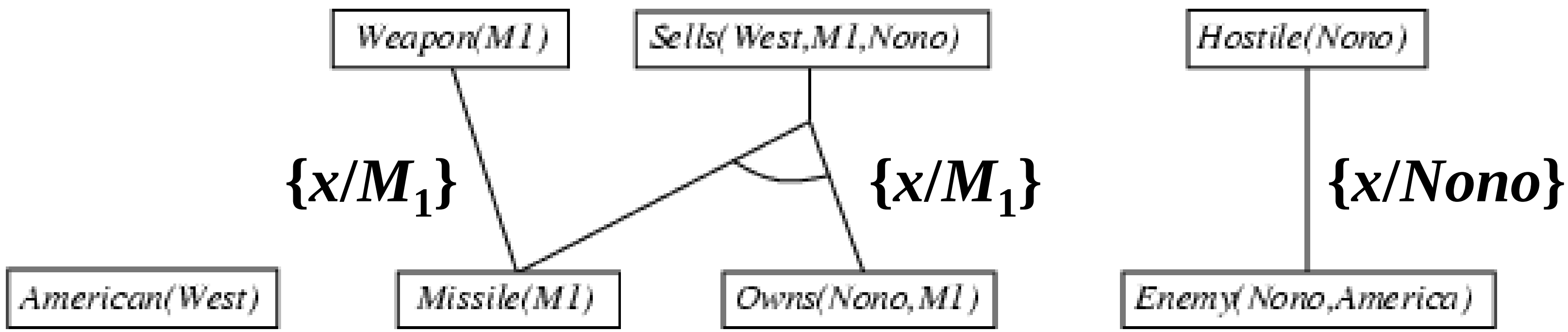
1 *Missile(x)* $\Rightarrow$ *Weapon(x)*
2 *Missile(x)* $\wedge$ *Owns(Nono,x)* $\Rightarrow$ *Sells(West,x,Nono)*
3 *Enemy(x,America)* $\Rightarrow$ *Hostile(x)*



Facts inferred after 1st iteration

Forward chaining example

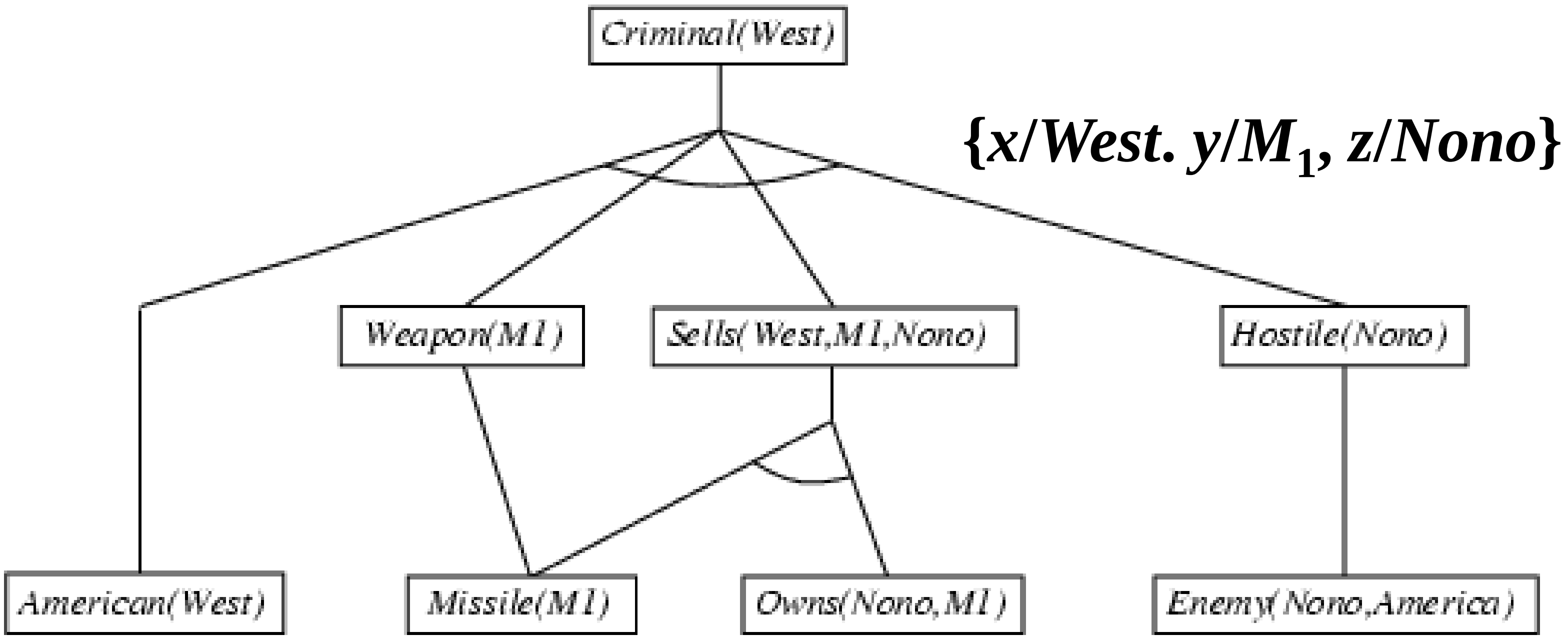
American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)



Facts inferred after 1st iteration

Forward chaining example

Col. West is a criminal



Inference I: Forward Chaining

- Sound? Complete? Decidable?

  Yes; yes for definite KB; no (see p. 331 in text)

- Speed concerns? Inefficiencies due to:

  Unification via exhaustive pattern matching; premise rechecking on every iteration; irrelevant fact generation. (see Section 9.3.3 for strategies to increase speed)

Inference II: Backward Chaining

- The algorithm:
  - Start with KB and goal.
  - Find all rules whose *results* unify with goal:
    - Add the *premises* of these rules to the goal list
    - Remove the corresponding result from the goal list
  - Stop when:
    - Goal list is empty (SUCCEED) or
    - Progress halts (FAIL)

Backward chaining example

Goal

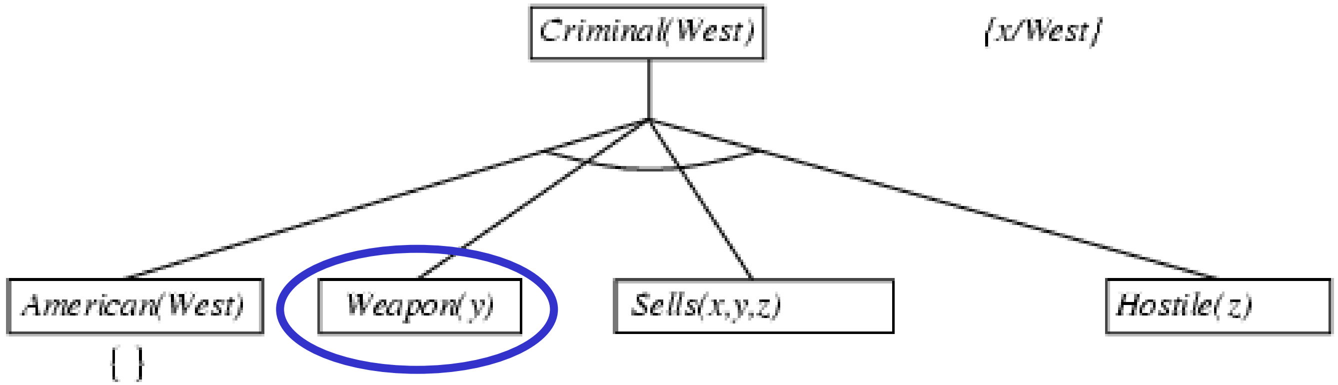
Criminal(West)

Backward chaining example

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

Backward chaining example

Depth-first traversal



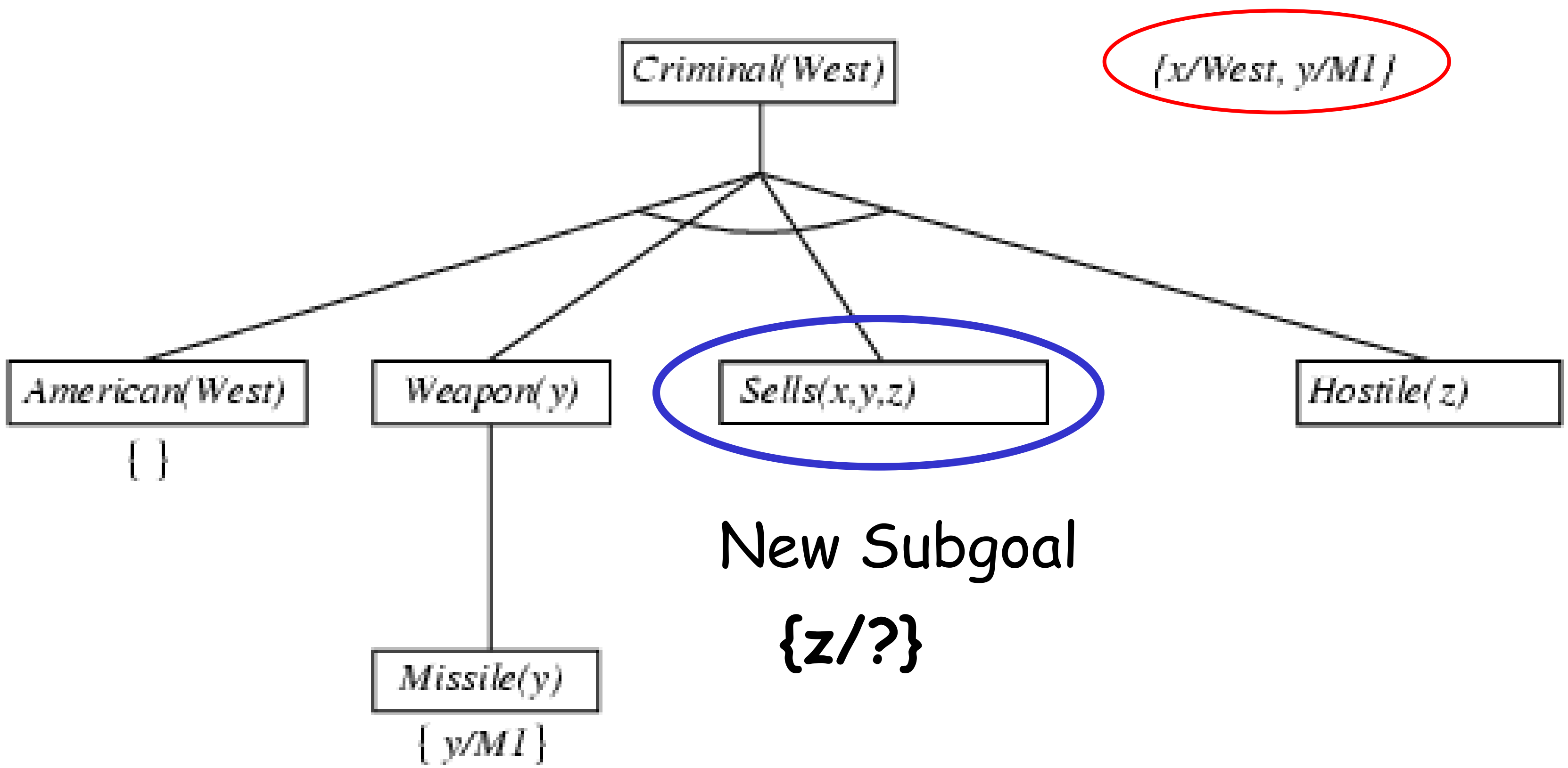
New Subgoal

KB has:

Missile(y) ⇒ Weapon(y)

Missile(M_1)

Backward chaining example



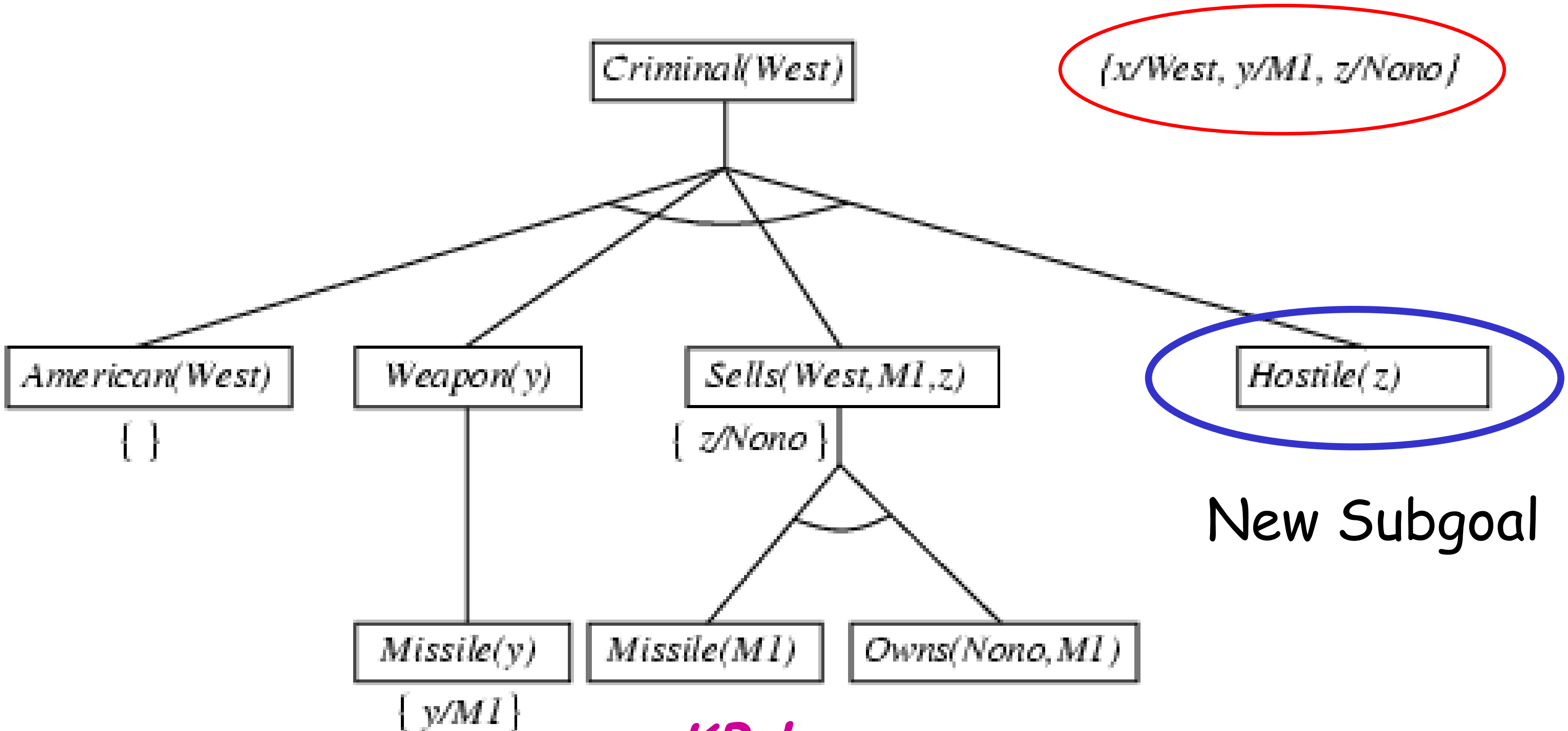
KB has:

Missile(y) ∧ Owns(Nono,y) ⇒ Sells(West,y,Nono)

Missile(M_1)

Owns(Nono,M_1)

Backward chaining example

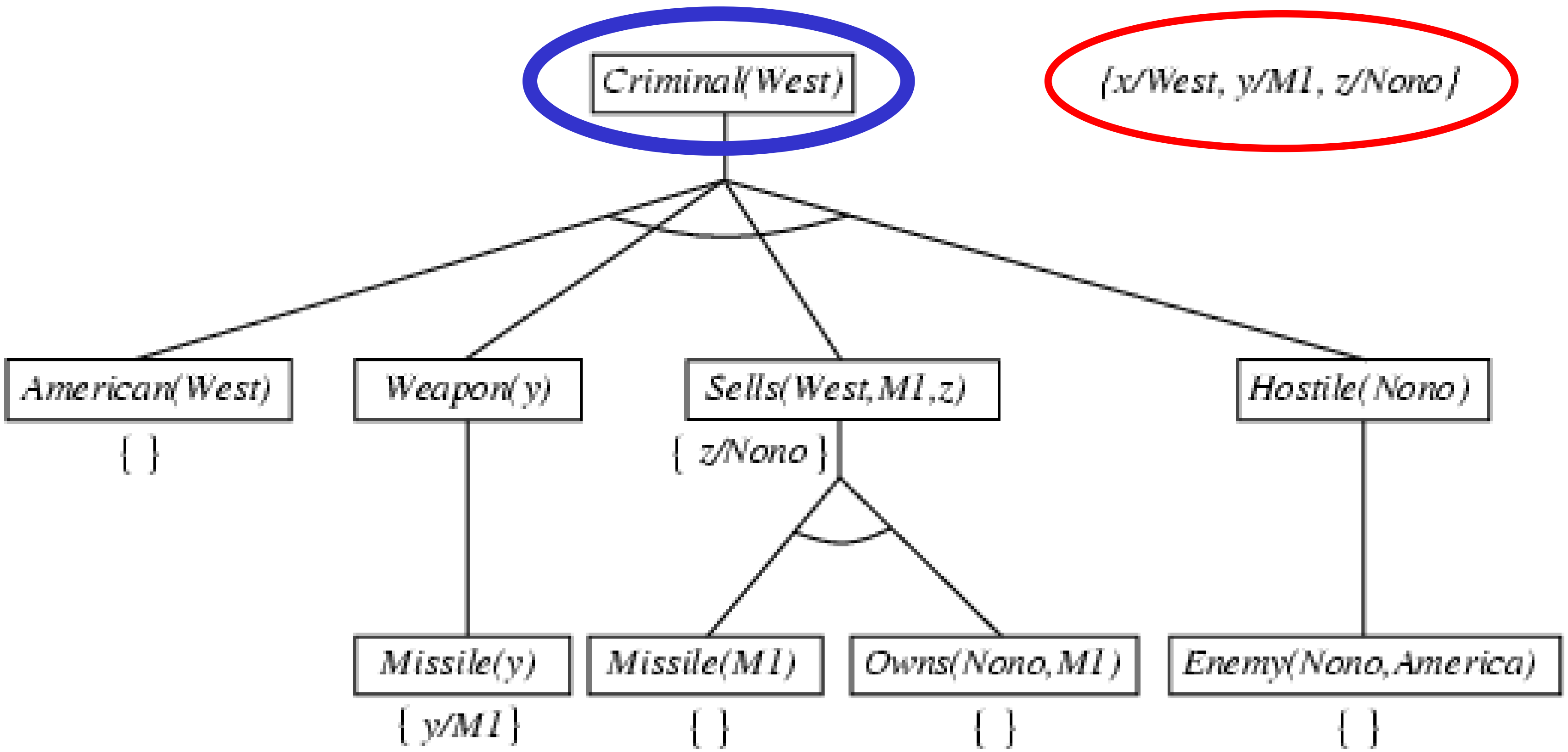


KB has:

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

Enemy(Nono,America)

Backward chaining example



Properties of backward chaining

- Depth-first recursive search: space is linear in size of proof
- Incomplete due to infinite loops (e.g. repeated states)
  - $\Rightarrow$ fix by checking current goal against goals on stack
  - $\Rightarrow$ Can't fix infinite paths though (similar to DFS)
- Inefficient due to repeated computations
  - $\Rightarrow$ fix using caching of previous results (extra space)
- Widely used for logic programming

  E.g., Prolog (logic programming language) – see Section 9.4 in text

Inference III: Resolution

$$\{ (p \vee q), (\neg p \vee r \vee s) \} \vdash_R (q \vee r \vee s)$$

Recall Propositional Case:

- **Literal in one clause**
- **Its negation in the other**
- **Result is disjunction of *other* literals**

First-Order Resolution

[Robinson 1965]

$$\{ (p(x) \vee q(A), \quad (\neg p(B) \vee r(x) \vee s(y)) \}$$

$$\vdash_R$$

$$(q(A) \vee r(B) \vee s(y))$$

Substitute MGU {x/B} in all literals

- Literal in one clause
- Negation of *something which unifies* in other
- Result is disjunction of all other literals with substitution based on MGU

Inference using First-Order Resolution

- As before, use "proof by contradiction"
  - To show $KB \models \alpha$, show $KB \wedge \neg\alpha$ unsatisfiable
- Method
  - Let $S = KB \wedge \neg goal$
  - Convert S to clausal form
    - Standardize variables (replace x in all with y, z, x_1, ...)
    - Move quantifiers to front, skolemize to remove $\exists$
    - Replace $\Rightarrow$ with $\vee$ and $\neg$
    - Use deMorgan's laws to get CNF (ands-of-ors)
  - Resolve clauses in S until empty clause (unsatisfiable) or no new clauses added

Next Time

- Wrap up of FOL
- FOL Wumpus Agent
- To Do

    Project #2 due this Saturday NOON

    Read chapter 9