CSE 473

Lecture 13

Chapter 9

Reasoning with First-Order Logic

Today's Agenda

Reasoning with First-Order Logic

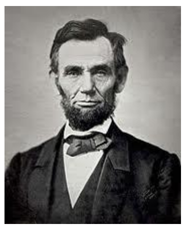
Chaining

Resolution

Compilation to SAT

Recall: Unification

- Match up expressions by *finding variable values that make the expressions identical*
  Unify city(x) and city(seattle) using {x/seattle}
- Unify(x, y) returns most general unifier (MGU)



  - MGU = unifier that places *fewest restrictions* on values of variables

3

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!

4

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 value may not ***contain*** itself in a substitution
  Directly or indirectly

5

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:

$$\frac{\text{Monkey}(x) \rightarrow \text{Curious}(x) \qquad \text{Monkey}(\text{George})}{\text{Curious}(\text{George})} \qquad \{x / \text{George}\}$$

Unify and then use modus ponens =
generalized modus ponens (GMP)
("Lifted" version of modus ponens)

6

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):

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

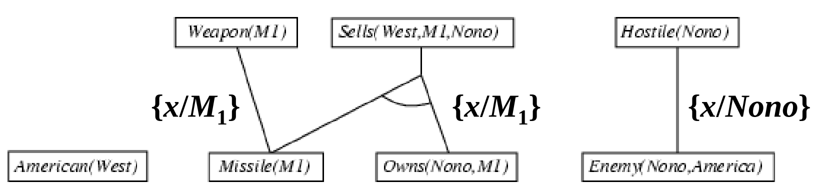
Forward chaining example

Missile(x) ⇒ Weapon(x)
Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)
Enemy(x,America) ⇒ Hostile(x)

Initial facts in KB

Forward chaining example

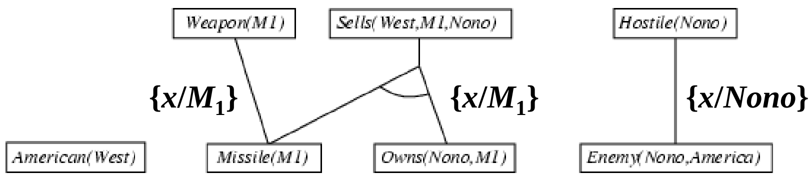
Missile(x) ⇒ Weapon(x)
Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)
Enemy(x,America) ⇒ Hostile(x)



Facts inferred after 1st iteration

Forward chaining example

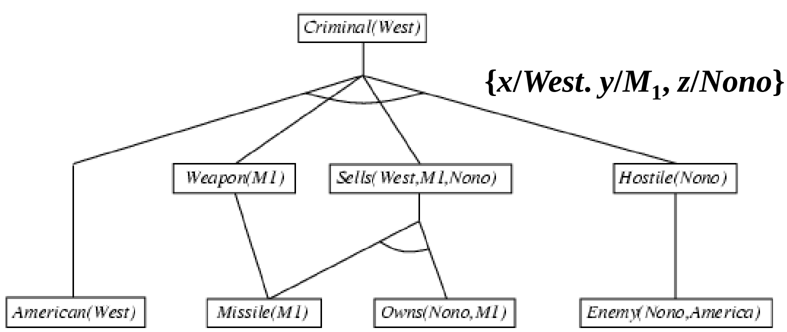
American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)



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)

13

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)

14

Backward chaining example

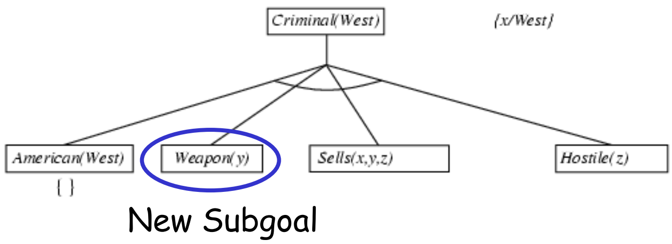
Goal

Backward chaining example

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

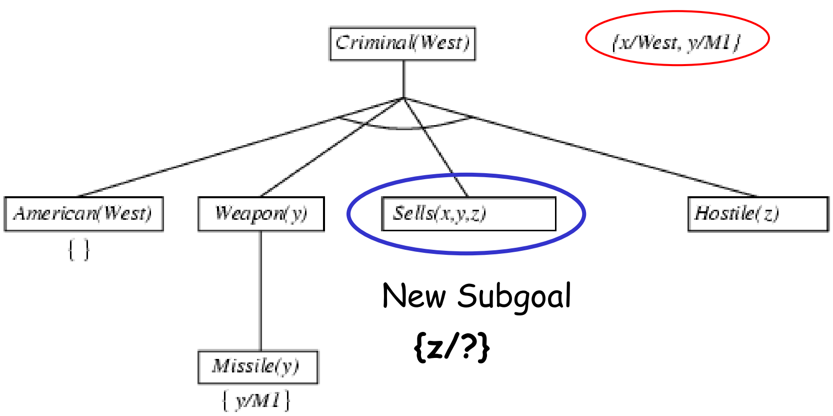
Backward chaining example

Depth-first traversal



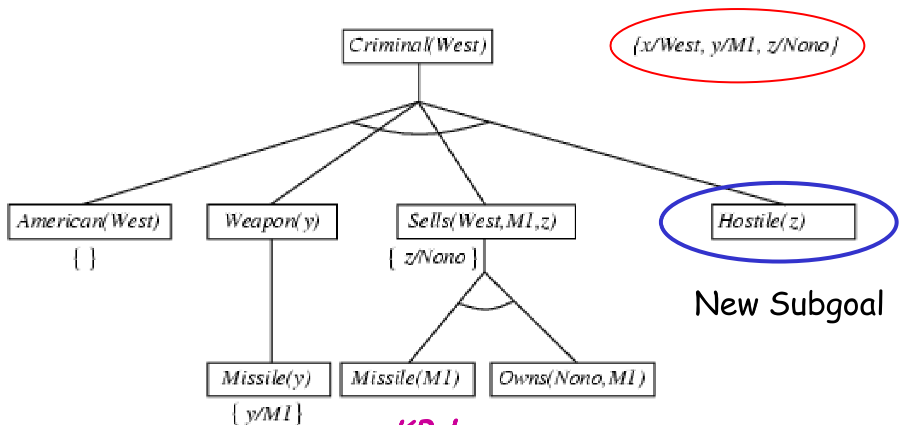
KB has:
Missile(y) ⇒ Weapon(y)
Missile(M_1)

Backward chaining example



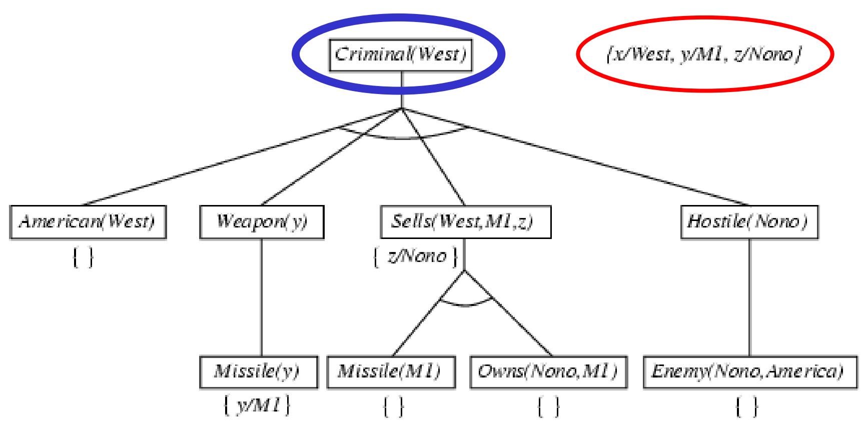
KB has:
Missile(y) ∧ Owns(Nono,y) ⇒ Sells(West,y,Nono)
Missile(M_1)
Owns(Nono,M_1)

Backward chaining example



KB has:
Enemy(z,America) ⇒ 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)
  - ⇒ fix by checking current goal against goals on stack
  - ⇒ Can't fix infinite paths though (similar to DFS)
- Inefficient due to repeated computations
  - ⇒ 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

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

Recall Propositional Case:
·Literal in one clause
·Its negation in the other
·Result is disjunction of *other* literals

22

First-Order Resolution

[Robinson 1965]

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

$\vdash_R$

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

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**

23

Inference using First-Order Resolution

- As before, use "proof by contradiction"
  To show KB $\models$ α, show KB $\wedge \neg$α 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

24

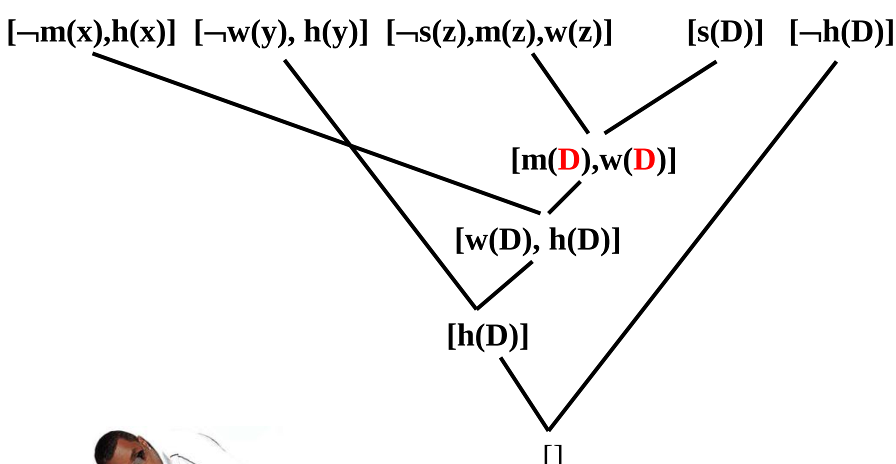
First-Order Resolution Example

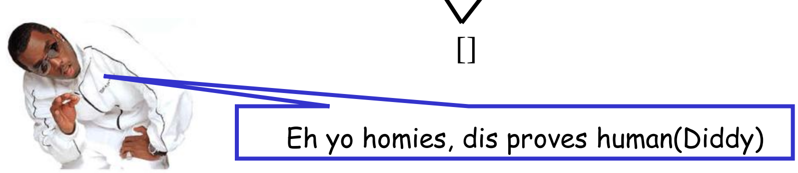
- Given
  - ∀x man(x) ⇒ human(x)
  - ∀x woman(x) ⇒ human(x)
  - ∀x singer(x) ⇒ man(x) ∨ woman(x)
  - singer(Diddy)
- Prove
  - human(Diddy)

CNF representation (list of clauses):
[¬m(x),h(x)] [¬w(y), h(y)] [¬s(z),m(z),w(z)] [s(D)] [¬h(D)]

25

FOL Resolution Example





26

Next Time

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

    Project #2 due this Thursday midnight

    Read chapter 9