CS 5100: Foundations of Artificial Intelligence

Logical Inference

Prof. Amy Sliva
September 29, 2011

Outline

- First-order logic
  - Review syntax and semantics
  - Conjunctive normal form
- Inference in FOL
  - Theoretical foundations
  - Practical implementation (forward and backward chaining)

Review: Basic syntax of FOL

• Constant symbols	*KingJohn*, *2*, *NU*, …
• Predicate symbols	*IsHappy*, *Likes*, >, …
• Function symbols	*Sqrt*, *Nationality*, …
• Variables	*x*, *y*, *a*, *b*, …
• Connectives	¬, ∧, ∨, ⇒, ⇔
• Equality	=
• Quantifiers	∀, ∃

Atomic sentences in FOL

- Atomic sentence = $predicate(term_1,\ldots,term_n)$
  or $term_1 = term_2$
- Term = $function(term_1,\ldots,term_n)$
  or constant or variable
- Examples
  - Brother(KingJohn, RichardTheLionheart)
  - >(AgeOf(Richard), AgeOf(John))
  - Brother(AgeOf(Richard), AgeOf(John))

Complex sentences

- Complex sentences are made from atomic sentences using connectives
  - Connectives have same semantics as propositional logic
  - $\neg S, S_1 \wedge S_2, S_1 \vee S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2$
- Examples
  - $Sibling(John,Richard) \Leftrightarrow Sibling(Richard,John)$
  - $>(1,2) \wedge \leq(1,2)$
  - $>(1,2) \wedge \neg>(1,2)$

Universal quantification

- ∀<*variables*> <*sentence*>
  - Everyone at NU is smart:
    $\forall x\ [\ At(x,NU) \Rightarrow Smart(x)\ ]$

- $\forall x$ P is true in a model m iff P is true with x being each possible object in the model

- Roughly speaking, equivalent to the **conjunction** of all possible **instantiations** of P

$$At(KingJohn,NU) \Rightarrow Smart(KingJohn) \quad \wedge$$
$$At(Richard,NU) \Rightarrow Smart(Richard) \quad \wedge$$
$$\mathrm{At}(NU,NU) \Rightarrow Smart(NU)$$

Existential quantification examples

- ∃<*variables*> <*sentence*>
  - Someone at NU is smart:
    $\exists x \; [\, At(x,NU) \wedge Smart(x) \,]$

- $\exists x \; P$ is true in a model m iff P is true with x being some possible object in the model

- Roughly speaking, equivalent to the **disjunction** of all possible **instantiations** of P

  $At(KingJohn,NU) \wedge Smart(KingJohn) \quad \vee$
  $At(Richard,NU) \wedge Smart(Richard) \quad \vee$
  $At(NU,NU) \wedge Smart(NU) \quad \vee \ldots$

FOL Example: Blocks World

- **Constants**
  Block, *Table*, *B1*, *B2*, *Cube*, *Brick*, *Sphere*, *Pyramid*, *Red*, *Blue*, *Large*, *Small*
- **Functions**
  Size
- **Relations**
  Cleartop, *Color*, *Loc*, *On*, *Held*, *Above*…

- **Atomic Sentences**
  $Loc(x,y)$; $On(x,y)$; $Cleartop(x)$; $Held(x)$; $Isa(x,z)$; $Color(x,c)$

FOL Example: Blocks world (cont.)

- Axioms for general **world knowledge**
  $\forall x\ [\ Isa(x, Block) \Rightarrow$
  $(\exists y\ Isa(y, Block) \wedge On(x, y)\ ) \vee On(x, Table)\ ]$
- To prevent something silly (we need "**common sense**")
  $\forall x,y\ [\ On(x, y) \Rightarrow\ \neg =(x, y)\ ]$
- Axioms for specific **state** of the world
  $Isa(B21,Block)$
  $Isa(\mathrm{B32,Block})$
  $On(B21,B32)$
  $On(B32,Table)$

Interacting with FOL KBs: substitution review

- Suppose a blocks world agent is using an FOL KB and perceives a new block configuration at time $t = 3$
  `Tell`(KB, $On(B21,B32)$, 3)
  `Ask`(KB, $\exists a\ BestAction(a,3)$)
- Does the KB entail some best action at time $t = 3$
  - Answer: {a/$Pickup(B21)$}

- **Substitution** (binding list)—ground binding for variables
  - Given sentence S and a substitution θ, $S\theta$ is result of plugging θ into S
- `Ask`(KB, S) returns some/all θ s.t. KB $\models S\theta$
- Formalize this more later today…

FOL conversion to CNF

- Everyone who loves all animals is loved by someone
  $\forall x\,[\,\forall y\,[Animal(y) \Rightarrow Loves(x,y)] \Rightarrow \exists y\; Loves(y,x)\,]$
- Converting quantified FOL sentence to CNF
  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 $\neg$ inwards: $\neg\forall x\; p \equiv \exists x\; \neg p,\;\; \neg\exists x\; p \equiv \forall x\; \neg p$
     - Use quantifier equivalence rules
       $\forall x\,[\exists y\,[\,\neg(\neg Animal(y) \vee Loves(x,y))] \vee\; \exists y\; Loves(y,x)]$
       $\forall x\,[\exists y\,[\neg\;\neg Animal(y) \wedge \neg Loves(x,y)] \vee\; \exists y\; Loves(y,x)]$
       $\forall x\,[\exists y\,[Animal(y) \wedge \neg Loves(x,y)] \vee \exists y\; Loves(y,x)]$
     - No more negated quantifiers

FOL conversion to CNF (cont.)

- Current formula:
  $\forall x\,[\exists y\,[Animal(y) \wedge \neg Loves(x,y)] \vee \exists y\,Loves(y,x)]$
- Continue CNF conversion
  3. **Standardize** variables—each quantifier should use a different one
     $\forall x_1\,[\exists y_1\,[Animal(y_1) \wedge \neg Loves(x_1,y_1)] \vee \exists y_2\,Loves(y_2,x_1)]$
  4. **Skolemize**—use only universal quantification
     - Each existential variable replaced by a Skolem function of the enclosing universal variables
       $\forall x_1\,[Animal(F_1(x_1)) \wedge \neg Loves(x_1,F_1(x_1))] \vee Loves(F_2(x_1),x_1)$
  5. Drop universal quantifiers
     $[Animal(F_1(x_1)) \wedge \neg Loves(x_1,F_1(x_1))] \vee Loves(F_2(x_1),x_1)$
  6. Distribute $\vee$ over $\wedge$
     $Animal(F_1(x_1)) \vee Loves(F_2(x_1),x_1) \wedge$
     $\neg Loves(x_1,F_1(x_1)) \vee Loves(F_2(x_1),x_1)$

Inference in FOL (RN chapter 9)

- Theoretical foundations
  - Rules of inference
  - Unification
  - Resolution as generalized modus ponens
- Implementation
  - Forward and backward chaining

Inference by universal instantiation (UI)

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

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

for any variable v and ground term g

- **Example**
$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ yields:

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

Formal notation for substitution

- **Substitution** is set of variable-term pairs: $\{x/term_1, y/term_2, \ldots\}$
  - Often denoted with symbol θ
  - No variable can occur more than once
- For any term or formula A
  Subst(θ, A) denotes result of replacing each variable with corresponding term (also written $A\theta$)
  - REMEMBER: a term is a constant symbol, a variable symbol, or a function symbol applied to one or more terms
  - **Ground term**—term with no variables
  - **Ground sentence**—sentence with no **free** variables (i.e., all have a corresponding term in a substitution)

Inference by existential instantiation (EI)

- For any sentence α, variable v, and constant symbol k that does **NOT** appear elsewhere in the knowledge base

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

- **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 symbol, called a **Skolem constant**

Implementing universal instantiation

All humans are mortal
Jack is human

Jack is mortal

- How do we represent this in FOL?
  R1. $\forall x\ Human(x) \Rightarrow Mortal(x)$
  F1. $Human(Jack)$
- **Modus ponens** for FOL
  - Let R1 be $p \Rightarrow q$, Let F1 be be p'
  - If p and p' **unify**, conclude q'
- **Unify** means we can match them up by replacing variables
  - Then apply the same replacement to q to get q'

Unification

- Two formulas A and B **unify** if there is a substitution θ such that $A\theta = B\theta$.
  - **θ is not unique!**
- Example—unify A = *Knows(John,x)* and B = *Knows(y,z)*
  θ_1 = {*y/John, x/z* } or θ_2 = {*y/John, x/Sue, z/Sue*}
- The first unifier is **more general** than the second
- There is a single **most general unifier** (MGU) that is unique up to renaming variables
  MGU = { *y/John, x/z* } or {*y/John, z/x* }

Most general unifier

- If θ_1 is a unifier for formulas A and B, it is a **MOST GENERAL UNIFIER** (MGU) iff:

  There is no other unifier θ_2 for A and B s.t. $A\theta_2$ subsumes $A\theta_1$

- A formula F **subsumes** a formula G if there is a non-trivial substitution Π s.t. $F\Pi = G$

- **Example**

  A = *Knows(John,x)* and B = *Knows(y,z)*

  $A\theta_1$ = *Knows*(*John, z*) $A\theta_2$ = *Knows*(*John, Sue*)

  $A\theta_1$ subsumes $A\theta_2$ therefore θ_2 = {*y/John, x/Sue, z/Sue*} is not an MGU

  NOTE: What is Π?

Unification examples

***Unify*(α,β) = θ if $\alpha\theta = \beta\theta$**

- Find the unifier θ

p	q	θ
Knows(*John*,*x*)	*Knows*(*John*,*Jane*)	
Knows(*John*,*x*)	*Knows*(*y*,*Barak*)	
Knows(*John*,*x*)	*Knows*(*y*,*Mother*(*y*))	
Knows(*John*,*x*)	*Knows*(*x*,*Barak*)	

Unification examples

***Unify*(α,β) = θ if $\alpha\theta = \beta\theta$**

- Find the unifier θ

p	q	θ
Knows(*John*,*x*)	*Knows*(*John*,*Jane*)	**{*x*/*Jane*}**
Knows(*John*,*x*)	*Knows*(*y*,*Barak*)	
Knows(*John*,*x*)	*Knows*(*y*,*Mother*(*y*))	
Knows(*John*,*x*)	*Knows*(*x*,*Barak*)	

Unification examples

***Unify*(α,β) = θ if $\alpha\theta = \beta\theta$**

- Find the unifier θ

p	q	θ
Knows(*John*,*x*)	*Knows*(*John*,*Jane*)	**{*x*/*Jane*}**
Knows(*John*,*x*)	*Knows*(*y*,*Barak*)	**{*x*/*Barak*,*y*/*John*}**
Knows(*John*,*x*)	*Knows*(*y*,*Mother*(*y*))	
Knows(*John*,*x*)	*Knows*(*x*,*Barak*)	

Unification examples

***Unify*(α,β) = θ if $\alpha\theta = \beta\theta$**

- Find the unifier θ

p	q	θ
Knows(*John*,*x*)	*Knows*(*John*,*Jane*)	**{*x*/*Jane*}**
Knows(*John*,*x*)	*Knows*(*y*,*Barak*)	**{*x*/*Barak*,*y*/*John*}**
Knows(*John*,*x*)	*Knows*(*y*,*Mother*(*y*))	**{x/*Mother*(*John*),y/*John*}**
Knows(*John*,*x*)	*Knows*(*x*,*Barak*)	

Unification examples

***Unify*(α,β) = θ if $\alpha\theta = \beta\theta$**

- Find the unifier θ

p	q	θ
Knows(*John*,*x*)	*Knows*(*John*,*Jane*)	**{*x*/*Jane*}**
Knows(*John*,*x*)	*Knows*(*y*,*Barak*)	**{*x*/*Barak*,*y*/*John*}**
Knows(*John*,*x*)	*Knows*(*y*,*Mother*(*y*))	**{x/*Mother*(*John*),y/*John*}**
Knows(*John*,*x*)	*Knows*(*x*,*Barak*)	**{fail}**

Unification examples

***Unify*(α,β) = θ if $\alpha\theta = \beta\theta$**

- Find the unifier θ

p	q	θ
Knows(*John*,*x*)	*Knows*(*John*,*Jane*)	**{*x*/*Jane*}**
Knows(*John*,*x*)	*Knows*(*y*,*Barak*)	**{*x*/*Barak*,*y*/*John*}**
Knows(*John*,*x*)	*Knows*(*y*,*Mother*(*y*))	**{x/*Mother*(*John*),y/*John*}**
Knows(*John*,*x*)	*Knows*(*x*,*Barak*)	**{fail}**

- Last one fails because we need to **standardize apart** the variables
  - Rename *x* in *Knows*(*x*,*Barak*) to *a*
  - Unify with θ = {*x*/*Barak*,*a*/*John*}

Unification algorithm

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

inputs: x, a variable, constant, list, or compound
y, a variable, constant, list, or compound
θ, the substitution built up so far

if θ = 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(ARGS[x], ARGS[y], UNIFY(OP[x], OP[y], θ))
else if LIST?(x) **and** LIST?(y) **then**
return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], θ))
else return failure

Unification algorithm (UNIFY-VAR function)

function UNIFY-VAR(var, x, θ) **returns** a substitution

inputs: var, a variable
x, any expression
θ, the substitution built up so far

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

Applying unification to reasoning

- **Modus ponens** says:

  Given $p \Rightarrow q$ and p

  Conclude q

- In FOL

  Given $p \Rightarrow q$, and p' (where p and p' unify by θ)

  Conclude q' s.t. $q' = q\theta$

- Suppose our KB includes:

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

  $King(John)$

- Modus ponens won't quite work since we have $p_1 \wedge p_2 \Rightarrow q$
  - But we can see using human reasoning that it should!

Generalized Modus Ponens (GMP)

- Follows from the resolution rule for FOL

$$\frac{(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q),\ p_1',\ p_2',\ \ldots,\ p_n'}{q\theta}$$ **where $p_i'\theta = p_i\theta$ for all i**

- **Example**

  p_1' is $King(John)$
  p_2' is $Greedy(y)$
  θ is {x/John,y/John}

  p_1 is $King(x)$
  p_2 is $Greedy(x)$
  q is $Evil(x)$

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

- GMP used with KB of **definite clauses** (**exactly** one positive literal)
- All variables assumed universally quantified (**Skolemized**)
- How do we get Greedy(y) in our KB ?

Soundness of GMP

- Need to show that

  $p_1', \ldots, p_n', (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models q\theta$

  provided that $p_i'\theta = p_i\theta$ for all p
- **Lemma:** For any sentence p, we have $p \models p\theta$ by UI

1. $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models (p_1 \wedge \ldots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \ldots \wedge p_n\theta \Rightarrow q\theta)$

2. $p_1', \ldots, p_n' \models p_1' \wedge \ldots \wedge p_n' \models p_1'\theta \wedge \ldots \wedge p_n'\theta$, *s.t.* $p_i'\theta = p_i\theta$ for all p

3. From 1 and 2, $q\theta$ follows by ordinary Modus Ponens

Forward chaining in FOL

- **Explicit knowledge** assumption
  - Assume percepts do not contain variables (may contain "generated symbol" constants)
  - Example: you see an unfamiliar dog in the building
    $isa(G33, dog)$
    $in(G33, WVH)$
- Add percept to KB if not already believed
- If percept **UNIFIES** with a rule premise (by θ) and all other premises $p\theta$ are believed, add $q\theta$ to KB
- This is an over-simplification, but we are moving on…

Backward chaining

- **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 of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

- Prove that Colonel West is a criminal

Arms trading KB 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., $\exists x\ Owns(Nono,x) \wedge Missile(x)$:
  $Owns(Nono,M_1) \wedge Missile(M_1)$
- …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:
  $Enemy(Nono,America)$

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** $\{\theta\}$
  $q' \leftarrow$ SUBST(θ, FIRST($goals$))
  for each r **in** KB where STANDARDIZE-APART(r) $= (p_1 \wedge \ldots \wedge p_n \Rightarrow q)$
    and $\theta' \leftarrow$ UNIFY(q, q') succeeds
    $ans \leftarrow$ FOL-BC-ASK(KB, $[p_1, \ldots, p_n | \text{REST}(goals)]$, COMPOSE($\theta, \theta'$)) $\cup$ ans
  return ans

- SUBST(COMPOSE(θ_1, θ_2), p) = SUBST(θ_2, SUBST(θ_1, p))

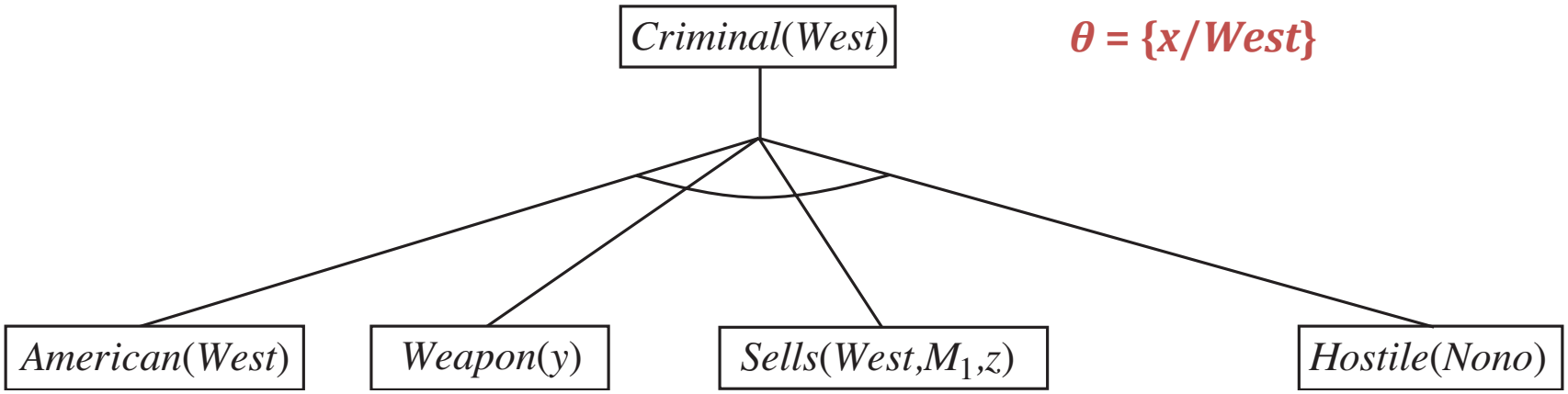
Backward chaining example

$$\boxed{Criminal(West)}$$

Goal: $Criminal(West)$

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

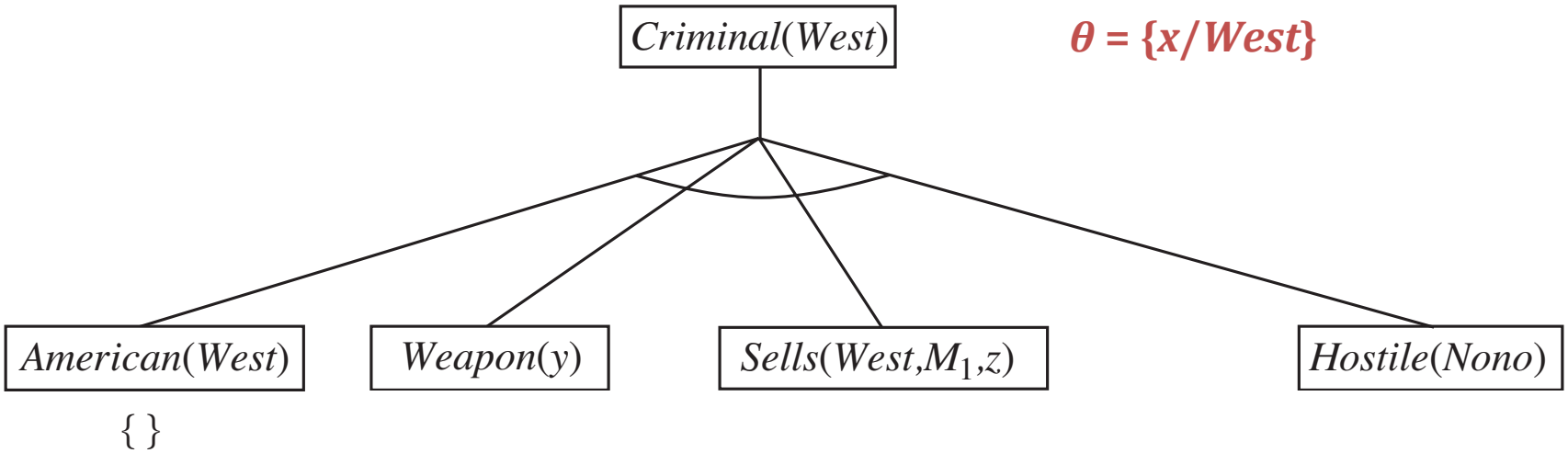
Backward chaining example



Goal: $Criminal(West)$

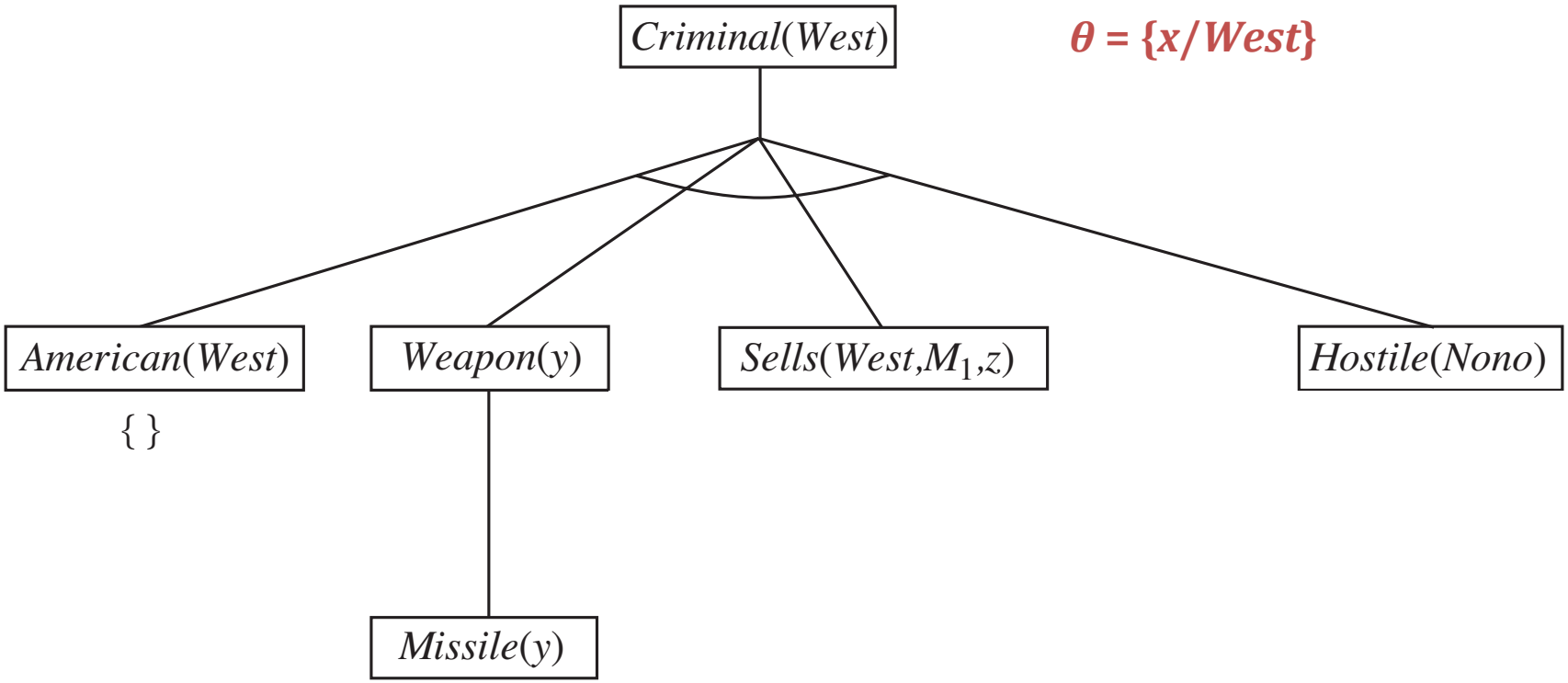
Rules: $American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Backward chaining example



Goal: ***Criminal(West)***
Rules: ***American(West) = American(x)θ***

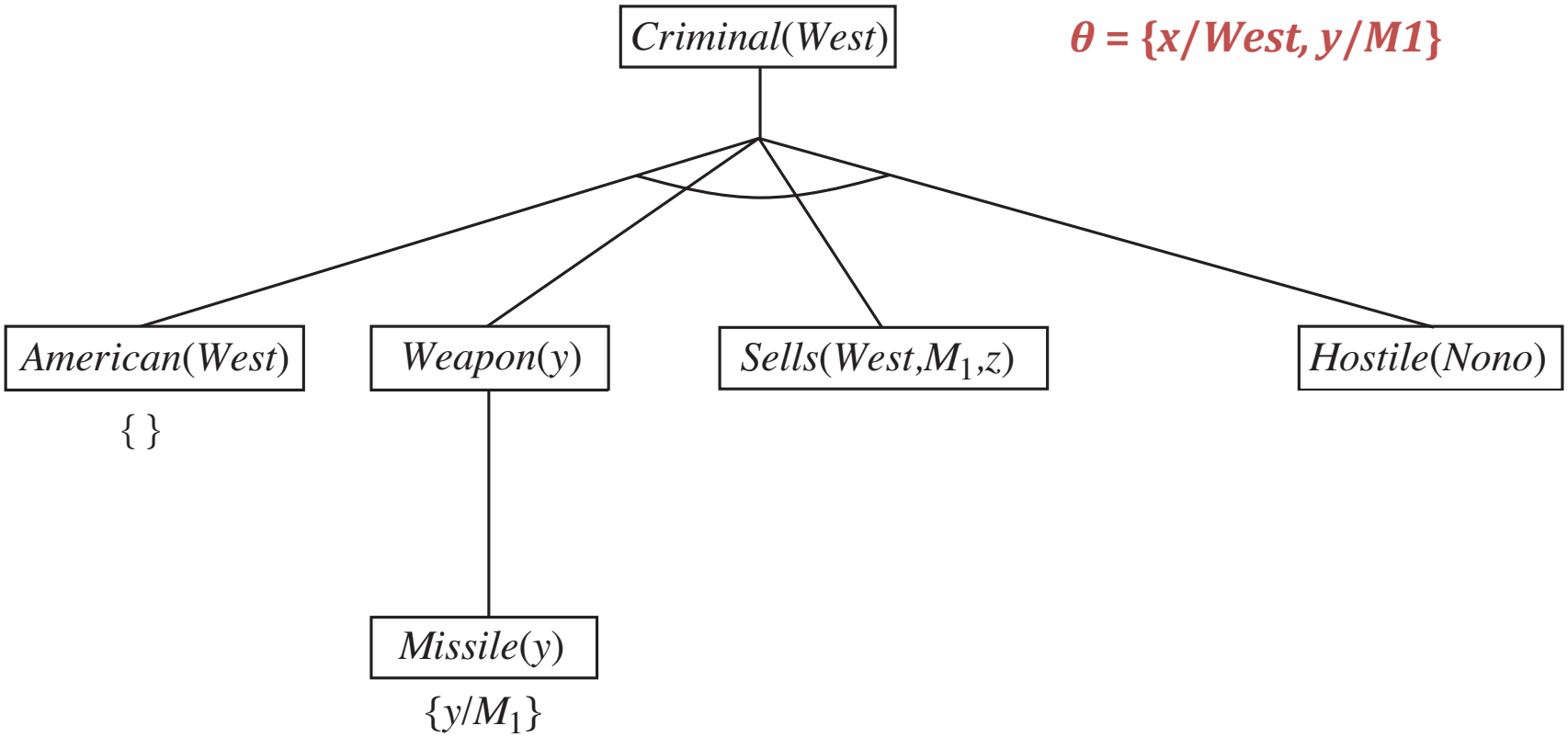
Backward chaining example



Goal: $Criminal(West)$

Rules: $Missile(y) \Rightarrow Weapon(y)$

Backward chaining example

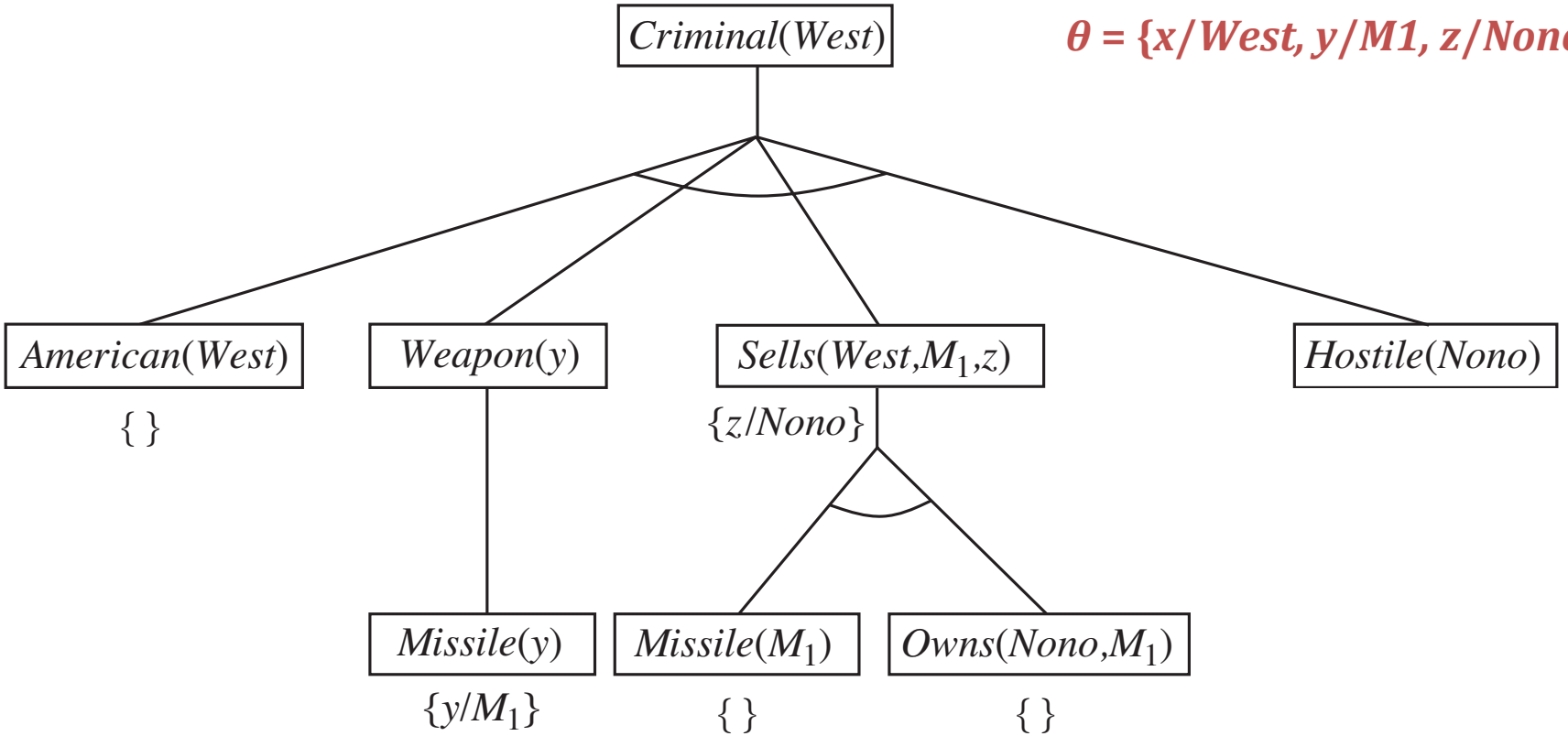


Goal: $Criminal(West)$

Rules: $Missile(y)\theta \Rightarrow Weapon(y)\theta$

$Missile(M1) = Missile(y)\theta$

Backward chaining example

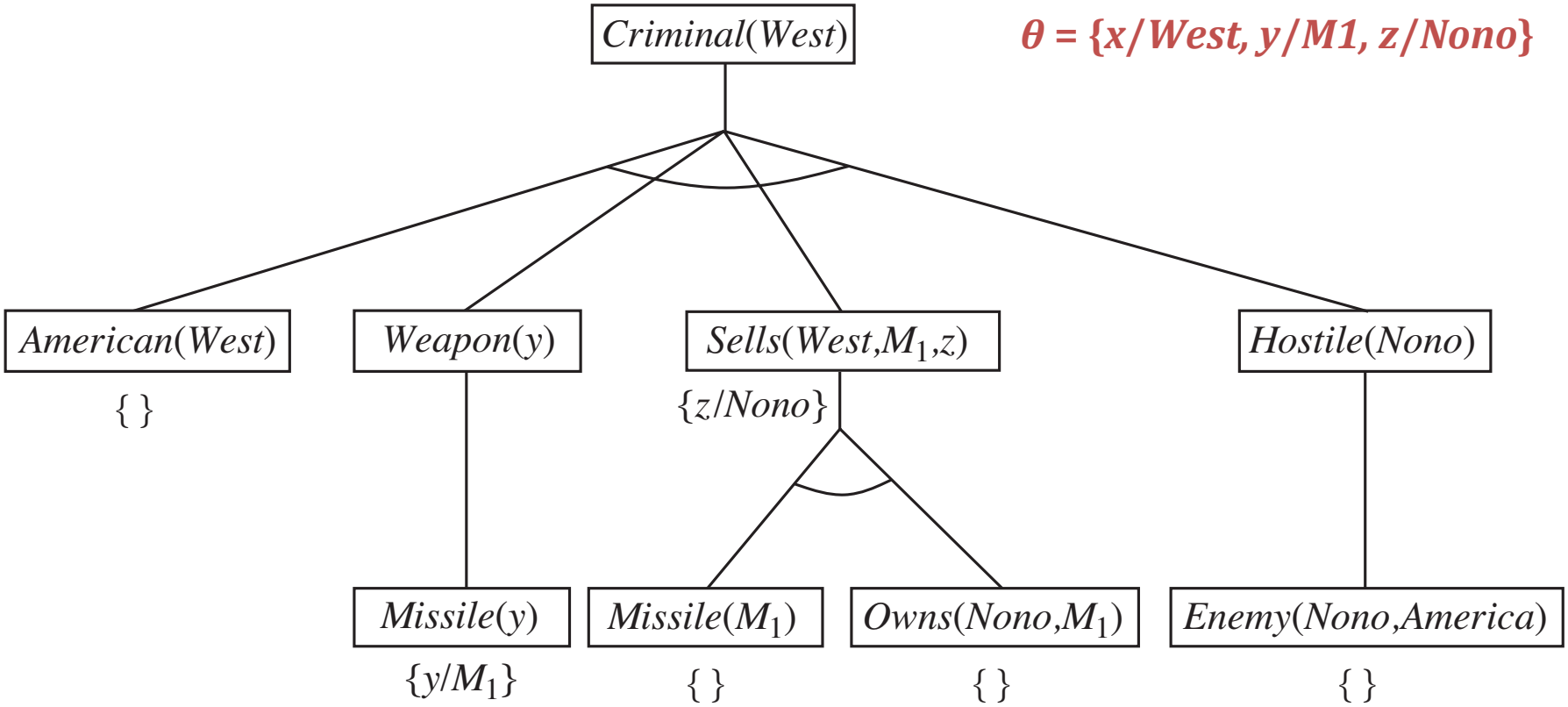


Goal: *Criminal(West)*

Rules: *Missile(y) ∧ Owns(Nono,y) ⇒ Sells(West,y,Nono)*

Missile(M1) ∧ Owns(Nono,M1) ⇒ Sells(West,M1,Nono)

Backward chaining example



Goal: *Criminal(West)*

Rules: *Enemy(z,America)θ ⇒ Hostile(z)θ*

Enemy(Nono, America)

Properties of backward chaining

- **Depth-first** recursive proof search
  - Space is linear in size of proof (we'll talk about search next week)
- **Incomplete** due to infinite loops
  - Fix by checking current goal against every goal on stack
- **Inefficient** due to repeated subgoals (both success and failure)
  - Fix using caching of previous results (extra space)
- Widely used for **logic programming**

Logic programming: Prolog

- *Algorithm* = *Logic* + *Control*
- Basis—backward chaining with Horn clauses (+ some bells & whistles)
  - Widely used in Europe, Japan (basis of 5th Generation Project)
  - Compilation techniques—60 million LIPS
- Program is a set of clauses
  - **`head :- literal`$_1$`, …,literal`$_n$`.`**
  - `criminal(x) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).`
- Depth-first, left-to-right backward chaining
- Built-in predicates for arithmetic etc., e.g., `X is Y*Z+3`
- Built-in predicates that have **side effects** (e.g., input and output)
- Predicates, assert/retract predicates
- Closed-world assumption ("negation as failure")
  - E.g., `alive(X) :- not dead(X)`
    `alive(joe)` succeeds if `dead(joe)` fails

Prolog

- Good for prototyping many small AI applications
- **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=[ ]
```

Resolution in FOL

- First-order resolution rule:

$$\frac{l_1 \vee \ldots \vee l_k,\ m_1 \vee \ldots \vee m_k}{(l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\,\theta}$$

  where $Unify(l_i, \neg m_j) = \theta$

- The two clauses are assumed to be **standardized apart**

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

  with = $\{x/Ken\}$

- Prove that KB ⊨ α by proving KB ∧ ¬α is unsatisfiable
  - Complete for FOL

Resolution proof for definite clauses