First-Order Logic

AIMA Chapter 8, 9

Pros of propositional logic

- ☺ Propositional logic allows partial/disjunctive/negated information
  - (unlike most data structures and databases)
- ☺ Propositional logic is compositional:
  - meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- ☺ Meaning in propositional logic is context-independent
  - (unlike natural language, where meaning depends on context)

Cons of propositional logic

- ☹ Hard to identify "individuals" (e.g., Mary, 3)
- ☹ Can't directly talk about properties of individuals or relations between individuals (e.g., "Bill is tall")
- ☹ Generalizations, patterns, regularities can't easily be represented (e.g., "all triangles have 3 sides")

First-order logic

- Whereas propositional logic assumes the world contains facts…
- First-order logic (like natural language) assumes the world contains
  - Objects: people, houses, numbers, colors, baseball games, wars, …
  - Relations: red, round, prime, bigger than, part of, comes between, …
  - Functions: father of, best friend of, one more than, …
- Also called first-order predicate logic

Syntax of FOL: Basic elements

- Logical symbols
  - Connectives $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
  - Quantifiers $\forall, \exists$
  - Variables x, y, a, b, ...
  - Equality =
- Non-logical symbols (ontology)
  - Constants KingArthur, 2, ShanghaiTech, …
  - Predicates Brother, >, ...
  - Functions Sqrt, LeftLegOf, ...

Atomic sentences

Atomic sentence = *predicate* (*$term_1$*,...,*$term_n$*)
or *$term_1$* = *$term_2$*

Term = *constant* or *variable*
or *function* (*$term_1$*,...,*$term_n$*)

Example:

Brother(KingJohn,RichardTheLionheart)

>(Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))

Complex sentences

Complex sentences are made from atomic sentences using connectives

$$\neg S,\ \ S_1 \wedge S_2,\ \ S_1 \vee S_2,\ \ S_1 \Rightarrow S_2,\ \ S_1 \Leftrightarrow S_2,$$

Example:

$Sibling(KingJohn,Richard) \Rightarrow Sibling(Richard,KingJohn)$

$>(1,2) \vee \leq (1,2)$

$>(1,2) \wedge \neg >(1,2)$

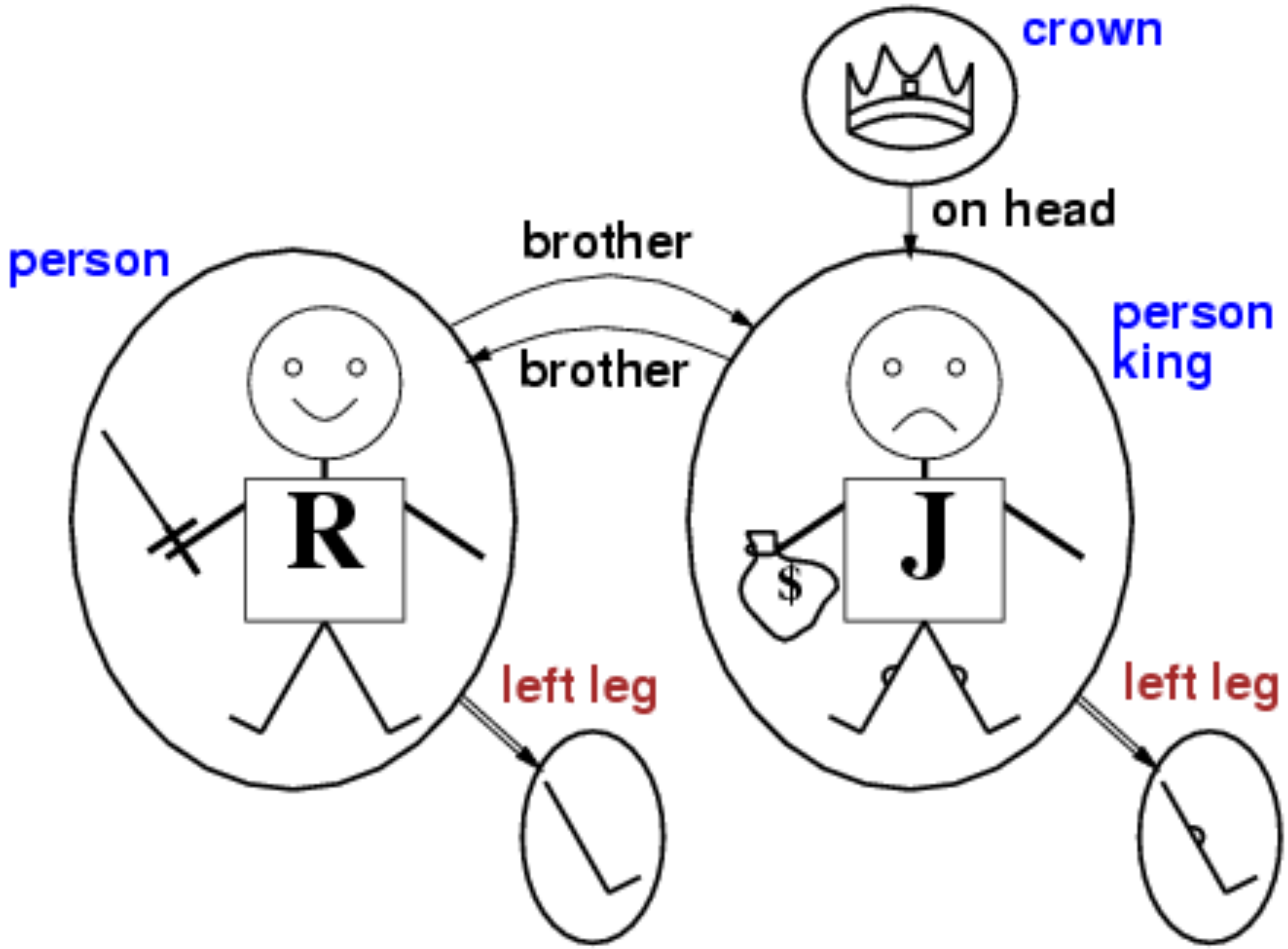
Semantics of FOL

- Sentences are true with respect to a model, which contains
  - Objects and relations among them
  - Interpretation specifying referents for

constant symbols	→	objects
predicate symbols	→	relations
function symbols	→	functional relations

- An atomic sentence $predicate(term_1,\ldots,term_n)$ is true iff the objects referred to by $term_1,\ldots,term_n$ are in the relation referred to by *predicate*

Models for FOL: Example



Models for FOL: Example

- Consider the interpretation:

  Richard → Person R

  John → Person J

  Brother → the brotherhood relation

  Under this interpretation, *Brother*(*Richard*, *John*) is true in the model.

Models for FOL

- How many models do we have?   Infinite!

  Models vary in:
  - the number of objects (1 to ∞)
  - the relations among the objects
  - the mapping from constants to objects
  - the mapping from predicates to relations
  - … …

Semantics of FOL

- Complex sentences
  - Exactly the same as in propositional logic

Rules for evaluating truth with respect to a model m:

- $\neg$S is true iff S is false
- S1 $\wedge$ S2 is true iff S1 is true and S2 is true
- S1 $\vee$ S2 is true iff S1 is true or S2 is true
- S1 $\Rightarrow$ S2 is true iff S1 is false or S2 is true
- S1 $\Leftrightarrow$ S2 is true iff S1$\Rightarrow$S2 is true and S2$\Rightarrow$S1 is true

Quantifiers

- Allows us to express properties of collections of objects instead of enumerating objects by name
- Universal: “for all” $\forall$
- Existential: “there exists” $\exists$

Universal quantification

$\forall$<variables> <sentence>

Example: $\forall x \;\; At(x,STU) \Rightarrow Smart(x)$

(Everyone at ShanghaiTech is smart)

$\forall x \; P$ 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 instantiations of P

$$
\begin{aligned}
& At(John,STU) \Rightarrow Smart(John) \\
\wedge \;\; & At(Richard,STU) \Rightarrow Smart(Richard) \\
\wedge \;\; & At(STU,STU) \Rightarrow Smart(STU) \\
\wedge \;\; & \ldots
\end{aligned}
$$

A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$:

  $\forall x \;\; At(x, STU) \wedge Smart(x)$

  means "Everyone is at STU and everyone is smart"

Existential quantification

$\exists$<variables> <sentence>

Example: $\exists x\ At(x, STU) \wedge Smart(x)$

(Someone at ShanghaiTech is smart)

$\exists x\ P$ 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 instantiations of P

$$(At(John, STU) \wedge Smart(John))$$
$$\vee\ (At(Richard, STU) \wedge Smart(Richard))$$
$$\vee\ (At(STU, STU) \wedge Smart(STU))$$
$$\vee\ \ldots$$

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

  $\exists x\ At(x,STU) \Rightarrow Smart(x)$

  is true if there is anyone who is not at STU!

Properties of quantifiers

- $\forall x\ \forall y$ is the same as $\forall y\ \forall x$
- $\exists x\ \exists y$ is the same as $\exists y\ \exists x$
- $\exists x\ \forall y$ is not the same as $\forall y\ \exists x$
  - $\exists x\ \forall y\ Loves(x,y)$

    "There is a person who loves everyone in the world"
  - $\forall y\ \exists x\ Loves(x,y)$

    "Everyone in the world is loved by at least one person"
- Quantifier duality: each can be expressed using the other

  $\forall x\ Likes(x,IceCream) \quad \equiv \quad \neg\exists x\ \neg Likes(x,IceCream)$

  $\exists x\ Likes(x,Broccoli) \quad \equiv \quad \neg\forall x\ \neg Likes(x,Broccoli)$

Sentences with variables

- A variable is free in a formula if it is not quantified
  - e.g., $\forall x \; P(x,y)$
- A variable is bound in a formula if it is quantified
  - e.g., $\forall x \, \exists y \; P(x,y)$
- In a FOL sentence, every variable must be bound.

FOL example: kinship

- Brothers are siblings

  $\forall x,y\ Brother(x,y) \Rightarrow Sibling(x,y).$
- "Sibling" is symmetric

  $\forall x,y\ Sibling(x,y) \Leftrightarrow Sibling(y,x).$
- One's mother is one's female parent

  $\forall x,y\ Mother(x,y) \Leftrightarrow (Female(x) \wedge Parent(x,y)).$
- A first cousin is a child of a parent's sibling

  $\forall x,y\ FirstCousin(x,y) \Leftrightarrow \exists p,ps\ Parent(p,x) \wedge Sibling(ps,p) \wedge Parent(ps,y)$

FOL example: kinship

- Siblings are people with the same parents

  $\forall x,y\ Sibling(x,y) \Leftrightarrow \exists m,f\ \ Parent(m,x) \wedge Parent(f,x) \wedge Parent(m,y) \wedge Parent(f,y)$

Is this correct?

Equality

$term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object

Example: Siblings are people with the same parents:

$\forall x,y\ Sibling(x,y) \Leftrightarrow [\neg(x = y) \wedge\ \exists m,f\ \ \neg(m = f) \wedge Parent(m,x) \wedge Parent(f,x) \wedge Parent(m,y) \wedge Parent(f,y)]$

FOL example

- True or false?

  $\forall x\ CollegeStudent(x) \Rightarrow Student(x)$

  $\forall x\ Student(x) \Rightarrow CollegeStudent(x)$

- Sentences are true/false with respect to a model
  - No truth-value without a model!
  - Symbols do not carry meanings by themselves

    David Hilbert: “One must be able to say at all times
    — instead of points, straight lines, and planes
    — tables, chairs, and beer mugs.”

Inference in first-order logic

Universal instantiation (UI)

(Term without variables)

- For any sentence α, variable *v* and <u>ground term</u> *g*:

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

Substitute *v* with *g* in *α*

- Every instantiation of a universally quantified sentence is entailed by it
- UI can be applied multiple times to add new sentences
- E.g., $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ 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 (EI)

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

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

- EI can be applied once to replace an existential sentence
- E.g., $\exists x\ Crown(x) \wedge OnHead(x, John)$ yields:

  $$Crown(C_1) \wedge OnHead(C_1, John)$$

  provided C_1 is a new constant symbol, called a Skolem constant

Skolemization

Reduction to propositional inference

- Suppose the KB contains just the following:
  - $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  - King(John)
  - Greedy(John)
  - Brother(Richard,John)

Reduction to propositional inference

- Instantiating the universal sentence in all possible ways, we have:
  - $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
  - $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
  - King(John)
  - Greedy(John)
  - Brother(Richard,John)
- The new KB is propositionalized: proposition symbols are
  - King(John), Greedy(John), Evil(John), King(Richard), etc.

Reduction to propositional inference

- Every FOL KB can be propositionalized so as to preserve entailment
  - i.e., a ground sentence is entailed by new KB *iff* entailed by original KB
- A naïve idea for FOL inference:
  - propositionalize KB and query, apply resolution, return result
- Problem: with function symbols, there are infinitely many ground terms,
  - e.g., *Father*(*Father*(*Father*(*John*)))

Reduction to propositional inference

- Theorem (Herbrand, 1930)
  - If a sentence α is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB
- Idea:

For $n = 0$ to ∞ do

1. create a propositional KB by instantiating with depth-n terms
2. if α is entailed by this KB, return true

Function nesting levels

Does this work?

Reduction to propositional inference

- Problem
  - works if α is entailed
  - infinite loops if α is not entailed

- Theorem (Turing, 1936; Church, 1936): entailment for FOL is semi-decidable
  - algorithms exist that say yes to every entailed sentence
  - but no algorithm exists that says no to every non-entailed sentence.

Problems with propositionalization

- Propositionalization generates many irrelevant sentences
- E.g., from:
  - $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  - $King(John)$
  - $\forall y\ Greedy(y)$

The query *Evil*(*John*) seems obviously true. But propositionalization produces lots of facts such as *Greedy*(*Richard*) that are irrelevant.

Unification

- Given:
  - $\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
  - King(John)
  - $\forall y$ Greedy(y)

Only variables can be substituted

- If we can find the substitution θ = {x/John,y/John}, then we get
  - King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)
  - King(John)
  - Greedy(John)

  and we can answer the query *Evil*(*John*) immediately
- Unification finds substitutions that make different expressions identical
  - E.g., *King(x)* vs. *King(John)*; *Greedy(x)* vs. *Greedy(y)*

Unification

- Unify(α,β) = θ if αθ = βθ

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

Unification

- Unify(α,β) = θ if αθ = βθ

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

Unification

- Unify(α,β) = θ if αθ = βθ

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))	
Knows(John,x)	Knows(x,OJ)	

Unification

- Unify(α,β) = θ if αθ = βθ

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)	

Unification

- Unify(α,β) = θ if αθ = βθ

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}

If the two x are bound to different quantifiers, then they can be standardized apart: eliminate overlap of variables, e.g., Knows(z,OJ)

Unification

- To unify Knows(John,x) and Knows(y,z),
  θ = {y/John, x/z } or θ = {y/John, x/John, z/John}
  - The first unifier is more general than the second.

- There is a single most general unifier (MGU) that is unique up to renaming of variables.
  - MGU = { y/John, x/z }

FOL Inference

- Horn logic (the FOL case)
  - Forward chaining
  - Backward chaining
- General FOL
  - Resolution

Horn clauses in FOL

- $p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q$
  - $p_1, p_2, \ldots, p_n$, q are atomic sentences
  - All variables assumed to be universally quantified
- E.g., $\mathit{human}(x) \Rightarrow \mathit{mortal}(x)$

Generalized Modus Ponens (GMP)

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

where $p_i'\theta = p_i\,\theta$ for all *i*

Example: King(John), Greedy(y), (King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x))

p_1' is *King*(*John*) p_1 is *King*(*x*)

p_2' is *Greedy*(*y*) p_2 is *Greedy*(*x*)

Therefore, θ is {x/John, y/John}

q is *Evil*(*x*), so qθ is *Evil*(*John*)

Example knowledge base

- The US 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 Col. West is a criminal

Example knowledge base contd.

... 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., $\exists x\ Owns(Nono,x) \wedge Missile(x)$:

$Owns(Nono,M_1)$ and $Missile(M_1)$

… 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 …

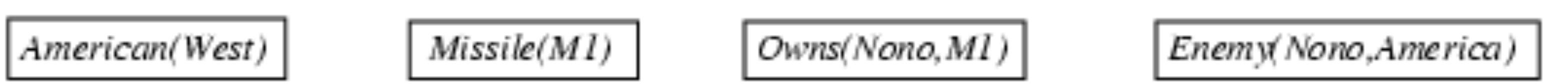
$American(West)$

The country Nono, an enemy of America …

$Enemy(Nono,America)$

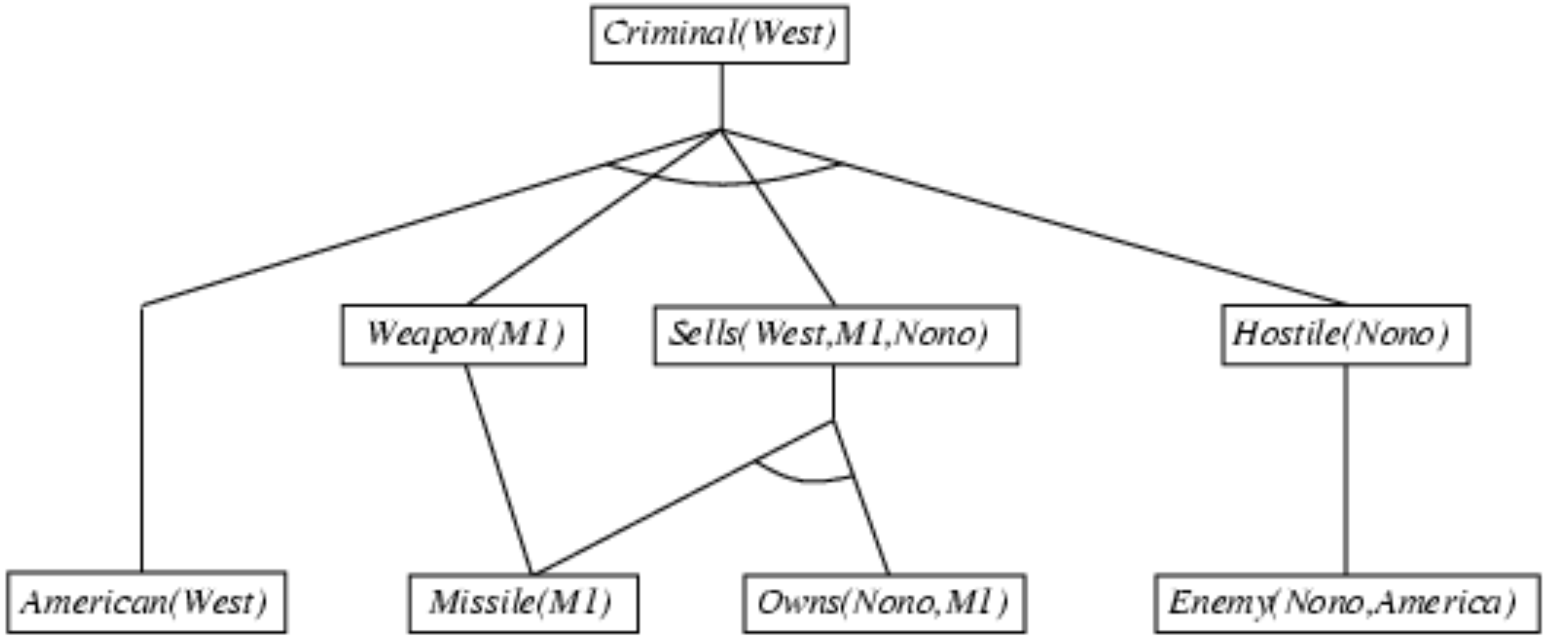
Forward chaining proof

- *American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
- *Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
- *Missile(x) ⇒ Weapon(x)*
- *Enemy(x,America) ⇒ Hostile(x)*
- *Owns(Nono,M_1), Missile(M_1), American(West), Enemy(Nono,America)*



Forward chaining proof

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



Properties of forward chaining

- Sound and complete for first-order Horn clauses
- FC terminates for first-order Horn clauses with no functions (Datalog) in finite number of iterations
- In general, FC may not terminate if α is not entailed
  - This is unavoidable: entailment with Horn clauses is also semi-decidable

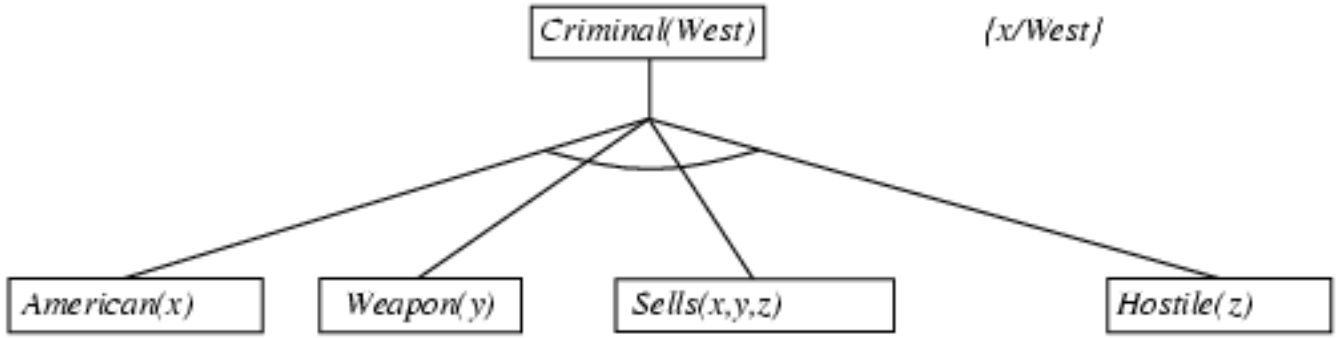
Backward chaining example

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

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

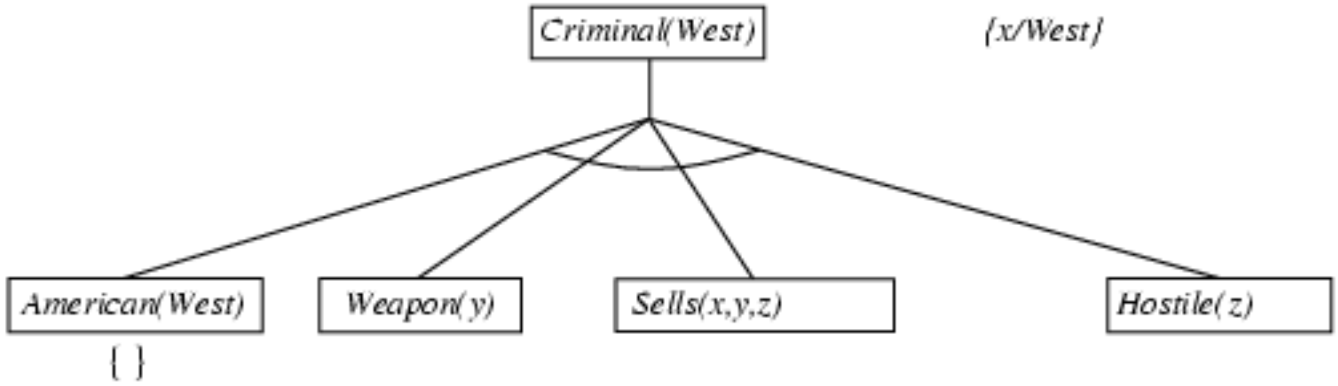
Backward chaining example

- *American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
- *Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
- *Missile(x) ⇒ Weapon(x)*
- *Enemy(x,America) ⇒ Hostile(x)*
- *Owns(Nono,M_1), Missile(M_1), American(West), Enemy(Nono,America)*



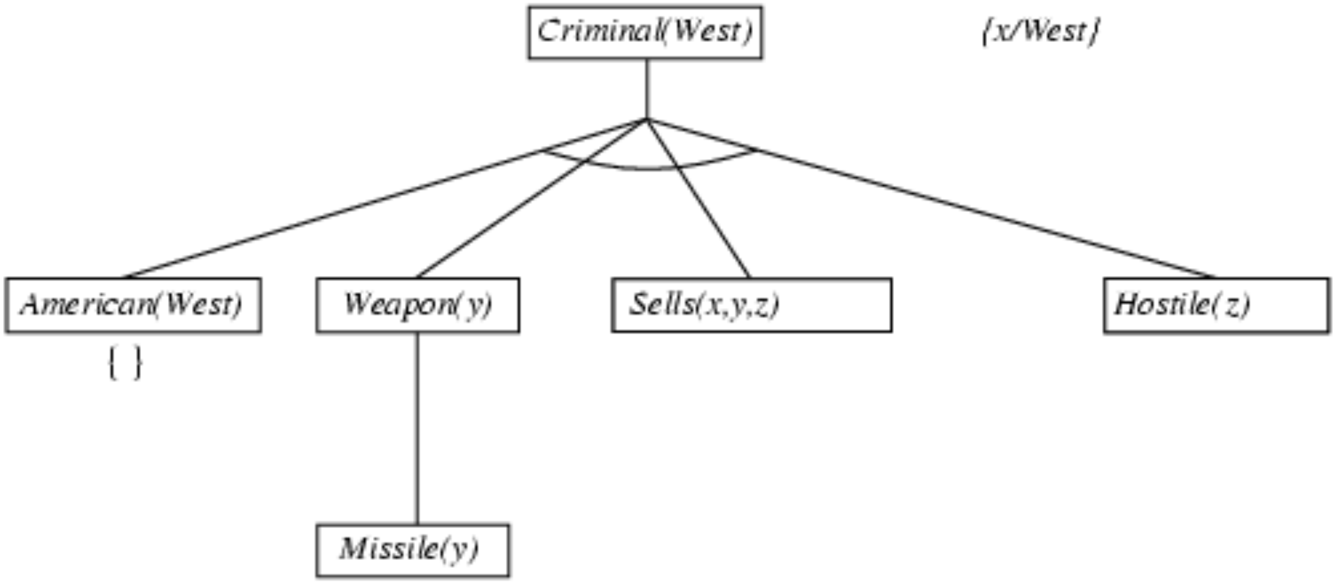
Backward chaining example

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



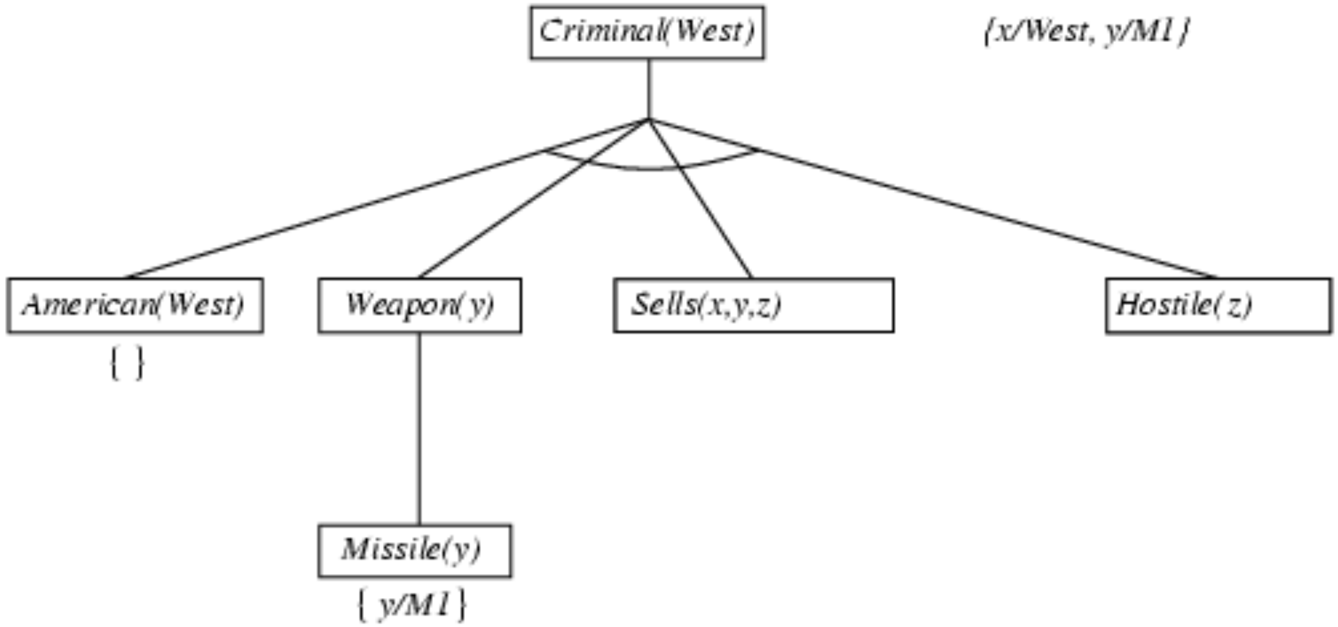
Backward chaining example

- *American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
- *Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
- *Missile(x) ⇒ Weapon(x)*
- *Enemy(x,America) ⇒ Hostile(x)*
- *Owns(Nono,M_1), Missile(M_1), American(West), Enemy(Nono,America)*



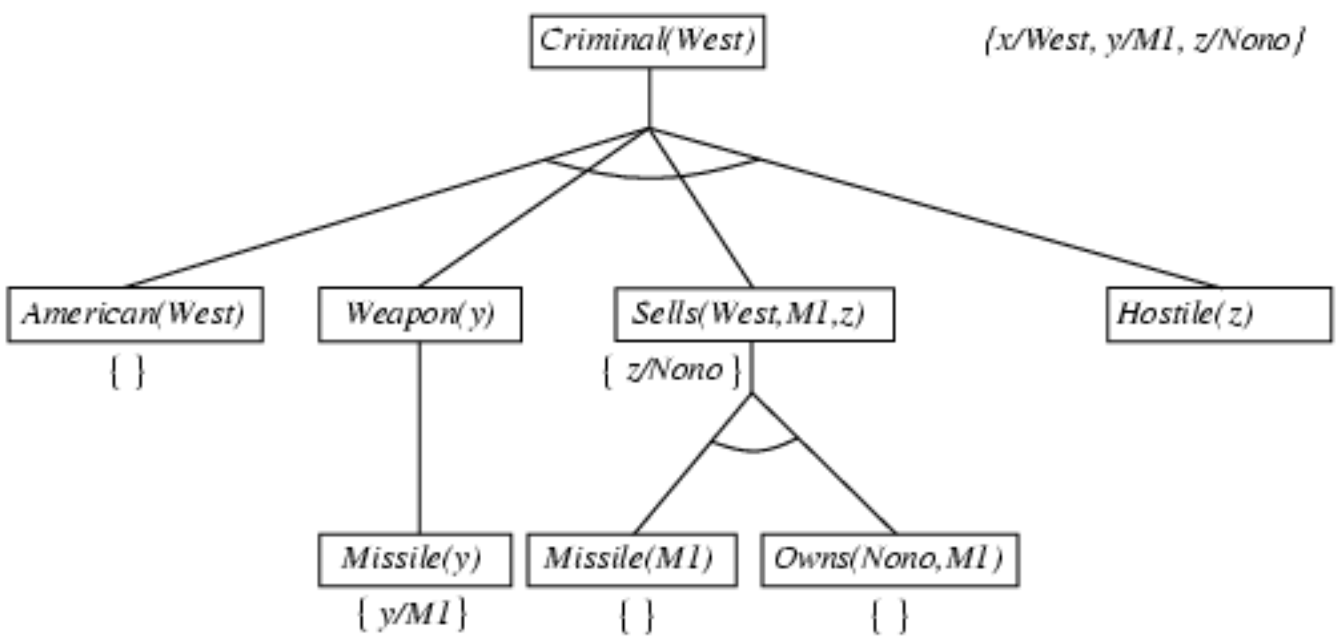
Backward chaining example

- *American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
- *Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
- *Missile(x) ⇒ Weapon(x)*
- *Enemy(x,America) ⇒ Hostile(x)*
- *Owns(Nono,M_1), Missile(M_1), American(West), Enemy(Nono,America)*



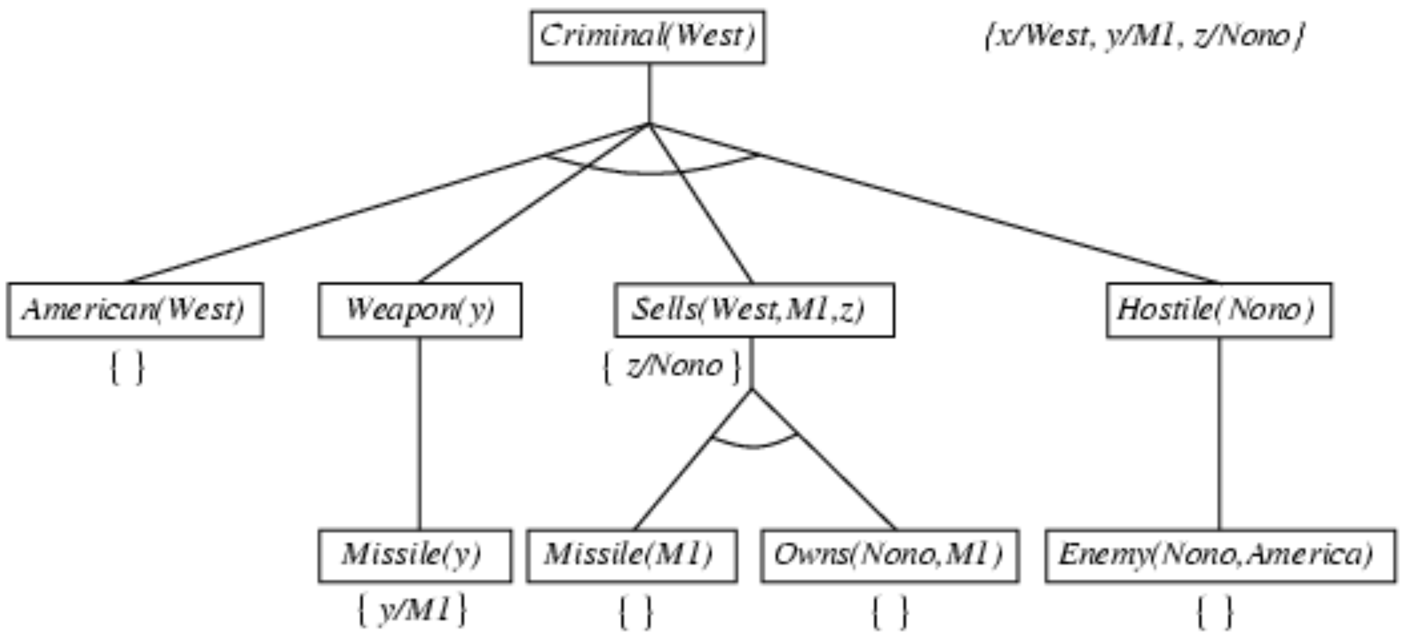
Backward chaining example

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



Backward chaining example

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



Backward chaining

- Depth-first recursive proof search: space is linear in size of proof
- Avoid infinite loops by checking current goal against every goal on stack
- Avoid repeated subgoals by caching previous results

- Widely used for logic programming

Logic programming

- Ordinary programming
  - Identify problem
  - Assemble information
  - Figure out solution
  - Encode solution
  - Encode problem instance as data
  - Apply program to data

- Logic programming
  - Identify problem
  - Assemble information
  - <coffee break> ☺
  - Encode info in KB
  - Encode problem instances as facts
  - Ask queries (run SAT solver)

Logic programming: Prolog

- Was widely used in Europe, Japan (basis of 5th Generation project)
- Basis: backward chaining with Horn clauses
  - Program = set of Horn clauses
  - Inference: depth-first, left-to-right backward chaining
- Additions:
  - 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")

Resolution

- Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

  where $\mathtt{Unify}(\ell_i, \neg m_j) = \theta$.

  - The two clauses are assumed to be standardized apart so that they share no variables.

- Example:

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

  with $\theta = \{x/Ken\}$

- Inference algorithm: applying resolution steps to CNF(KB $\wedge \neg\alpha$)
- Resolution is sound and complete for FOL

Conversion to CNF

$\forall x\ [\forall y\ \mathit{Animal}(y) \Rightarrow \mathit{Loves}(x,y)] \Rightarrow [\exists y\ \mathit{Loves}(y,x)]$

1. Eliminate biconditionals and implications

   $\forall x\ [\neg\forall y\ \neg\mathit{Animal}(y) \vee \mathit{Loves}(x,y)] \vee [\exists y\ \mathit{Loves}(y,x)]$

2. Move $\neg$ inwards: $\neg\forall x\ p \equiv \exists x\ \neg p,\ \ \neg\exists x\ p \equiv \forall x\ \neg p$

   $\forall x\ [\exists y\ \neg(\neg \mathrm{Animal}(y) \vee \mathrm{Loves}(x,y))] \vee [\exists y\ \mathrm{Loves}(y,x)]$

   $\forall x\ [\exists y\ \neg\neg \mathrm{Animal}(y) \wedge \neg \mathrm{Loves}(x,y)] \vee [\exists y\ \mathrm{Loves}(y,x)]$

   $\forall x\ [\exists y\ \mathrm{Animal}(y) \wedge \neg \mathrm{Loves}(x,y)] \vee [\exists y\ \mathrm{Loves}(y,x)]$

3. Standardize variables: each quantifier should use a different variable

   $\forall x\ [\exists y\ \mathrm{Animal}(y) \wedge \neg \mathrm{Loves}(x,y)] \vee [\exists z\ \mathrm{Loves}(z,x)]$

Conversion to CNF contd.

$\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z\ Loves(z,x)]$

4. Skolemize: a more general form of existential instantiation. Each existential 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. Distribute $\vee$ over $\wedge$:

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

6. Drop universal quantifiers