

Announcement

- Project 1b due 11:59pm today!
- Homework 2
 - Due: March 17, 11:59pm
- Programming Assignment 2
 - Due: March 24, 11:59pm

First-Order Logic

AIMA Chapter 8, 9

Syntax of FOL: Basic elements

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

Atomic sentences

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

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

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:

$$\textit{Sibling}(\textit{KingJohn}, \textit{Richard}) \Rightarrow \textit{Sibling}(\textit{Richard}, \textit{KingJohn})$$

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

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

Semantics of FOL

- Sentences are true with respect to a **model**, which contains
 - **Objects** and **relations** among them
 - **Interpretation** specifying referents for
 - constant symbols** \rightarrow **objects**
 - predicate symbols** \rightarrow **relations**
 - function symbols** \rightarrow **functional relations**
- An atomic sentence ***predicate(term₁, ..., term_n)*** is true iff the **objects** referred to by ***term₁, ..., term_n*** are in the **relation** referred to by ***predicate***

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
-

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: $\forall x \text{ } At(x, STU) \Rightarrow Smart(x)$

(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

$At(John, STU) \Rightarrow Smart(John)$

$\wedge At(Richard, STU) \Rightarrow Smart(Richard)$

$\wedge At(STU, STU) \Rightarrow Smart(STU)$

$\wedge \dots$

A common mistake to avoid

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

$$\forall x \text{ } At(x, STU) \wedge Smart(x)$$

means “Everyone is at STU and everyone is smart”

Existential quantification

\exists <variables> <sentence>

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

(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

$(\text{At}(\text{John}, \text{STU}) \wedge \text{Smart}(\text{John}))$

$\vee (\text{At}(\text{Richard}, \text{STU}) \wedge \text{Smart}(\text{Richard}))$

$\vee (\text{At}(\text{STU}, \text{STU}) \wedge \text{Smart}(\text{STU}))$

$\vee \dots$

Another common mistake to avoid

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

$$\exists x \text{ } At(x, STU) \Rightarrow Smart(x)$$

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 $\forall y \exists x$
 - $\exists x \forall y \text{ Loves}(x,y)$

“There is a person who loves everyone in the world”
 - $\forall y \exists x \text{ Loves}(x,y)$

“Everyone in the world is loved by at least one person”
- Quantifier duality: each can be expressed using the other
 - $\forall x \text{ Likes}(x, \text{IceCream}) \quad \equiv \quad \neg \exists x \neg \text{Likes}(x, \text{IceCream})$
 - $\exists x \text{ Likes}(x, \text{Broccoli}) \quad \equiv \quad \neg \forall x \neg \text{Likes}(x, \text{Broccoli})$

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

$$\forall x,y \text{ Brother}(x,y) \Rightarrow \text{Sibling}(x,y).$$

- "Sibling" is symmetric

$$\forall x,y \text{ Sibling}(x,y) \Leftrightarrow \text{Sibling}(y,x).$$

- One's mother is one's female parent

$$\forall x,y \text{ Mother}(x,y) \Leftrightarrow (\text{Female}(x) \wedge \text{Parent}(x,y)).$$

- A first cousin is a child of a parent's sibling

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

FOL example: kinship

- Siblings are people with the same parents

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

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:

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

TODO: no truth-value without model

- Give a FOL sentence that looks wrong on the surface
- 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 ground term g :

$$\frac{\forall v \alpha}{\text{Subst}(\{v/g\}, \alpha)} \longleftarrow \text{Substitute } v \text{ with } g \text{ in } \alpha$$

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

Existential instantiation (EI)

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

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

- EI can be applied once to replace an existential sentence
- E.g., $\exists x \text{Crown}(x) \wedge \text{OnHead}(x, \text{John})$ yields:
 $\text{Crown}(C_1) \wedge \text{OnHead}(C_1, \text{John})$
provided C_1 is a new constant symbol, called a **Skolem constant**

Reduction to propositional inference

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

Reduction contd.

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


Reduction contd.

- 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 contd.

- 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 contd.

- 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:
 - $\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$
 - $\text{King}(\text{John})$
 - $\forall y \text{ Greedy}(y)$

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

Unification

- Given:
 - $\forall x \text{ King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$
 - $\text{King}(\text{John})$
 - $\forall y \text{ Greedy}(y)$
 - If we can find the substitution $\theta = \{x/\text{John}, y/\text{John}\}$, then we get
 - $\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$
 - $\text{King}(\text{John})$
 - $\text{Greedy}(\text{John})$
- and we can answer the query *Evil(John)* immediately
- Unification finds substitutions that make different expressions identical
 - E.g., $\text{King}(x)$ vs. $\text{King}(\text{John})$; $\text{Greedy}(x)$ vs. $\text{Greedy}(y)$

Only variables can
be substituted



Unification

- $\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

- $\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

- $\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

- $\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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

- $\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

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}

Can be avoided by **standardizing apart**: eliminate overlap of variables, e.g., Knows(z,OJ)

Unification

- To unify $\text{Knows}(\text{John}, x)$ and $\text{Knows}(y, z)$,
 $\theta = \{y/\text{John}, x/z\}$ or $\theta = \{y/\text{John}, x/\text{John}, z/\text{John}\}$
 - 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.
 - $\text{MGU} = \{y/\text{John}, x/z\}$

FOL Inference

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

Horn clauses in FOL

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

Generalized Modus Ponens (GMP)

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

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 $\{x/\text{John}, y/\text{John}\}$

q is *Evil(x)*, so $q\theta$ 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

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

American(West)

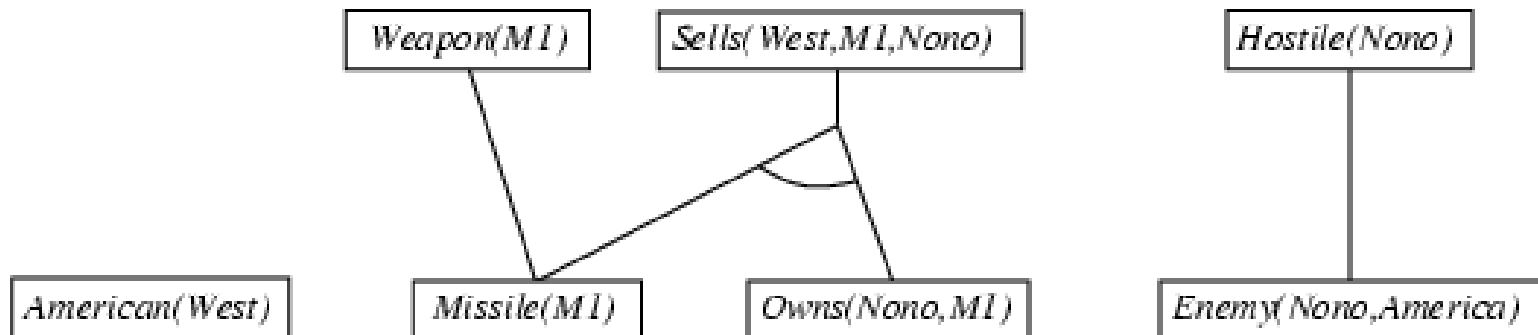
Missile(M1)

Owns(Nono,M1)

Enemy(Nono,America)

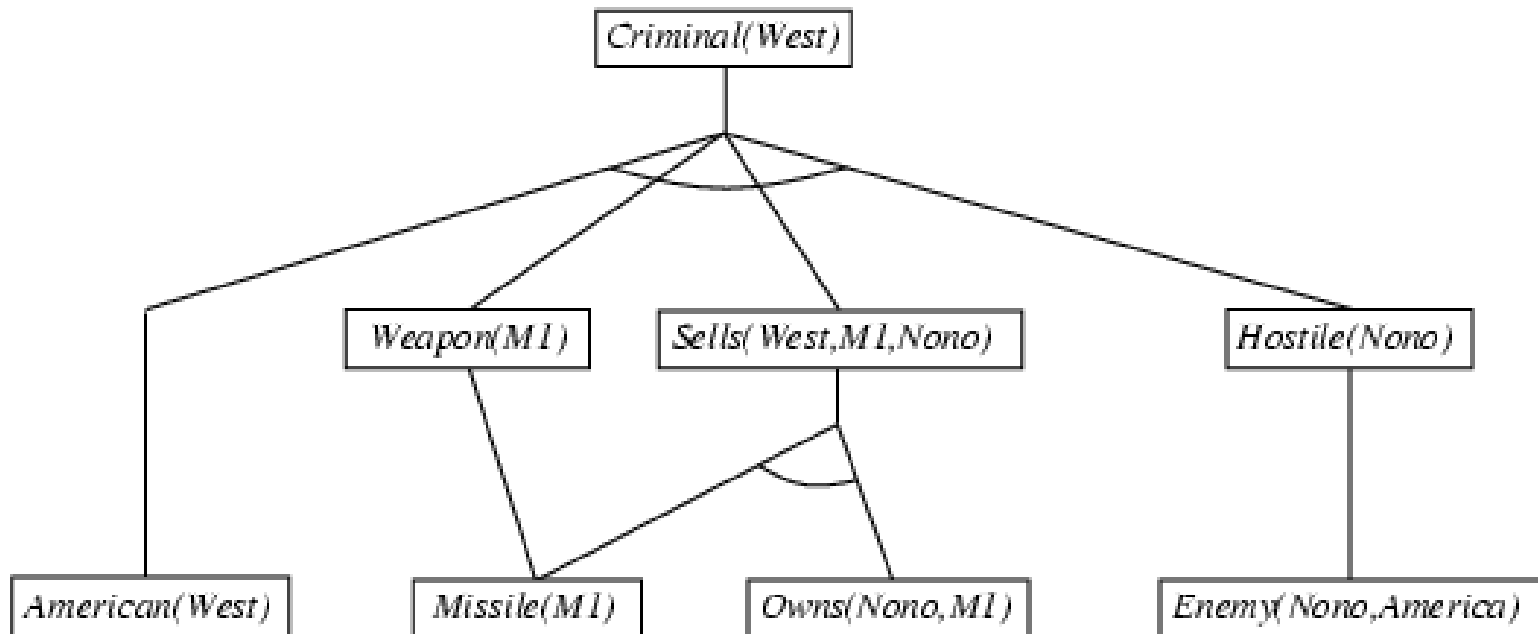
Forward chaining proof

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



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

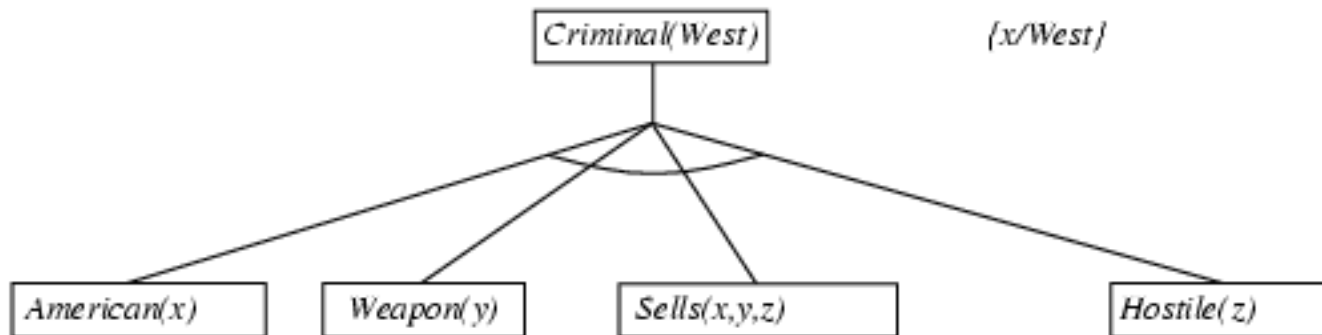
Backward chaining example

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

$Criminal(West)$

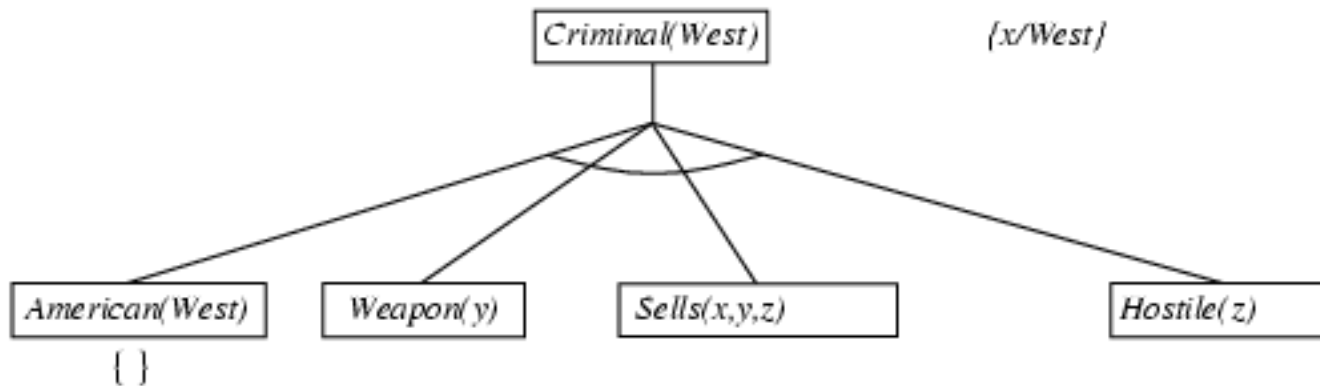
Backward chaining example

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



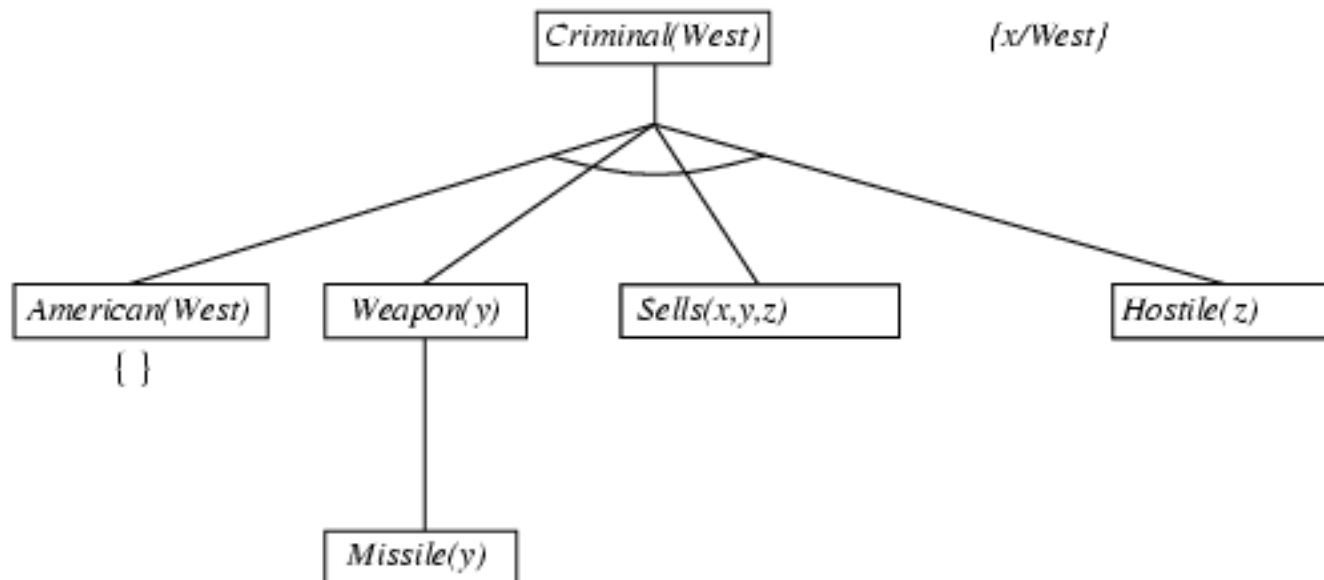
Backward chaining example

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



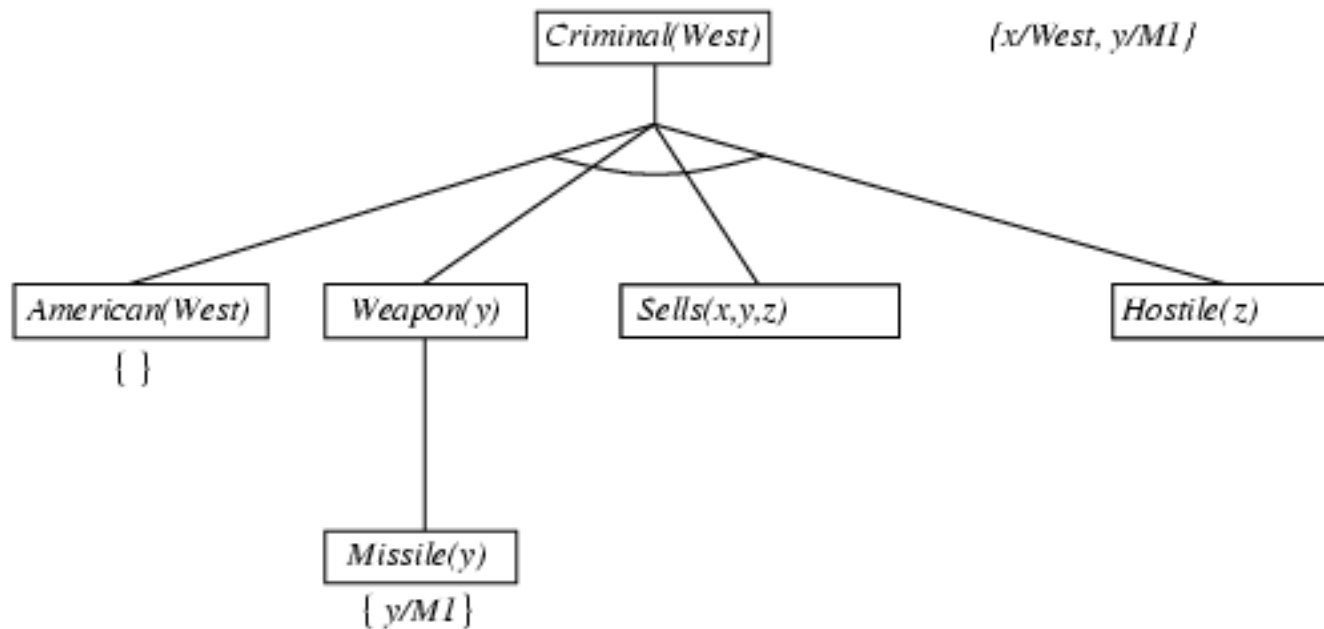
Backward chaining example

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



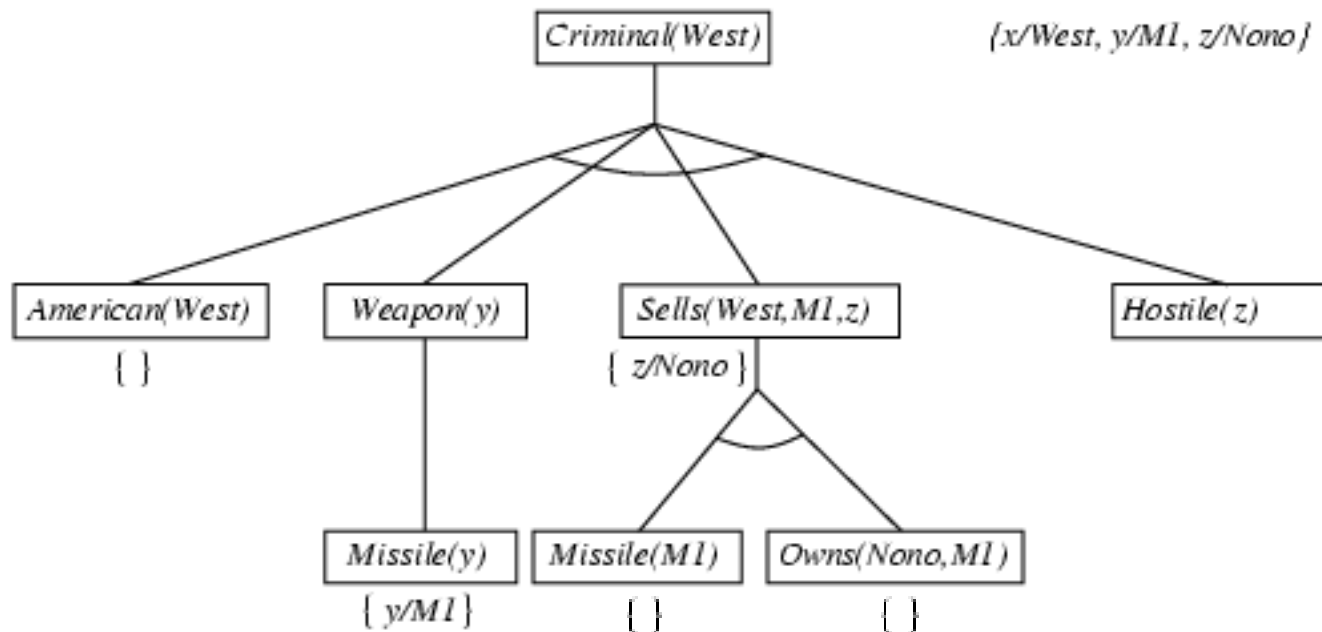
Backward chaining example

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



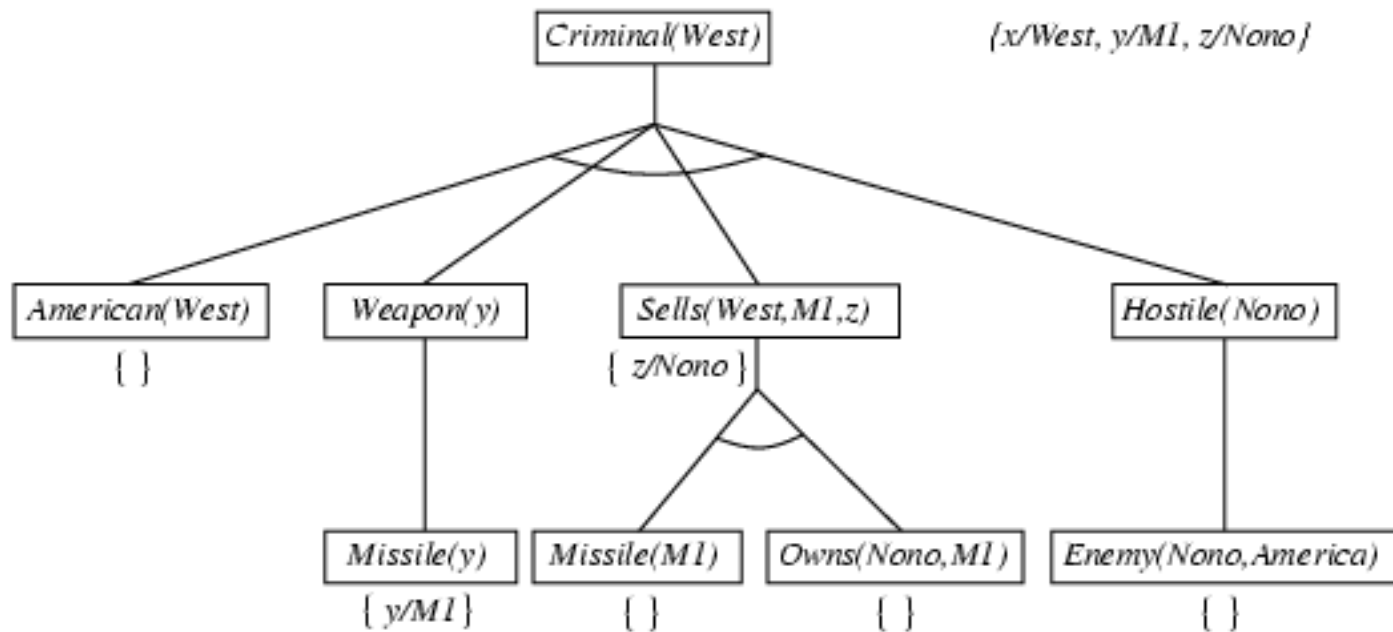
Backward chaining example

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



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., $X \text{ is } Y * Z + 3$
 - 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:

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

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

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

- Example:

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

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

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

Conversion to CNF

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

1. Eliminate biconditionals and implications

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

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

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

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

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

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

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

Conversion to CNF contd.

4. Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

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

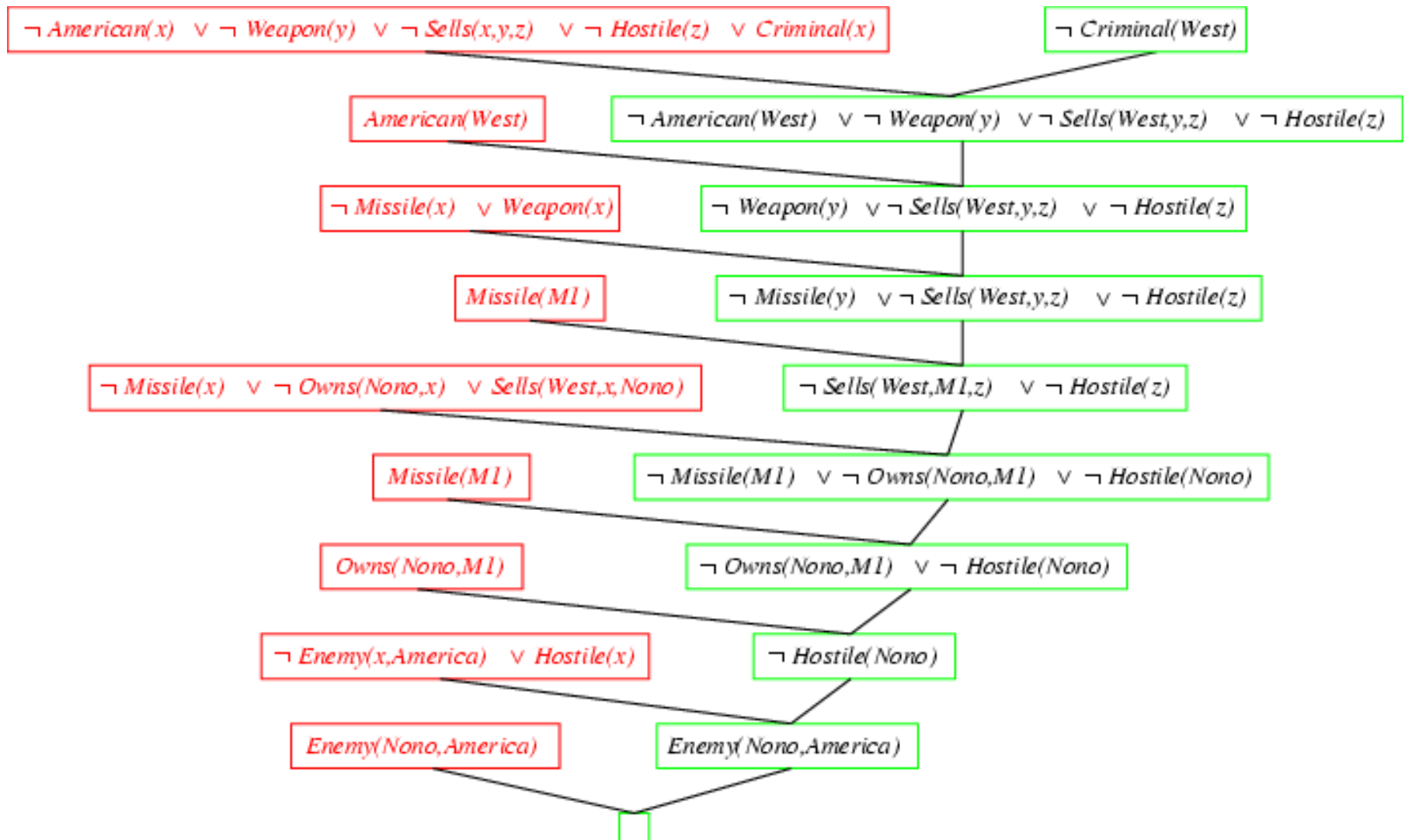
5. Drop universal quantifiers:

$$[\text{Animal}(F(x)) \wedge \neg \text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$$

6. Distribute \vee over \wedge :

$$[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)] \wedge [\neg \text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)]$$

Resolution proof: Horn clauses



Summary

- First-order logic:
 - objects and relations are semantic primitives
 - syntax: constants, functions, predicates, quantifiers
- Inference
 - Unification
 - Forward/backward chaining
 - Resolution
- Semantic web
 - Application of predicate logic to WWW