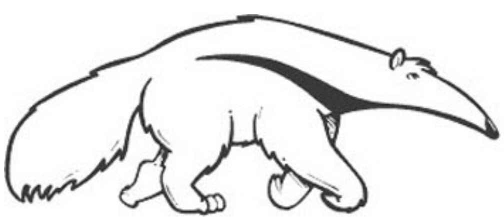
First Order Logic

CS171, Fall 2016

Introduction to Artificial Intelligence

Prof. Alexander Ihler



BREN:ICS INFORMATION AND COMPUTER SCIENCES

UNIVERSITY of CALIFORNIA IRVINE

Outline

- New ontology
  - objects, relations, properties, functions.
- New Syntax
  - Constants, predicates, properties, functions
- New semantics
  - meaning of new syntax
- Inference rules for Predicate Logic (FOL)
  - Resolution
  - Forward-chaining, Backward-chaining
  - Unification
- Reading: Russell and Norvig Chapters 8 & 9

Pros and cons of propositional logic

- Propositional logic is *declarative*: pieces of syntax correspond to facts
- 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)
- Propositional logic has very limited expressive power (unlike natural language)
  E.g., cannot say "pits cause breezes in adjacent squares"
  except by writing one sentence for each square

Building a more expressive language

Want to develop a better, more expressive language:

- Needs to refer to objects in the world,
- Needs to express general rules
  - On(x,y) → ~ clear(y)
  - All men are mortal
  - Everyone over age 21 can drink
  - One student in this class got a perfect score
  - Etc….
- First order logic, or "predicate calculus" allows more expressiveness

Logics in general

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0, 1]$
Fuzzy logic	degree of truth $\in [0, 1]$	known interval value

First-order logic

Whereas propositional logic assumes world contains *facts*,
first-order logic (like natural language) assumes the world contains

- Objects: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries . . .
- Relations: red, round, bogus, prime, multistoried . . .,
  brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, . . .
- Functions: father of, best friend, third inning of, one more than, beginning of . . .

Syntax of FOL: Basic elements

Constants	$KingJohn,\ 2,\ UCB, \ldots$
Predicates	$Brother,\ >, \ldots$
Functions	$Sqrt,\ LeftLegOf, \ldots$
Variables	$x,\ y,\ a,\ b, \ldots$
Connectives	$\wedge\ \vee\ \neg\ \Rightarrow\ \Leftrightarrow$
Equality	$=$
Quantifiers	$\forall\ \exists$

Atomic sentences

$$
\begin{aligned}
\text{Atomic sentence} &= predicate(term_1, \ldots, term_n) \\
&\quad \text{or } term_1 = term_2
\end{aligned}
$$

$$
\begin{aligned}
\text{Term} &= function(term_1, \ldots, term_n) \\
&\quad \text{or } constant \text{ or } variable
\end{aligned}
$$

E.g., $Brother(KingJohn, RichardTheLionheart)$
$> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))$

Complex sentences

Complex sentences are made from atomic sentences using connectives

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

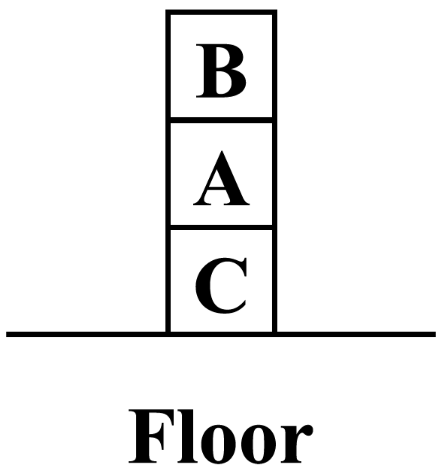
E.g. $Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)$
$>(1,2) \vee \leq(1,2)$
$>(1,2) \wedge \neg>(1,2)$

Semantics: Worlds

- The **world** consists of **objects** that have **properties**.
  - There are **relations** and **functions** between these objects
  - Objects in the world, individuals: people, houses, numbers, colors, baseball games, wars, centuries
    - Clock A, John, 7, the-house in the corner, Tel-Aviv
  - Functions on individuals:
    - father-of, best friend, third inning of, one more than
  - Relations:
    - brother-of, bigger than, inside, part-of, has color, occurred after
  - Properties (a relation of arity 1):
    - red, round, bogus, prime, multistoried, beautiful

Semantics: Interpretation

- An interpretation of a sentence (wff) is an assignment that maps
  - Object constants to objects in the worlds,
  - n-ary function symbols to n-ary functions in the world,
  - n-ary relation symbols to n-ary relations in the world
- Given an interpretation, an atom has the value "true" if it denotes a relation that holds for those individuals denoted in the terms. Otherwise it has the value "false"
  - Example: Block world:
    - A,B,C,floor, On, Clear
  - World:
  - On(A,B) is false, Clear(B) is true, On(C,F1) is true…


B
A
C
Floor

Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- Model contains objects (domain elements) and relations among them
- Interpretation specifies referents for
  constant symbols → objects

  predicate symbols → relations

  function symbols → functional relations

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

Semantics: Models

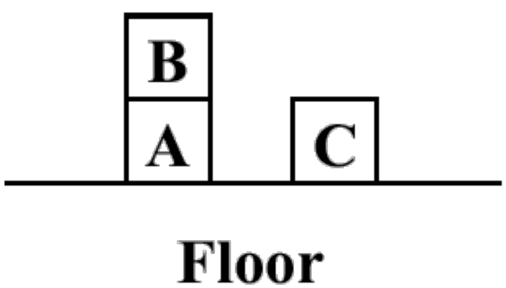
- An interpretation satisfies a wff (sentence) if the wff has the value "true" under the interpretation.
- Model: An interpretation that satisfies a wff is a model of that wff
- Validity: Any wff that has the value "true" under all interpretations is valid
- Any wff that does not have a model is inconsistent or unsatisfiable
- If a wff w has a value true under all the models of a set of sentences KB then KB logically entails w

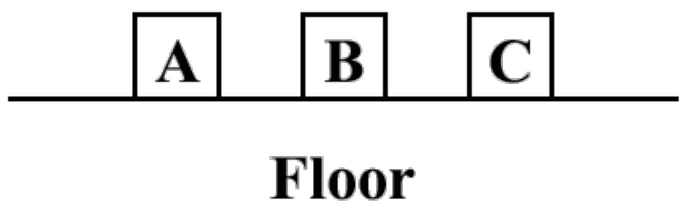
Example of models (blocks world)

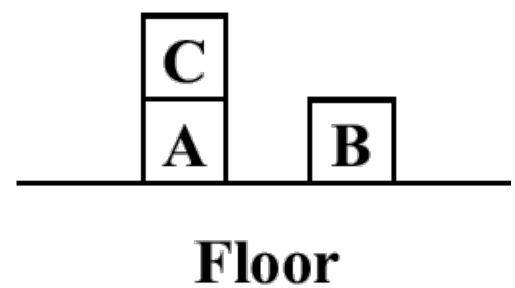
The formulas:

- On(A,F1) → Clear(B)
- Clear(B) and Clear(C) → On(A,F1)
- Clear(B) or Clear(A)
- Clear(B)
- Clear(C)

Possible interpretations that are models:

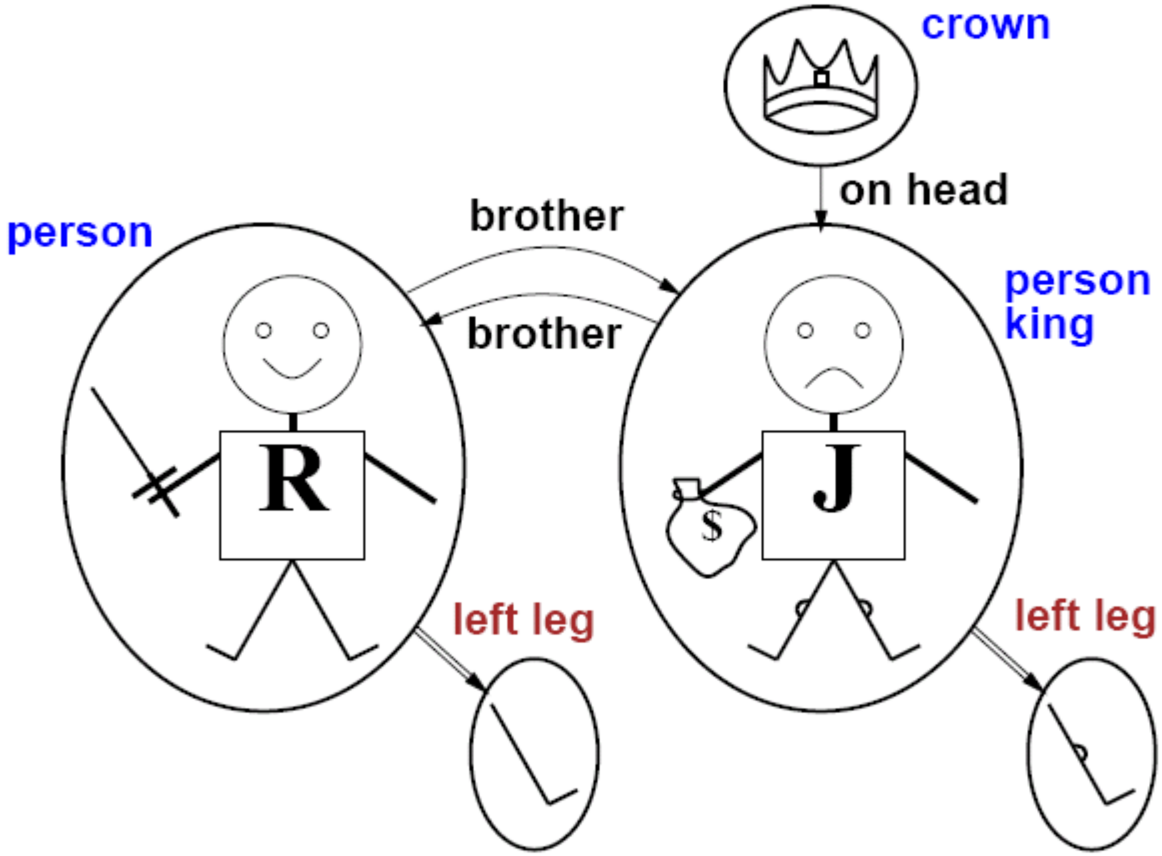






- On = {<B,A>,<A,floor>,<C,floor>}
- Clear = {<C>,<B>}

Models for FOL: Example



Quantification

- Universal and existential quantifiers allow expressing general rules with variables
- Universal quantification
  - All cats are mammals

$$\forall x\, Cat(x) \rightarrow Mammal(x)$$

  - It is equivalent to the conjunction of all the sentences obtained by substitution the name of an object for the variable x.
- Syntax: if w is a wff then (forall x) w is a wff.

$$Cat(Spot) \rightarrow Mammal(Spot) \wedge$$

$$Cat(Rebbeka) \rightarrow Mammal(Rebbeka) \wedge$$

$$Cat(Felix) \rightarrow Mammal(Felix) \wedge$$

$$,,,,$$

Quantification: Universal

- Universal quantification $\forall$:   a universally quantified sentence is true if it is true for every object in the model

    Everyone in Irvine has a tan:

- Roughly equivalent to conjunction:

A common mistake

- Typically, "implies" = "$\Rightarrow$" is the main connective operator with $\forall$
- Everyone in Irvine has a tan:

$$\forall x : \text{InIrvine}(x) \Rightarrow \text{Tan}(x)$$

- Operator $\wedge$ is uncommon

$$\forall x : \text{InIrvine}(x) \wedge \text{Tan}(x)$$

means that everyone lives in Irvine and is tan.

Quantification: Existential

- Existential quantification $\exists$: an existentially quantified sentence is true in case one of the disjunct is true

  Spot has a sister who is a cat:

  $$\exists x Sister(x, spot) \wedge Cat(x)$$

- Roughly quivalent to disjunction:

  $$Sister(Spot, Spot) \wedge Cat(Spot) \vee$$
  $$Sister(Rebecca, Spot) \wedge Cat(Rebecca) \vee$$
  $$Sister(Felix, Spot) \wedge Cat(Felix) \vee$$
  $$Sister(Richard, Spot) \wedge Cat(Richard) \ldots$$

- We can mix existential and universal quantification.

A common mistake

- Typically, "and" = "$\wedge$" is the main connective operator with $\exists$
- Spot has a sister who is a cat:

$$\exists\, x : Sister(x,Spot) \wedge Cat(x)$$

- Operator $\Rightarrow$ is uncommon

$$\exists\, x : Sister(x,Spot) \Rightarrow Cat(x)$$

is true if there is anyone who is not Spot's sister

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) $\qquad \neg\exists x\ \neg$Likes(x,IceCream)

  $\exists x$ Likes(x,Broccoli) $\qquad \neg\forall x\ \neg$Likes(x,Broccoli)

Fun with sentences

Brothers are siblings

Fun with sentences

Brothers are siblings

$$\forall\, x,y \;\; Brother(x,y) \;\Rightarrow\; Sibling(x,y).$$

"Sibling" is symmetric

Fun with sentences

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

Fun with sentences

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

$$\forall\, x, y\;\; Mother(x, y) \;\Leftrightarrow\; (Female(x) \land Parent(x, y)).$$

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

Equality

- term1 = term2 is true under a given interpretation if and only if term1 and term2 refer to the same object

- E.g., definition of Sibling in terms of Parent:

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

Using FOL

- The kinship domain:
  - object are people
  - Properties include gender and they are related by relations such as parenthood, brotherhood,marriage
  - predicates: Male, Female (unary) Parent,Sibling,Daughter,Son...
  - Function: Mother Father
- Brothers are siblings
  - $\forall x,y \; Brother(x,y) \Leftrightarrow Sibling(x,y)$
- One's mother is one's female parent
  - $\forall m,c \; Mother(c) = m \Leftrightarrow (Female(m) \wedge Parent(m,c))$
- “Sibling” is symmetric
  - $\forall x,y \; Sibling(x,y) \Leftrightarrow Sibling(y,x)$

Using FOL

- The set domain:
- $\forall s\ Set(s) \Leftrightarrow (s = \{\}\ ) \vee (\exists x,s2\ Set(s2) \wedge s = \{x|s2\})$
- $\neg\exists x,s\ \{x|s\} = \{\}$
- (Adjoining an element already in the set has no effect)
- $\forall x,s\ x \in s \Leftrightarrow s = \{x|s\}$
- (the only members of a set are the elements that were adjoint into it)
- $\forall x,s\ x \in s \Leftrightarrow [\ \exists y,s2\}\ (s = \{y|s2\} \wedge (x = y \vee x \in s2))]$
- $\forall s1,s2\ s1 \subseteq s2 \Leftrightarrow (\forall x\ x \in s1 \Rightarrow x \in s2)$
- $\forall s1,s2\ (s1 = s2) \Leftrightarrow (s1 \subseteq s2 \wedge s2 \subseteq s1)$
- $\forall x,s1,s2\ x \in (s1 \cap s2) \Leftrightarrow (x \in s1 \wedge x \in s2)$
- $\forall x,s1,s2\ x \in (s1 \cup s2) \Leftrightarrow (x \in s1 \vee x \in s2)$

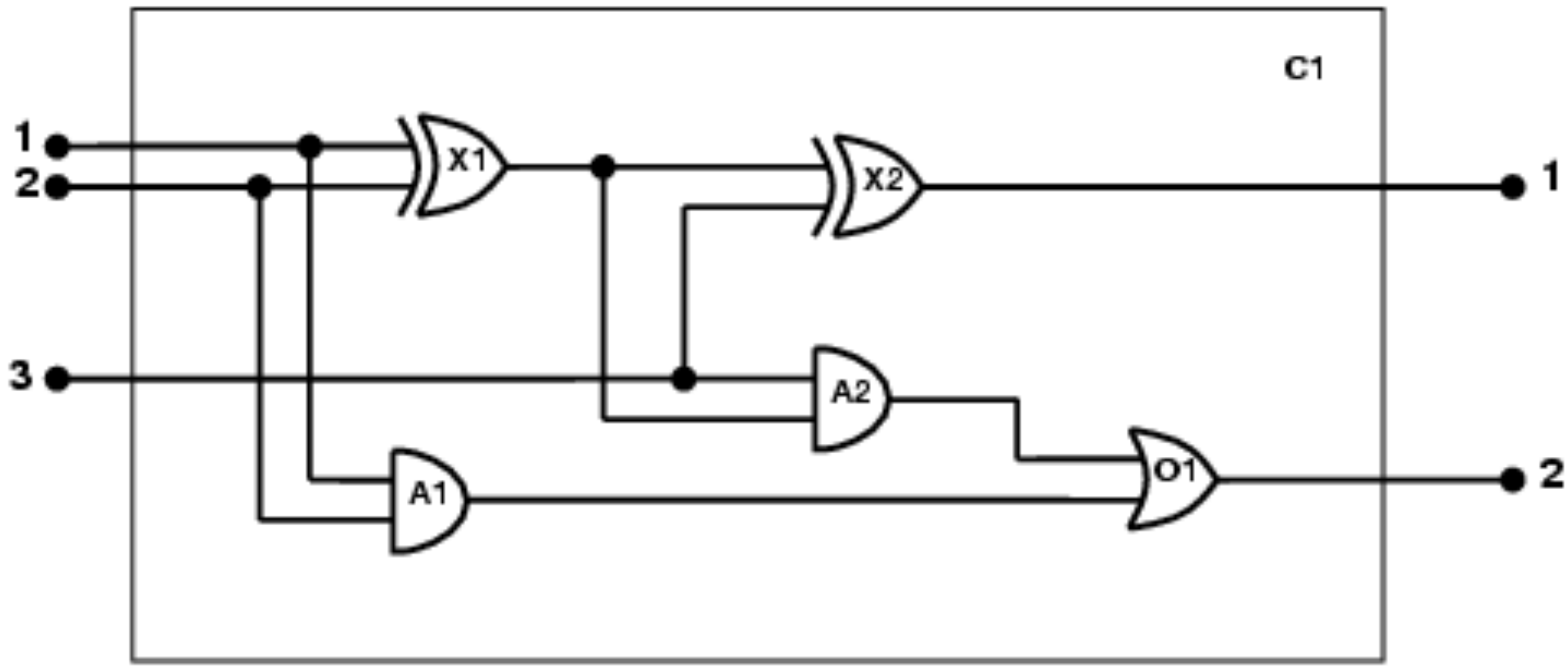
Objects are sets
Predicates: unary predicate “set:, binary predicate membership (x is a member of set), “subset” (s1 is a subset of s2)
Functions: intersections, union, adjoining an eleiment to a set.

Knowledge engineering in FOL

- Identify the task
- Assemble the relevant knowledge
- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain
- Encode a description of the specific problem instance
- Pose queries to the inference procedure and get answers
- Debug the knowledge base

The electronic circuits domain

One-bit full adder



The electronic circuits domain

- Identify the task
  - Does the circuit actually add properly? (circuit verification)
- Assemble the relevant knowledge
  - Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
  - Irrelevant: size, shape, color, cost of gates
- Decide on a vocabulary
  - Alternatives:
    - Type(X1) = XOR
    - Type(X1, XOR)
    - XOR(X1)

The electronic circuits domain

- Encode general knowledge of the domain
  - $\forall t1,t2\ Connected(t1, t2) \Rightarrow Signal(t1) = Signal(t2)$
  - $\forall t\ Signal(t) = 1 \lor Signal(t) = 0$
  - $1 \neq 0$
  - $\forall t1,t2\ Connected(t1, t2) \Rightarrow Connected(t2, t1)$
  - $\forall g\ Type(g) = OR \Rightarrow Signal(Out(1,g)) = 1 \Leftrightarrow \exists n\ Signal(In(n,g)) = 1$
  - $\forall g\ Type(g) = AND \Rightarrow Signal(Out(1,g)) = 0 \Leftrightarrow \exists n\ Signal(In(n,g)) = 0$
  - $\forall g\ Type(g) = XOR \Rightarrow Signal(Out(1,g)) = 1 \Leftrightarrow Signal(In(1,g)) \neq Signal(In(2,g))$
  - $\forall g\ Type(g) = NOT \Rightarrow Signal(Out(1,g)) \neq Signal(In(1,g))$

The electronic circuits domain

- Encode the specific problem instance
  - Type(X1) = XOR Type(X2) = XOR
  - Type(A1) = AND Type(A2) = AND
  - Type(O1) = OR

  - Connected(Out(1,X1),In(1,X2)) Connected(In(1,C1),In(1,X1))
  - Connected(Out(1,X1),In(2,A2)) Connected(In(1,C1),In(1,A1))
  - Connected(Out(1,A2),In(1,O1)) Connected(In(2,C1),In(2,X1))
  - Connected(Out(1,A1),In(2,O1)) Connected(In(2,C1),In(2,A1))
  - Connected(Out(1,X2),Out(1,C1)) Connected(In(3,C1),In(2,X2))
  - Connected(Out(1,O1),Out(2,C1)) Connected(In(3,C1),In(1,A2))

The electronic circuits domain

6. Pose queries to the inference procedure

   What are the possible sets of values of all the terminals for the adder circuit?

$\exists i_1,i_2,i_3,o_1,o_2$ Signal(In(1,C_1)) = $i_1 \land$ Signal(In(2,C_1)) = $i_2 \land$ Signal(In(3,C_1)) = $i_3 \land$ Signal(Out(1,C_1)) = $o_1 \land$ Signal(Out(2,C_1)) = o_2

7. Debug the knowledge base

   (May have omitted assertions like $1 \neq 0$)

Interacting with FOL KBs

Suppose a wumpus-world agent is using an FOL KB
and perceives a smell and a breeze (but no glitter) at $t = 5$:

$Tell(KB, Percept([Smell, Breeze, None], 5))$
$Ask(KB, \exists\, a \;\; Action(a, 5))$

I.e., does the KB entail any particular actions at $t = 5$?

Answer: $Yes,\ \{a/Shoot\}$ $\quad\leftarrow$ substitution (binding list)

Given a sentence S and a substitution σ,
$S\sigma$ denotes the result of plugging σ into S; e.g.,
$S = Smarter(x, y)$
$\sigma = \{x/Hillary, y/Bill\}$
$S\sigma = Smarter(Hillary, Bill)$

$Ask(KB, S)$ returns some/all σ such that $KB \models S\sigma$

Knowledge base for the wumpus world

"Perception"

$\forall\, b, g, t \;\; Percept([Smell, b, g], t) \;\Rightarrow\; Smelt(t)$

$\forall\, s, b, t \;\; Percept([s, b, Glitter], t) \;\Rightarrow\; AtGold(t)$

Reflex: $\forall\, t \;\; AtGold(t) \;\Rightarrow\; Action(Grab, t)$

Reflex with internal state: do we have the gold already?

$\forall\, t \;\; AtGold(t) \wedge \neg Holding(Gold, t) \;\Rightarrow\; Action(Grab, t)$

$Holding(Gold, t)$ cannot be observed

$\Rightarrow$ keeping track of change is essential

Deducing hidden properties

Properties of locations:
$\forall x, t \;\; At(Agent, x, t) \land Smelt(t) \;\Rightarrow\; Smelly(x)$
$\forall x, t \;\; At(Agent, x, t) \land Breeze(t) \;\Rightarrow\; Breezy(x)$

Squares are breezy near a pit:

Diagnostic rule—infer cause from effect
$$\forall y \;\; Breezy(y) \;\Rightarrow\; \exists x \;\; Pit(x) \land Adjacent(x, y)$$

Causal rule—infer effect from cause
$$\forall x, y \;\; Pit(x) \land Adjacent(x, y) \;\Rightarrow\; Breezy(y)$$

Neither of these is complete—e.g., the causal rule doesn't say whether squares far away from pits can be breezy

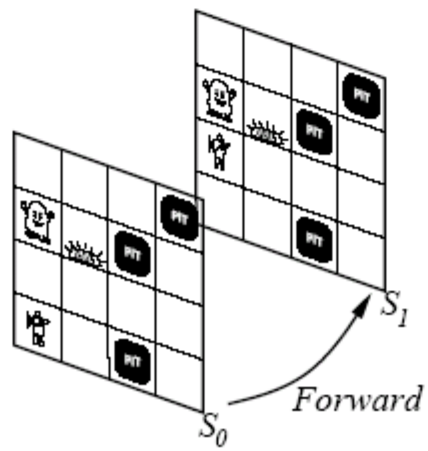
Definition for the $Breezy$ predicate:
$$\forall y \;\; Breezy(y) \;\Leftrightarrow\; [\exists x \;\; Pit(x) \land Adjacent(x, y)]$$

Keeping track of change

Facts hold in situations, rather than eternally
E.g., $Holding(Gold, Now)$ rather than just $Holding(Gold)$

Situation calculus is one way to represent change in FOL:
Adds a situation argument to each non-eternal predicate
E.g., Now in $Holding(Gold, Now)$ denotes a situation

Situations are connected by the $Result$ function
$Result(a, s)$ is the situation that results from doing a in s



Describing actions I

"Effect" axiom—describe changes due to action
$\forall s \;\; AtGold(s) \;\Rightarrow\; Holding(Gold, Result(Grab, s))$

"Frame" axiom—describe non-changes due to action
$\forall s \;\; HaveArrow(s) \;\Rightarrow\; HaveArrow(Result(Grab, s))$

Frame problem: find an elegant way to handle non-change
(a) representation—avoid frame axioms
(b) inference—avoid repeated "copy-overs" to keep track of state

Qualification problem: true descriptions of real actions require endless caveats—what if gold is slippery or nailed down or . . .

Ramification problem: real actions have many secondary consequences—what about the dust on the gold, wear and tear on gloves, . . .

Describing actions II

Successor-state axioms solve the representational frame problem

Each axiom is "about" a predicate (not an action per se):

$$\begin{array}{lcl} \text{P true afterwards} & \Leftrightarrow & [\text{an action made P true} \\ & \vee & \text{P true already and no action made P false}] \end{array}$$

For holding the gold:

$$\forall a, s \;\; Holding(Gold, Result(a,s)) \Leftrightarrow$$
$$[(a = Grab \wedge AtGold(s))$$
$$\vee (Holding(Gold, s) \wedge a \neq Release)]$$

Some more notation

- Instantiation: specify values for variables

- Ground term
  - A term without variables
- Substitution
  - Setting a variable equal to something
  - θ = {x / John, y / Richard}
  - Read as "x := John, y:=Richard"
- Write a subsitution into sentence α as

  Subst(θ, α)   or just as $\alpha\theta$

Universal instantiation (UI)

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

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

  for any variable *v* and ground term *g*

- 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:

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

- E.g., $\exists x\, Crown(x) \land OnHead(x, John)$ yields:

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

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

Reduction to propositional inference

Suppose the KB contains just the following:

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
King(John)
Greedy(John)
Brother(Richard,John)

- **Instantiating the universal sentence in all possible ways, we have:**

  King(John) ∧ Greedy(John) ⇒ Evil(John)
  King(Richard) ∧ Greedy(Richard) ⇒ 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 contd.

- Every FOL KB can be propositionalized so as to preserve entailment
- (A ground sentence is entailed by new KB iff entailed by original KB)
- Idea: 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

create a propositional KB by instantiating with depth-n terms

see if α is entailed by this KB

Problem: works if α is entailed, loops if α is not entailed

Theorem: Turing (1936), Church (1936) Entailment for FOL is semidecidable (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.)

Problems with propositionalization

- Propositionalization seems to generate lots of irrelevant sentences.

- E.g., from:

  $\forall$x King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
  King(John)
  $\forall$y Greedy(y)
  Brother(Richard,John)

- Given query "evil(x) it seems obvious that *Evil*(*John*), but propositionalization produces lots of facts such as *Greedy*(*Richard*) that are irrelevant

- With *p* *k*-ary predicates and *n* constants, there are $p \cdot n^k$ instantiations.

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*

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

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 *I*

- 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. $p_1', \backslash; \ldots, \backslash;p_n' \models p_1' \wedge \ldots \wedge p_n' \models p_1'\theta \wedge \ldots \wedge p_n'\theta$
3. From 1 and 2, $q\theta$ follows by ordinary Modus Ponens

Unification

- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

- 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)	

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

- 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)	

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

- 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)	

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

- 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)	

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification

- We can get the inference immediately if we can find a substitution θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

- 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}

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},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 }

The 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

The unification algorithm

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

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 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 …

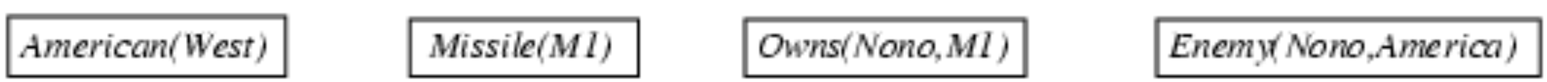
$Enemy(Nono,America)$

Forward chaining algorithm

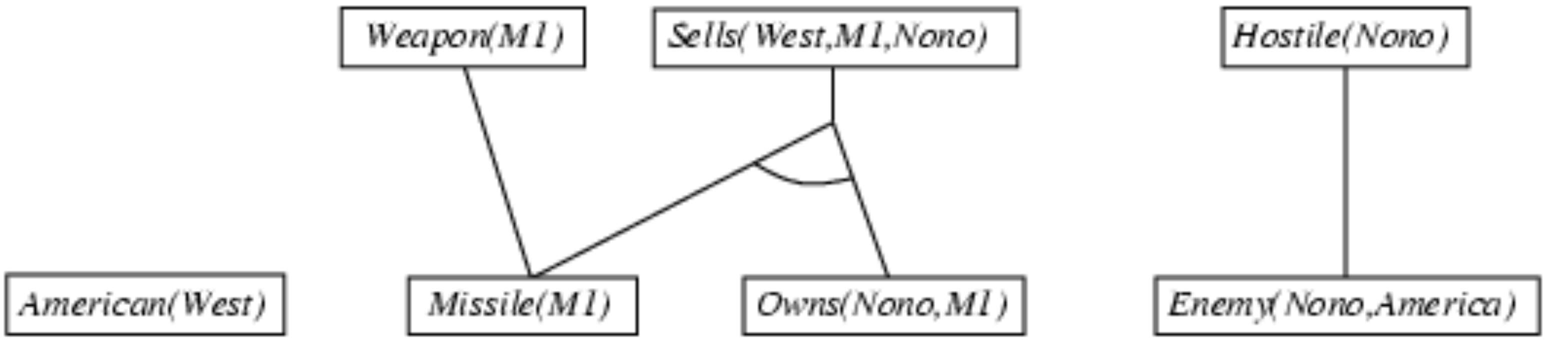
function FOL-FC-ASK(KB, α) **returns** a substitution or *false*

repeat until *new* is empty
    $new \leftarrow \{ \}$
    for each sentence r **in** KB **do**
        $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART(r)
        for each θ such that $(p_1 \wedge \ldots \wedge p_n)\theta = (p'_1 \wedge \ldots \wedge p'_n)\theta$
                for some $p'_1, \ldots, p'_n$ in KB
            $q' \leftarrow$ SUBST(θ, q)
            if q' is not a renaming of a sentence already in KB or new **then do**
                add q' to new
                $\phi \leftarrow$ UNIFY(q', α)
                if ϕ is not *fail* **then return** ϕ
    add new to KB
return *false*

Forward chaining proof



Forward chaining proof

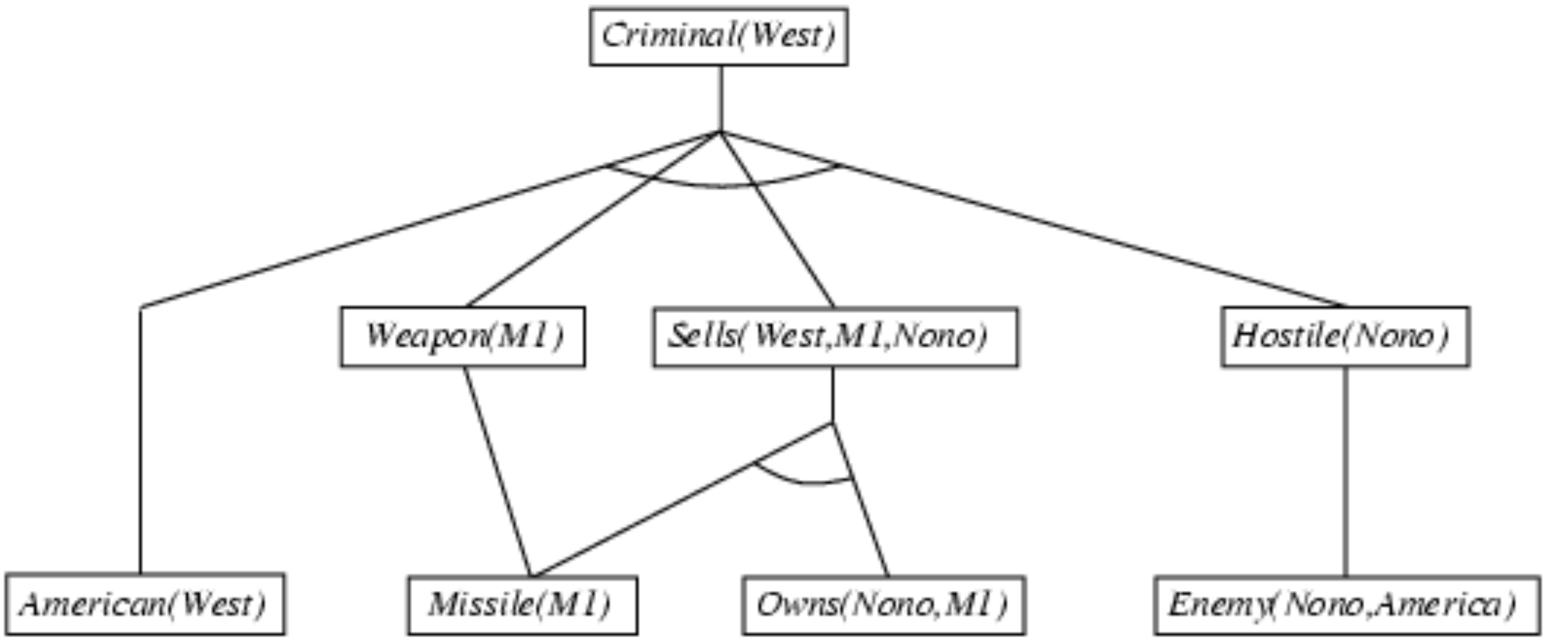


Enemy(x,America) $\Rightarrow$ *Hostile(x)*

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

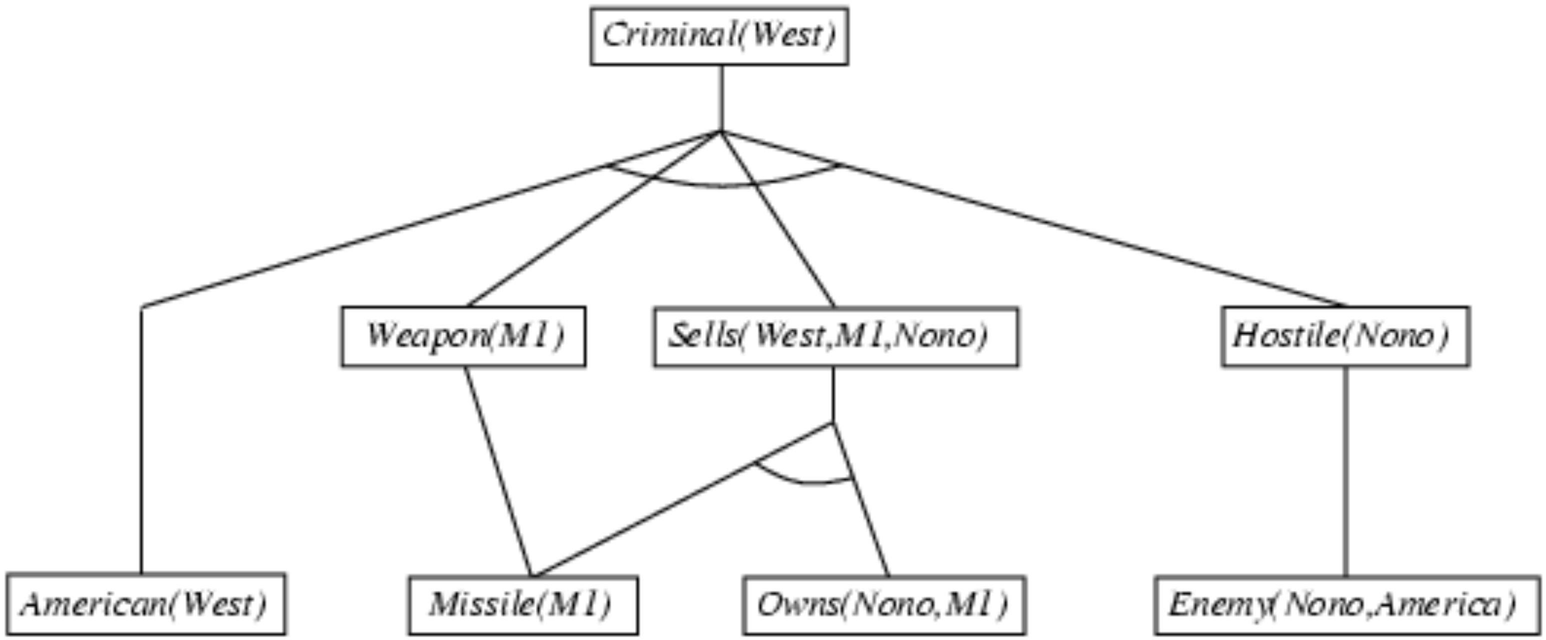
Missile(x) $\Rightarrow$ *Weapon(x)*

Forward chaining proof



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

Forward chaining proof



*American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)
*Owns(Nono,M1) and Missile(M1)
*Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)
*Missile(x) ⇒ Weapon(x)
*Enemy(x,America) ⇒ Hostile(x)
*American(West)
*Enemy(Nono,America)

Properties of forward chaining

- **Sound and complete for first-order definite clauses**
- **Datalog = first-order definite clauses + no functions**
- **FC terminates for Datalog in finite number of iterations**
- **May not terminate in general if α is not entailed**
- **This is unavoidable: entailment with definite clauses is semidecidable**

Efficiency of forward chaining

Incremental forward chaining: no need to match a rule on iteration *k* if a premise wasn't added on iteration *k-1*

$\Rightarrow$ match each rule whose premise contains a newly added positive literal

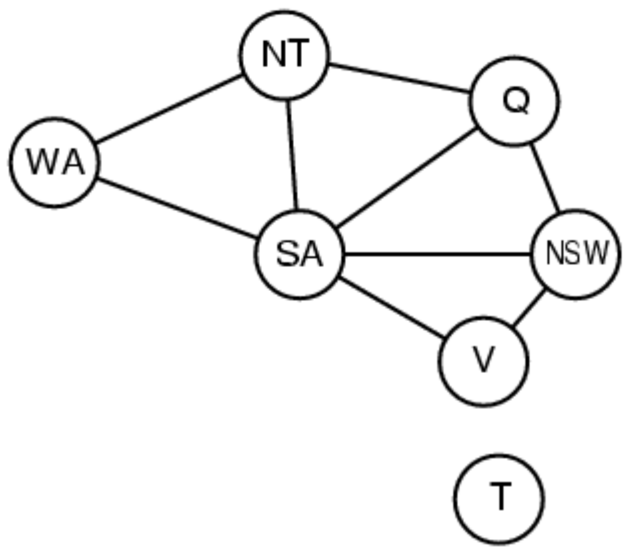
Matching itself can be expensive:

Database indexing allows O(1) retrieval of known facts

– e.g., query *Missile(x)* retrieves *Missile*(M_1)

Forward chaining is widely used in deductive databases

Hard matching example



Diff(wa,nt) ∧ *Diff(wa,sa)* ∧ *Diff(nt,q)* ∧
Diff(nt,sa) ∧ *Diff(q,nsw)* ∧ *Diff(q,sa)* ∧
Diff(nsw,v) ∧ *Diff(nsw,sa)* ∧ *Diff(v,sa)* ⇒
Colorable()

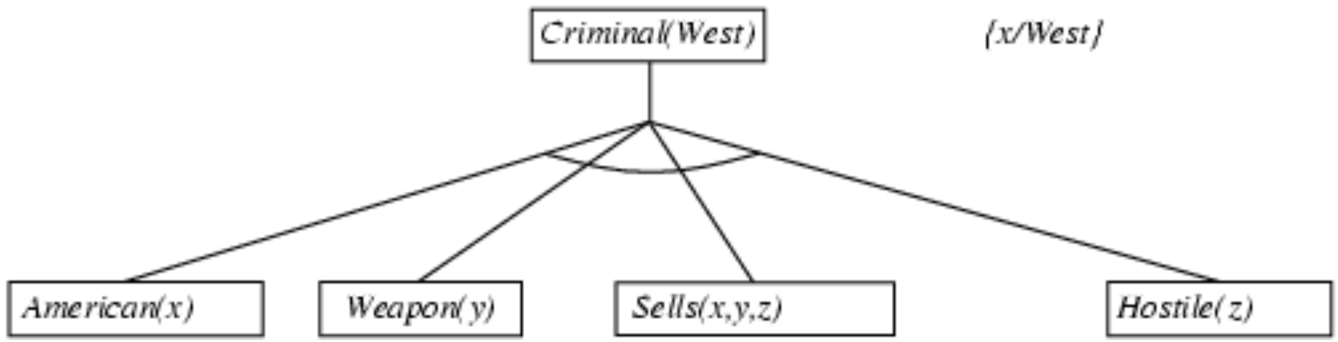
Diff(Red,Blue) *Diff (Red,Green)*
Diff(Green,Red) *Diff(Green,Blue)*
Diff(Blue,Red) *Diff(Blue,Green)*

- ***Colorable*() is inferred iff the CSP has a solution**
- **CSPs include 3SAT as a special case, hence matching is NP-hard**

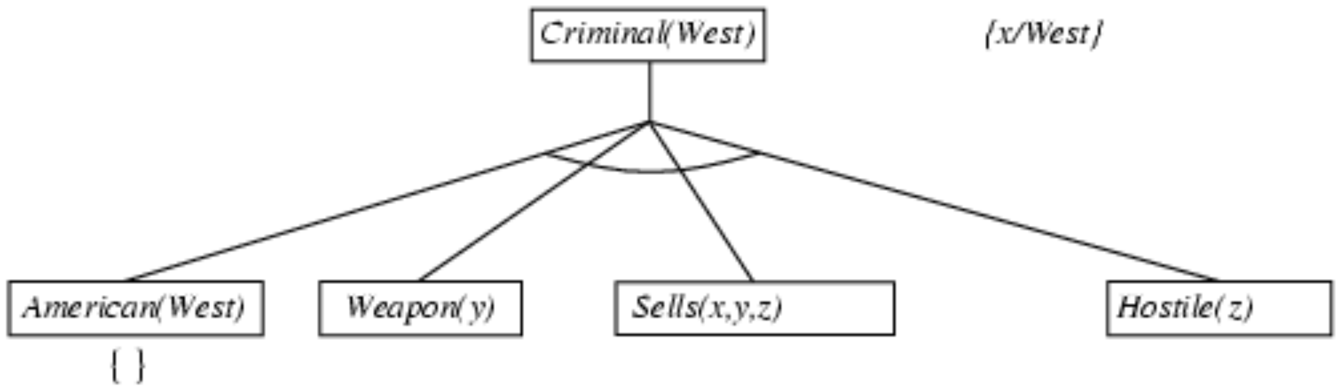
Backward chaining example

$\boxed{Criminal(West)}$

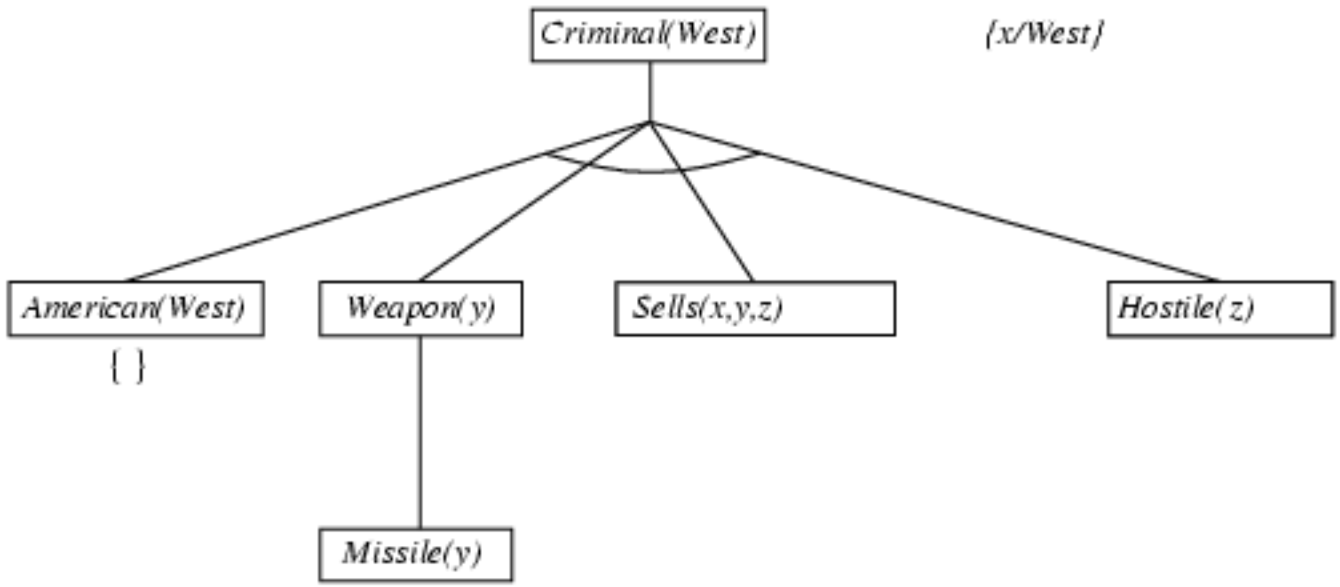
Backward chaining example



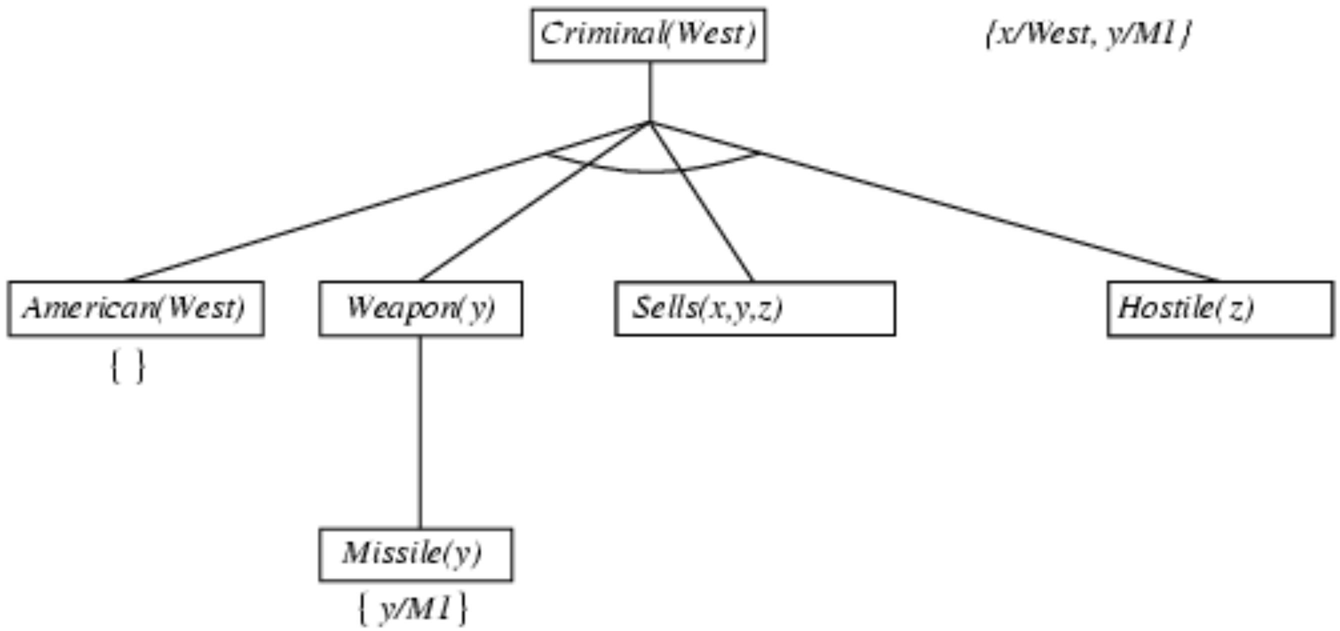
Backward chaining example



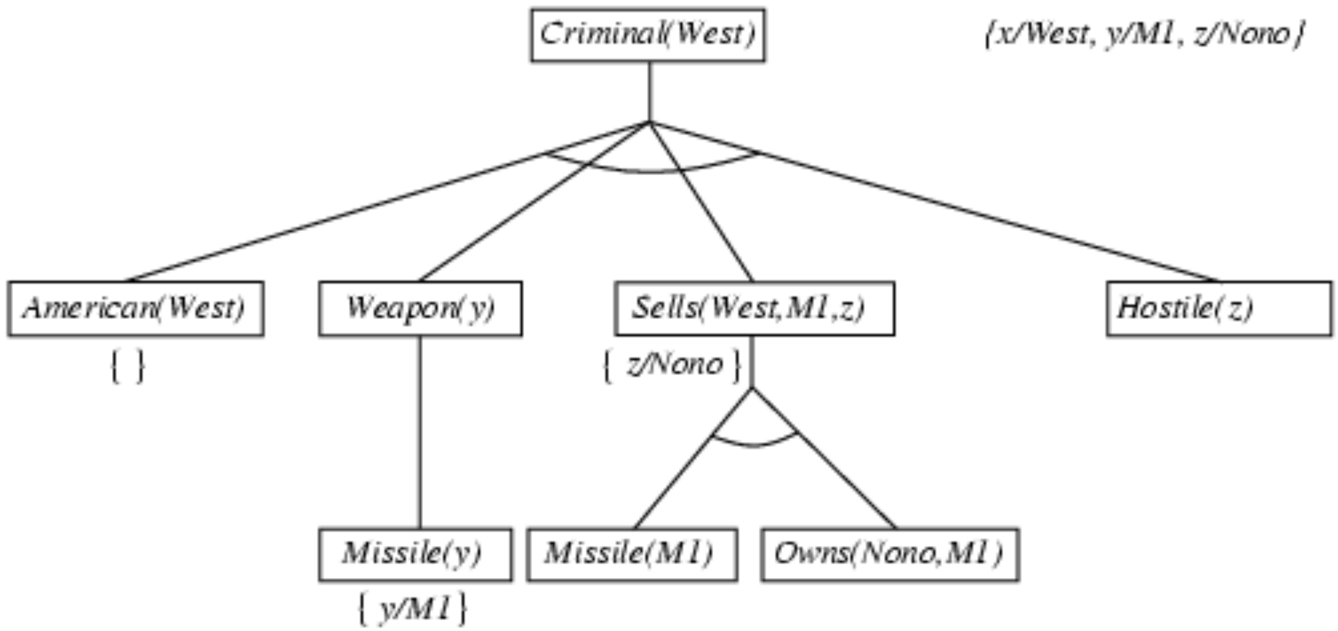
Backward chaining example



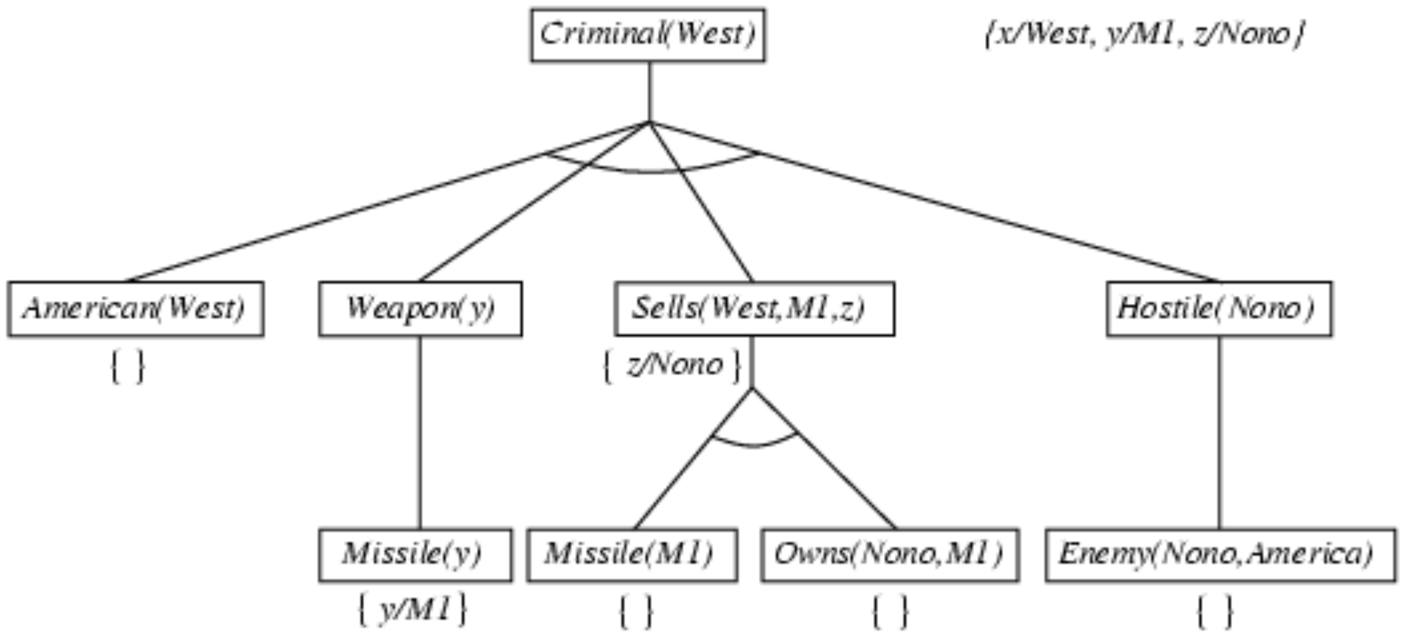
Backward chaining example



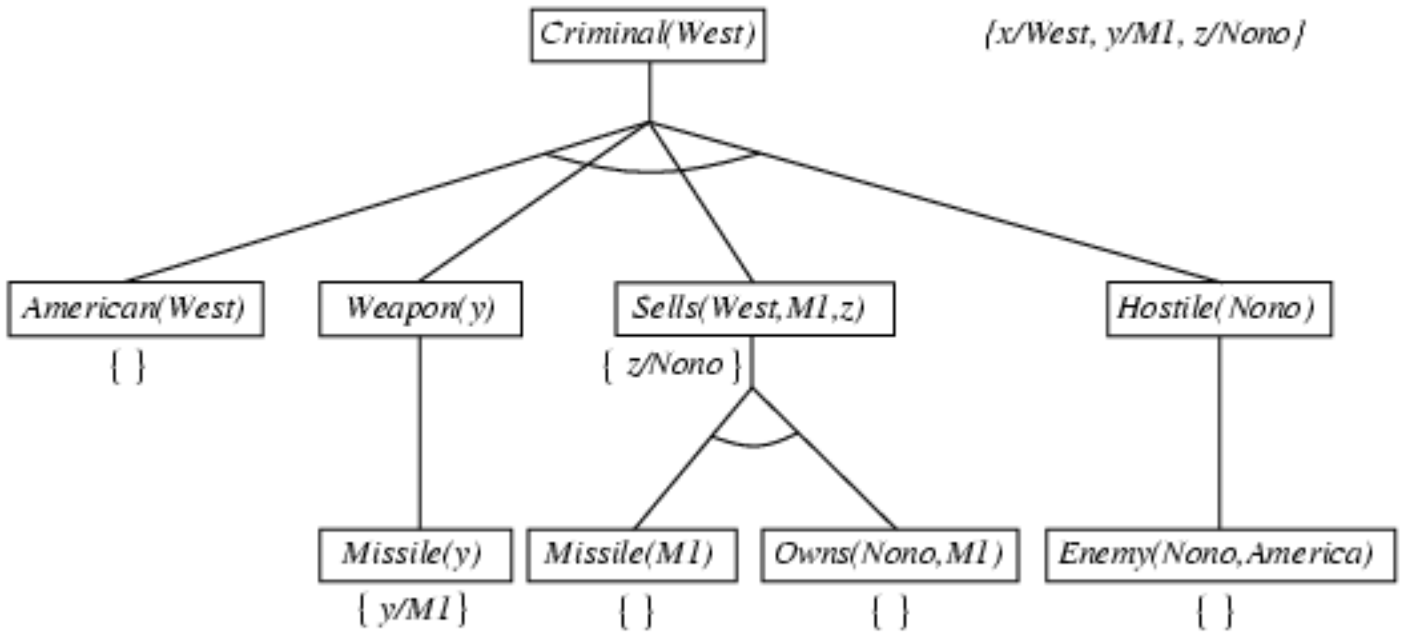
Backward chaining example



Backward chaining example



Backward chaining example



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

$\text{SUBST}(\text{COMPOSE}(\theta_1, \theta_2), p) = \text{SUBST}(\theta_2, \text{SUBST}(\theta_1, p))$

Properties of backward chaining

- Depth-first recursive proof search: space is linear in size of proof
- Incomplete due to infinite loops
  - $\Rightarrow$ fix by checking current goal against every goal on stack
- Inefficient due to repeated subgoals (both success and failure)
  - $\Rightarrow$ 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 + bells & whistles
  Widely used in Europe, Japan (basis of 5th Generation project)
  Compilation techniques $\Rightarrow$ 60 million LIPS

- Program = 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., given `alive(X) :- not dead(X).`
  - `alive(joe)` succeeds if `dead(joe)` fails

Prolog

- 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: brief summary

- 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 `Unify`(ℓ_i, $\neg m_j$) = θ.

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

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

  with θ = {x/Ken}

- Apply resolution steps to CNF(KB $\wedge$ $\neg\alpha$); complete for FOL

Conversion to CNF

- Everyone who loves all animals is loved by someone:
  $\forall x\ [\forall y\ \mathit{Animal}(y) \Rightarrow \mathit{Loves}(x,y)] \Rightarrow [\exists y\ \mathit{Loves}(y,x)]$

1. Eliminate biconditionals and implications
   $\forall x\ [\neg \forall y\ \neg \mathit{Animal}(y) \lor \mathit{Loves}(x,y)] \lor [\exists y\ \mathit{Loves}(y,x)]$

2. Move $\neg$ inwards: $\neg \forall x\ p \equiv \exists x\ \neg p$, $\neg \exists x\ p \equiv \forall x\ \neg p$
   $\forall x\ [\exists y\ \neg(\neg \mathit{Animal}(y) \lor \mathit{Loves}(x,y))] \lor [\exists y\ \mathit{Loves}(y,x)]$
   $\forall x\ [\exists y\ \neg\neg \mathit{Animal}(y) \land \neg \mathit{Loves}(x,y)] \lor [\exists y\ \mathit{Loves}(y,x)]$
   $\forall x\ [\exists y\ \mathit{Animal}(y) \land \neg \mathit{Loves}(x,y)] \lor [\exists y\ \mathit{Loves}(y,x)]$

Conversion to CNF contd.

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

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

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. Drop universal quantifiers:

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

6. Distribute $\vee$ over $\wedge$:

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

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“:

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

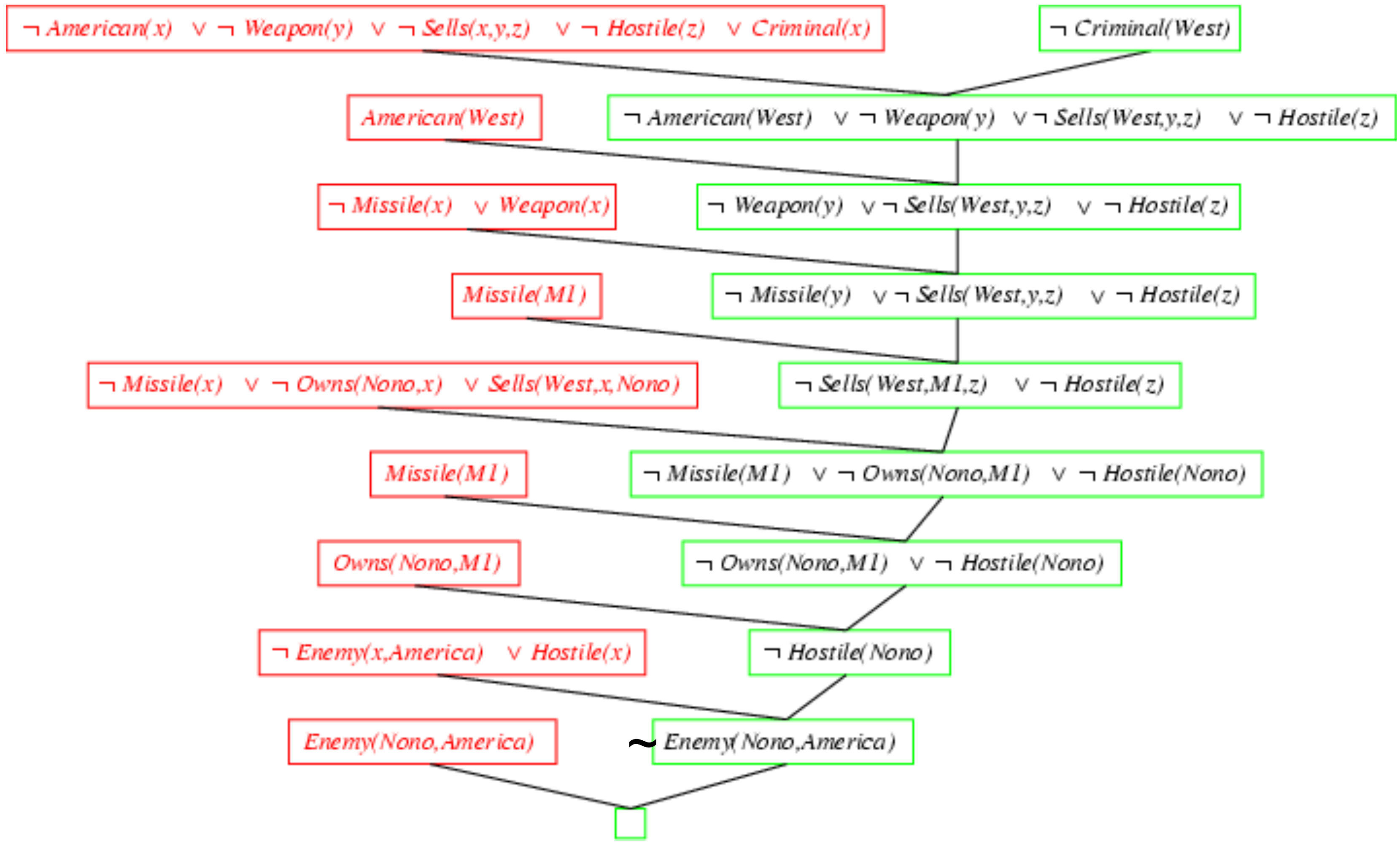
West, who is American …

$American(West)$

The country Nono, an enemy of America …

$Enemy(Nono,America)$

Resolution proof: definite clauses



Converting to clause form

$$\forall x, y\ P(x) \wedge P(y) \wedge I(x,27) \wedge I(y,28) \rightarrow S(x,y)$$

$$P(A), P(B)$$

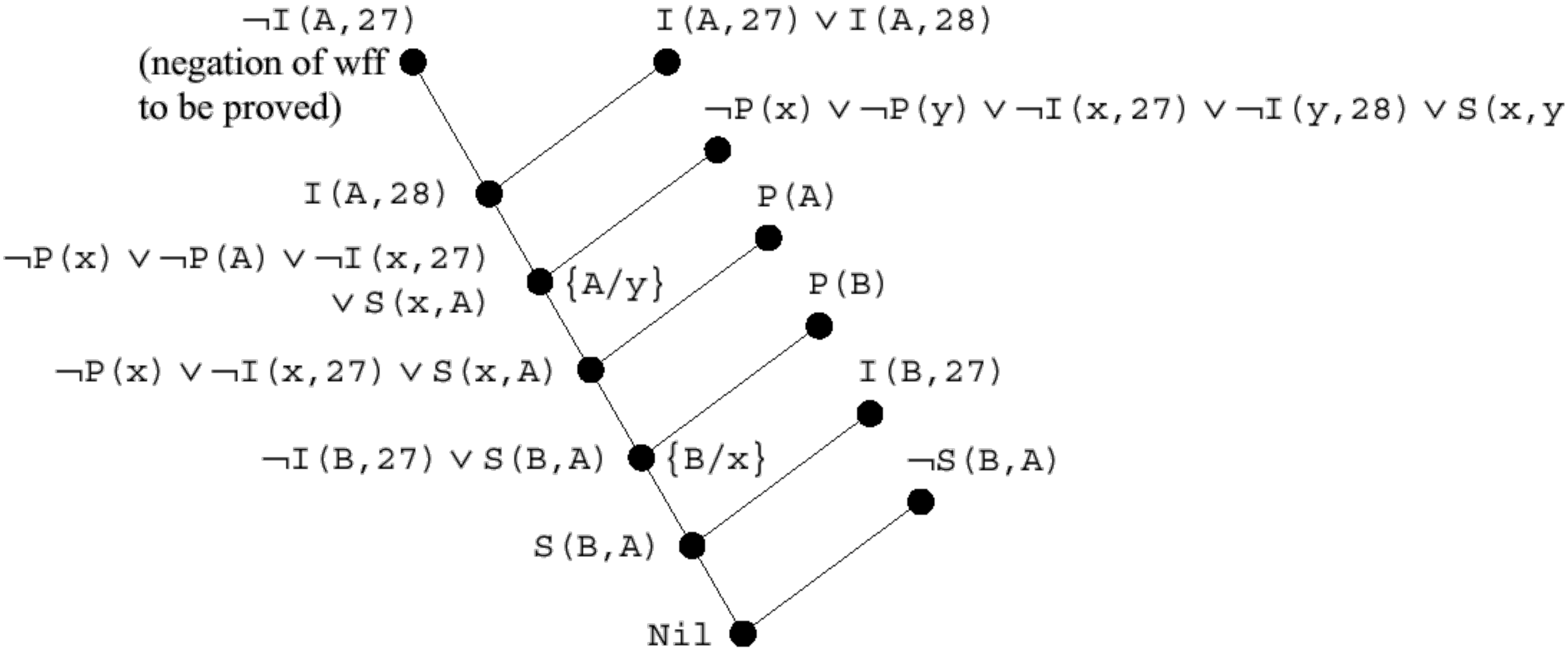
$$I(A,27) \vee I(A,28)$$

$$I(B,27)$$

$$\neg S(B,A)$$

Prove I(A,27)

Example: Resolution Refutation Prove *I(A,27)*



Example: Answer Extraction