Inference in First Order Logic

Fundamentals of Artificial Intelligence

Slides are mostly adapted from AIMA and MIT Open Courseware,

Milos Hauskrecht (U. Pittsburgh)

and Max Welling (UC Irvine)

Logical Inference

Logical inference problem:

- Given a knowledge base KB (a set of sentences) and a sentence α , does the KB semantically entail α ?

$$KB \models \alpha \quad ?$$

In other words: In all interpretations in which sentences in the KB are true, is also α true?

Inference in Propositional Logic

Computational procedures that answer:

$$KB \models \alpha \ ?$$

Three approaches:

- **Truth-table approach**
- **Inference rules**
- **Conversion to the inverse SAT problem**
  - **Resolution-refutation**

Inference in FOL : Truth Table Approach

- Is the Truth-table approach a viable approach for the FOL?
  ?
- **NO!**
- Why?
- It would require us to enumerate and list all possible interpretations **I**
- I = (assignments of symbols to objects, predicates to relations and functions to relational mappings)
- Simply there are too many interpretations

Inference Rules

- **Inference rules from the propositional logic:**
  - Modus ponens

$$\frac{A \Rightarrow B, \quad A}{B}$$

  - Resolution

$$\frac{A \vee B, \quad \neg B \vee C}{A \vee C}$$

  - and others: And-introduction, And-elimination, Or-introduction, Negation elimination
- **Additional inference rules are needed for sentences with quantifiers and variables**
  - **Rules must involve variable substitutions**

Sentences with variables

First-order logic sentences can include variables.

- **Variable** is:
  - **Bound** – if it is in the scope of some quantifier

$$\forall x \; P(x)$$

  - **Free** – if it is not bound.

$$\exists x \; P(y) \wedge Q(x) \qquad y \text{ is free}$$

Examples:

$$\forall x \, \exists y \; Likes\,(x, y)$$

- Bound

$$\forall x \left(Likes\,(x, y) \wedge \exists y \; Likes\,(y, Raymond\,) \right)$$

- Free

Sentences with variables

First-order logic sentences can include variables.

- **Sentence** (formula) is:
  - **Closed** – if it has no free variables

    $$\forall y \exists x\ P(y) \Rightarrow Q(x)$$

  - **Open** – if it is not closed

    $$\exists x\ P(y) \wedge Q(x) \qquad y \text{ is free}$$

  - **Ground** – if it does not have any variables

    $$Likes(John, Jane)$$

Variable Substitutions

- Variables in the sentences can be substituted with terms. (terms = constants, variables, functions)
- **Substitution:**
  - Is represented by a mapping from variables to terms

$$\theta = \{x_1 / t_1, x_2 / t_2, \ldots\} \qquad \text{SUBST}(\theta, \alpha)$$

  - Application of the substitution to sentences

$$SUBST(\{x / Sam, y / Pam\}, Likes(x, y)) = Likes(Sam, Pam)$$

$$SUBST(\{x / z, y / fatherof(John)\}, Likes(x, y)) =$$
$$Likes(z, fatherof(John))$$

Universal elimination

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

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

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

- E.g., $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ yields:

$King(John) \wedge Greedy(John) \Rightarrow Evil(John),\quad \{x/John\}$

$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard),\quad \{x/Richard\}$

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

$\{x/Father(John)\}$

Example:

$$\forall x\ Likes(x, IceCream) \xrightarrow{\{x/Ben\}} Likes(Ben, IceCream)$$

$$\frac{\forall x\ \phi(x)}{\phi(a)}$$

Existential elimination

- 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

$$\exists x\ Kill(x, Victim) \longrightarrow Kill(Murderer, Victim) \qquad \frac{\exists x\ \phi(x)}{\phi(a)}$$

Special constant called a **Skolem** constant

$$\exists x\ Crown(x) \land OnHead(x, John) \longrightarrow$$

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

Inference rules for quantifiers

- **Universal instantiation (introduction)**

$$\frac{\phi}{\forall x\ \phi} \qquad x - \text{is not free in } \phi$$

  - Introduces a universal variable which does not affect ϕ or its assumptions

$$Sister(Amy, Jane) \qquad \forall x\ Sister(Amy, Jane)$$

- **Existential instantiation (introduction)**

$$\frac{\phi(a)}{\exists x \phi(x)} \qquad \begin{array}{l} a - \text{is a ground term in } \phi \\ x - \text{is not free in } \phi \end{array} \qquad \frac{\alpha}{\exists v\ \text{Subst}(\{g/v\}, \alpha)}$$

  - Substitutes a ground term in the sentence with a variable and an existential statement

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

Example Proof

- 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:

$\forall x,y,z\ American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Nation(z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Nono … has some missiles, i.e.,

$\exists x\ \mathrm{Owns}(\mathrm{Nono},x) \wedge \mathrm{Missile}(x)$:

… all of its missiles were sold to it by Colonel West

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

Missiles are weapons:

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

An enemy of America counts as "hostile“:

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

West, who is American …

American(West)

The country Nono

Nation(Nono)

Nono, an enemy of America …

Enemy(Nono,America), Nation(America)

Example knowledge base contd.

1. $\forall$ x,y,z *American(x)* $\wedge$ *Weapon(y)* $\wedge$ *Sells(x,y,z)* $\wedge$ *Nation(z)* $\wedge$ *Hostile(z)* $\Rightarrow$ *Criminal(x)*
2. $\exists$x Owns(Nono,x) $\wedge$ Missile(x):
3. $\forall$x *Missile(x)* $\wedge$ *Owns(Nono,x)* $\Rightarrow$ *Sells(West,x,Nono)*
4. $\forall$ x *Missile(x)* $\Rightarrow$ *Weapon(x)*
5. $\forall$x *Enemy(x,America)* $\Rightarrow$ *Hostile(x)*
6. *American(West)*
7. *Nation(Nono)*
8. *Enemy(Nono,America)*
9. *Nation(America)*
10. *Owns(Nono,M_1) and Missile(M_1) Existential elimination 2*
11. *Owns(Nono,M_1) And elimination 10*
12. *Missile(M_1) And elimination 10*
13. *Missile(M1)* $\Rightarrow$ *Weapon(M1) Universal elimination 4*
14. *Weapon(M1) Modus Ponens, 12, 13*
15. *Missile(M1)* $\wedge$ *Owns(Nono,M1)* $\Rightarrow$ *Sells(West,M1,Nono) Universal Elimination 3*
16. *Sells(West,M1,Nono) Modus Ponens 10,15*
17. *American(West)* $\wedge$ *Weapon(M1)* $\wedge$ *Sells(West,M1,Nono)* $\wedge$ *Nation(Nono)* $\wedge$ *Hostile(Nono)* $\Rightarrow$ *Criminal(Nono) Universal elimination, three times 1*
18. *Enemy(Nono,America)* $\Rightarrow$ *Hostile(Nono) Universal Elimination 5*
19. *Hostile(Nono) Modus Ponens 8, 18*
20. *American(West)* $\wedge$ *Weapon(M1)* $\wedge$ *Sells(West,M1,Nono)* $\wedge$ *Nation(Nono)* $\wedge$ *Hostile(Nono) And Introduction 6,7,14,16,19*
21. *Criminal(West) Modus Ponens 17, 20*

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 (there are only two ground terms: John and Richard) , 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 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 for doing inference in FOL:
  - propositionalize KB and query
  - apply inference
  - return result

- Problem: with function symbols, there are infinitely many ground terms,
  - e.g., *Father*(*Father*(*Father*(*John*))), etc

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)

- it seems obvious that *Evil*(*John*) is entailed, 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.

- Lets see if we can do inference directly with FOL sentences

Generalized Modus Ponens (GMP)

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

where we can unify p_i‘ and p_i for all i
i.e. $p_i'\theta = p_i\,\theta$ for all i

Example:

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*)
Subst(θ,q) is *Evil*(*John*)

Example:

p_1' is *Missile*(*M1*) p_1 is *Missile*(*x*)
p_2' is *Owns*(*y*, *M1*) p_2 is Owns(*Nono*,*x*)
θ is {x/M1, y/Nono} q is *Sells*(West, Nono, *x*)
Subst(θ,q) is *Sells*(*West*, *Nono*, *M1*)

- Implicit assumption that all variables universally quantified

GMP used with KB of definite clauses (exactly one positive literal)

Soundness and completeness of GMP

GMP is sound

Only derives sentences that are logically entailed

- 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', \; …, \;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

GMP is complete for a KB consisting of definite clauses

- Complete: derives all sentences that entailed
- OR…answers every query whose answers are entailed by such a KB
-
- Definite clause: disjunction of literals of which exactly 1 is positive,

  e.g., King(x) AND Greedy(x) -> Evil(x)

  NOT(King(x)) OR NOT(Greedy(x)) OR Evil(x)

Generalized Modus Ponens (GMP)

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

Substitution that satisfies the generalized inference rule can be build via *unification process*

Advantage of the generalized rules: **they are focused**

- only substitutions that allow the inferences to proceed are tried

Use substitutions that let us make inferences !!!!

Convert each sentence into cannonical form prior to inference: Either an atomic sentence or an implication with a conjunction of atomic sentences on the left hand side and a single atom on the right (Horn clauses)

Unification

- **Problem in inference:** Universal elimination gives us many opportunities for substituting variables with ground terms

$$\frac{\forall x\ \phi(x)}{\phi(a)} \qquad a \text{ - is a constant symbol}$$

- **Solution: make only substitutions that may help**
  - Use substitutions of "similar" sentences in KB

- **Unification** – takes two similar sentences and computes the substitution that **makes them look the same**, if it exists

$$UNIFY\ (p,q) = \sigma \ \text{ s.t. } SUBST(\sigma, p) = SUBST\ (\sigma, q)$$

Unification

- **Unification:**

  $$UNIFY(p, q) = \sigma \text{ s.t. } SUBST(\sigma, p) = SUBST(\sigma, q)$$

- **Examples:**

$$UNIFY(Knows(John, x), Knows(John, Jane)) = \{x / Jane\}$$

$$UNIFY(Knows(John, x), Knows(y, Ann)) = \{x / Ann, y / John\}$$

$$UNIFY(Knows(John, x), Knows(y, MotherOf(y))) \\ = \{x / MotherOf(John), y / John\}$$

$$UNIFY(Knows(John, x), Knows(x, Elizabeth)) = fail$$

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, Elizabeth)	{x/ Elizabeth,y/John}}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x, Elizabeth)	{fail}

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

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.
- Most general unifier is the substitution that makes the least commitment about the bindings of the variables
- 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 revisited

1. $\forall$ x,y,z *American(x)* $\wedge$ *Weapon(y)* $\wedge$ *Sells(x,y,z)* $\wedge$ *Nation(z)* $\wedge$ *Hostile(z)* $\Rightarrow$ *Criminal(x)*
2. $\exists$x Owns(Nono,x) $\wedge$ Missile(x):
3. $\forall$x *Missile(x)* $\wedge$ *Owns(Nono,x)* $\Rightarrow$ *Sells(West,x,Nono)*
4. $\forall$ x *Missile(x)* $\Rightarrow$ *Weapon(x)*
5. $\forall$x *Enemy(x,America)* $\Rightarrow$ *Hostile(x)*
6. *American(West)*
7. *Nation(Nono)*
8. *Enemy(Nono,America)*
9. *Nation(America)*

Convert the sentences into Horn form

1. *American(x)* $\wedge$ *Weapon(y)* $\wedge$ *Sells(x,y,z)* $\wedge$ *Nation(z)* $\wedge$ *Hostile(z)* $\Rightarrow$ *Criminal(x)*
2. *Owns(Nono,* M_1*)*
3. *Missile(* M_1*)*
4. *Missile(x)* $\wedge$ *Owns(Nono,x)* $\Rightarrow$ *Sells(West,x,Nono)*
5. *Missile(x)* $\Rightarrow$ *Weapon(x)*
6. *Enemy(x,America)* $\Rightarrow$ *Hostile(x)*
7. *American(West)*
8. *Nation(Nono)*
9. *Enemy(Nono,America)*
10. *Nation(America)*
11. *Proof*
12. *Weapon(M1)*
13. *Hostile(Nono)*
14. *Sells(West,M1,Nono)*

---

15. *Criminal(West)*

Inference appoaches in FOL

- Forward-chaining
  - Uses GMP to add new atomic sentences
  - Useful for systems that make inferences as information streams in
  - Requires KB to be in form of first-order definite clauses

- Backward-chaining
  - Works backwards from a query to try to construct a proof
  - Can suffer from repeated states and incompleteness
  - Useful for query-driven inference

- Note that these methods are generalizations of their propositional equivalents

Forward chaining algorithm

```
function FOL-FC-Ask(KB, α) returns a substitution or false
    repeat until new is empty
        new ← { }
        for each sentence r in KB do
            ( p1 ∧ . . . ∧ pn ⇒ q) ← Standardize-Apart(r)
            for each θ such that (p1 ∧ . . . ∧ pn)θ = (p'1 ∧ . . . ∧ p'n)θ
                            for some p'1, . . . , p'n in KB
                q' ← Subst(θ, q)
                if q' is not a renaming of a sentence already in KB or new then do
                    add q' to new
                    φ ← Unify(q', α)
                    if φ is not fail then return φ
        add new to KB
    return false
```

Forward chaining proof

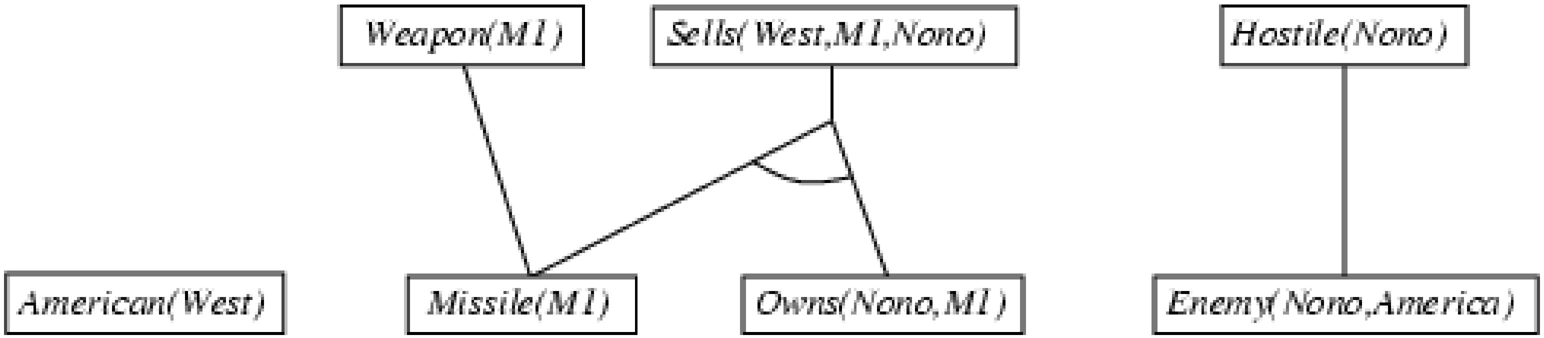
American(West)

Missile(M1)

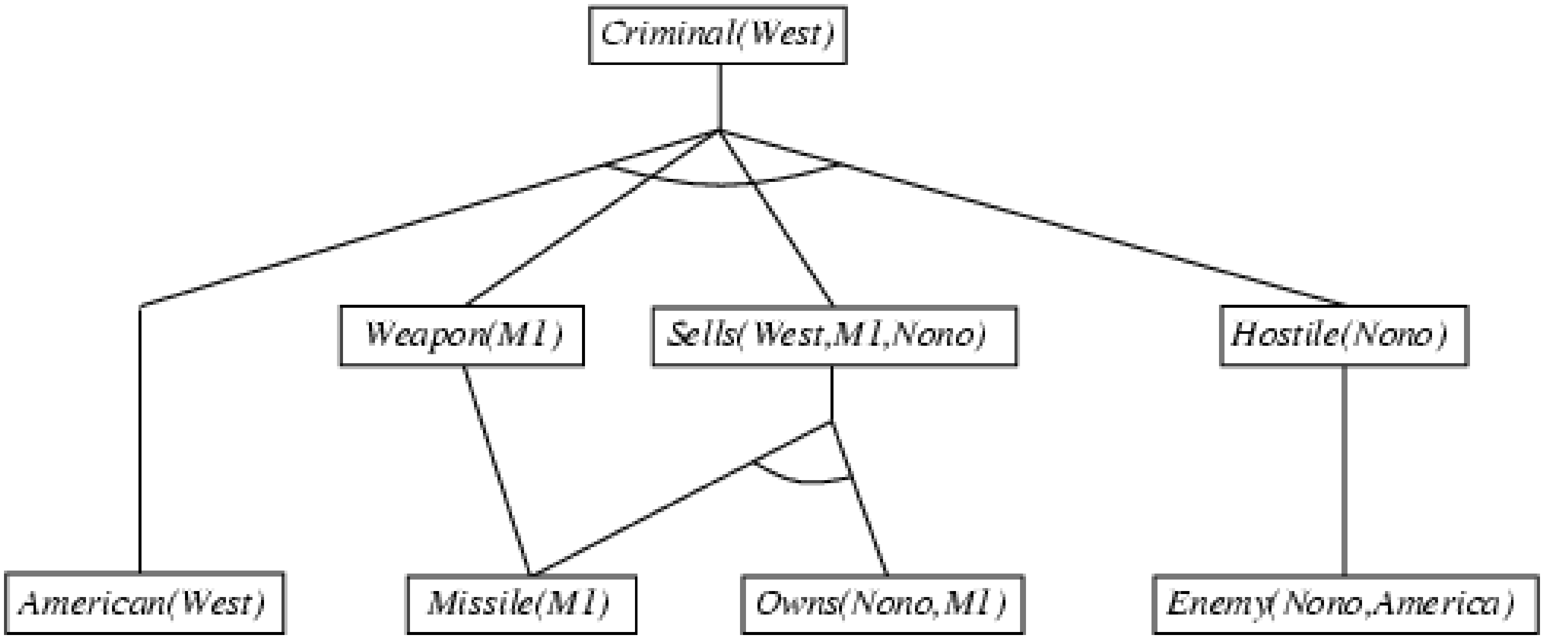
Owns(Nono,M1)

Enemy(Nono,America)

Forward chaining proof



Forward chaining proof



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

Backward chaining algorithm

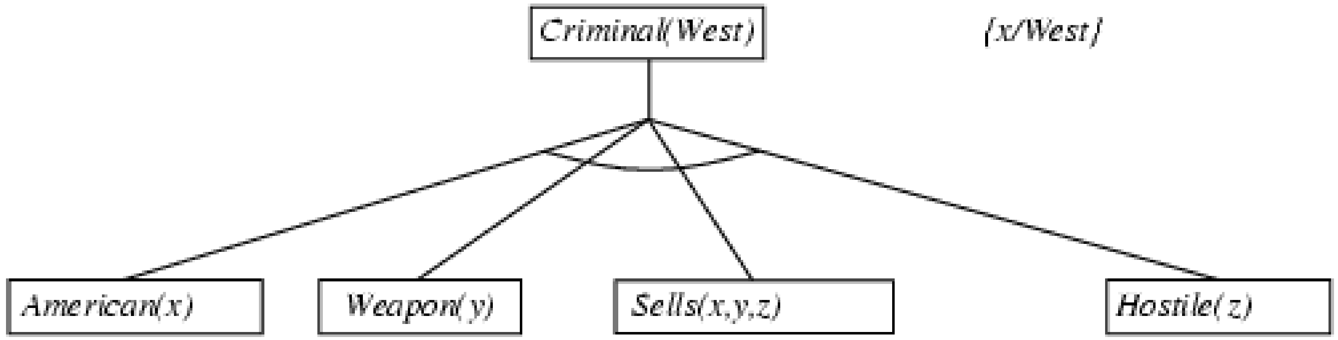
function FOL-BC-ASK(KB, goals, θ) **returns** a set of substitutions
  inputs: KB, a knowledge base
    $goals$, a list of conjuncts forming a query
    θ, the current substitution, initially the empty substitution $\{\,\}$
  local variables: ans, a set of substitutions, initially empty

  if $goals$ is empty **then return** $\{\theta\}$
  $q' \leftarrow \textsc{Subst}(\theta, \textsc{First}(goals))$
  for each r **in** KB where $\textsc{Standardize-Apart}(r) = (\, p_1 \ \wedge\ \ldots\ \wedge\ p_n \ \Rightarrow\ q)$
    and $\theta' \leftarrow \textsc{Unify}(q, q')$ succeeds
   $ans \leftarrow \text{FOL-BC-}\textsc{Ask}(KB, [p_1, \ldots, p_n | \textsc{Rest}(goals)], \textsc{Compose}(\theta, \theta')) \cup ans$
  return ans

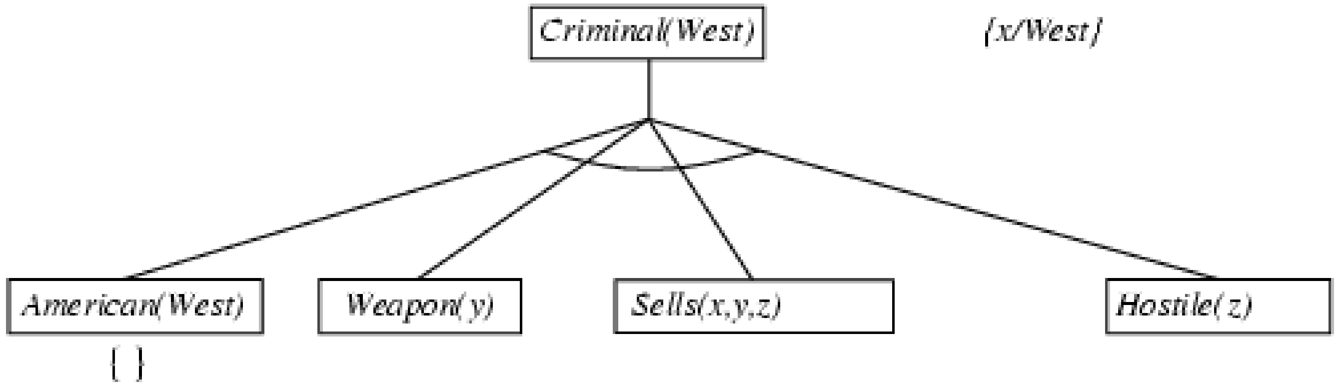
$$\text{SUBST}(\text{COMPOSE}(\theta_1, \theta_2), p) = \text{SUBST}(\theta_2, \text{SUBST}(\theta_1, p))$$

Backward chaining example

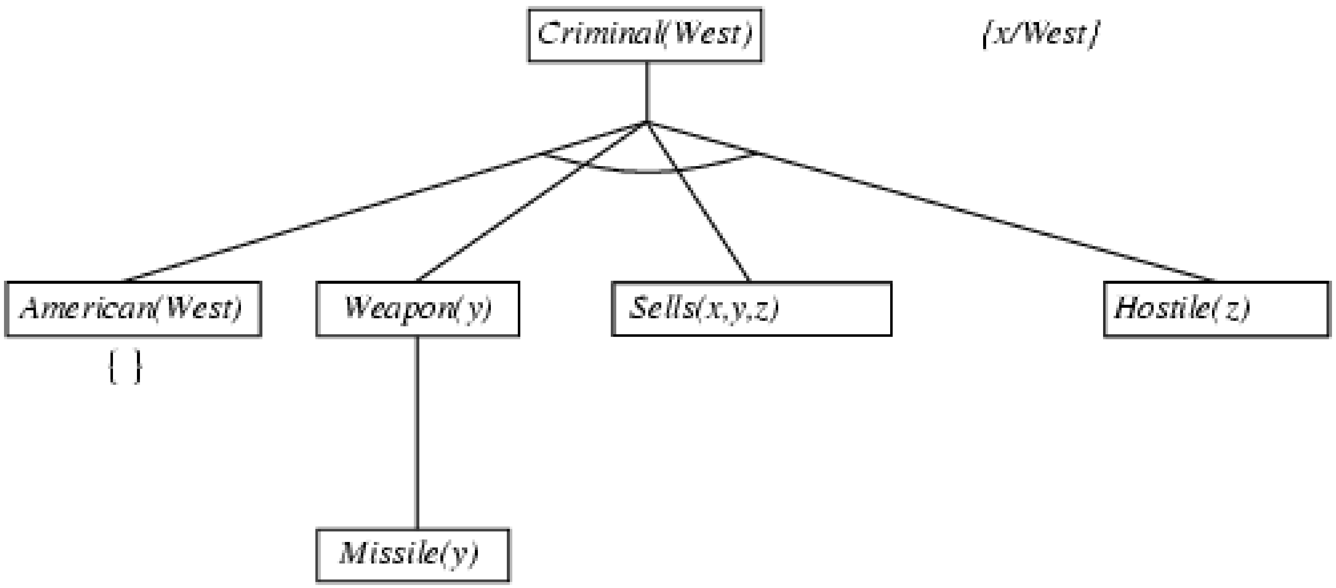
Backward chaining example



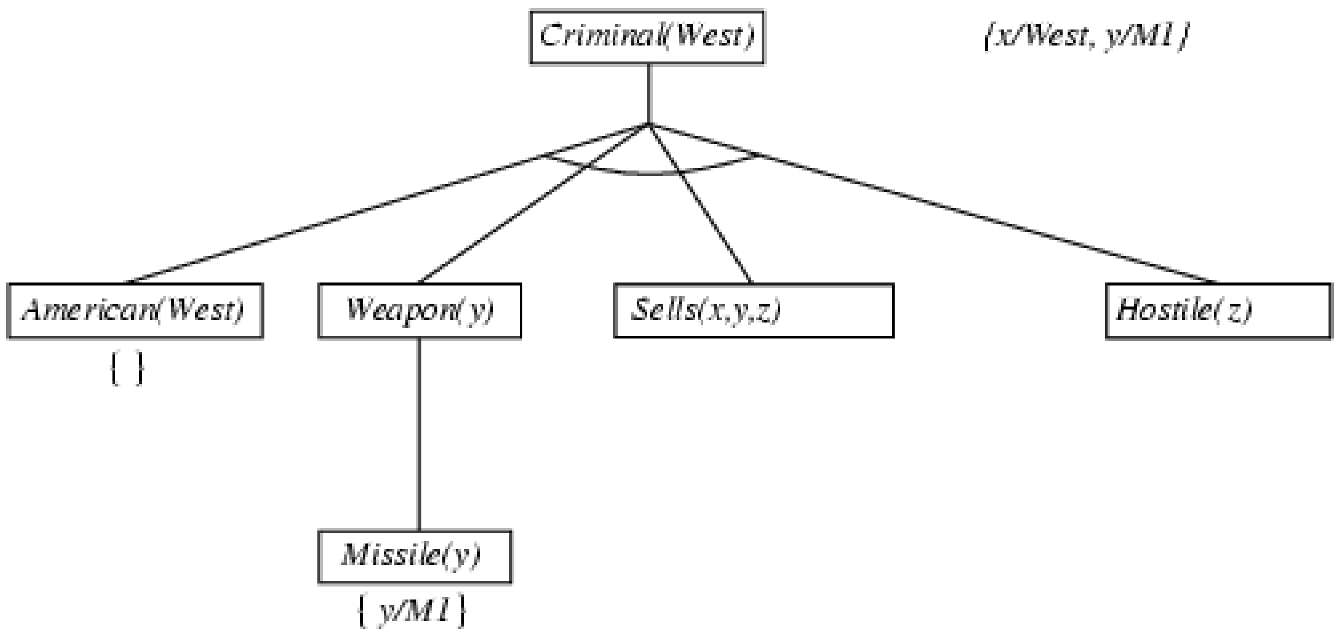
Backward chaining example



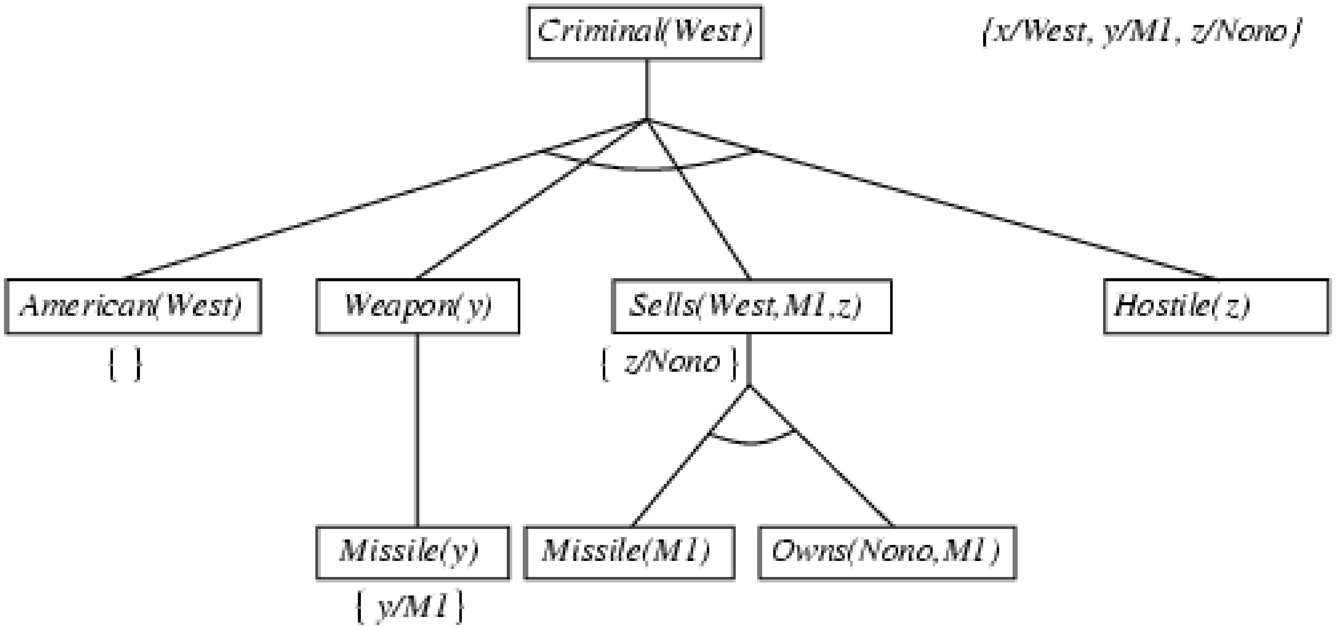
Backward chaining example



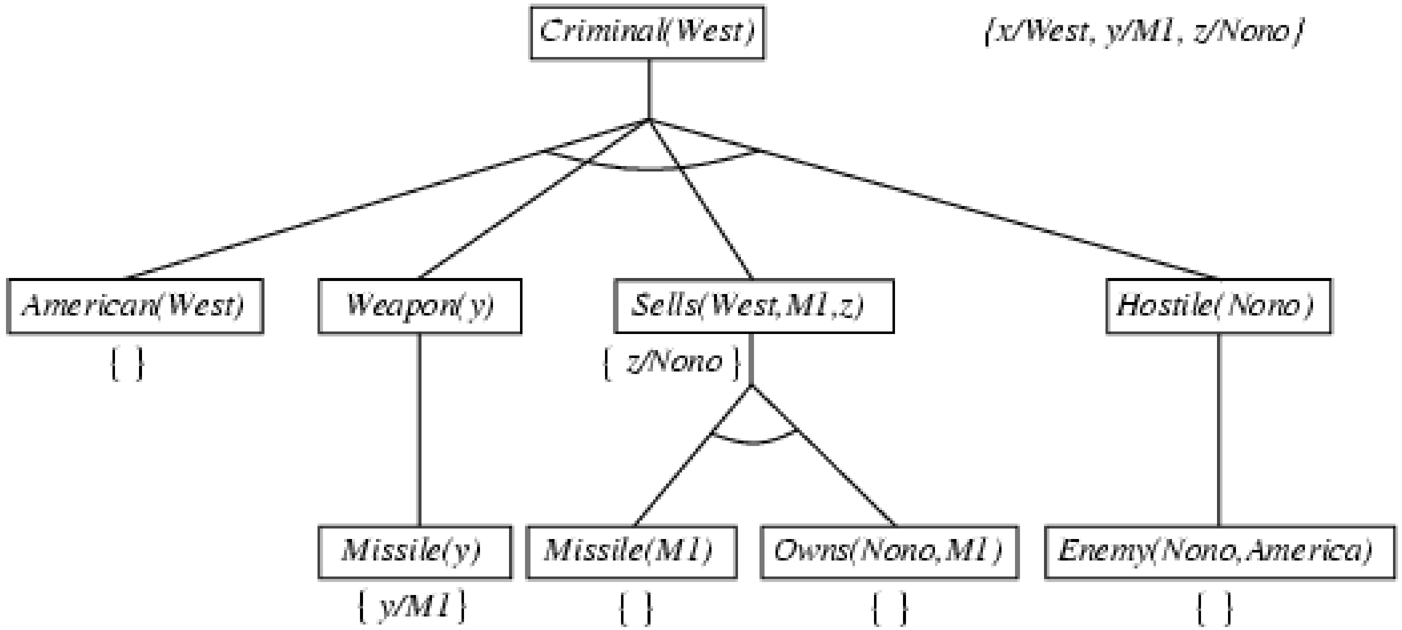
Backward chaining example



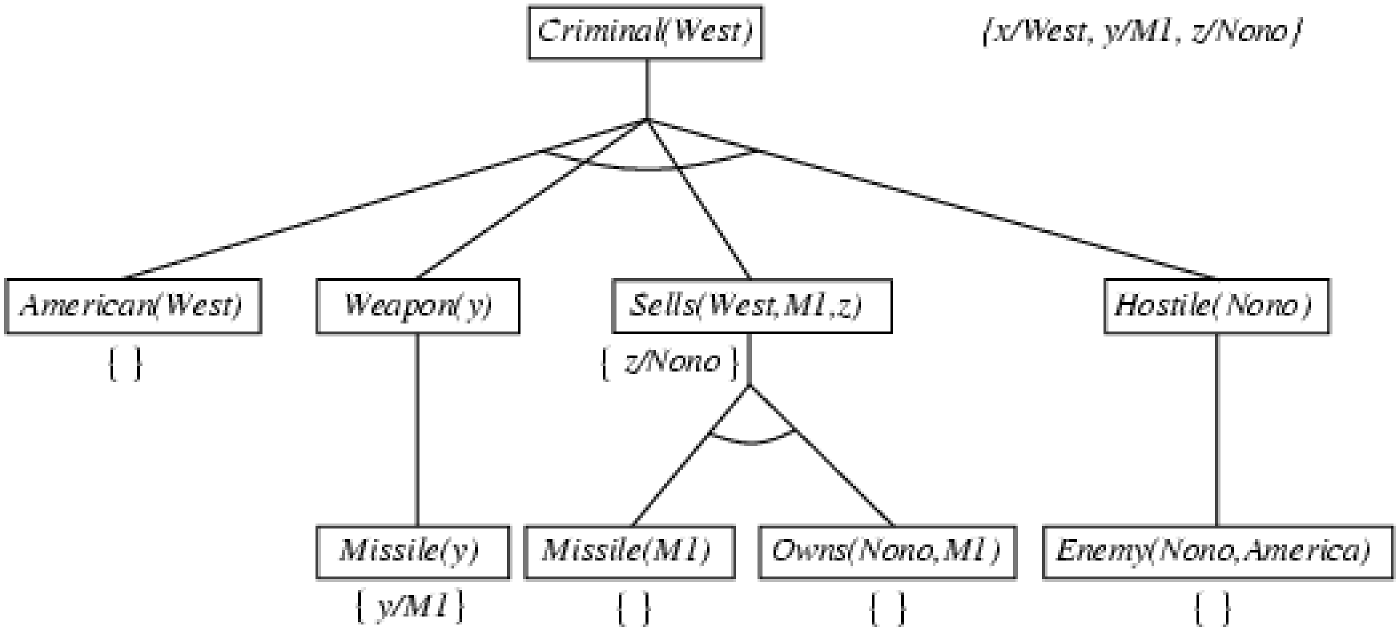
Backward chaining example



Backward chaining example



Backward chaining example



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
- 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