Deduction in First Order Horn KB[1]

Lecture 5

[1]The slides have been prepared using the textbook material available on the web, and the slides of the previous editions of the course by Prof. Luigia Carlucci Aiello

Inference in first order logic

- inference rules for quantifiers [RN 9.1]
- unification [simplified]
- generalized modus ponens
- forward chaining [RN 9.3]
- backward chaining [RN 9.4]

Substitutions

Inference in fol requires the **substitution** of variables with terms: a function mapping $\textsc{Var}$ into $\textsc{Term}$.

The substitution σ of variable x with a term t is denoted by $\sigma = \{x/t\}$ or $\sigma = \{t = x\}$

The **application** of a substitution σ to the expression α is denoted by $\alpha\sigma$ (simultaneous substitutions of the variables with the corresponding terms), or $\alpha[t/x]$ for a single variable.

Notation

R&N: application of a substitution σ to the expression α.

$$\textsc{Subst}(\sigma, \alpha)$$

Example:

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

Universal instantiation (UI)

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

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

for any variable v and ground term g

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

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

Existential instantiation (EI)

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

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

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

Another example: $\exists x \;\; Killed(x, Victim)$
$Killed(k, Victim)$
provided k is a new constant symbol chosen to denote the (unknown) "killer"

Summarizing ...

UI can be applied several times to *add* new sentences;
the new KB is logically equivalent to the old

EI can be applied once to *replace* the existential sentence;
the new KB is *not* equivalent to the old,
but is satisfiable iff the old KB was satisfiable

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:

$$King(John) \wedge Greedy(John) \;\Rightarrow\; Evil(John)$$
$$King(Richard) \wedge Greedy(Richard) \;\Rightarrow\; Evil(Richard)$$
$$King(John) \quad Greedy(John) \quad Brother(Richard, John)$$

The new KB is propositionalized: proposition symbols are:
$King(John)$, $Greedy(John)$, $Evil(John)$, $King(Richard)$ etc.

Reduction contd.

Claim*: a ground sentence is entailed by new propositional KB iff it is entailed by original KB

$KB_{prop} \models A$ iff $KB \models A$

Approach: propositionalize KB and query and apply any propositional reasoning method

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 a FOL KB,

it is entailed by a *finite* subset of the propositional 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

Entailment in FOL is semidecidable (Turing (1936), Church (1936))

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 $Evil(John)$, 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!

Horn Clauses – Quick recap

♢ *Literals*: predicate symbols (*atoms*) or negated atoms (rather than just propositional variables).

♢ A *clause* is a disjuntion of literals $L_1 \vee L_2 \vee \cdots \vee L_n$. usually written as: $\{L_1, L_2, \ldots, L_p\}$.

♢ A formula is in *conjunctive normal form (CNF)* or in *clausal form* or it is called a *set of clauses* if:
it is $C_1 \wedge C_2 \wedge \cdots \wedge C_n$ where C_i are clauses.

♢ A clause is **Horn** if $n \leq 1$, with n = number of pos. lit.

♢ If $n = 1$: we have a *definite (Horn) clause*

CNF – variable

♢ Existentially quantified variables are eliminated by **EI** (skolemization)

♢ Universally quantifiers are removed (all variables are assumed universally quantified)

Later we will extend the construction of the clausal form (CNF) with nested quantifiers.

Horn Clauses

Horn Clauses

$Facts\ \top \Rightarrow A(x)$ (or simply $A(x)$)
$Rules\ A_1(x) \wedge \ldots \wedge A_n(x) \Rightarrow B(x)$
$Goals\ A_1(x) \wedge \ldots \wedge A_n(x) \Rightarrow \bot$ (negation of $A_1(x) \wedge \ldots \wedge A_n(x)$)

Variables are universally quantified (existentials replaced by EI).

Horn Data Bases (HDB) – Datalog emphasize that:

- ♢ no function symbols;
- ♢ domain restricted to the "known" individuals (finite).

For Horn Clauses **Forward** reasoning and **Backward** reasoning work as in the propositional case, except for the introduction of variables.

Unification

Unification of two expressions is the process of finding a substitution that, when applied to them, makes them identical.

Example: $\{x/b\}$ is a unifier for $p(a, x)$ and $p(a, b)$.

The **application** of the substitution σ to the expression t (*without function symbols*) is defined as follows:

- if c is a constant symbol, $c\sigma = c$;
- if x is a variable symbol, $x\sigma = b$ (with $\sigma = \{x/b, \ldots\}$);
- if x is a variable symbol, $x\sigma = x$ if x is not a variable in σ.

Example: $p(a, x)\sigma = p(a, b)$ (with $\sigma = \{x/b\}$)

Unification (without function symbols)

Unification is applied to expressions of the form $P(t_1, t_2, \ldots, t_n)$.

$unify(P(t_1, t_2, \ldots, t_n), P(s_1, s_2, \ldots, s_n))$: find a substitution that makes $P(t_1, t_2, \ldots, t_n)$ and $P(s_1, s_2, \ldots, s_n)$ identical.

Examples:

$unify(P(X), P(a)) = \{X/a\}$

$unify(P(X), P(Y)) = \{X/Y\}$

$unify(P(a), Q(a)) =?$ NO

$unify(P(X, b), P(a, Y)) = \{X/a, Y/b\}$

$unify(P(X, X), P(a, b))$ NO

Unification simplified (final comment)

An expression s is *more general* than an expression t if t is an instance of s, but not viceversa.

Examples:

$P(a, x)$ is more general than $P(a, b)$

$P(y, x)$ is more general than $P(a, b)$.

Intuition

We choose the unifier so that the expression resulting from the substitution is the most general.

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 missils, and all of its missils were sold to it by Colonel West, who is American.

Prove that Col. West is a criminal

. . . selling weapons to hostile nations is a crime for an American:

Example knowledge base contd.

. . . selling weapons to hostile nations is a crime for an American:

$$Amer(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Host(z) \Rightarrow Criminal(x)$$

Nono . . . has some missils

Example knowledge base contd.

. . . selling weapons to hostile nations is a crime for an American:

$$Amer(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Host(z) \Rightarrow Criminal(x)$$

Nono . . . has some missils, i.e., $\exists\, x\, Owns(Nono, x) \wedge missil(x)$:

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

. . . all of its missils were sold to it by Colonel West

Example knowledge base contd.

. . . selling weapons to hostile nations is a crime for an American:

$$Amer(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Host(z) \Rightarrow Criminal(x)$$

Nono . . . has some missils, i.e., $\exists\, x\, Owns(Nono, x) \wedge missil(x)$:

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

. . . all of its missils were sold to it by Colonel West

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

missils are weapons:

Example knowledge base contd.

. . . selling weapons to hostile nations is a crime for an American:

$$Amer(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Host(z) \Rightarrow Criminal(x)$$

Nono . . . has some missils, i.e., $\exists\, x\, Owns(Nono, x) \wedge missil(x)$:

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

. . . all of its missils were sold to it by Colonel West

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

missils are weapons:

$$missil(x) \Rightarrow Weapon(x)$$

An enemy of America counts as "Host":

Example knowledge base contd.

. . . selling weapons to hostile nations is a crime for an American:

$(1) Amer(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Host(z) \Rightarrow Criminal(x$

Nono . . . has some missils, i.e., $\exists\, x\, Owns(Nono, x) \wedge missil(x)$:

$(2) Owns(Nono, M_1)$ and $(3) missil(M_1)$

. . . all of its missils were sold to it by Colonel West

$(4) missil(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

missils are weapons:

$(5) missil(x) \Rightarrow Weapon(x)$

An enemy of America counts as "Host":

$(6) Enemy(x, America) \Rightarrow Host(x)$

West, who is American . . .

$(7) American(West)$

The country Nono, an enemy of America . . .

$(8) Enemy(Nono, America)$

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 θ is the unifier of all the pairs $p_i' = p_i\theta$ for all i

p_1' is $missil(M_1)$ $\quad$ p_1 is $missil(x)$
θ is $\{x/M_1\}$ $\quad$ q is $Weapon(x)$
$q\theta$ is $Weapon(M_1)$

$\diamondsuit$ The Generalized version of MP is a sound and complete rule for Definite Clauses

The proof with GMP

From (3) and (5):
$(9) Weapon(M1)$

From (8) and (6):
$(10) Host((Nono)$

From (2), (3) and (4):
$(11) Sells(West, Nono, M1)$

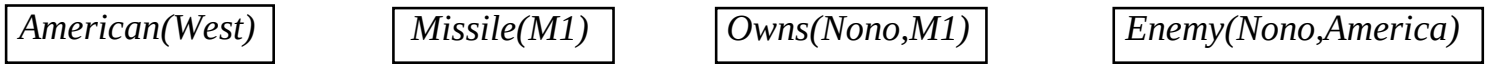
From (7), (8), (10), (11) and (1):
$Criminal(West)$

Proof search

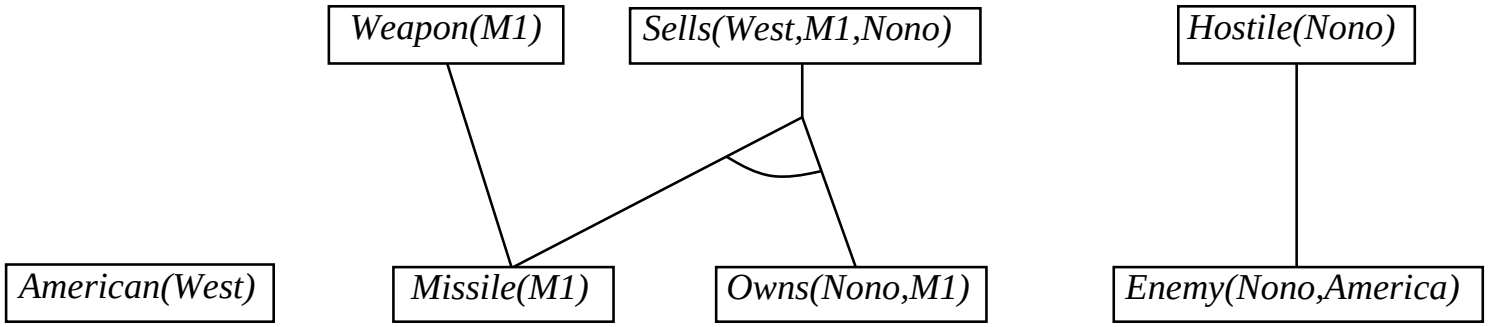
Initial State = KB
Operators = inference rules
Goal Test = Criminal(West) $\in$ KB

- The proof is 12 steps.
- Quantifier elimination can be extremely expensive.
- Checking for applicability of Generalized Modus Ponens requires some work.

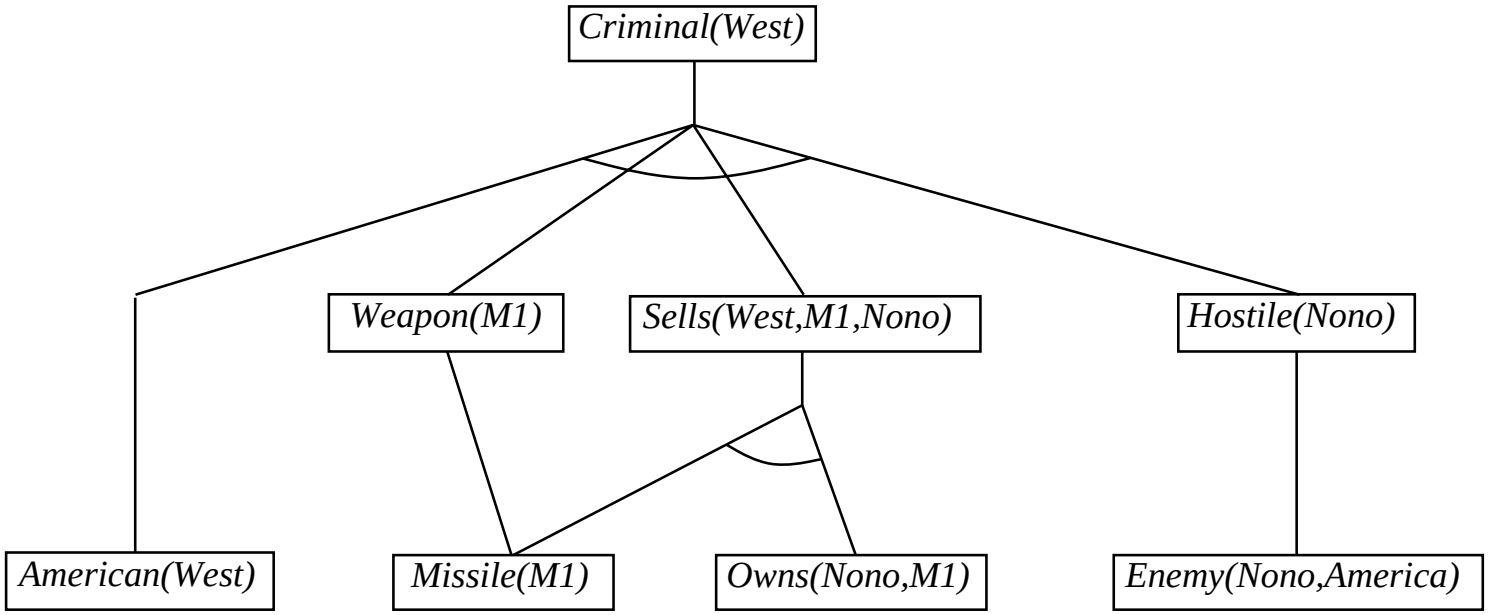
Forward chaining proof



Forward chaining proof



Forward chaining proof



Notation

Standardize-Apart(r)

denotes a renaming of the variables in a rule that avoids name conflicts of the variables.

Notice: rules (clauses) derive from closed formulae, hence the variables in a clause should have names different from those in other rules.

function FOL-FC-ASK(KB, α) **returns** subst 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 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*

Properties of forward chaining

FC is sound and complete for first-order definite clauses (proof similar to propositional case)

- In the case HDB (no functions and finite domain), it terminates in poly iterations: $n \times d$ (n the maximum number of premises in the rules and d the size of the HDB).
- However, matching conjunctive premises against known facts is NP-hard as a CSP-problem can be formulated as a rule: $diff(X, Y) \wedge \ldots \wedge diff(W, Z) \Rightarrow Colorable$
- If functions symbols are allowed, it may not terminate, if α is not entailed. This is unavoidable: entailment with definite clauses is **semidecidable**.

Efficiency of forward chaining

1. **conjunct ordering** – Most Constrained Variable
2. **incremental matching** at iteration k, focus on premises added on iteration $k-1$ (i.e. match each rule whose premise contains a newly added literal)
3. **Database indexing** allows $O(1)$ retrieval of known facts
   e.g., query $Missile(x)$ retrieves $Missile(M_1)$

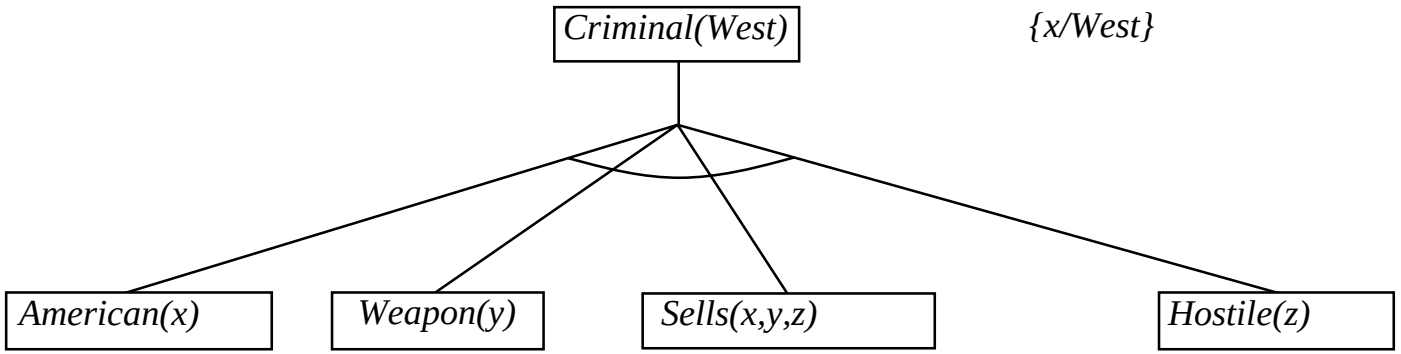
4. restrict process to **relevant facts**

Forward chaining is widely used in deductive databases

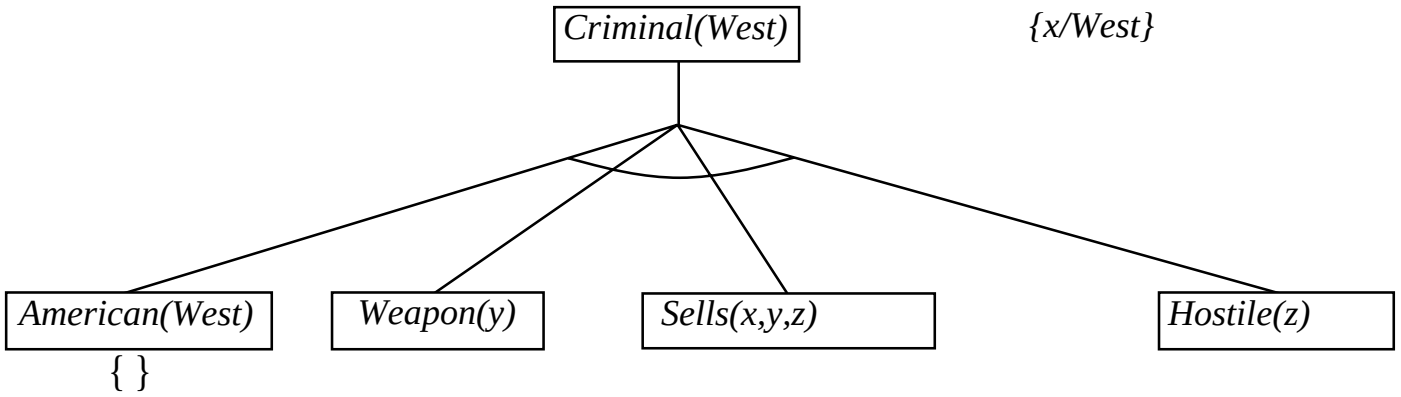
Backward chaining example

Criminal(West)

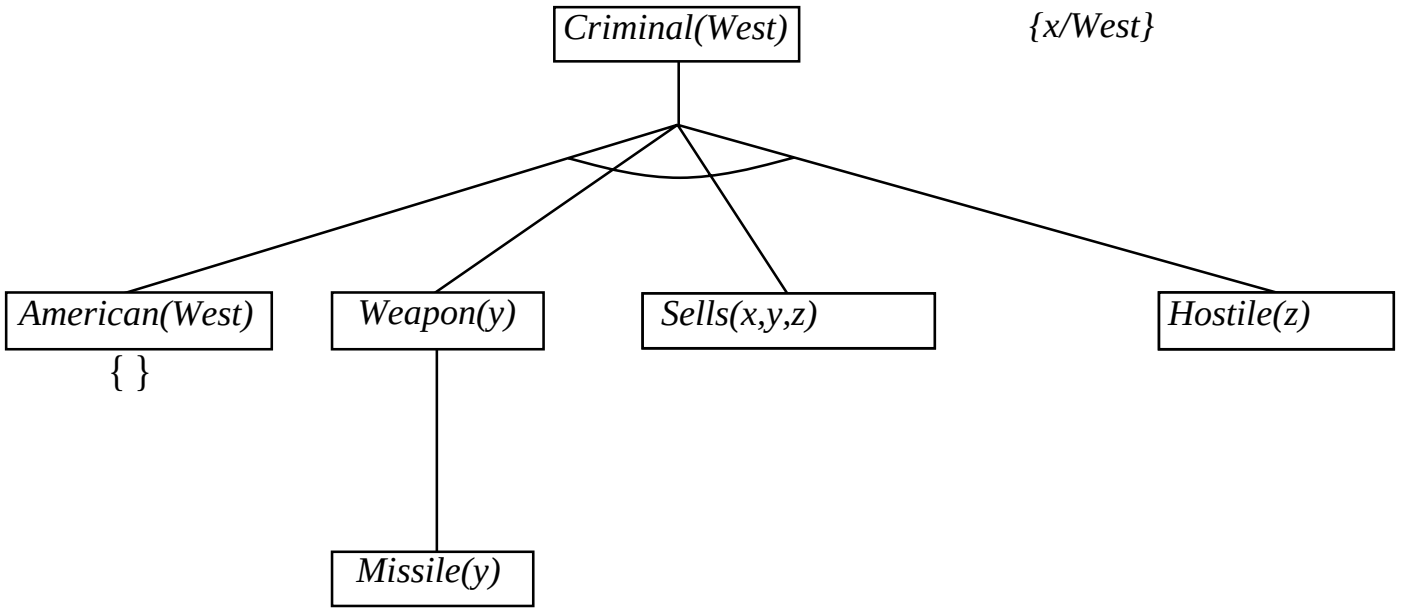
Backward chaining example



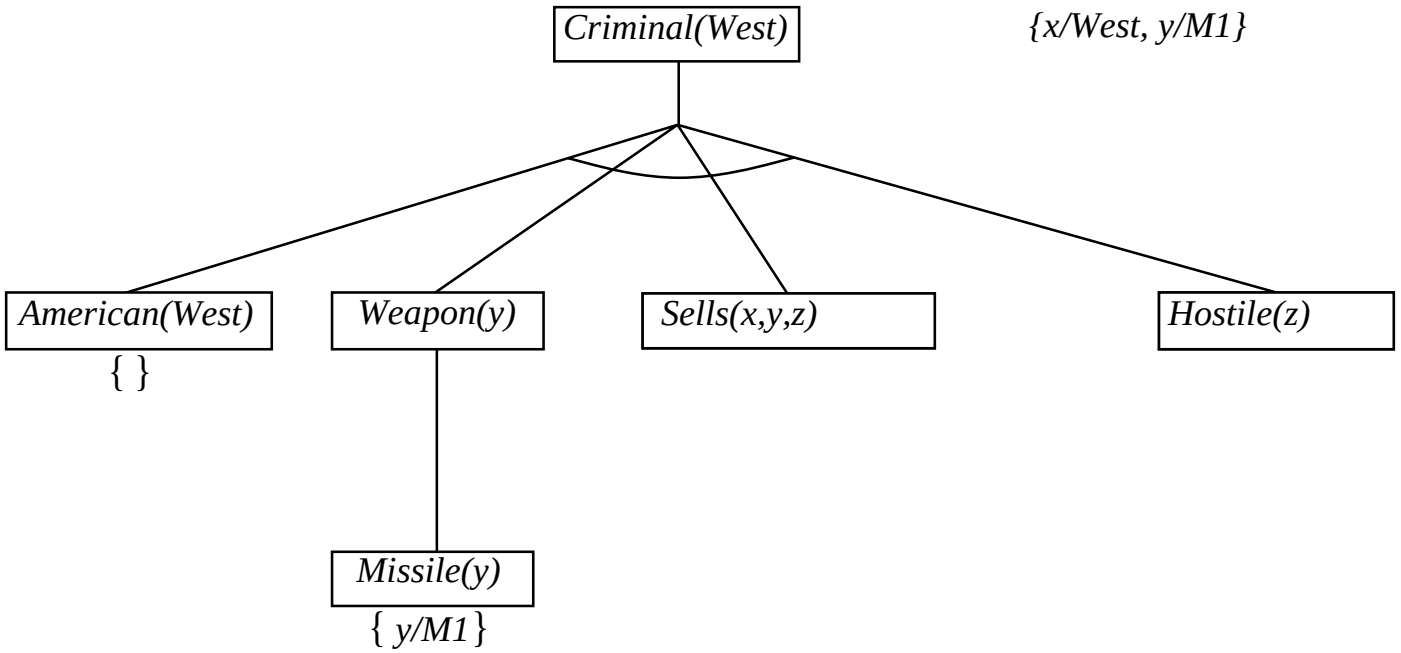
Backward chaining example



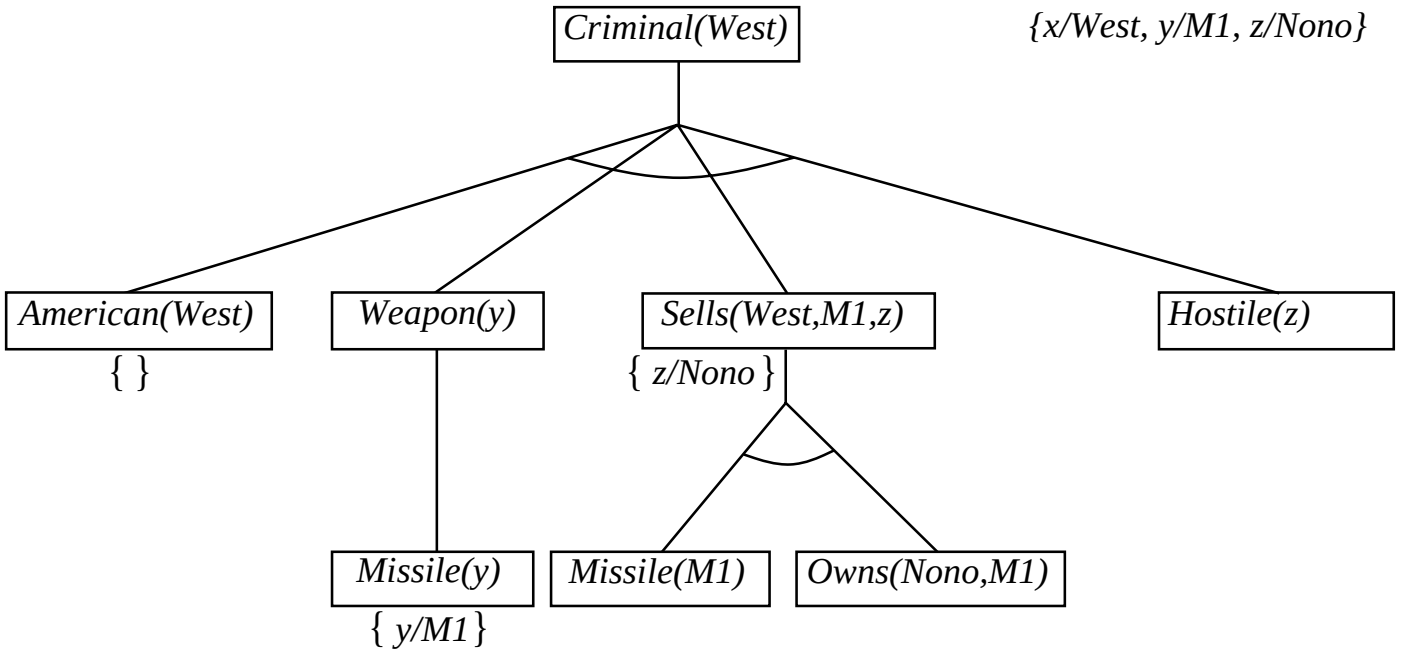
Backward chaining example



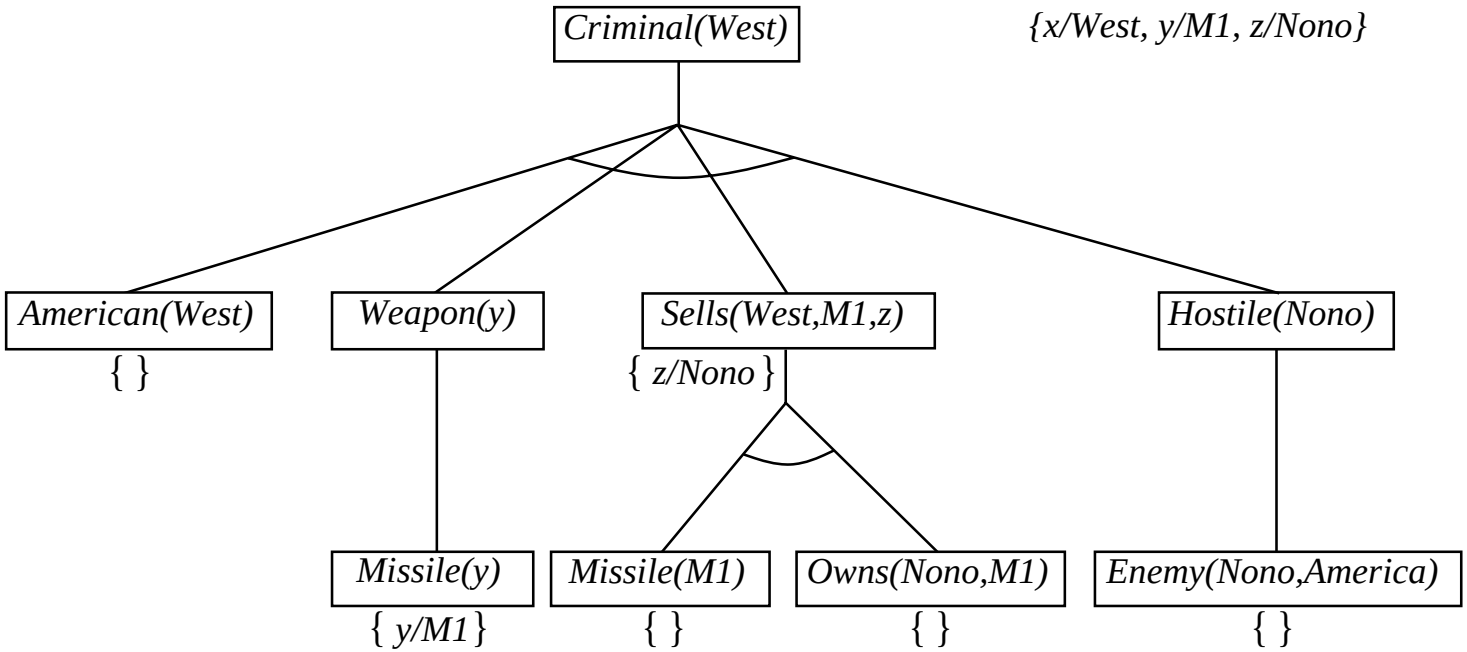
Backward chaining example



Backward chaining example



Backward chaining example



function FOL-BC-ASK(KB, $goals$, θ) **returns** a set of substitutions

inputs: KB, a knowledge base
$goals$, list of conjuncts(θ already applied)
θ, current substitution, init empty $\{\}$

local variables: $answers$, set of substs, init empty

if $goals$ is empty **then return** $\{\theta\}$
$q' \leftarrow \text{SUBST}(\theta, \text{FIRST}(goals))$
for each sentence 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
$new\_goals \leftarrow [p_1, \ldots, p_n | \text{REST}(goals)]$
$answers \leftarrow \text{FOL-BC-ASK}(KB, new\_goals, \text{COMPOSE}(\theta', \theta)) \cup answers$
return $answers$

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!)

At the basis of logic programming (PROLOG).

Horn Clause programming: PROLOG

Program = set of Horn clauses = `head :- literal`$_1$`, ... literal`$_n$`.`

Basis: backward chaining with Horn clauses + (**full Unification**):
Compare with abstract interpreter in PROLOG-1

Widely used in Europe, Japan (basis of 5th Generation project)

Compilation techniques

Nono in Prolog

```
criminal(X) :- american(X), weapon(Y),
                sells(X,Y,Z), hostile(Z).
sells(west,X,nono):- missil(X), owns(nono,X).
weapon(X):- missil(X).
hostile(nono).
owns(nono,m1).
missil(m1).
enemy(nono,america).
american(west).
```

Goal:

```
? criminal(X).
```

Prolog technology

- Compilation (Warren Abstract Machine)
  - clause indexing
  - conjunct ordering
  - memoization
  - continuation
- PROLOG + CSP (Constraint Logic Programming), recall $MapColoring$

  "Of course the fact that one can write a planner ot o NL parser in a few lines of code makes it somewhat more desirable than C for prototyping ... most small scale AI research projects"