# Lecture 9: Inference in First Order Logic

## CS 580 (001) - Spring 2018

Amarda Shehu

Department of Computer Science
George Mason University, Fairfax, VA, USA

April 11, 2018

# A Brief History of Reasoning

| | | |
|---|---|---|
| 450B.C. | Stoics | propositional logic, inference (maybe) |
| 322B.C. | Aristotle | "syllogisms" (inference rules), quantifiers |
| 1565 | Cardano | probability theory (propositional logic + uncertainty) |
| 1847 | Boole | propositional logic (again) |
| 1879 | Frege | first-order logic |
| 1922 | Wittgenstein | proof by truth tables |
| 1930 | Gödel | $\exists$ complete algorithm for FOL |
| 1930 | Herbrand | complete algorithm for FOL (reduce to propositional) |
| 1931 | Gödel | $\neg\exists$ complete algorithm for arithmetic |
| 1960 | Davis/Putnam | "practical" algorithm for propositional logic |
| 1965 | Robinson | "practical" algorithm for FOL—resolution |

# Universal Instantiation (UI)

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

$$\frac{\forall v\ \alpha}{\textsc{Subst}(\{v/g\}, \alpha)} \text{ for any variable } v \text{ and ground term } g$$

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

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

# Existential Instantiation (EI)

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

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

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: from $\exists x\, d(x^y)/dy = x^y$ we obtain

$$d(e^y)/dy = e^y$$

provided $e$ is a new constant symbol

# EI Continued

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

Suppose the KB contains just the following:

$$\forall x King(x) \land Greedy(x) \implies Evil(x)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

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

$$King(John) \land Greedy(John) \implies Evil(John)$$
$$King(Richard) \land Greedy(Richard) \implies Evil(Richard)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

The new KB is propositionalized: proposition symbols are

$King(John)$, $Greedy(John)$, $Evil(John)$, $King(Richard)$ etc.

**Claim:** a ground sentence* is entailed by new KB iff entailed by original KB
**Claim:** every FOL KB can be propositionalized so as to preserve entailment

**Idea:** propositionalize KB and query, apply resolution, return result

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

Theorem: Herbrand (1930). If a sentence $\alpha$ is entailed by an FOL KB,
it is entailed by a **finite** subset of the propositional KB

**Idea:** For $n = 0$ to $\infty$ do
create a propositional KB by instantiating with depth-$n$ terms
see if $\alpha$ is entailed by this KB

Problem: works if $\alpha$ is entailed, loops if $\alpha$ is not entailed

Theorem: Turing (1936), Church (1936).
Entailment in FOL is *semidecidable*

# Problems with Propositionalization

Propositionalization seems to generate lots of irrelevant sentences.
E.g., from

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

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

With function symbols, it gets nuch much worse!

We can get the inference immediately if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

$\theta = \{x/John, y/John\}$ works
$\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, OJ)$ | |

We can get the inference immediately if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

$\theta = \{x/John, y/John\}$ works
$\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, OJ)$ | $\{x/OJ, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | |

We can get the inference immediately if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

$\theta = \{x/John, y/John\}$ works
$\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, OJ)$ | $\{x/OJ, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | $\{y/John, x/Mother(John)\}$ |
| $Knows(John, x)$ | $Knows(x, OJ)$ | |

We can get the inference immediately if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

$\theta = \{x/John, y/John\}$ works
$\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, OJ)$ | $\{x/OJ, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | $\{y/John, x/Mother(John)\}$ |
| $Knows(John, x)$ | $Knows(x, OJ)$ | *fail* |

We can get the inference immediately if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

$\theta = \{x/John, y/John\}$ works
$\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, OJ)$ | $\{x/OJ, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | $\{y/John, x/Mother(John)\}$ |
| $Knows(John, x)$ | $Knows(x, OJ)$ | *fail* |

Standardizing apart eliminates overlap of variables, e.g.,

# Unification

We can get the inference immediately if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

$\theta = \{x/John, y/John\}$ works
$\text{UNIFY}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, OJ)$ | $\{x/OJ, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | $\{y/John, x/Mother(John)\}$ |
| $Knows(John, x)$ | $Knows(x, OJ)$ | $fail$ |

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

We can get the inference immediately if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

$\theta = \{x/John, y/John\}$ works

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

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, OJ)$ | $\{x/OJ, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | $\{y/John, x/Mother(John)\}$ |
| $Knows(John, x)$ | $Knows(x, OJ)$ | $fail$ |

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

# Generalized Modus Ponens (GMP)

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

$p_1{}'$ is $King(John)$ $\quad$ $p_1$ is $King(x)$

$p_2{}'$ is $Greedy(y)$ $\quad$ $p_2$ is $Greedy(x)$

$\theta$ is $\{x/John, y/John\}$ $\quad$ $q$ 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 definite clause $p$, 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. $p_1{}', \ \ldots, \ p_n{}' \models p_1{}' \wedge \ldots \wedge p_n{}' \models p_1{}'\theta \wedge \ldots \wedge p_n{}'\theta$
3. From 1 and 2, $q\theta$ follows by ordinary Modus Ponens

# 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

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

# Example Knowledge Base

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

$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$

. . . it is a crime for an American to sell weapons to hostile nations:
$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$

Nono . . . has some missiles

# Example Knowledge Base

... it is a crime for an American to sell weapons to hostile nations:
$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$

Nono ... has some missiles
i.e., $\exists x \; Owns(Nono, x) \wedge Missile(x)$:

. . . it is a crime for an American to sell weapons to hostile nations:
$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \implies Criminal(x)$

Nono . . . has some missiles
i.e., $\exists x \; Owns(Nono, x) \land Missile(x)$:
$Owns(Nono, M_1)$ and $Missile(M_1)$

# Example Knowledge Base

... it is a crime for an American to sell weapons to hostile nations:
$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \implies Criminal(x)$

Nono ... has some missiles
i.e., $\exists x \; Owns(Nono, x) \land Missile(x)$:
$Owns(Nono, M_1)$ and $Missile(M_1)$

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

# Example Knowledge Base

... it is a crime for an American to sell weapons to hostile nations:
$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$

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

# Example Knowledge Base

... it is a crime for an American to sell weapons to hostile nations:
$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \implies Criminal(x)$

Nono ... has some missiles
i.e., $\exists x \ Owns(Nono, x) \land Missile(x)$:
$Owns(Nono, M_1)$ and $Missile(M_1)$

... all of its missiles were sold to it by Colonel West
$\forall x Missile(x) \land Owns(Nono, x) \implies Sells(West, x, Nono)$

Missiles are weapons:

# Example Knowledge Base

... it is a crime for an American to sell weapons to hostile nations:
$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$

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

Missiles are weapons:
$Missile(x) \Rightarrow Weapon(x)$

# Example Knowledge Base

... it is a crime for an American to sell weapons to hostile nations:
$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$

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

Missiles are weapons:
$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile":

# Example Knowledge Base

... it is a crime for an American to sell weapons to hostile nations:
$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$

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

Missiles are weapons:
$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as “hostile”:
$Enemy(x, America) \implies Hostile(x)$

# Example Knowledge Base

... it is a crime for an American to sell weapons to hostile nations:
$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$

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

Missiles are weapons:
$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile":
$Enemy(x, America) \implies Hostile(x)$

West, who is American ...

... it is a crime for an American to sell weapons to hostile nations:
$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$

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

Missiles are weapons:
$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile":
$Enemy(x, America) \implies Hostile(x)$

West, who is American ...
$American(West)$

... it is a crime for an American to sell weapons to hostile nations:
$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \implies Criminal(x)$

Nono ... has some missiles
i.e., $\exists x \; Owns(Nono, x) \land Missile(x)$:
$Owns(Nono, M_1)$ and $Missile(M_1)$

... all of its missiles were sold to it by Colonel West
$\forall x Missile(x) \land Owns(Nono, x) \implies Sells(West, x, Nono)$

Missiles are weapons:
$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile":
$Enemy(x, America) \implies Hostile(x)$

West, who is American ...
$American(West)$

The country Nono, an enemy of America ...
$Enemy(Nono, America)$

# Example Knowledge Base

... it is a crime for an American to sell weapons to hostile nations:
$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \implies Criminal(x)$

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

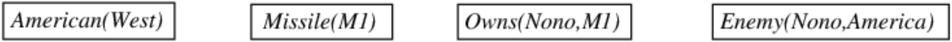
Missiles are weapons:
$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as "hostile":
$Enemy(x, America) \implies Hostile(x)$

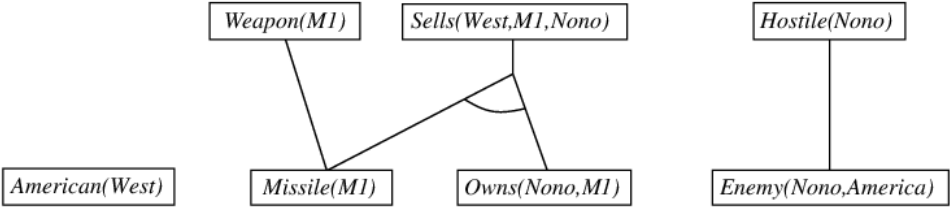
West, who is American ...
$American(West)$

The country Nono, an enemy of America ...
$Enemy(Nono, America)$

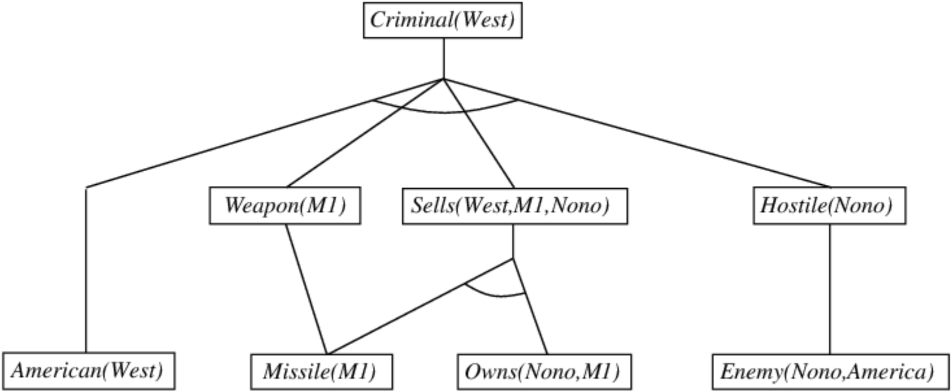
```
function FOL-FC-ASK(KB, α) returns a substitution or false

    repeat until new is empty
        new ← { }
        for each sentence r in KB do
            ( p_1 ∧ . . . ∧ p_n ⟹ q) ← STANDARDIZE-APART(r)
            for each θ such that (p_1 ∧ . . . ∧ p_n)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
```


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
(proof similar to propositional proof)

Datalog = first-order definite clauses + **no functions** (e.g., crime KB)
FC terminates for Datalog in poly iterations: at most $p \cdot n^k$ literals

May not terminate in general if $\alpha$ is not entailed

This is unavoidable: entailment with definite clauses is semidecidable

# Efficiency of Forward Chaining

Simple observation: no need to match a rule on iteration $k$
if a premise wasn't added on iteration $k - 1$
$\Longrightarrow$ match each rule whose premise contains a newly added literal

Matching itself can be expensive

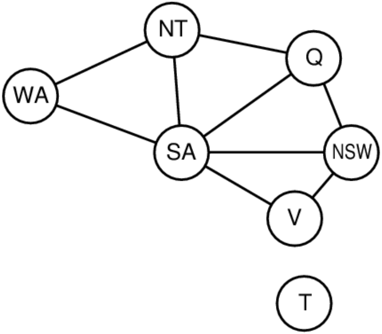
Database indexing allows $O(1)$ retrieval of known facts

e.g., query $Missile(x)$ retrieves $Missile(M_1)$

Matching conjunctive premises against known facts is NP-hard

Forward chaining is widely used in deductive databases

# Hard Matching Example




$Diff(wa, nt) \wedge Diff(wa, sa) \wedge$
$Diff(nt, q) \wedge Diff(nt, sa) \wedge$
$Diff(q, nsw) \wedge Diff(q, sa) \wedge$
$Diff(nsw, v) \wedge Diff(nsw, sa) \wedge$
$Diff(v, sa)$
$\implies Colorable()$

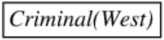
$Diff(Red, Blue)$ $Diff(Red, Green)$
$Diff(Green, Red)$ $Diff(Green, Blue)$
$Diff(Blue, Red)$ $Diff(Blue, Green)$

$Colorable()$ is inferred iff the CSP has a solution

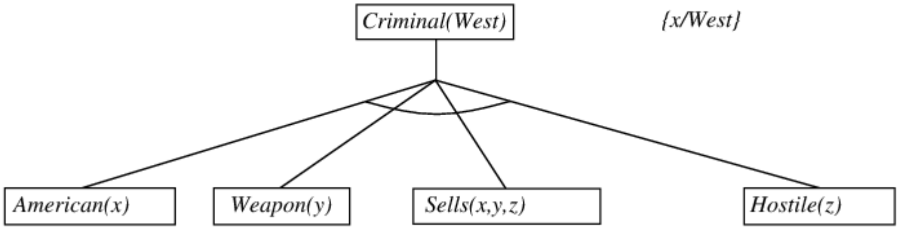
CSPs include 3SAT as a special case, hence matching is NP-hard

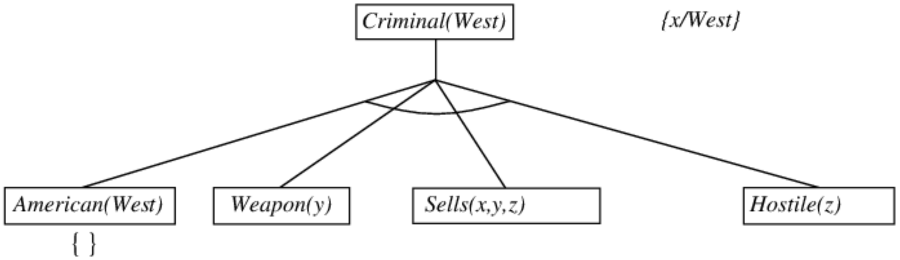
**function** FOL-BC-ASK($KB$, $goals$, $\theta$) **returns** a set of substitutions
**inputs**: $KB$, a knowledge base
$goals$, a list of conjuncts forming a query ($\theta$ already applied)
$\theta$, the current substitution, initially the empty substitution $\{\ \}$
**local variables**: $answers$,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

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

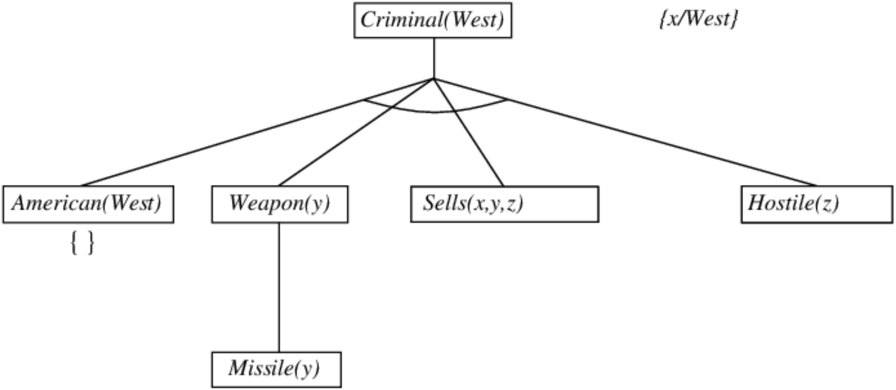

Criminal(West)

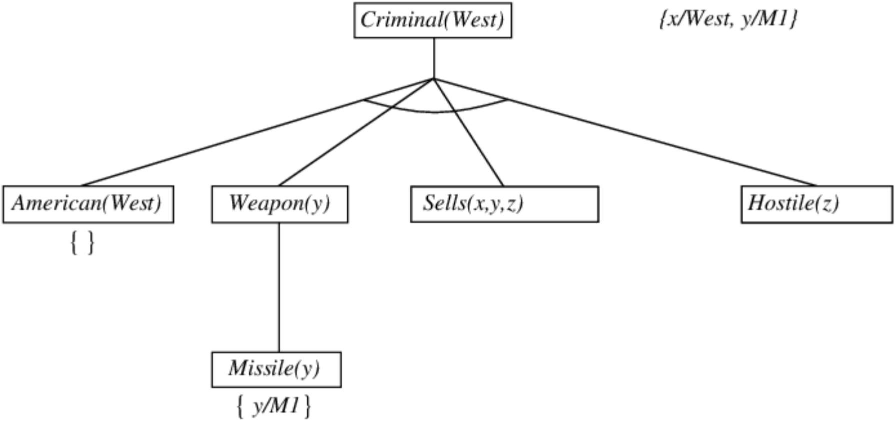
# Backward Chaining Example



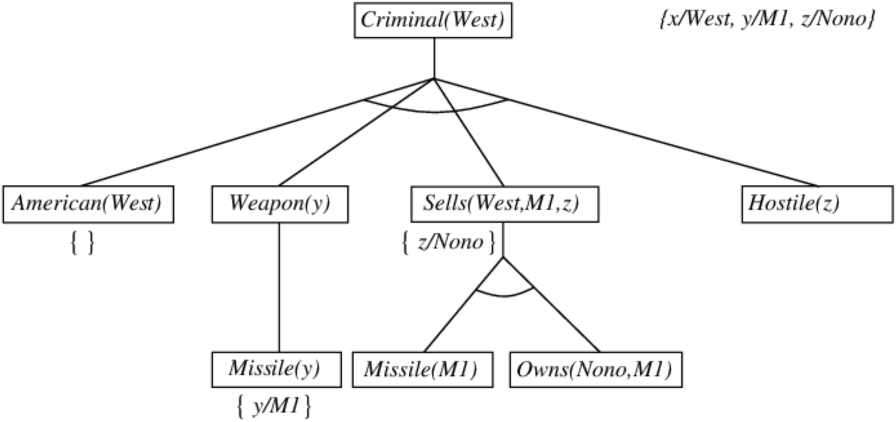

Criminal(West)
{x/West}
American(West)
{ }
Weapon(y)
Sells(x,y,z)
Hostile(z)

# Backward Chaining Example

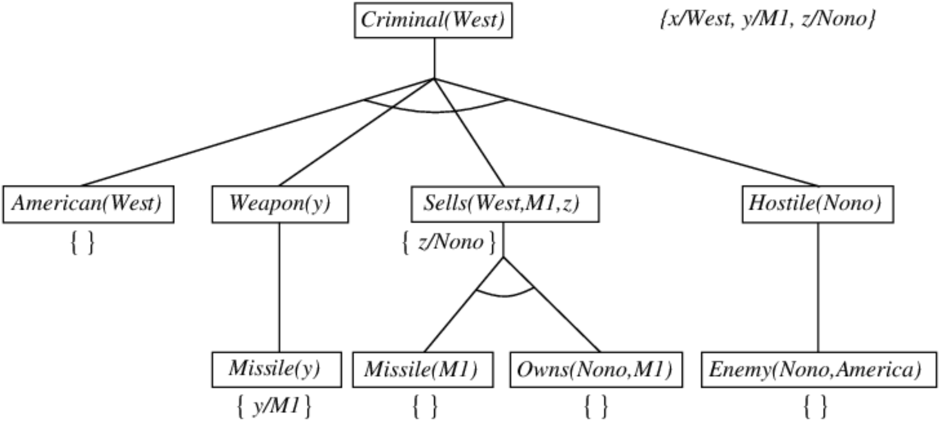



Criminal(West)
{x/West, y/M1}
American(West)
{ }
Weapon(y)
Sells(x,y,z)
Hostile(z)
Missile(y)
{ y/M1 }

# 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
$\Longrightarrow$ fix by checking current goal against every goal on stack

Inefficient due to repeated subgoals (both success and failure)
$\Longrightarrow$ fix using caching of previous results (extra space!)

Widely used (without improvements!) for logic programming

**Sound bite: computation as inference on logical KBs**

| | Logic programming | Ordinary programming |
|---|---|---|
| 1. | Identify problem | Identify problem |
| 2. | Assemble information | Assemble information |
| 3. | Tea break | Figure out solution |
| 4. | Encode information in KB | Program solution |
| 5. | Encode problem instance as facts | Encode problem instance as data |
| 6. | Ask queries | Apply program to data |
| 7. | Find false facts | Debug procedural errors |

Should be easier to debug $Capital(NewYork, US)$ than $x := x + 2$ !

# Prolog Systems

Basis: backward chaining with Horn clauses + bells & whistles
Widely used in Europe, Japan (basis of 5th Generation project)
Compilation techniques ⇒ approaching a billion LIPS

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

```
criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).
```

Efficient unification by open coding
Efficient retrieval of matching clauses by direct linking
Depth-first, left-to-right backward chaining

Built-in predicates for arithmetic etc., e.g., `X is Y*Z+3`
Closed-world assumption ("negation as failure")

e.g., given `alive(X) :- not dead(X).`

`alive(joe)` succeeds if `dead(joe)` fails

Depth-first search from a start state X:

```
dfs(X) :- goal(X).
dfs(X) :- successor(X,S),dfs(S).
```

No need to loop over S: successor succeeds for each

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

For example,

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

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

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

Everyone who loves all animals is loved by someone:
$\forall x[\forall y\, Animal(y) \implies Loves(x,y)] \implies [\exists y\, Loves(y,x)]$

1. Eliminate biconditionals and implications

$$\forall x[\neg\forall y\, \neg Animal(y) \lor Loves(x,y)] \lor [\exists y\, Loves(y,x)]$$

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

$$\forall x[\exists y\, \neg(\neg Animal(y) \lor Loves(x,y))] \lor [\exists y\, Loves(y,x)]$$
$$\forall x[\exists y\, \neg\neg Animal(y) \land \neg Loves(x,y)] \lor [\exists y\, Loves(y,x)]$$
$$\forall x[\exists y\, Animal(y) \land \neg Loves(x,y)] \lor [\exists y\, Loves(y,x)]$$

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

$$\forall x [\exists y \; Animal(y) \land \neg Loves(x, y)] \lor [\exists z \; Loves(z, x)]$$

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

$$\forall x [Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x)$$

5. Drop universal quantifiers:

$$[Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x)$$

6. Distribute $\land$ over $\lor$:

$$[Animal(F(x)) \lor Loves(G(x), x)] \land [\neg Loves(x, F(x)) \lor Loves(G(x), x)]$$

# Resolution Proof: Definite Clauses