Last update: March 30, 2010

Inference in first-order logic

CMSC 421: Chapter 9

Outline

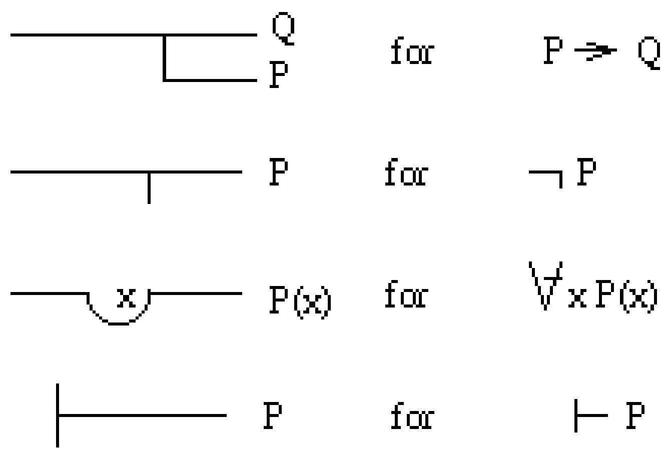
- ♢ Reducing first-order inference to propositional inference
- ♢ Unification
- ♢ Generalized Modus Ponens
- ♢ Forward and backward chaining
- ♢ Logic programming
- ♢ Resolution

A brief history of first-order logic

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

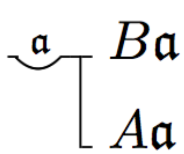
Frege's notation for FOL

In Frege's notation, formulas looked like tree structures. He used



Example: $\forall x(A(x) \rightarrow B(x))$

Frege would have written



Universal instantiation (UI)

Every instantiation of a universally quantified sentence is entailed by it

For every variable v and ground term g, if θ is the substitution $\{v \leftarrow g\}$ then

$$\frac{\forall\, v \;\; \alpha}{\alpha\, \theta}$$

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 doesn't appear elsewhere in the knowledge base**, if $\theta = \{v \leftarrow k\}$ then

$$\frac{\exists\, v \;\; \alpha}{\alpha\, \theta}$$

E.g., $\exists\, x \;\; Crown(x) \wedge OnHead(x, John)$ yields

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

where C_1 is a new constant symbol (i.e., doesn't already appear somewhere)

In words:

If there is a crown on John's head, then we can call the crown C_1

C_1 is called a *Skolem constant*

Existential instantiation, 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

Mathematicians use these techniques informally every day.
Example: proofs involving limits

Given $\lim_{x \to 5} f(x) = 2$, i.e.,
$\forall \epsilon > 0 \; \exists \delta > 0 \; \forall x \; |x - 5| < \delta, \; |f(x) - 2| < \epsilon$.

Let ϵ be any number > 0.
Then $\exists \delta > 0 \; \forall x \; |x - 5| < \delta, \; |f(x) - 2| < \epsilon$.

Let $\delta_1 > 0$ be such that $\forall x \; |x - 5| < \delta, \; |f(x) - 2| < \epsilon$.

Let x be any number such that $|x - 5| < \delta_1$. Then $|f(x) - 2| < \epsilon$.

. . .

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

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

The new KB is *propositionalized*: proposition symbols are

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

Reduction, continued

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 1: propositionalization can create lots of irrelevant sentences.
E.g., suppose we are given

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

To prove $Evil(John)$, we first use propositionalization to get $Greedy(John)$

But propositionalization also produces $Greedy(Richard)$ and $Greedy(Joanna)$

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

Reduction, continued

Problem 2: with function symbols, propositionalization can create infinitely many sentences!

$Greedy(John)$
$Greedy(father(John))$
$Greedy(father(father(John)))$
$\ldots$

Theorem: Herbrand (1930). If a sentence α is entailed by an FOL KB, then it is entailed by a **finite** subset of the propositionalized KB

Idea: For $n = 0$ to ∞ do
  create a propositional KB by instantiating with all terms of depth $\leq n$
    (e.g., up to n nested occurrences of $Father$)
  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 in FOL is *semidecidable*

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 $Greedy(y)$

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

A *unifier* for α and β is a substitution θ such that $\alpha\theta = \beta\theta$
  α and β are *unifiable* if such a θ exists

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	
$Knows(John, x)$	$Knows(y, Joanna)$	
$Knows(John, x)$	$Knows(y, mother(y))$	
$Knows(John, x)$	$Knows(x, Joanna)$	
$Knows(John, x)$	$Knows(x_{17}, Joanna)$	
$Knows(x, x)$	$Knows(z, mother(z))$	

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 $Greedy(y)$

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

A *unifier* for α and β is a substitution θ such that $\alpha\theta = \beta\theta$

α and β are *unifiable* if such a θ exists

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x \leftarrow Jane\}$
$Knows(John, x)$	$Knows(y, Joanna)$	
$Knows(John, x)$	$Knows(y, mother(y))$	
$Knows(John, x)$	$Knows(x, Joanna)$	
$Knows(John, x)$	$Knows(x_{17}, Joanna)$	
$Knows(x, x)$	$Knows(z, mother(z))$	

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 $Greedy(y)$

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

A *unifier* for α and β is a substitution θ such that $\alpha\theta = \beta\theta$

α and β are *unifiable* if such a θ exists

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x \leftarrow Jane\}$
$Knows(John, x)$	$Knows(y, Joanna)$	$\{x \leftarrow Joanna, y \leftarrow John\}$
$Knows(John, x)$	$Knows(y, mother(y))$	
$Knows(John, x)$	$Knows(x, Joanna)$	
$Knows(John, x)$	$Knows(x_{17}, Joanna)$	
$Knows(x, x)$	$Knows(z, mother(z))$	

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 $Greedy(y)$

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

A *unifier* for α and β is a substitution θ such that $\alpha\theta = \beta\theta$
α and β are *unifiable* if such a θ exists

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x \leftarrow Jane\}$
$Knows(John, x)$	$Knows(y, Joanna)$	$\{x \leftarrow Joanna, y \leftarrow John\}$
$Knows(John, x)$	$Knows(y, mother(y))$	$\{y \leftarrow John, x \leftarrow mother(John)\}$
$Knows(John, x)$	$Knows(x, Joanna)$	
$Knows(John, x)$	$Knows(x_{17}, Joanna)$	
$Knows(x, x)$	$Knows(z, mother(z))$	

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 $Greedy(y)$

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

A *unifier* for α and β is a substitution θ such that $\alpha\theta = \beta\theta$
α and β are *unifiable* if such a θ exists

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x \leftarrow Jane\}$
$Knows(John, x)$	$Knows(y, Joanna)$	$\{x \leftarrow Joanna, y \leftarrow John\}$
$Knows(John, x)$	$Knows(y, mother(y))$	$\{y \leftarrow John, x \leftarrow mother(John)\}$
$Knows(John, x)$	$Knows(x, Joanna)$	$fail$
$Knows(John, x)$	$Knows(x_{17}, Joanna)$	
$Knows(x, x)$	$Knows(z, mother(z))$	

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 $Greedy(y)$

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

A *unifier* for α and β is a substitution θ such that $\alpha\theta = \beta\theta$
α and β are *unifiable* if such a θ exists

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x \leftarrow Jane\}$
$Knows(John, x)$	$Knows(y, Joanna)$	$\{x \leftarrow Joanna, y \leftarrow John\}$
$Knows(John, x)$	$Knows(y, mother(y))$	$\{y \leftarrow John, x \leftarrow mother(John)\}$
$Knows(John, x)$	$Knows(x, Joanna)$	$fail$
$Knows(John, x)$	$Knows(x_{17}, Joanna)$	$\{x_{17} \leftarrow John, x \leftarrow Joanna\}$
$Knows(x, x)$	$Knows(z, mother(z))$	

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

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 $Greedy(y)$

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

A *unifier* for α and β is a substitution θ such that $\alpha\theta = \beta\theta$
α and β are *unifiable* if such a θ exists

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x \leftarrow Jane\}$
$Knows(John, x)$	$Knows(y, Joanna)$	$\{x \leftarrow Joanna, y \leftarrow John\}$
$Knows(John, x)$	$Knows(y, mother(y))$	$\{y \leftarrow John, x \leftarrow mother(John)\}$
$Knows(John, x)$	$Knows(x, Joanna)$	$fail$
$Knows(John, x)$	$Knows(x_{17}, Joanna)$	$\{x_{17} \leftarrow John, x \leftarrow Joanna\}$
$Knows(x, x)$	$Knows(z, mother(z))$	$fail$

Standardizing apart eliminates overlap of variables, e.g., $Knows(x_{17}, Joanna)$
Can't unify a variable with a term that contains the variable

Unification (continued)

A *most general unifier (mgu)* for α and β is a substitution θ such that
(1) θ is a unifier for α and β;
(2) for every unifier θ' of α and β and for every expression e,
$e\theta'$ is a substitution instance of $e\theta$

E.g., let $\alpha = Knows(w, father(x))$ and $\beta = Knows(mother(y), y)$

$\theta_1 = \{w \leftarrow mother(father(x)), y \leftarrow father(x)\}$ is an mgu

$\theta_2 = \{w \leftarrow mother(father(v)), y \leftarrow father(v), x \leftarrow v\}$ is an mgu

$\theta_3 = \{w \leftarrow mother(father(John)), y \leftarrow father(John)\}$
is a unifier but it is not an mgu

If θ and θ' are mgus for α and β, then they are identical except for renaming of variables

Algorithm to find an mgu

Compare the expressions element by element, building up a substitution along the way. Here's the basic idea (the book gives additional details):

For each pair of corresponding elements:
  Apply the substitution we've built so far
  If the two elements are the same after substituting, keep going
  Else if one of them is a variable x and the other is an expression e,
    and if x doesn't appear anywhere in e (the "occur check")
    then incorporate $x = e$ into the substitution
  Else FAIL

$$
\begin{array}{llll}
Knows(& John, & x &) \\
\quad\updownarrow & \quad\updownarrow & \quad\updownarrow & \\
Knows(& y, & mother(y) &) \\
\theta = \{\,\} & \theta = \{y \leftarrow John\} & \theta = \{y \leftarrow John, x \leftarrow mother(John)\} &
\end{array}
$$

Runs in quadratic time (would be linear time if it weren't for the occur check)

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 a substitution such that $p_i'\theta = p_i\theta$ for all i,
and all variables are assumed to be universally quantified.
Example:

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

with $\theta = \{x \leftarrow John, y \leftarrow John\}$, $q\theta = Evil(x)\theta = Evil(John)$

Equivalent formulation using *definite clauses* (**exactly** one positive literal)

$$\frac{p_1{}',\;\; p_2{}',\; \ldots,\; p_n{}',\;\; (\neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_n \vee q)}{q\theta}$$

Soundness of GMP

Need to show that

$$p_1', \ldots, p_n', \ (p_1 \land \ldots \land p_n \Rightarrow q) \ \models \ q\theta$$

provided that $p_i'\theta = p_i\theta$ for all i

We know that for any definite clause p, universal instantiation gives us $p \ \models \ p\theta$. Thus

1. $(p_1 \land \ldots \land p_n \Rightarrow q) \ \models \ (p_1 \land \ldots \land p_n \Rightarrow q)\theta = (p_1\theta \land \ldots \land p_n\theta \Rightarrow q\theta)$

2. $p_1', \ldots, p_n' \ \models \ p_1' \land \ldots \land p_n' \ \models \ p_1'\theta \land \ldots \land p_n'\theta$

If $p_i'\theta = p_i\theta$ for all i, then $q\theta$ follows from 1 and 2 and ordinary Modus Ponens

Example knowledge base

The law says it is a crime for an American to sell weapons to hostile nations.
The country Nono, an enemy of America, has some missiles.
All of its missiles were sold to it by Colonel West, who is American.

Prove one of the following:

1. Russell & Norvig have a sense of humor
2. Col. West is a criminal

Example knowledge base, continued

The law says it is a crime for an American to sell weapons to hostile nations.
The country Nono, an enemy of America, has some missiles.
All of its missiles were sold to it by Colonel West, who is American.

$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$

Example knowledge base, continued

The law says it is a crime for an American to sell weapons to hostile nations.
The country Nono, an enemy of America, has some missiles.
All of its missiles were sold to it by Colonel West, who is American.

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

$\exists\, x\ Owns(Nono, x) \wedge Missile(x)$

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

Example knowledge base, continued

The law says it is a crime for an American to sell weapons to hostile nations.
The country Nono, an enemy of America, has some missiles.
All of its missiles were sold to it by Colonel West, who is American.

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

$\exists\, x\; Owns(Nono, x) \wedge Missile(x)$

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

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

Example knowledge base, continued

The law says it is a crime for an American to sell weapons to hostile nations.
The country Nono, an enemy of America, has some missiles.
All of its missiles were sold to it by Colonel West, who is American.

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

$\exists\, x\ Owns(Nono, x) \wedge Missile(x)$

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

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

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

Example knowledge base, continued

The law says it is a crime for an American to sell weapons to hostile nations.
The country Nono, an enemy of America, has some missiles.
All of its missiles were sold to it by Colonel West, who is American.

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

$\exists\, x\ Owns(Nono, x) \wedge Missile(x)$

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

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

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

$Enemy(x, America)\ \Rightarrow\ Hostile(x)$ An enemy of America is "hostile"

Example knowledge base, continued

The law says it is a crime for an American to sell weapons to hostile nations.
The country Nono, an enemy of America, has some missiles.
All of its missiles were sold to it by Colonel West, who is American.

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

$\exists\, x\ Owns(Nono, x) \wedge Missile(x)$

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

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

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

$Enemy(x, America)\ \Rightarrow\ Hostile(x)$ An enemy of America is "hostile"

$American(West)$

Example knowledge base, continued

The law says it is a crime for an American to sell weapons to hostile nations.
The country Nono, an enemy of America, has some missiles.
All of its missiles were sold to it by Colonel West, who is American.

$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)$

$\exists\, x\ Owns(Nono, x) \land Missile(x)$

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

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

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

$Enemy(x, America)\ \Rightarrow\ Hostile(x)$ An enemy of America is "hostile"

$American(West)$

$Enemy(Nono, America)$

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)

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

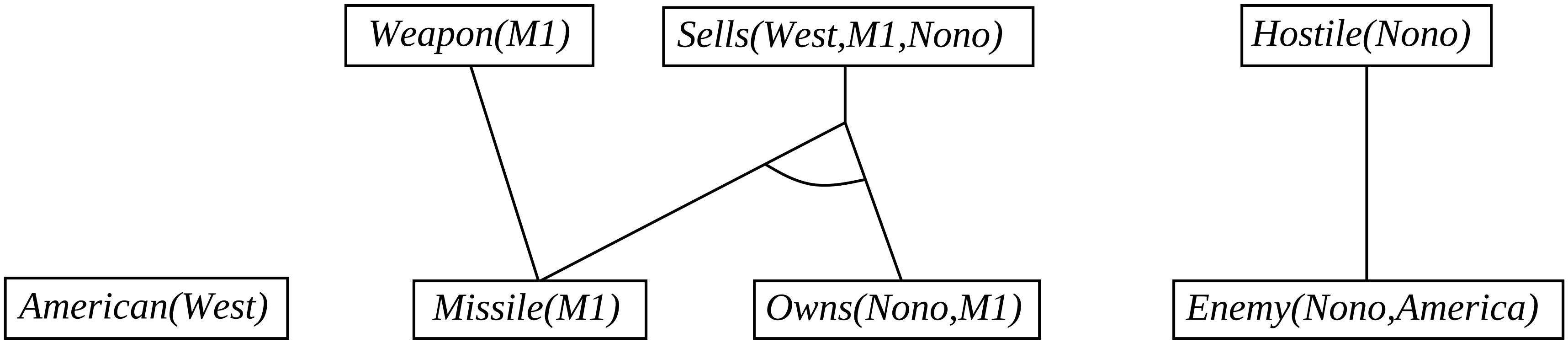
$\forall\, x \;\; Missile(x) \wedge Owns(Nono, x) \;\Rightarrow\; Sells(West, x, Nono)$

$Owns(Nono, M_1)$ $\qquad$ $Missile(M_1)$

$Missile(x) \Rightarrow Weapon(x)$ $\qquad$ $Enemy(x, America) \;\Rightarrow\; Hostile(x)$

$American(West)$ $\qquad$ $Enemy(Nono, America)$

Forward chaining proof



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

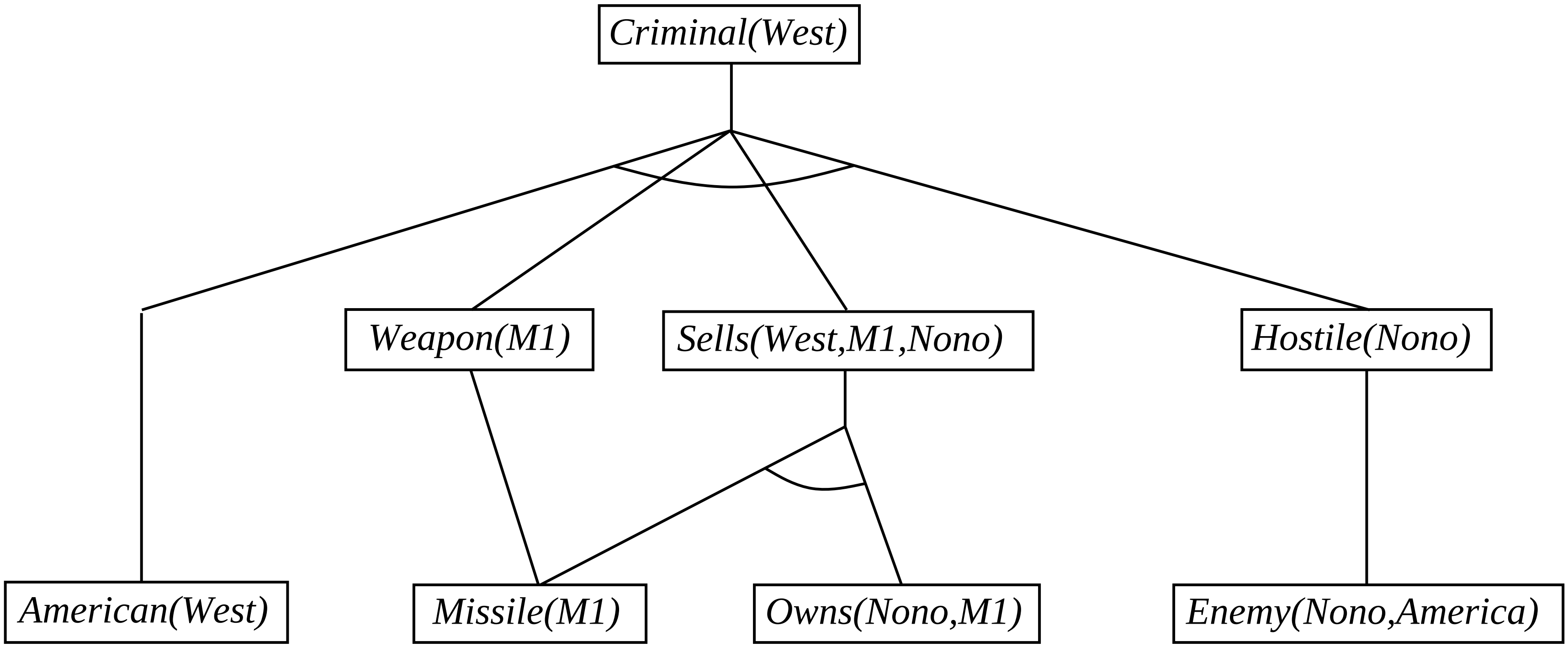
$\forall x \;\; Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

$Owns(Nono, M_1)$ $\quad$ $Missile(M_1)$

$Missile(x) \Rightarrow Weapon(x)$ $\quad$ $Enemy(x, America) \Rightarrow Hostile(x)$

$American(West)$ $\quad$ $Enemy(Nono, America)$

Forward chaining proof



$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \;\Rightarrow\; Criminal(x)$

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

$Owns(Nono, M_1)$ $\qquad$ $Missile(M_1)$

$Missile(x) \Rightarrow Weapon(x)$ $\qquad$ $Enemy(x, America) \;\Rightarrow\; Hostile(x)$

$American(West)$ $\qquad$ $Enemy(Nono, America)$

Properties of forward chaining

Sound and complete for first-order definite clauses
(proof similar to propositional proof)

May not terminate in general if α is not entailed

This is unavoidable: entailment with definite clauses is semidecidable
(i.e., equivalent to the halting problem)

Can guarantee termination if restrictions are satisfied, e.g.,

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

Efficiency of forward chaining

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

Matching itself can be expensive

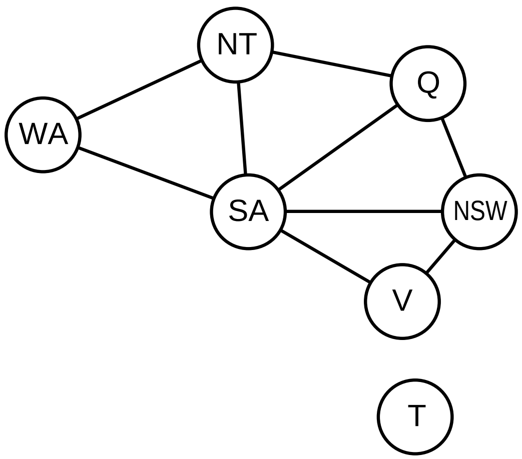
◇ *Database indexing* allows $O(1)$ retrieval of known facts
e.g., query $Missile(x)$ retrieves $Missile(M_1)$

◇ But matching conjunctive premises against known facts is NP-hard
(see next page)

◇ Partial fix: store partial matches in data structures such as *rete networks*

Forward chaining is widely used in *deductive databases* and *expert systems*

Hard matching example



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

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

Don't need statements like $nt = Red \vee nt = Blue \vee nt = Green$. *Why?*

$Colorable()$ is inferred iff the CSP has a solution
Need to try many combinations of variable values

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

Backward chaining algorithm

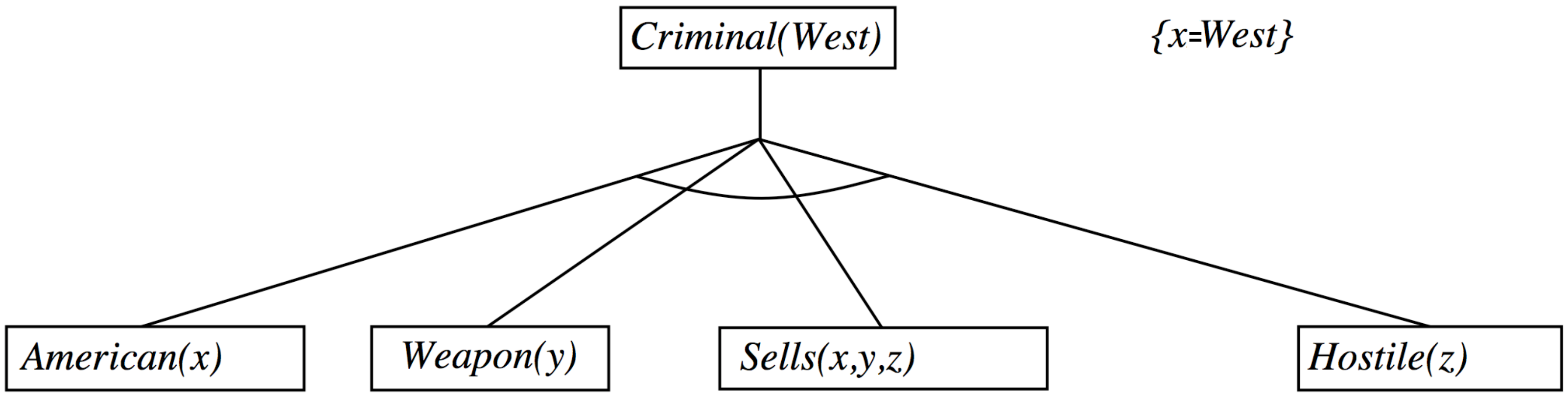
```
function FOL-BC-Ask(KB, goals, θ) returns a set of substitutions
   inputs: KB, a knowledge base
           goals, a list of conjuncts forming a query (θ already applied)
           θ, the current substitution, initially the empty substitution { }
   local variables: answers, a set of substitutions,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

   if goals is empty then return {θ}
   q′ ← Subst(θ, First(goals))
   for each sentence r in KB
           where Standardize-Apart(r) = ( p1 ∧ ... ∧ pn ⇒ q)
           and θ′ ← Unify(q, q′) succeeds
       new_goals ← [ p1, ..., pn|Rest(goals)]
       answers ← FOL-BC-Ask(KB, new_goals, Compose(θ′, θ)) ∪ answers
   return answers
```

Backward chaining example

Criminal(West)

$American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \;\Rightarrow\; Criminal(x)$
$\forall\, x \;\; Missile(x) \land Owns(Nono,x) \;\Rightarrow\; Sells(West,x,Nono)$
$Owns(Nono,M_1)$ $\quad$ $Missile(M_1)$
$Missile(x) \Rightarrow Weapon(x)$ $\quad$ $Enemy(x,America) \;\Rightarrow\; Hostile(x)$
$American(West)$ $\quad$ $Enemy(Nono,America)$

Backward chaining example



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

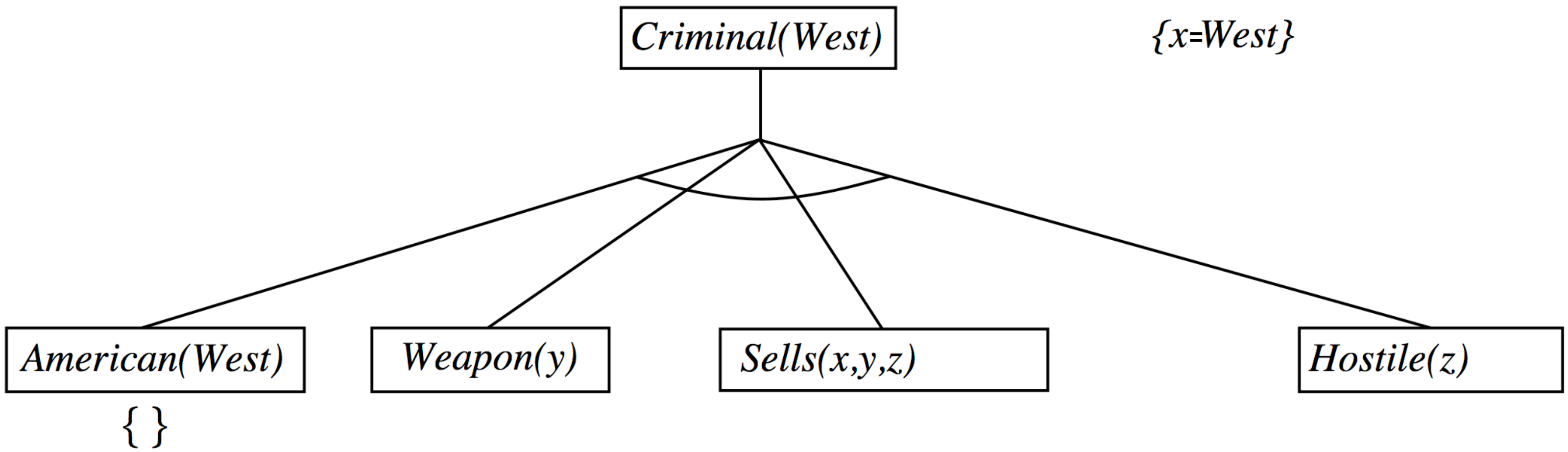
$\forall\, x \;\; Missile(x) \wedge Owns(Nono, x) \;\Rightarrow\; Sells(West, x, Nono)$

$Owns(Nono, M_1)$ $\qquad$ $Missile(M_1)$

$Missile(x) \Rightarrow Weapon(x)$ $\qquad$ $Enemy(x, America) \;\Rightarrow\; Hostile(x)$

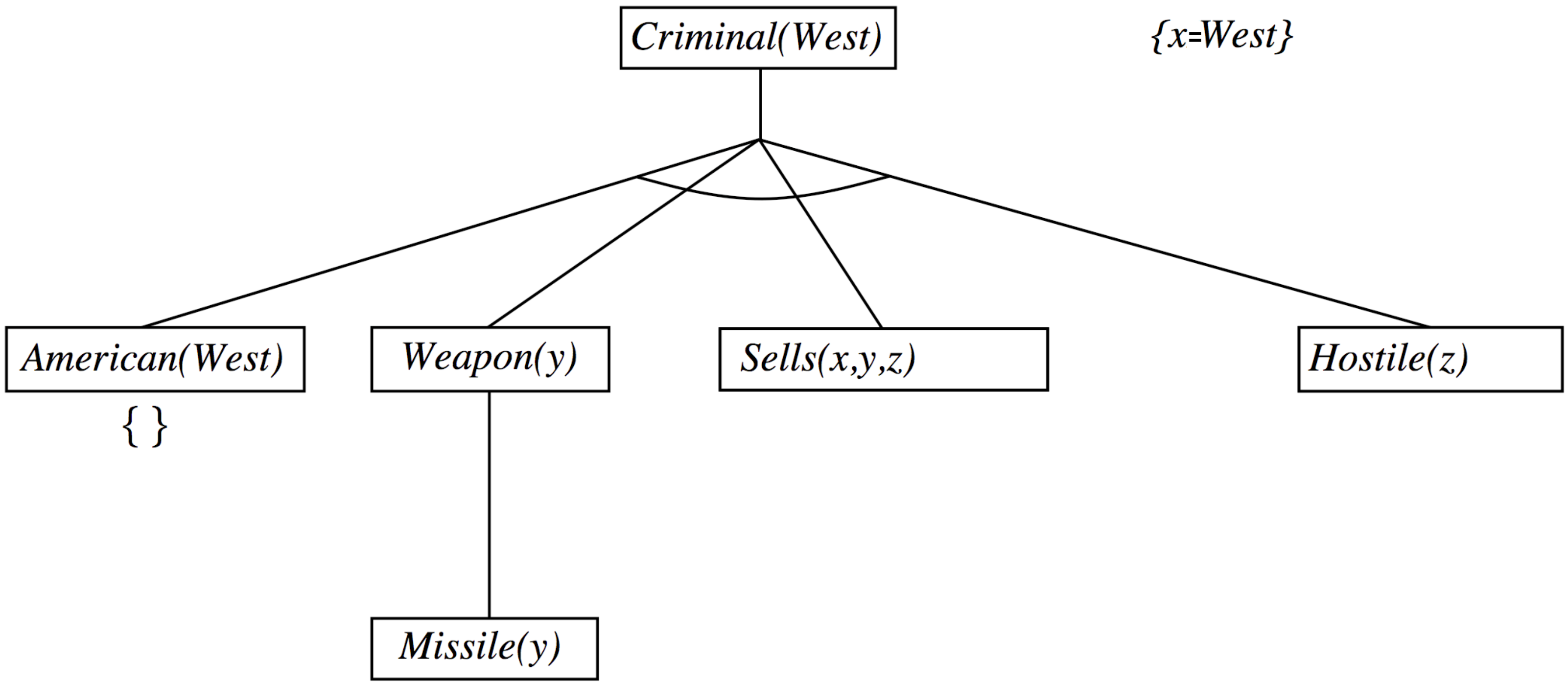
$American(West)$ $\qquad$ $Enemy(Nono, America)$

Backward chaining example



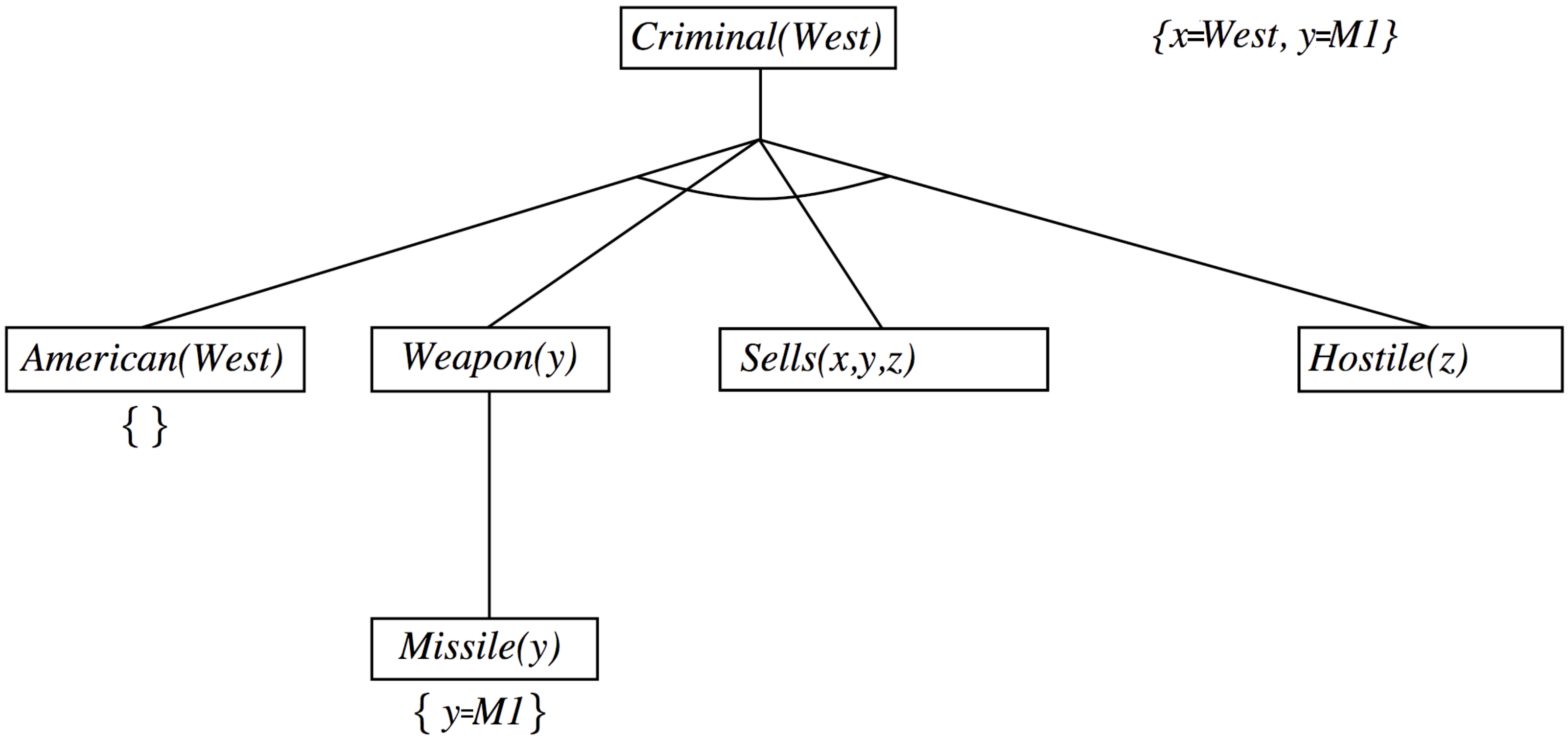
$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \;\Rightarrow\; Criminal(x)$
$\forall\, x \;\; Missile(x) \wedge Owns(Nono, x) \;\Rightarrow\; Sells(West, x, Nono)$
$Owns(Nono, M_1)$ $\qquad Missile(M_1)$
$Missile(x) \Rightarrow Weapon(x)$ $\qquad Enemy(x, America) \;\Rightarrow\;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

Backward chaining example



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

Backward chaining example



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

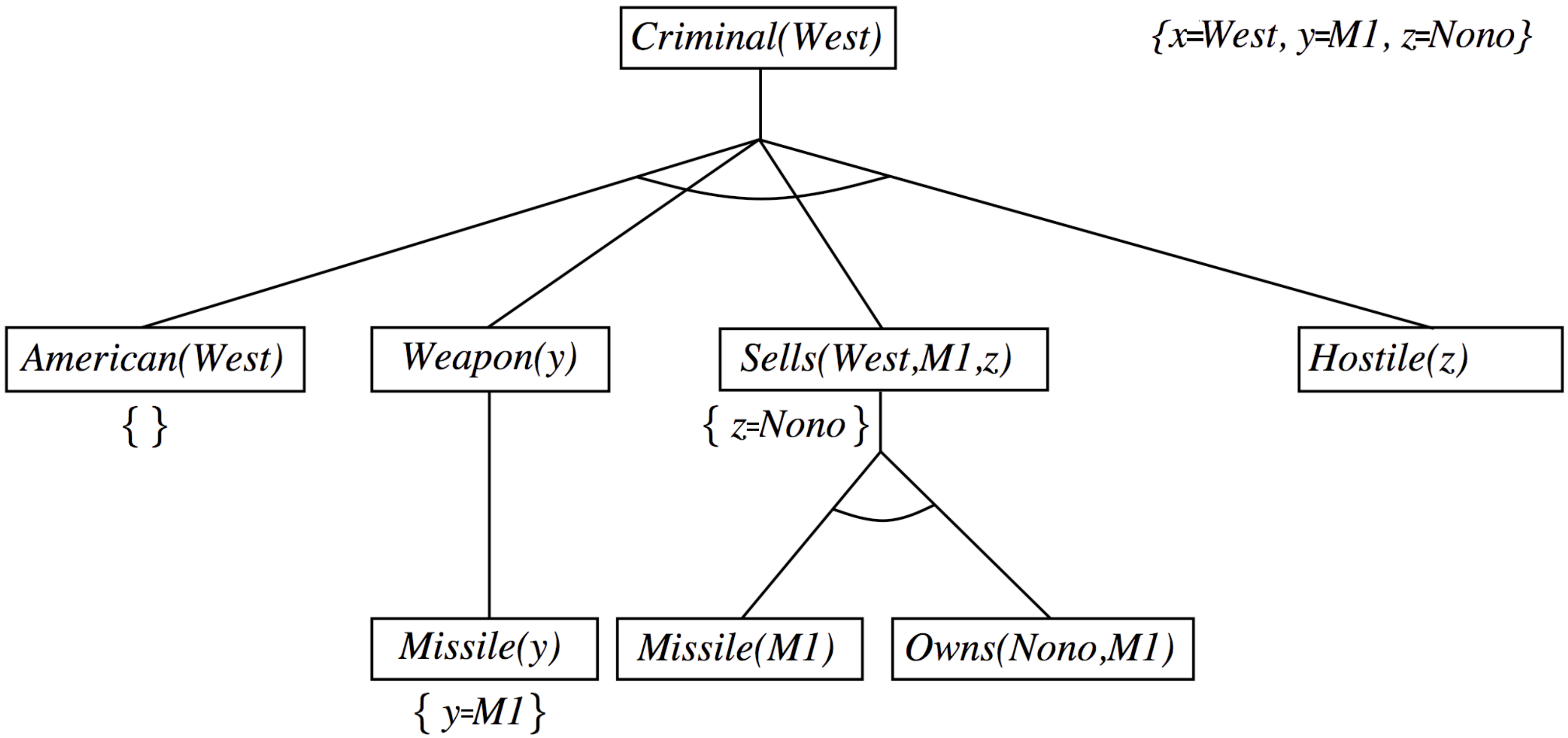
$\forall\, x \;\; Missile(x) \wedge Owns(Nono, x) \;\Rightarrow\; Sells(West, x, Nono)$

$Owns(Nono, M_1)$ $Missile(M_1)$

$Missile(x) \Rightarrow Weapon(x)$ $Enemy(x, America) \;\Rightarrow\; Hostile(x)$

$American(West)$ $Enemy(Nono, America)$

Backward chaining example



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

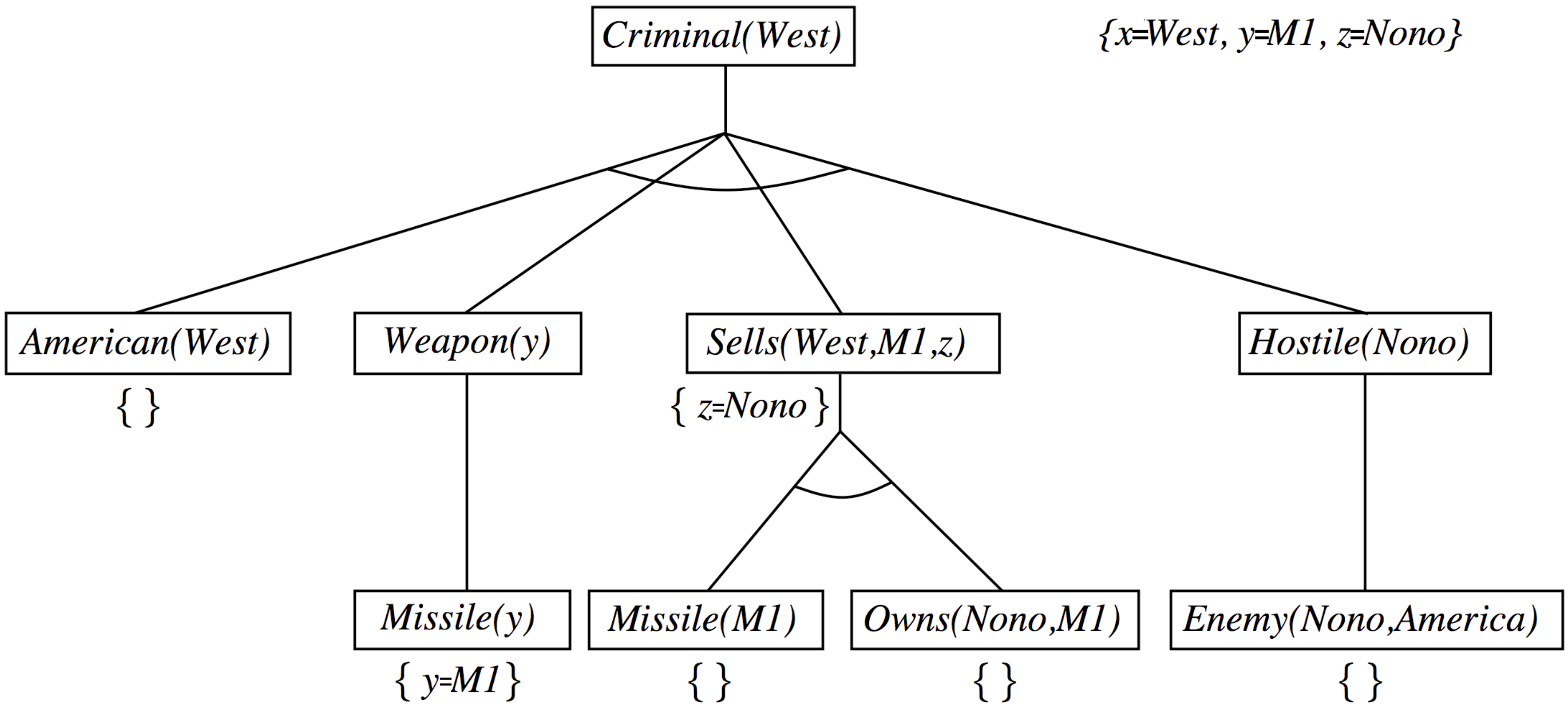
$\forall\, x \;\; Missile(x) \wedge Owns(Nono,x) \;\Rightarrow\; Sells(West,x,Nono)$

$Owns(Nono,M_1)$ $\quad$ $Missile(M_1)$

$Missile(x) \Rightarrow Weapon(x)$ $\quad$ $Enemy(x,America) \;\Rightarrow\; Hostile(x)$

$American(West)$ $\quad$ $Enemy(Nono,America)$

Backward chaining example



$American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \;\Rightarrow\; Criminal(x)$

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

$Owns(Nono,M_1)$ $\qquad$ $Missile(M_1)$

$Missile(x) \Rightarrow Weapon(x)$ $\qquad$ $Enemy(x,America) \;\Rightarrow\; Hostile(x)$

$American(West)$ $\qquad$ $Enemy(Nono,America)$

Properties of backward chaining

◇ Depth-first recursive proof search: space is linear in size of proof

◇ Incomplete due to infinite loops

Partial fix: check current goal against every goal on stack

This prevents looping here:
$P(x) \Rightarrow P(x)$

But it doesn't prevent looping here:
$Q(f(x)) \Rightarrow Q(x)$

◇ Inefficient due to repeated subgoals (both success and failure)
Fix using caching of previous results (extra space!)

◇ Widely used (without improvements!) for *logic programming*

Prolog systems

Basis: backward chaining with Horn clauses
+ extras (e.g., built-in "predicates" that do arithmetic, printing, etc.)

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

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

Capitalized words (e.g., `X`) are variables, and
lower-case words (e.g., `nono`) are constants
this is the opposite of what we've been doing

Depth-first, left-to-right backward chaining
Closed-world assumption ("negation as failure")
e.g., given `alive(X) :- not dead(X).`
`alive(joe)` succeeds if `dead(joe)` fails

Compilation techniques ⇒ approaching a billion LIPS
Efficient unification by *open coding* (generate unification code inline)
Efficient retrieval of matching clauses by direct linking

Prolog examples

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 successor of `X`

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 in FOL

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

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

For example,

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

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

To prove that $KB \models$ an instance of α, convert $KB \wedge \neg\alpha$ to CNF and do resolution repeatedly

This is a complete proof procedure for FOL

- If there's a substitution θ such that $KB \models \theta\alpha$, then it will return θ
- If there's no such θ, then the procedure won't necessarily terminate

Conversion to CNF

Everyone who loves all animals is loved by someone:

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

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: $\neg\forall\, x, p\ \ \equiv \exists\, x\ \ \neg p$, $\neg\exists\, x, p\ \ \equiv \forall\, x\ \ \neg p$:

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

Conversion to CNF, continued

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:

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

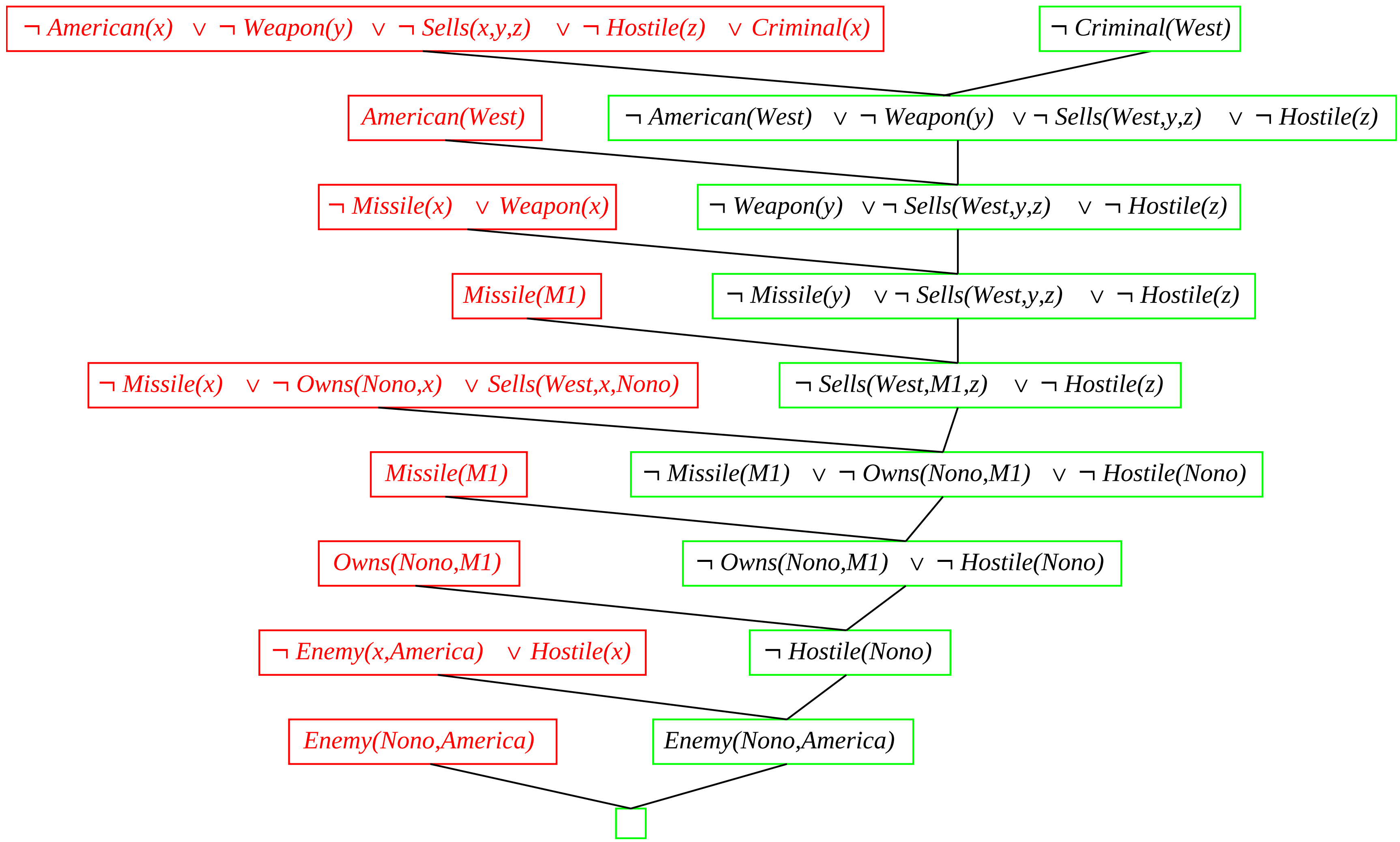
5. Drop universal quantifiers:

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

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



The figure omits all resolvents except for the ones in the proof

Homework

Problems 20.11, 20.17, 9.12 and 9.19

10 points each, 40 points total

Due in one week (i.e., April 6)