Inference in First-Order Logic

Philipp Koehn

5 March 2024

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 systems
1960	Davis/Putnam	"practical" algorithm for propositional logic
1965	Robinson	"practical" algorithm for FOL—resolution

The Story So Far

- Propositional logic
- Subset of propositional logic: horn clauses
- Inference algorithms
  - forward chaining
  - backward chaining
  - resolution (for full propositional logic)
- First order logic (FOL)
  - variables
  - functions
  - quantifiers
  - etc.
- Today: inference for first order logic

Outline

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

reduction to propositional inference

Universal Instantiation

- 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., $\forall\, x \;\; King(x) \land Greedy(x) \implies Evil(x)$ yields

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

Existential Instantiation

- For any sentence α, 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**

Instantiation

- Universal Instantiation
  - can be applied several times to **add** new sentences
  - the new KB is logically equivalent to the old

- Existential Instantiation
  - 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:

$$\begin{array}{l} \forall\, x \;\; King(x) \land Greedy(x) \implies Evil(x) \\ King(John) \\ Greedy(John) \\ Brother(Richard, John) \end{array}$$

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

$$\begin{array}{l} King(John) \land Greedy(John) \implies Evil(John) \\ King(Richard) \land Greedy(Richard) \implies Evil(Richard) \\ King(John) \\ Greedy(John) \\ Brother(Richard, John) \end{array}$$

- The new KB is propositionalized: proposition symbols are

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

Reduction to Propositional Inference

- 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 α is entailed by an 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
- Theorem: Turing (1936), Church (1936), entailment in FOL is semidecidable

Practical Problems with Propositionalization

- Propositionalization seems to generate lots of irrelevant sentences.

- E.g., from

$$\begin{array}{l} \forall\, x \;\; King(x) \land Greedy(x) \implies Evil(x) \\ King(John) \\ \forall\, y \;\; Greedy(y) \\ Brother(Richard, John) \end{array}$$

  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!

unification

Plan

- We have the inference rule
  - $\forall x \ King(x) \land Greedy(x) \implies Evil(x)$
- We have facts that (partially) match the precondition
  - $King(John)$
  - $\forall y \ Greedy(y)$
- We need to match them up with substitutions: $\theta = \{x/John, y/John\}$ works
  - unification
  - generalized modus ponens

Unification

- UNIFY$(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	
$Knows(John, x)$	$Knows(y, Mary)$	
$Knows(John, x)$	$Knows(y, Mother(y))$	
$Knows(John, x)$	$Knows(x, Mary)$	

Unification

- $\text{UNIFY}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, Mary)$	
$Knows(John, x)$	$Knows(y, Mother(y))$	
$Knows(John, x)$	$Knows(x, Mary)$	

Unification

- $\text{UNIFY}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, Mary)$	$\{x/Mary, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	
$Knows(John, x)$	$Knows(x, Mary)$	

Unification

- $\text{UNIFY}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, Mary)$	$\{x/Mary, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, Mary)$	

Unification

- $\text{UNIFY}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, Mary)$	$\{x/Mary, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, Mary)$	$fail$

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

$Knows(John, x)$	$Knows(z_{17}, Mary)$	$\{z_{17}/John, x/Mary\}$

generalized modus ponens

Generalized Modus Ponens

- Generalized modus ponens used with KB of **definite clauses** (exactly one positive literal)
- All variables assumed universally quantified

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

- Rule: $King(x) \wedge Greedy(x) \implies Evil(x)$
- Precondition of rule: p_1 is $King(x)$ p_2 is $Greedy(x)$
- Implication: q is $Evil(x)$
- Facts: $p_1{}'$ is $King(John)$ $p_2{}'$ is $Greedy(y)$
- Substitution: θ is $\{x/John, y/John\}$

$\Rightarrow$ Result of modus ponens: $q\theta$ is $Evil(John)$

forward chaining

Example Knowledge

- *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

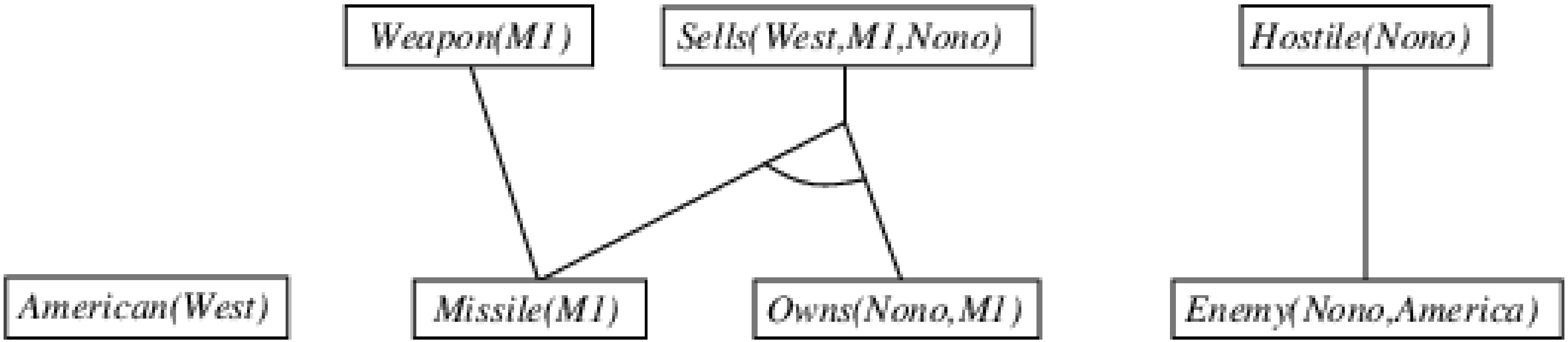
Example Knowledge Base

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

Forward Chaining Proof

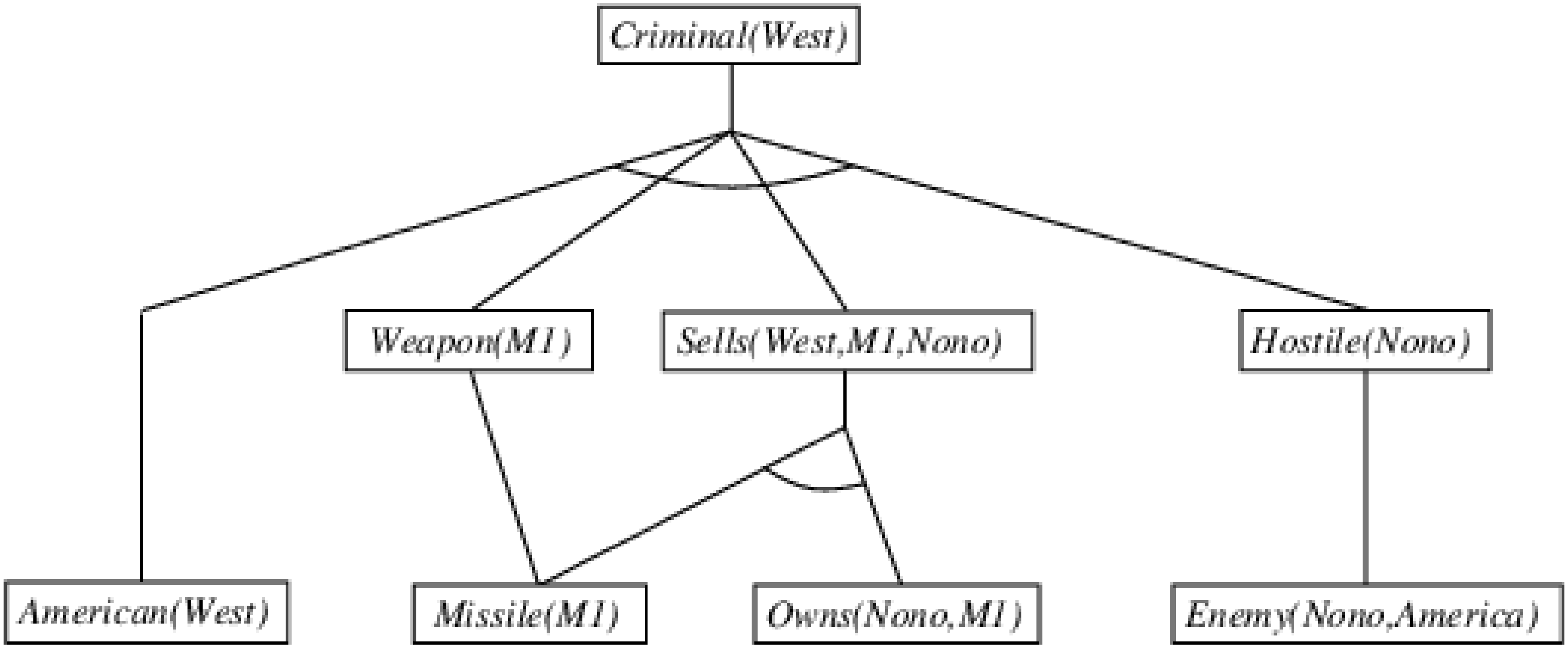
$American(West)$ $\quad$ $Missile(M1)$ $\quad$ $Owns(Nono,M1)$ $\quad$ $Enemy(Nono,America)$

Forward Chaining Proof



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

Forward Chaining Proof



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

Properties of Forward Chaining

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

- **Datalog** (1977) = first-order definite clauses + no functions (e.g., crime example)
  Forward chaining terminates for Datalog in poly iterations: at most $p \cdot n^k$ literals

- May not terminate in general if α 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

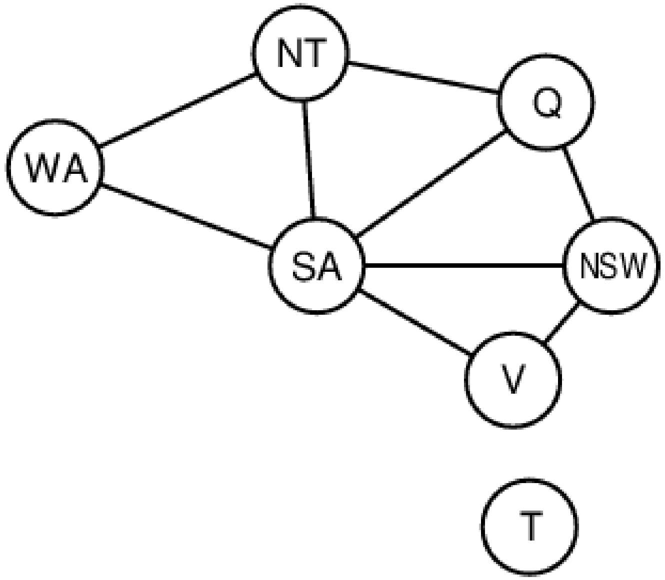
- Matching itself can be expensive

- 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



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

$$\textit{Diff}(Red, Blue) \quad \textit{Diff}(Red, Green)$$
$$\textit{Diff}(Green, Red) \quad \textit{Diff}(Green, Blue)$$
$$\textit{Diff}(Blue, Red) \quad \textit{Diff}(Blue, Green)$$

- $Colorable()$ is inferred iff the constraint satisfaction problem has a solution
- CSPs include 3SAT as a special case, hence matching is NP-hard

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
```

backward chaining

Backward Chaining

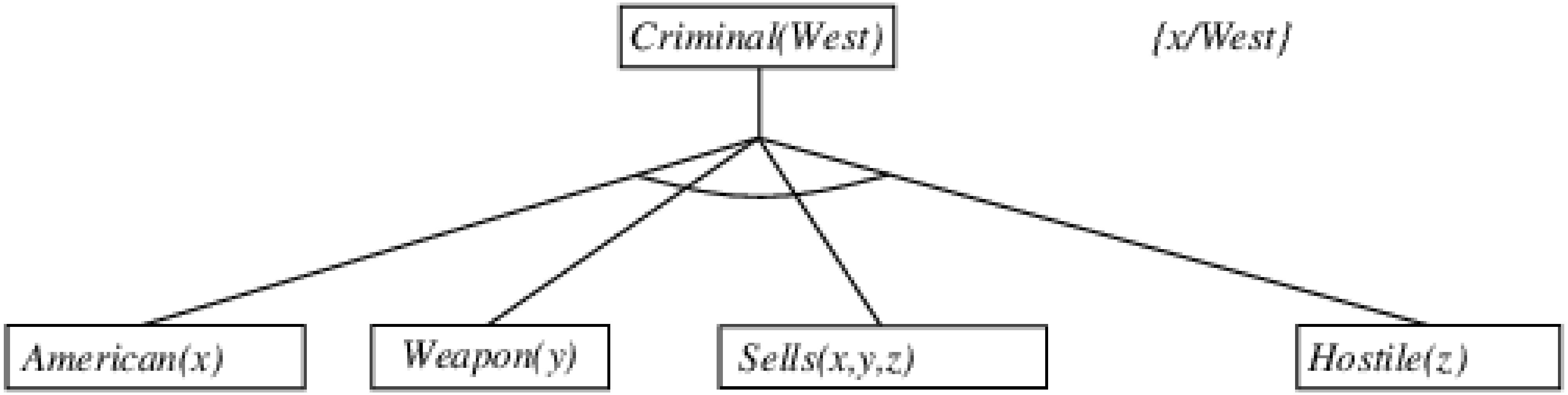
- Start with query

- Check if it can be derived by given rules and facts
  - apply rules that infer the query
  - recurse over pre-conditions

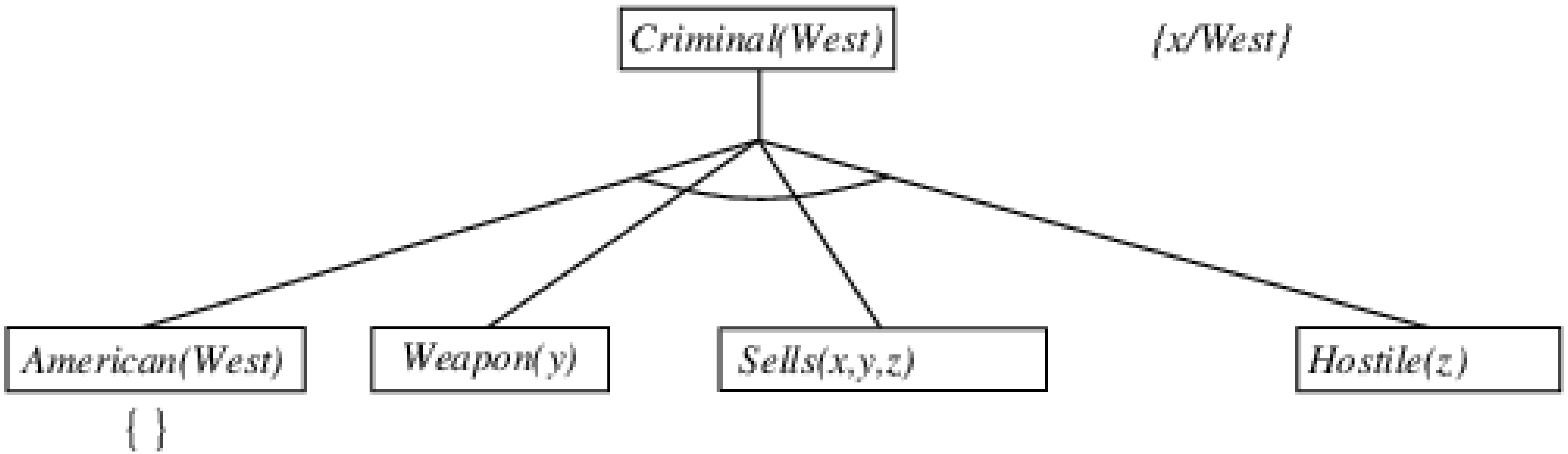
Backward Chaining Example

$\boxed{Criminal(West)}$

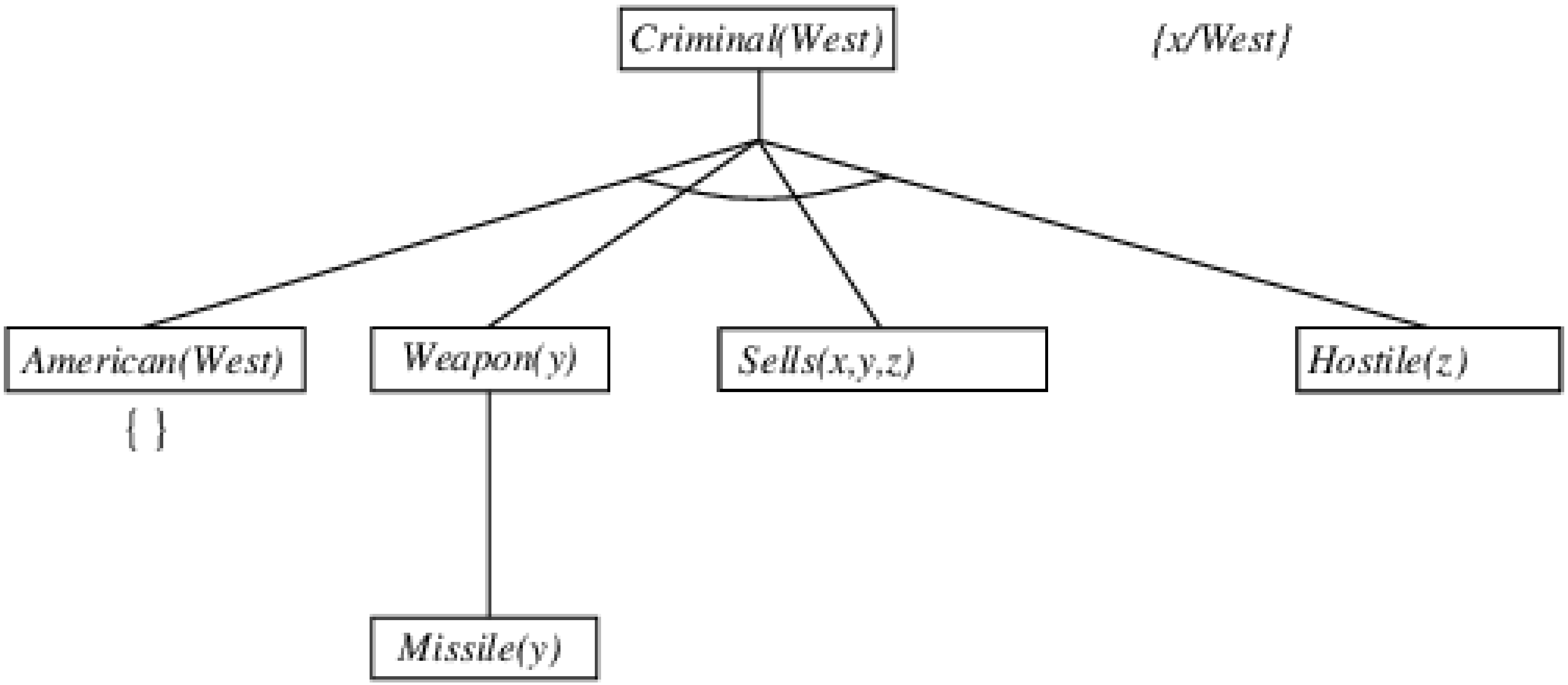
Backward Chaining Example



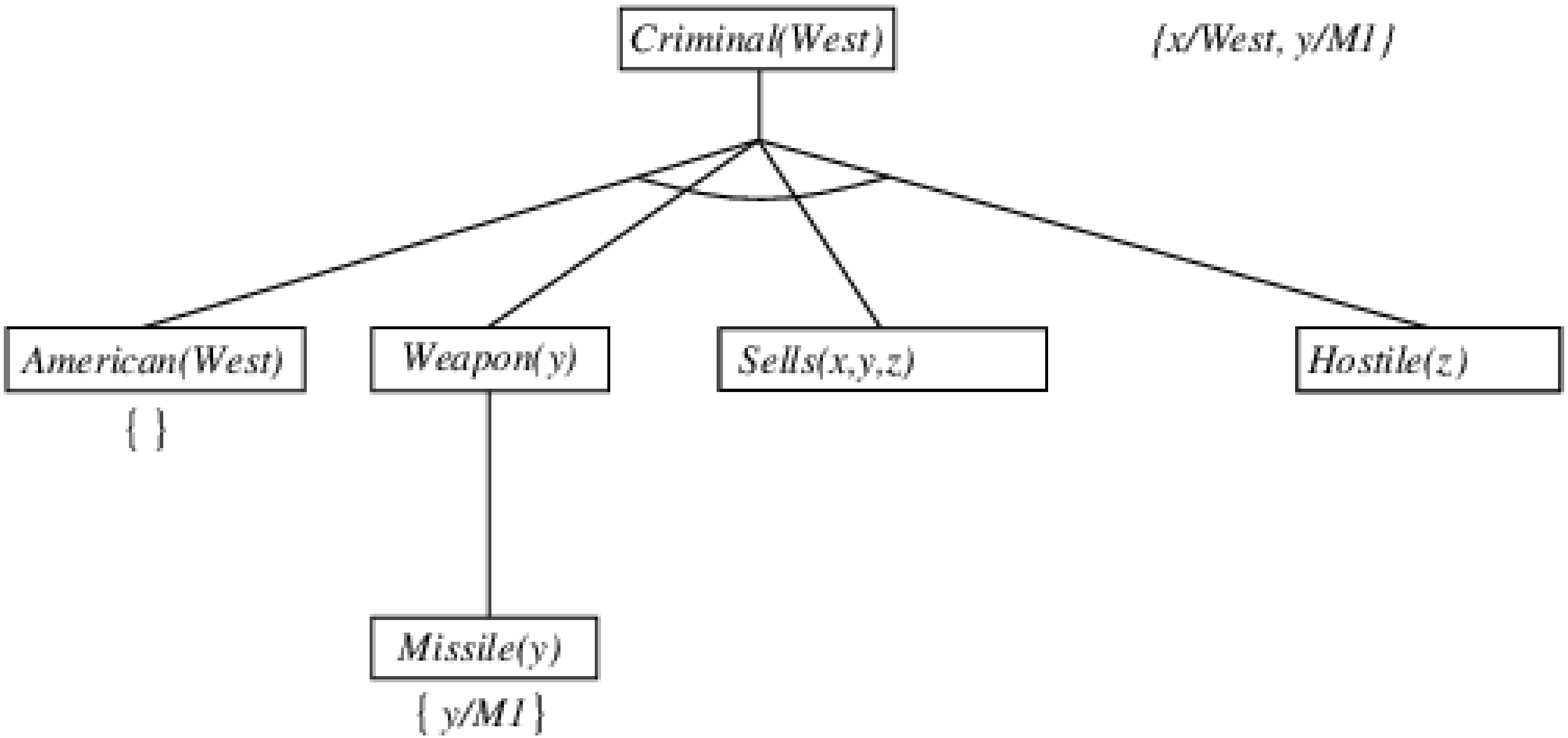
Backward Chaining Example



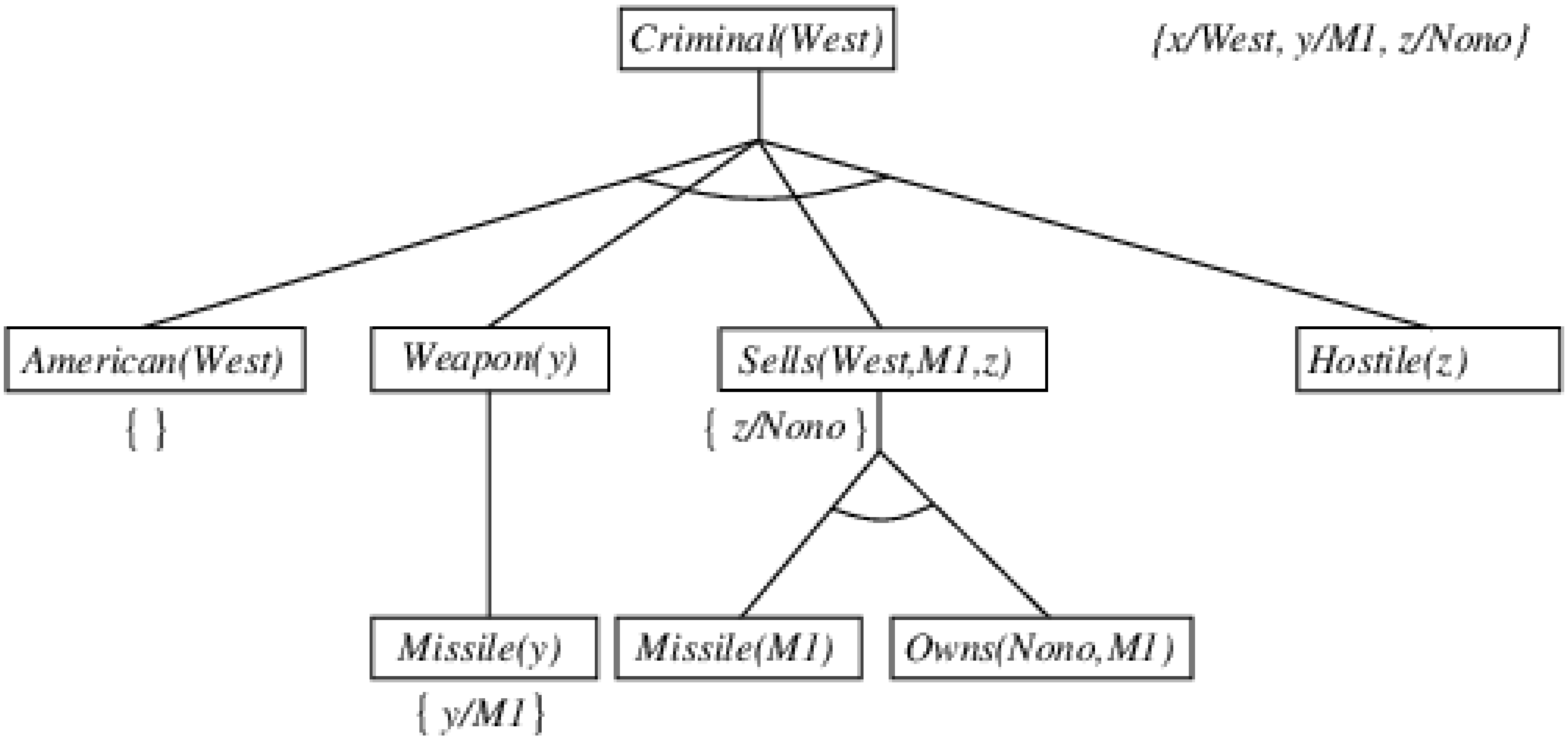
Backward Chaining Example



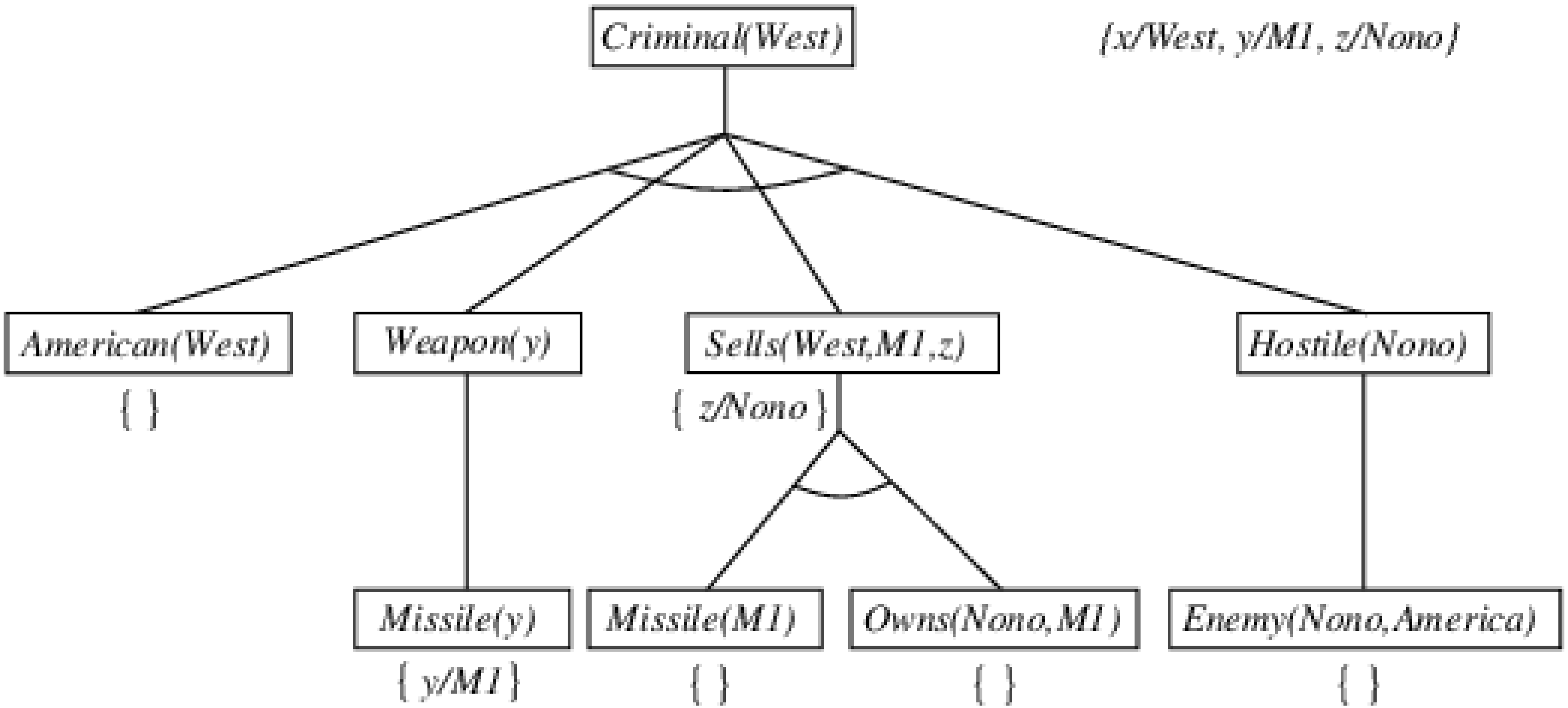
Backward Chaining Example



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
  $\Longrightarrow$ 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

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 $\varnothing$
local variables: *answers*, a set of substitutions, initially empty

if *goals* is empty **then return** $\{\theta\}$
$q' \leftarrow$ SUBST(θ, 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 $\leftarrow [p_1, \ldots, p_n |$ REST(*goals*)$]$
answers $\leftarrow$ FOL-BC-ASK(*KB*, *new_goals*, COMPOSE(θ', θ)) $\cup$ *answers*
return *answers*

logic programming

Logic Programming

- 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

- Basis: backward chaining with Horn clauses + bells & whistles
- Widely used in Europe, Japan (basis of 5th Generation project)
- Compilation techniques ⇒ approaching a billion logical inferences per second
- Program = set of clauses = `head :- literal`$_1$`, ... literal`$_n$`.`

```
criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).
missile(M1).
owns(Nono,M1).
sells(West,X,Nono) :- missile(X), owns(Nono,X).
weapon(X) :- missile(X).
hostile(X) :- enemy(X,America).
American(West).
Enemy(Nono,America).
```

Prolog Systems

- 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

resolution

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,

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

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

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

Conversion to CNF

Everyone who loves all animals is loved by someone:

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

1. Eliminate biconditionals and implications

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

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

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

Conversion to CNF

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

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

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

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

5. Drop universal quantifiers:

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

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

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

Our Previous Example

- Rules
  - $American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \implies Criminal(x)$
  - $Missile(M_1)$ and $Owns(Nono, M_1)$
  - $Missile(x) \wedge Owns(Nono, x) \implies Sells(West, x, Nono)$
  - $Missile(x) \Rightarrow Weapon(x)$
  - $Enemy(x, America) \implies Hostile(x)$
  - $American(West)$
  - $Enemy(Nono, America)$
- Converted to CNF
  - $\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x,y,z) \vee \neg Hostile(z) \vee Criminal(x)$
  - $Missile(M_1)$ and $Owns(Nono, M_1)$
  - $\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$
  - $\neg Missile(x) \vee Weapon(x)$
  - $\neg Enemy(x, America) \vee Hostile(x)$
  - $American(West)$
  - $Enemy(Nono, America)$
- Query: $\neg Criminal(West)$

Resolution Proof