Inference in first-order logic

Chapter 9

Outline

- ♢ Reducing first-order inference to propositional inference
- ♢ Unification
- ♢ Generalized Modus Ponens
- ♢ Forward and backward chaining
- ♢ Resolution
- ♢ Logic programming - separately in chp9-Prolog.ppt

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

FOL Inference

◇ One way to do inference in FOL is to convert the FOL KB into a propositional KB.

→ Propositionalization was an important historical algorithm (1930), but is not complete.

◇ As alternative, we will study Generalized Modus Ponens - with Forward and Backward chaining -and and Resolution, similar to PL.

◇ Inference in FOL requires the concept of substitution and instantiation. We will first see these two concepts.

Substitution

Given a sentence S and a substitution σ,
$S\sigma$ denotes the result of plugging σ into S; e.g.,
$S = Loves(x, y)$
$\sigma = \{x/John, y/Jane\}$
$S\sigma = Loves(John, Jane)$

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 (a term without variables)

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

$$\begin{array}{l} King(John) \land Greedy(John) \;\Rightarrow\; Evil(John) \\ King(Richard) \land Greedy(Richard) \;\Rightarrow\; Evil(Richard) \\ King(Father(John)) \land Greedy(Father(John)) \;\Rightarrow\; Evil(Father(John)) \\ \quad\vdots \end{array}$$

Existential instantiation (EI)

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

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

◇ E.g., $\exists\, x \;\; Spy(x) \wedge Knows(x, Turkish)$ yields

$Spy(S_1) \wedge Knows(S_1, Turkish)$

provided S_1 is a new constant symbol, called a Skolem constant.

◇ E.g., from $\exists\, x \;\; d(x^y)/dy = x^y$ we obtain

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

provided e is a new constant symbol

Reduction to propositional inference

Once we have rules for inferring non-quantified sentences, it becomes possible to reduce FOL to PL:

Suppose the KB contains just the following:

$$\forall x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x)$$
$$King(John)$$
$$Greedy(John)$$

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

$$King(John) \land Greedy(John) \;\Rightarrow\; Evil(John)$$
$$King(Richard) \land Greedy(Richard) \;\Rightarrow\; Evil(Richard)$$
$$King(John)$$
$$Greedy(John)$$

Reduction to propositional inference

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

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

The new KB is propositionalized: Proposition symbols are:

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

Or you can think the following, as they look more like propositions:

$KingJohn,\ GreedyJohn,\ EvilJohn, KingRichard$ etc.

Reduction contd.

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

Reduction contd.

Problem: With function symbols, there are infinitely many ground terms.

Assume KB contains:

$$\forall x \;\; King(x) \;\Rightarrow\; King(Father((x))$$
$$A = Father(B)$$
$$B = Father(C)$$
$$King(C)$$

In order to prove $King(A)$, the above method requires instantiation with depth-2 ground terms Father(Father(C)).

Notice that this could go on indefinitely.

Reduction contd.

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:

- you can prove it it if a sentence is entailed
- no algorithm exists that rejects every non-entailed sentence.

Problems with propositionalization

Propositionalization seems to generate lots of irrelevant sentences.

Assume we have the query Evil(x) and the following KB:

$$\forall x \;\; King(x) \Rightarrow Evil(x)$$
$$King(John)$$

It seems obvious that $Evil(John)$ holds, 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 much much worse!

$\rightarrow$ In addition to the fact that we will never know what depth should we use for our ground terms (Father(Father....(Father(John))

Unification

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

We can get the inference quickly if we can find a substitution θ such that $King(x)$ matches $King(John)$, and $Greedy(x)$ matches $Greedy(y)$.

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

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

Unification finds substitutions that makes different logical expressions look identical.

Unification

Assume we have the following KB; what is the unification of each pair (row) of expression:

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

Unification

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

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

Unification-Standardizing Apart

p	q	θ
$Knows(John, x)$	$Knows(x, OJ)$	♢ $fail$

♢ $Knows(John, x)$ means "John knows everyone" (universally quantified) and since "Everyone knows OJ" as well, so the unification should NOT fail.

♢ The problem is due to both predicates using the same variable, which is solved by "standardizing apart" (renaming variables) before unification.

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

Unification-Most General Unifier

$UNIFY(Knows(John, x), Knows(y, z))$?

$\{y/John, x/z\}$ or $\{y/John, x/John, z/John\}$?

Unification-Most General Unifier

$UNIFY(Knows(John, x), Knows(y, z))?$

$\{y/John, x/z\}$ (gives $(Knows(John, z))$ or
$\{y/John, x/John, z/John\}$ (gives $(Knows(John, John))$.

◊ When there are more than one possible unifier, pick the one that leaves the most choice for the variables.

◊ There is always a **single** Most General Unifier for every pair of expressions.

Proof with Basic inference rules

The next few slides show how a proof can be made with basic inference rules.

Then we present Generelized Modeus Ponens (GMP) that applies a common pattern of 3 steps in one.

Example proof using basic inference rules

Using the unification and inference rules, we can prove some statements:

Assuming that the KB contains the following:

Buffy is a buffalo	1. $Buffalo(Bob)$
Pat is a pig	2. $Pig(Pat)$
Buffaloes outrun pigs	3. $\forall\, x, y \;\; Buffalo(x) \land Pig(y) \;\Rightarrow\; Faster(x, y)$

Prove that Bob outruns Pat

Example proof

Prove Bob outruns Pat

Bob is a buffalo	1. $Buffalo(Bob)$
Pat is a pig	2. $Pig(Pat)$
Buffaloes outrun pigs	3. $\forall\, x, y \;\; Buffalo(x) \land Pig(y) \;\Rightarrow\; Faster(x, y)$
And-Intro. 1 & 2	4. $Buffalo(Bob) \land Pig(Pat)$

Example proof

Prove Bob outruns Pat

Bob is a buffalo	1. $Buffalo(Bob)$
Pat is a pig	2. $Pig(Pat)$
Buffaloes outrun pigs	3. $\forall\, x, y \;\; Buffalo(x) \wedge Pig(y) \;\Rightarrow\; Faster(x, y)$
And-Intro. to 1 & 2	4. $Buffalo(Bob) \wedge Pig(Pat)$
Univ.I. to 3, $\{x/Bob, y/Pat\}$	5. $Buffalo(Bob) \wedge Pig(Pat) \;\Rightarrow\; Faster(Bob, Pa$

Example proof

Prove Bob outruns Pat

Bob is a buffalo	1. $Buffalo(Bob)$
Pat is a pig	2. $Pig(Pat)$
Buffaloes outrun pigs	3. $\forall\, x, y \;\; Buffalo(x) \wedge Pig(y) \;\Rightarrow\; Faster(x, y)$
And-Intro. to 1 & 2	4. $Buffalo(Bob) \wedge Pig(Pat)$
Univ.I. to 3, $\{x/Bob, y/Pat\}$	5. $Buffalo(Bob) \wedge Pig(Pat) \;\Rightarrow\; Faster(Bob, Pa$
Mod.Pon. to 4 & 5	6. $Faster(Bob, Pat)$

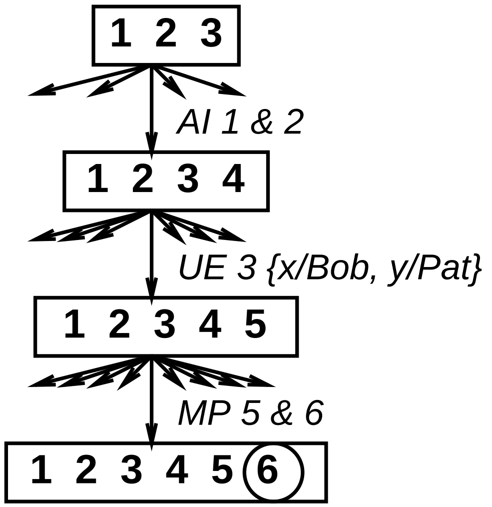
Search with primitive inference rules

The above can be implemented as a search process where:

◇ Operators are inference rules

◇ States are sets of sentences

◇ Goal test checks state to see if it contains query sentence



Problem: branching factor huge, esp. for UE

Idea: AI, UE, MP is a common inference pattern. Find a substitution that makes the rule premise match some known facts

$\Rightarrow$ a single, more powerful inference rule: Generalized Modus Ponens

Generalized Modus Ponens (GMP)

Idea (Gen. Mod. Ponens): Unify rule premises with known facts, apply unifier to conclusion

E.g., KB contains:
$Knows(John, Jane)$
$Knows(John, x) \Rightarrow Likes(John, x)$

We can infer: $Likes(John, Jane)$ after the unification with substitution $\{x/Jane\}$

Generalized Modus Ponens (GMP)

GMP does this in one step: And-Introduction, Universal Elimination, Modus Ponens

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

$$\begin{aligned}
\text{E.g. } p_1{}' = &\ \ \text{Faster(Bob,Pat)} \\
p_2{}' = &\ \ \text{Faster(Pat,Steve)} \\
p_1 \wedge p_2 \ \Rightarrow\ q \ = &\ \ Faster(x,y) \wedge Faster(y,z) \ \Rightarrow\ Faster(x,z) \\
\\
\sigma = &\ \ \{x/Bob, y/Pat, z/Steve\} \\
q\sigma = &\ \ Faster(Bob, Steve)
\end{aligned}$$

GMP

GMP is sensible because:

◊ It takes big steps

◊ It takes sensible steps: it uses substitutions that are guaranteed to help ($\{x/Bob, y/Pat, z/Steve\}$)

Generalized Modus Ponens (GMP)

◇ With the knowledgebase containing:

$$\forall x \;\; King(x) \wedge Greedy(x) \;\Rightarrow\; Evil(x)$$
$$King(John)$$
$$\forall y \;\; Greedy(y)$$

◇ Applying GMP, we can entail Evil(John):

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

where $p_i{}'\theta = p_i\theta$ for all i

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

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

θ is $\{x/John, y/John\}$ $\qquad$ q is $Evil(x)$

$q\theta$ is $Evil(John)$

Canonical Form

Inference mechanism with one inference rule (GMP) $\Rightarrow$ all the sentences in the KB should be in a form to match the premise of Modus Ponens.

Convert the KB into Canonical Form:

◇ Apply Existential Elimination

$\exists x \;\; Missile(x)$ is converted to
$Missile(M1)$

◇ Remove Universal Quantifiers (*all variables are assumed universally quantified*)

$\forall x \;\; Missile(x) \;\Rightarrow\; Weapon(x)$ is written as
$Missile(x) \;\Rightarrow\; Weapon(x)$

◇ Put sentences in Horn form

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.

◊ First of all you shd. be able to extract the facts and axioms to put in the knowledge base, from this paragraph

Example knowledge base contd.

"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."

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

Use $American(x), Weapon(x), Sells(x, y, z), Hostile(x), Criminal(x)$ predicates

Nono . . . has some missiles.

Use $UseOwns(x, y), Missile(x)$ predicates

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

Use $UseOwns(x, y), Missile(x), Sells(x, y, z)$ predicates

Example knowledge base contd.

"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."

Missiles are weapons:

Use $Missile(x), Weapon(x)$ predicates

An enemy of America counts as "hostile":

Use $Enemy(x, y), Hostile(x)$ predicates

West, who is American . . .

Use $American(x)$ predicate

The country Nono, an enemy of America . . .

Use $Enemy(x, y)$ predicate

Conversion to FOL-Summary

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

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

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

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

Missiles are weapons:

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

An enemy of America counts as "hostile":

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

West, who is American . . .

$American(West)$

The country Nono, an enemy of America . . .

$Enemy(Nono, America)$

Forward and Backward Chaining

GMP can be used to ways:

◇ forward chaining (new fact is added to the KB and we want to generate its consequences)

◇ backward chaining (we start with something we want to prove, find implication sentences that would conclude it, and attemp to establish their premises)

Forward chaining

When a new fact p is added to the KB
  for each rule such that p unifies with a premise
    if the other premises are <u>known</u>
    then add the conclusion to the KB and continue chaining

Forward chaining is <u>data-driven</u>
    e.g., inferring properties and categories from percepts

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

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

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

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

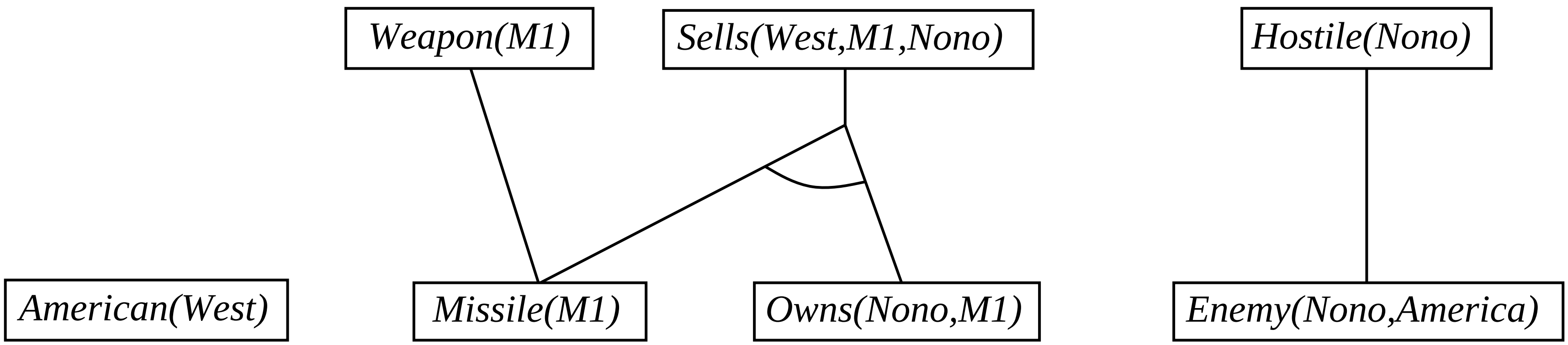
$Missile(x) \Rightarrow Weapon(x)$

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

$American(West)$

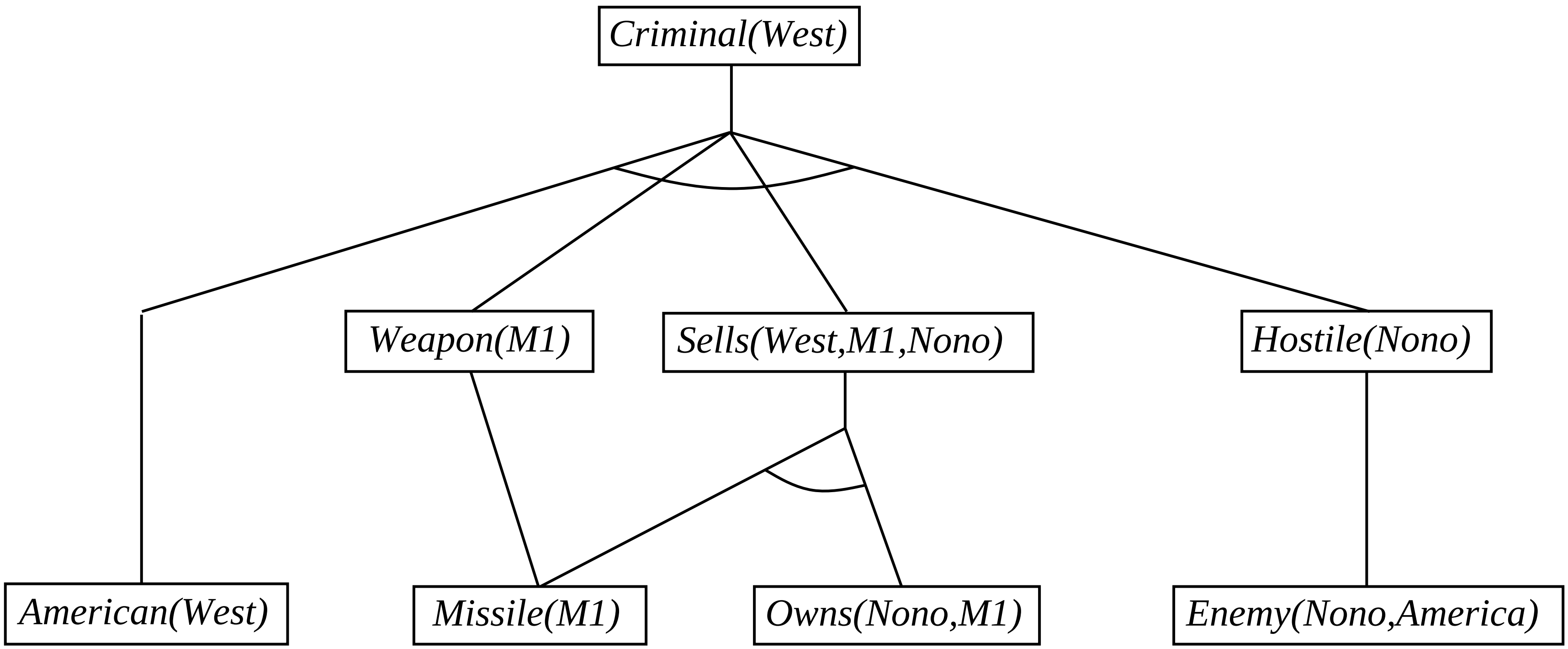
$Enemy(Nono, America)$

Forward chaining proof



◇ Three sentences was found to be of the form of "premises implies conclusion" and unified with known facts; then resulting substitution is applied to the conclusion of each rule (e.g. Hostile(Nono)).

Forward chaining proof



Properties of forward chaining

◇ Sound and complete for first-order definite clauses.

◇ FC may not terminate in general if α is not entailed.
This is unavoidable since entailment with definite clauses is semidecidable
can find a proof of α if $KB \models \alpha$
cannot always prove that $KB \not\models \alpha$

◇ Terminates for **Datalog** in polynomial iterations.
There can be at most $p \cdot n^k$ facts to be added, where k is the maximum arity (num. arguments) of any predicate and p is the number of predicates and n is the number of constant symbols.

Definite clause $=$ Horn clause with exactly one positive literal
Datalog $=$ First-order definite clauses $+$ **no functions** (e.g., crime KB)

◇ With function symbols, infinitely many new facts can be added.

Efficiency of forward chaining

♢ Matching itself can be expensive: matching conjunctive premises against known facts is NP-hard!

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

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 literal

♢ Forward chaining is widely used in deductive databases

Backward chaining

When a query q is asked
  if a matching fact q' is known, return the unifier
  for each rule whose consequent q' matches q
    attempt to prove each premise of the rule by backward chaining

(Some added complications in keeping track of the unifiers)

(More complications help to avoid infinite loops)

Backward chaining is the basis for logic programming, e.g., Prolog

Backward chaining algorithm

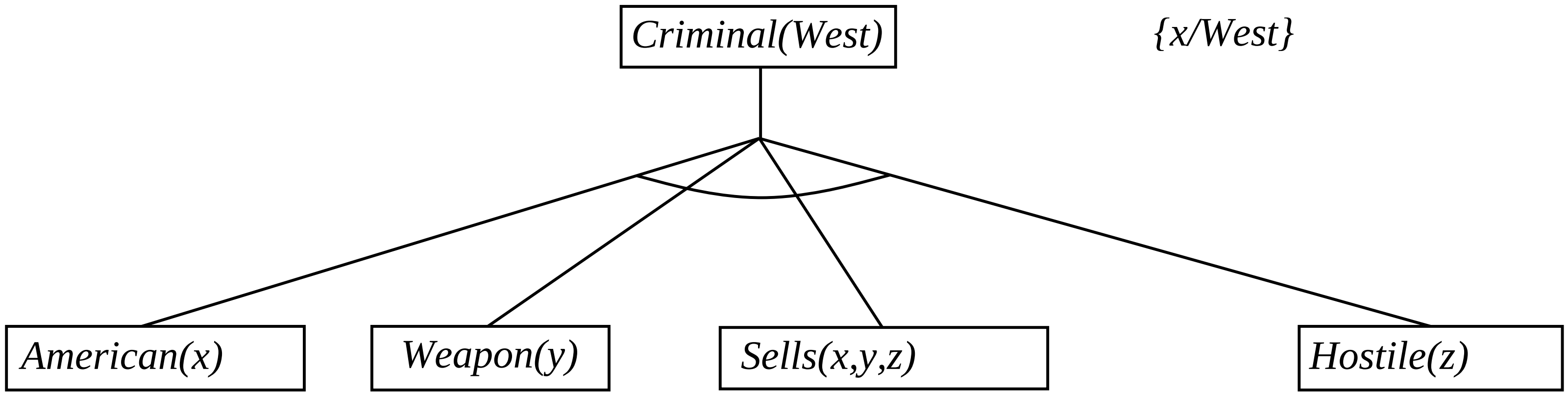
function FOL-BC-ASK($KB, goals, \theta$) **returns** a set of substitutions
  inputs: KB, a knowledge base
    $goals$, a list of conjuncts forming a query
    θ, the current substitution, initially the empty substitution $\{\,\}$
  local variables: $answers$, a set of substitutions, initially empty

  if $goals$ is empty **then return** $\{\theta\}$
  $q' \leftarrow$ SUBST(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 $answers$

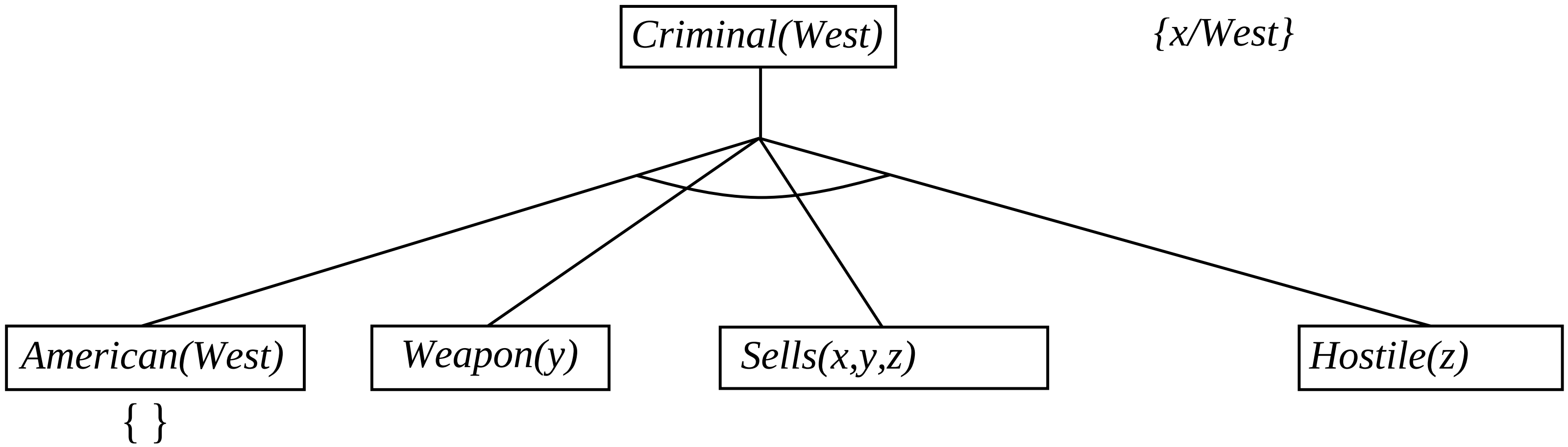
Backward chaining example

$\boxed{Criminal(West)}$

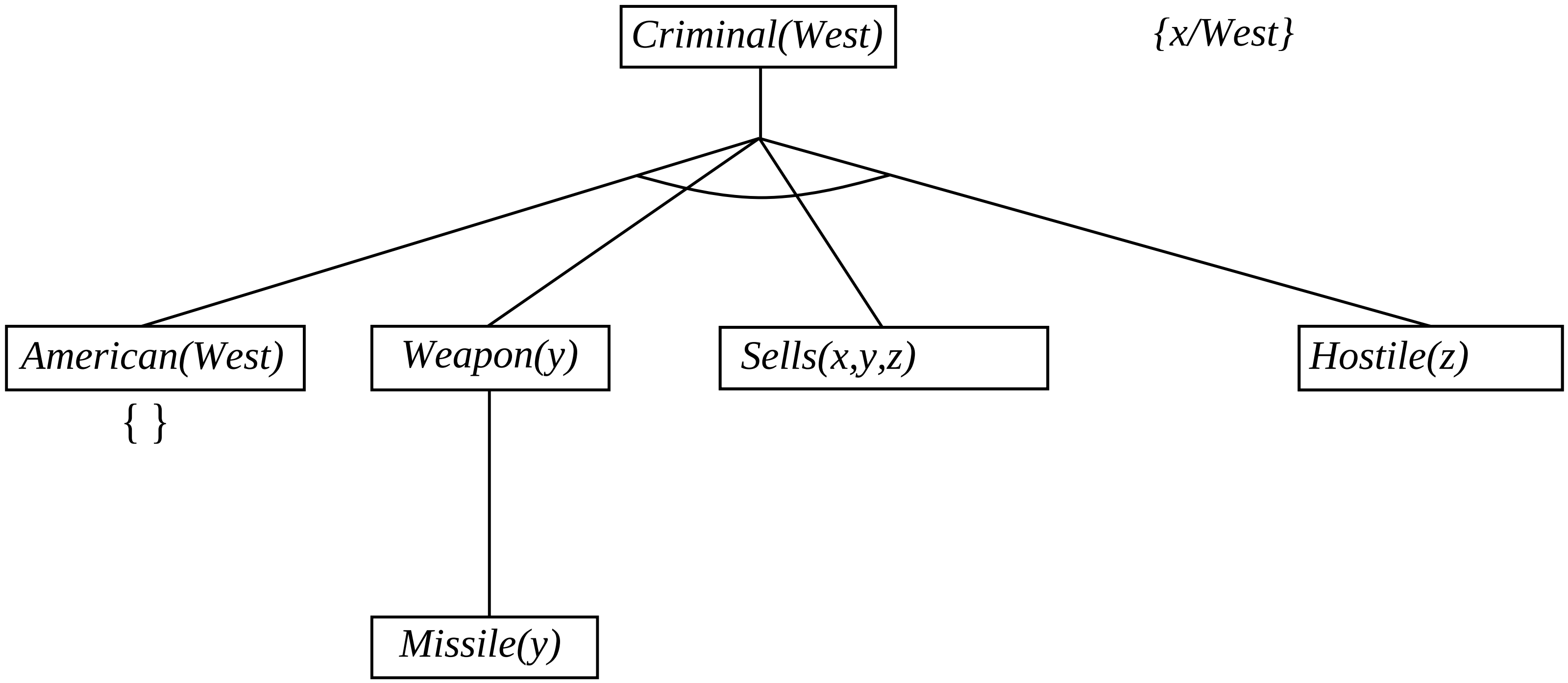
Backward chaining example



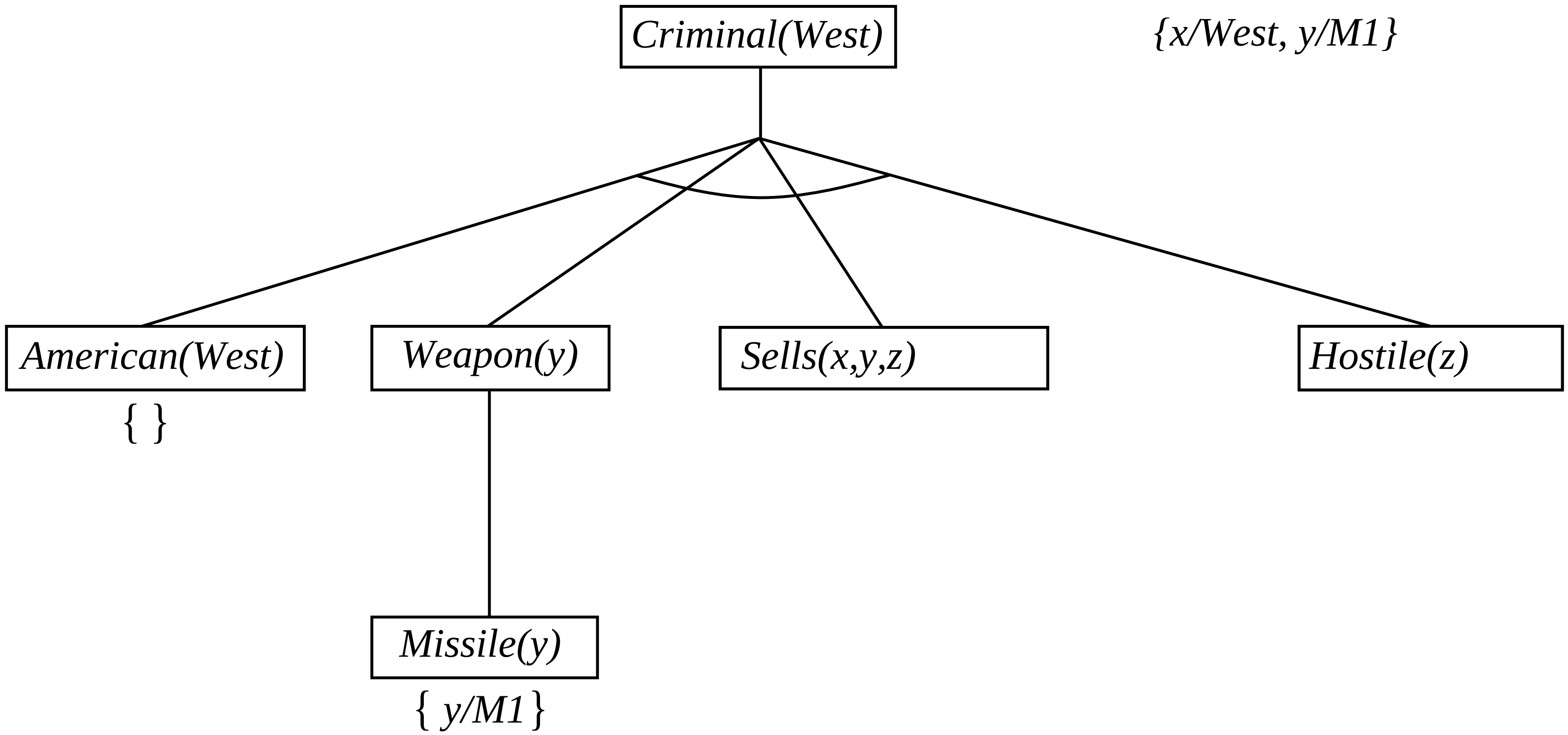
Backward chaining example



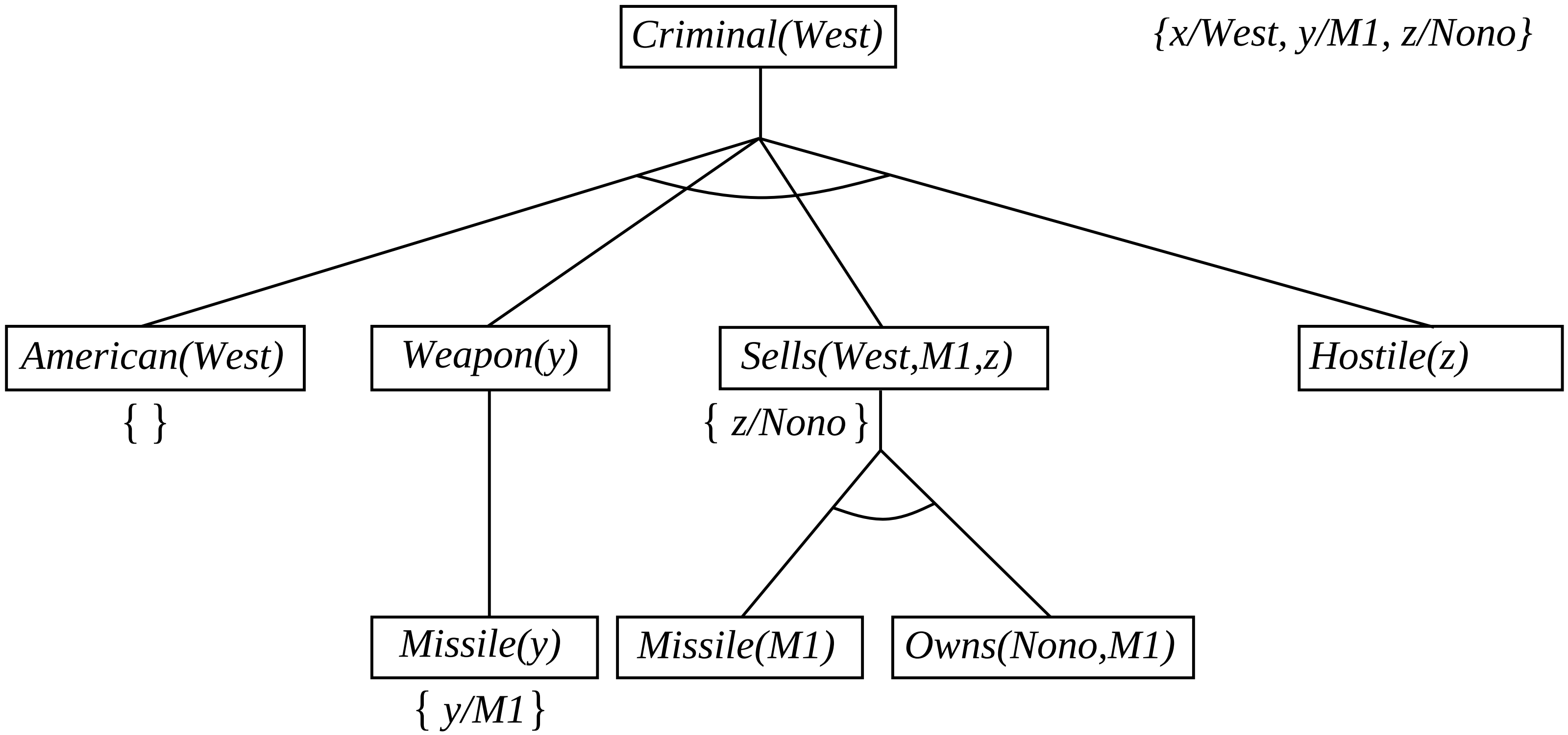
Backward chaining example



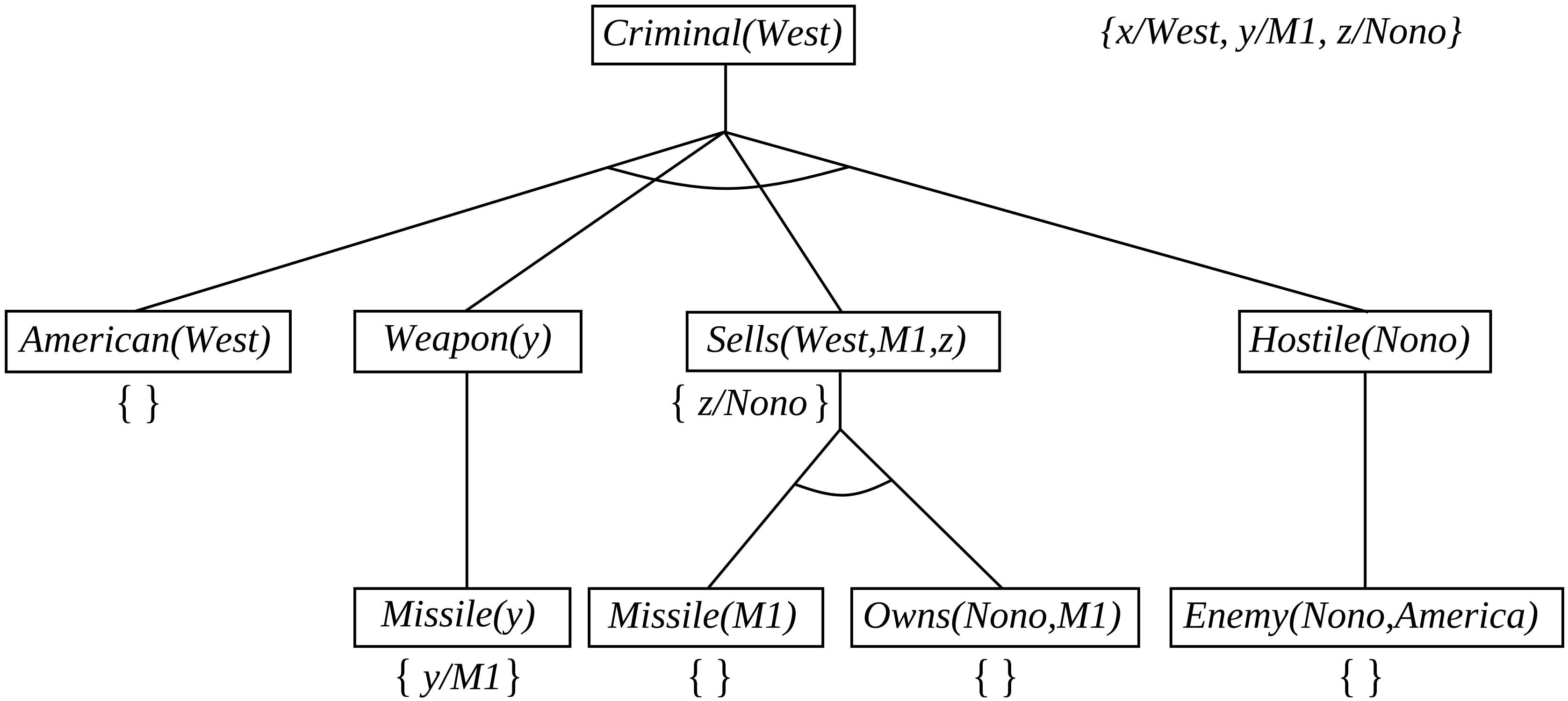
Backward chaining example



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
 $\Rightarrow$ 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 (without improvements!) for logic programming

Completeness of Modus Ponens

Unfortunately, Modus Ponens is **incomplete**. Assume we have the following KB:

$$P(x) \Rightarrow Q(x)$$
$$\neg P(x) \Rightarrow Q(x)$$
$$Q(x) \Rightarrow S(x)$$
$$R(x) \Rightarrow S(x)$$

We should be able to conclude $S(A)$ but we cannot because $\neg P(x) \Rightarrow Q(x)$ cannot be put in Horn form ($P1 \land P2 \land P3 \Rightarrow Q$ where P_i are non-negated atoms)

What other inference mechanism can we use, which will be complete?

Resolution: A complete Inference procedure

Resolution for FOL is complete as in Propositional Logic.

Basic propositional version:

$$\frac{\alpha \lor \beta, \quad \neg\beta \lor \gamma}{\alpha \lor \gamma} \qquad \text{or equivalently} \qquad \frac{\neg\alpha \;\Rightarrow\; \beta, \quad \beta \;\Rightarrow\; \gamma}{\neg\alpha \;\Rightarrow\; \gamma}$$

Resolution with CNF

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

♢ In POL, two literals are complementary if one is the negation of another one

♢ In FOL, they are complementary if one can be unified with the negation of the other one.

Resolution with CNF

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

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

Example:

$$\frac{\begin{array}{l}\neg Rich(x) \vee Unhappy(x) \\ Rich(Ken)\end{array}}{Unhappy(Ken)}$$

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

Conjunctive Normal Form

Resolution with disjunctions requires the CNF:

Literal = (possibly negated) atomic sentence, e.g., $\neg Rich(Me)$

Clause = disjunction of literals, e.g., $\neg Rich(Me) \vee Unhappy(Me)$

The KB is a conjunction of clauses.

Any FOL KB can be converted to CNF as follows:

1. Replace $P \Rightarrow Q$ by $\neg P \vee Q$
2. Move $\neg$ inwards, e.g., $\neg \forall x\, P$ becomes $\exists x\, \neg P$
3. Standardize variables apart, e.g., $\forall x\, P \vee \exists x\, Q$ becomes $\forall x\, P \vee \exists y\, Q$
4. Move quantifiers left in order, e.g., $\forall x\, P \vee \exists y\, Q$ becomes $\forall x \exists y\, P \vee Q$
5. Eliminate $\exists$ by Skolemization (next slide)
6. Drop universal quantifiers
7. Distribute $\wedge$ over $\vee$, e.g., $(P \wedge Q) \vee R$ becomes $(P \vee Q) \wedge (P \vee R)$

Skolemization

$\exists x\, Rich(x)$ becomes $Rich(G1)$ where $G1$ is a new "Skolem constant"

More tricky when $\exists$ is inside $\forall$

E.g., "Everyone has a heart"

$\forall\, x \;\; Person(x) \;\Rightarrow\; \exists\, y \;\; Heart(y) \wedge Has(x,y)$

Skolemization

More tricky when $\exists$ is inside $\forall$

E.g., "Everyone has a heart"

$$\forall x \;\; Person(x) \;\Rightarrow\; \exists y \;\; Heart(y) \land Has(x,y)$$

Incorrect:

$$\forall x \;\; Person(x) \;\Rightarrow\; Heart(H1) \land Has(x,H1)$$

Correct:

$$\forall x \;\; Person(x) \;\Rightarrow\; Heart(H(x)) \land Has(x,H(x))$$

where H is a new symbol ("Skolem function")

Skolem function arguments: all enclosing universally quantified variables

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 contd.

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 $\lor$:

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

Chaining with Resolution

Chaining with resolution is more powerful than Modus Ponens but still not complete.

Ex. How to prove $P \vee \neg P$ with an empty KB.

Solution: refutation (proof by contradiction):
To prove P, assume P is false (add $\neg P$ to the KB) and prove a contradiction.

In other words, to prove $KB \models \alpha$, show that $KB \wedge \neg\alpha$ is unsatisfiable

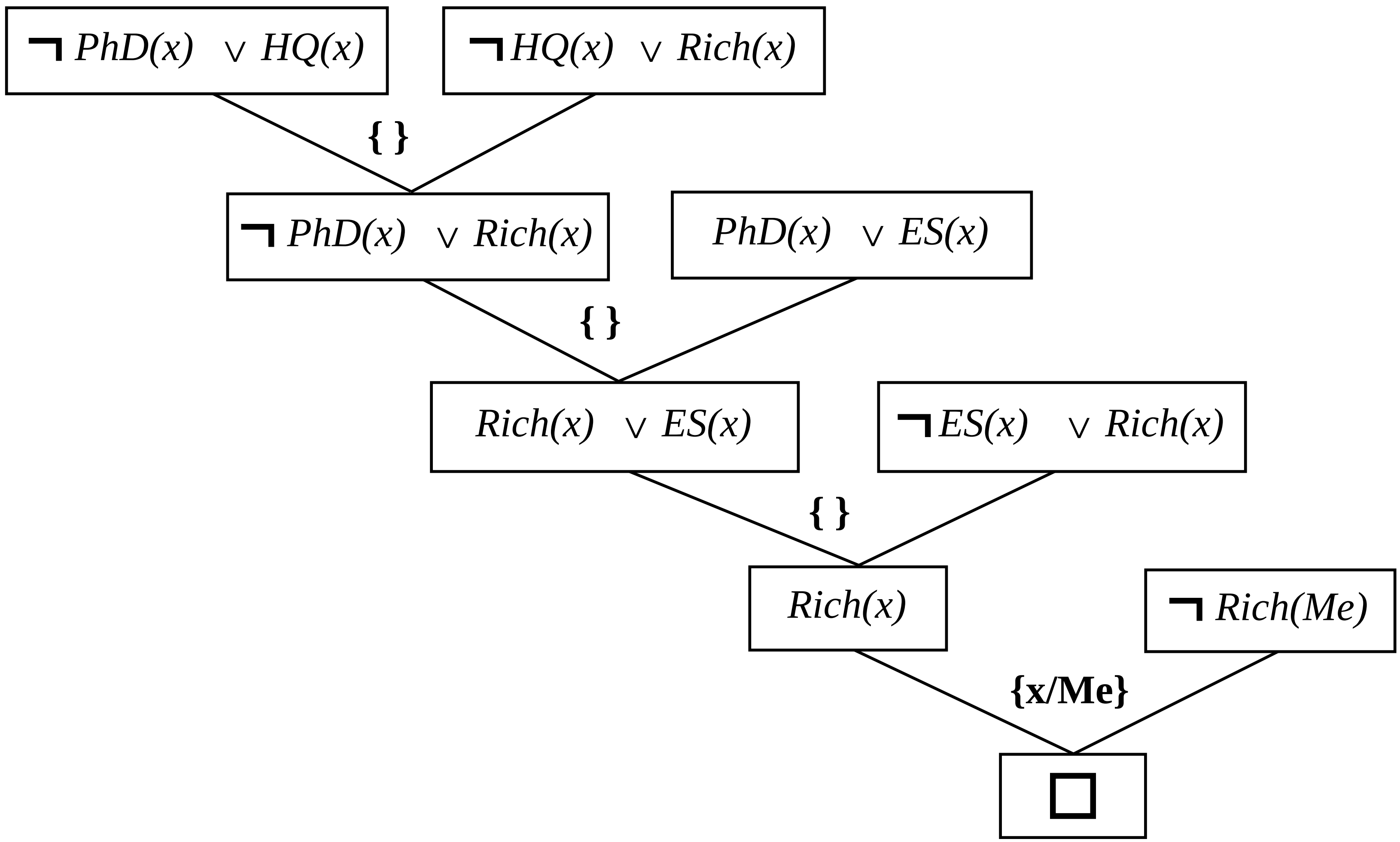
Refutation

To prove α:
- negate it
- convert to CNF
- add to CNF KB
- infer contradiction

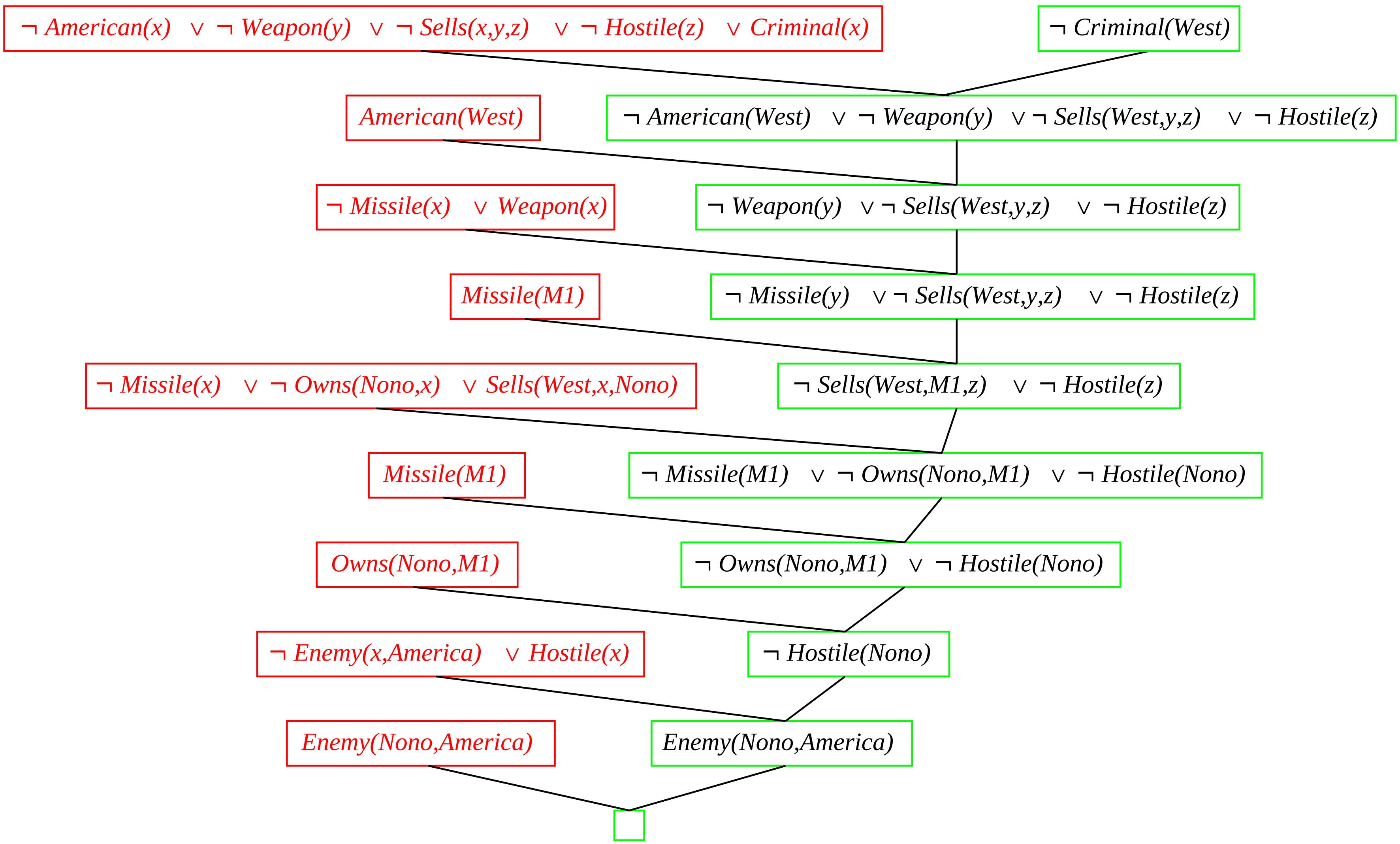
E.g., to prove $Rich(Me)$, add $\neg Rich(Me)$ to the CNF KB

$$\neg PhD(x) \lor HighlyQualified(x)$$
$$PhD(x) \lor EarlyEarnings(x)$$
$$\neg HighlyQualified(x) \lor Rich(x)$$
$$\neg EarlyEarnings(x) \lor Rich(x)$$

Resolution proof



Resolution proof: definite clauses



Resolution in practice

Resolution is complete and usually necessary for mathematics

Automated theorem provers are starting to be useful to mathematicians and have proved several new theorems

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 is a set of clauses of the form: `head :- literal`$_1$`, ... literal`$_n$`.`

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

◇ Properties:
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")
`alive(X) :- not dead(X).`
`alive(joe)` succeeds if `dead(joe)` fails