Inference in first-order logic

Chapter 9, Sections 1–5

Outline

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

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:

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

for any variable v and ground term g

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

Existential instantiation contd.

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

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

Reduction to propositional inference

Suppose the KB contains just the following:

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

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

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

The new KB is propositionalized: proposition symbols are

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

Reduction contd.

Claim: a ground sentence is entailed by new 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

Problems with propositionalization

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

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

it seems obvious that $Evil(John)$, but propositionalization produces lots of facts such as $Greedy(Richard)$ that are irrelevant

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

With function symbols, it gets much much worse!

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 $Greedy(y)$

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

$\text{Unify}(\alpha, \beta) = \theta$ where $\alpha\theta = \beta\theta$

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

Standardizing apart eliminates overlap of variables, e.g., x/z_{17} in q:

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

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

$p_1{}'$ is $King(John)$ p_1 is $King(x)$
$p_2{}'$ is $Greedy(y)$ p_2 is $Greedy(x)$
θ is $\{x/John, y/John\}$ q is $Evil(x)$
$q\theta$ is $Evil(John)$

GMP is used with a KB of definite clauses (**exactly** one positive literal)
All variables are assumed to be universally quantified

Theorem: GMP is sound

Soundness of GMP

We 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 Colonel West is a criminal.

Example knowledge base contd.

. . . 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 chaining algorithm

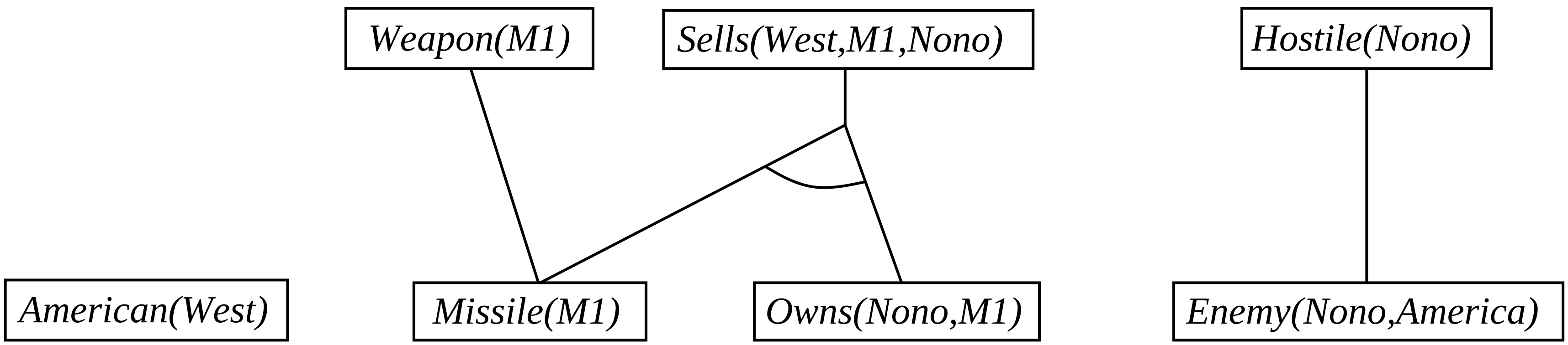
```
function FOL-FC-Ask(KB, α) returns a substitution or false

    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
```

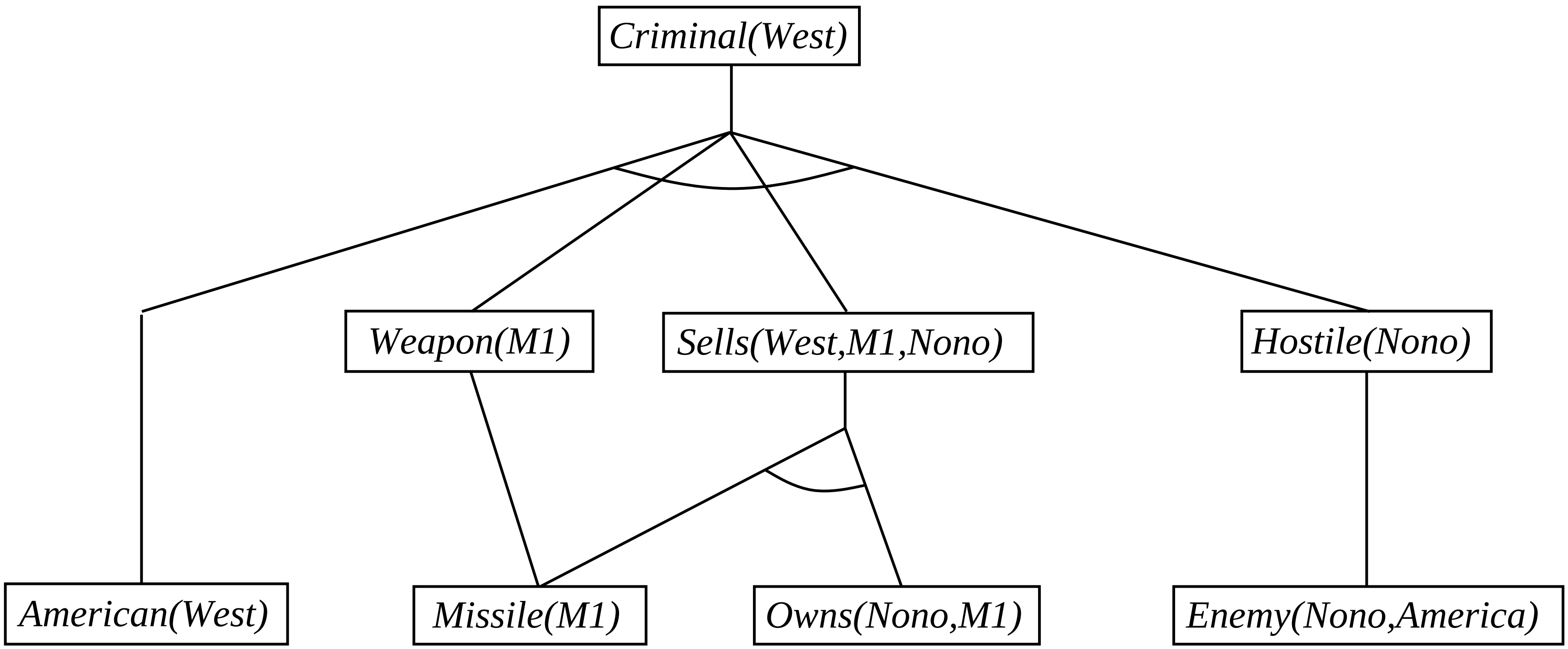
Forward chaining proof

American(West) *Missile(M1)* *Owns(Nono,M1)* *Enemy(Nono,America)*

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 polynomial time: 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

Backward chaining algorithm

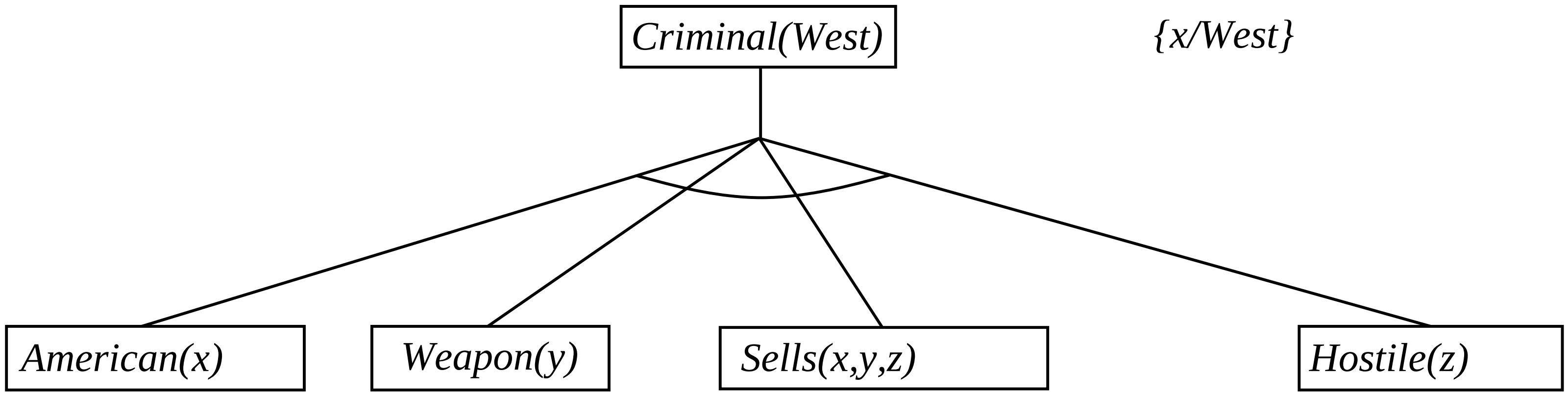
function FOL-BC-ASK(KB, $goals$, θ) **returns** a set of substitutions
inputs: KB, a knowledge base
$goals$, a list of conjuncts forming a query (θ already applied)
θ, the current substitution, initially the empty substitution $\{\,\}$
local variables: $answers$, a set of substitutions, initially empty

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

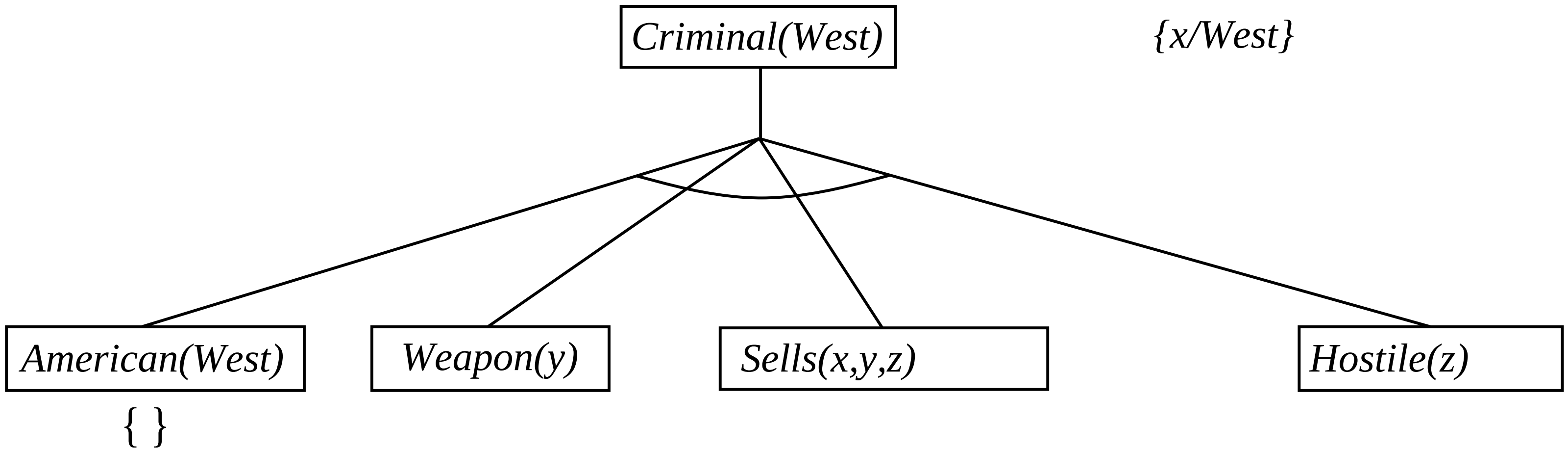
Backward chaining example

$\boxed{Criminal(West)}$

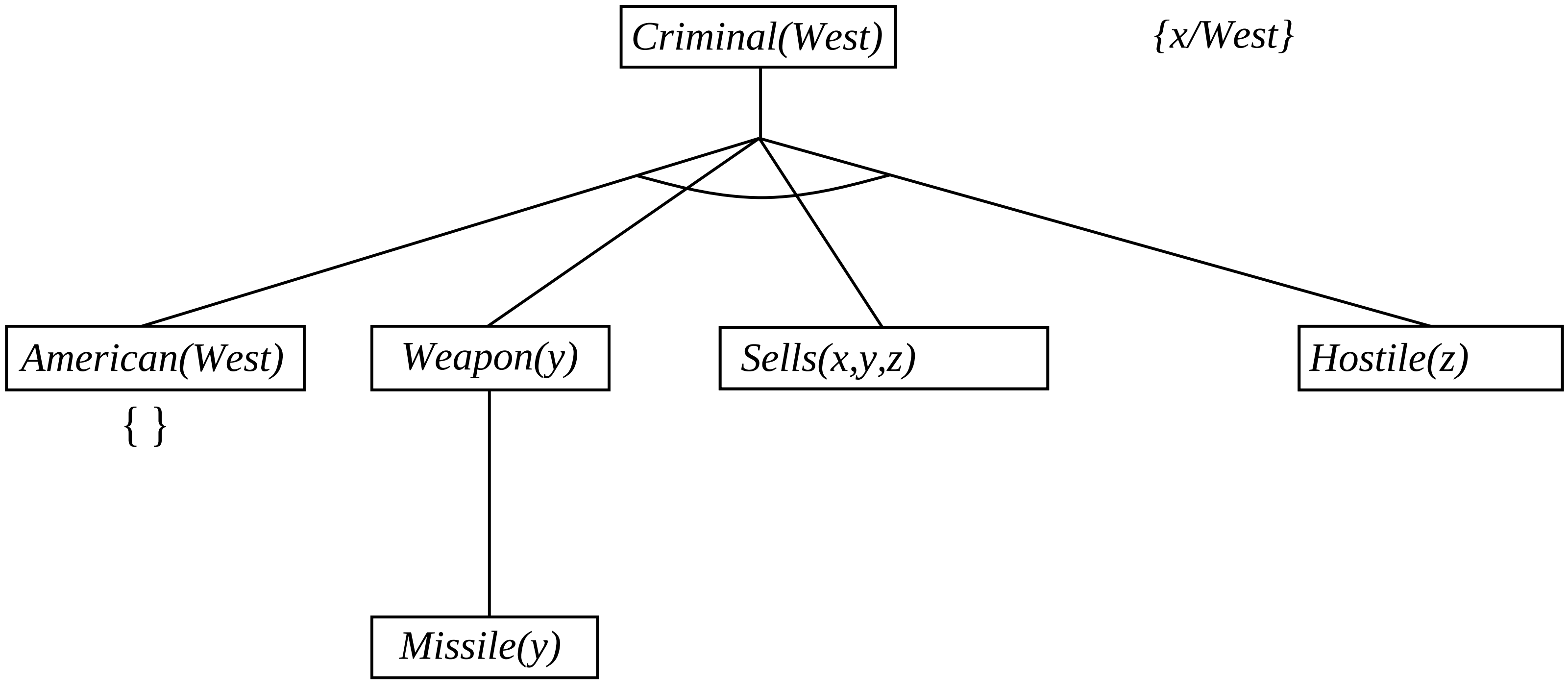
Backward chaining example



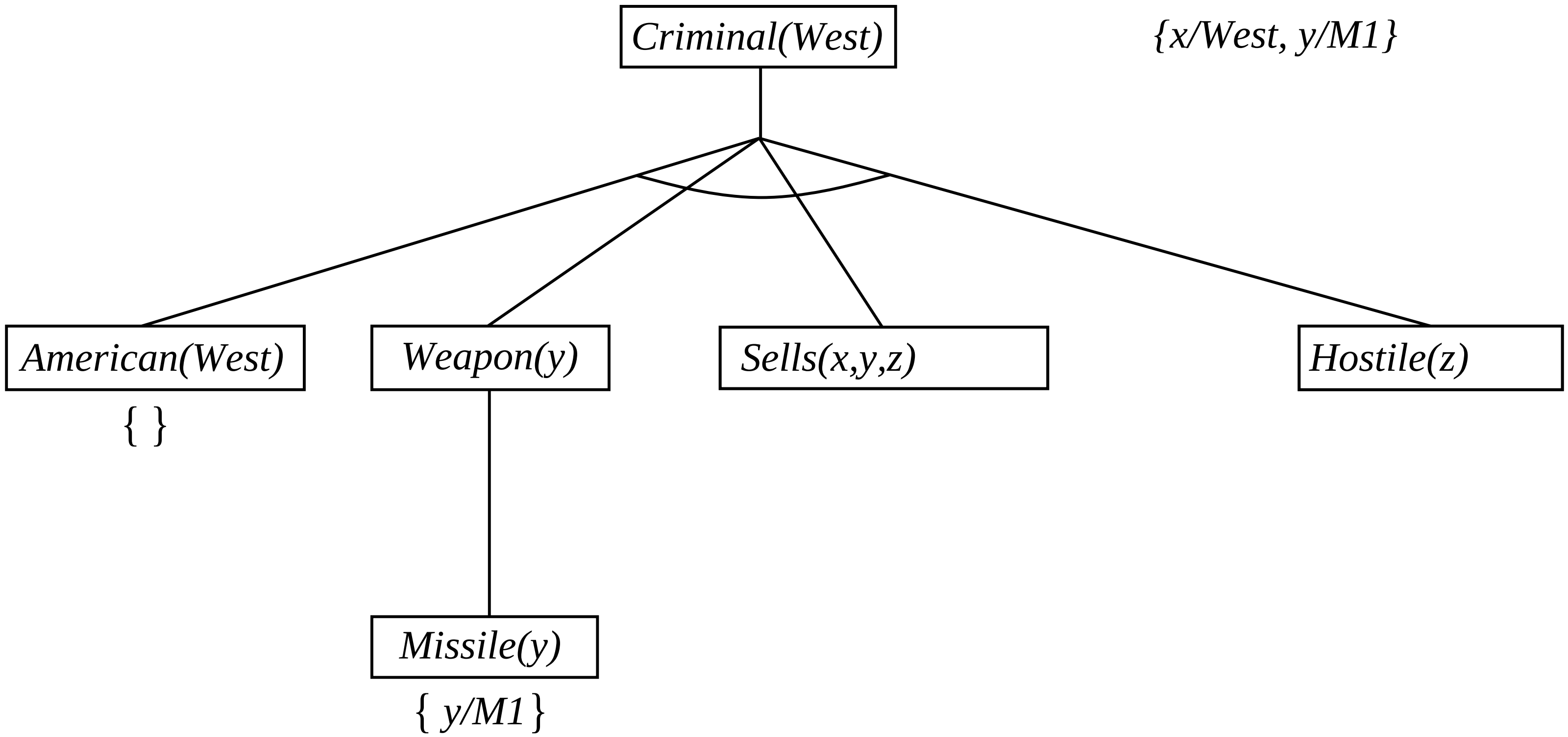
Backward chaining example



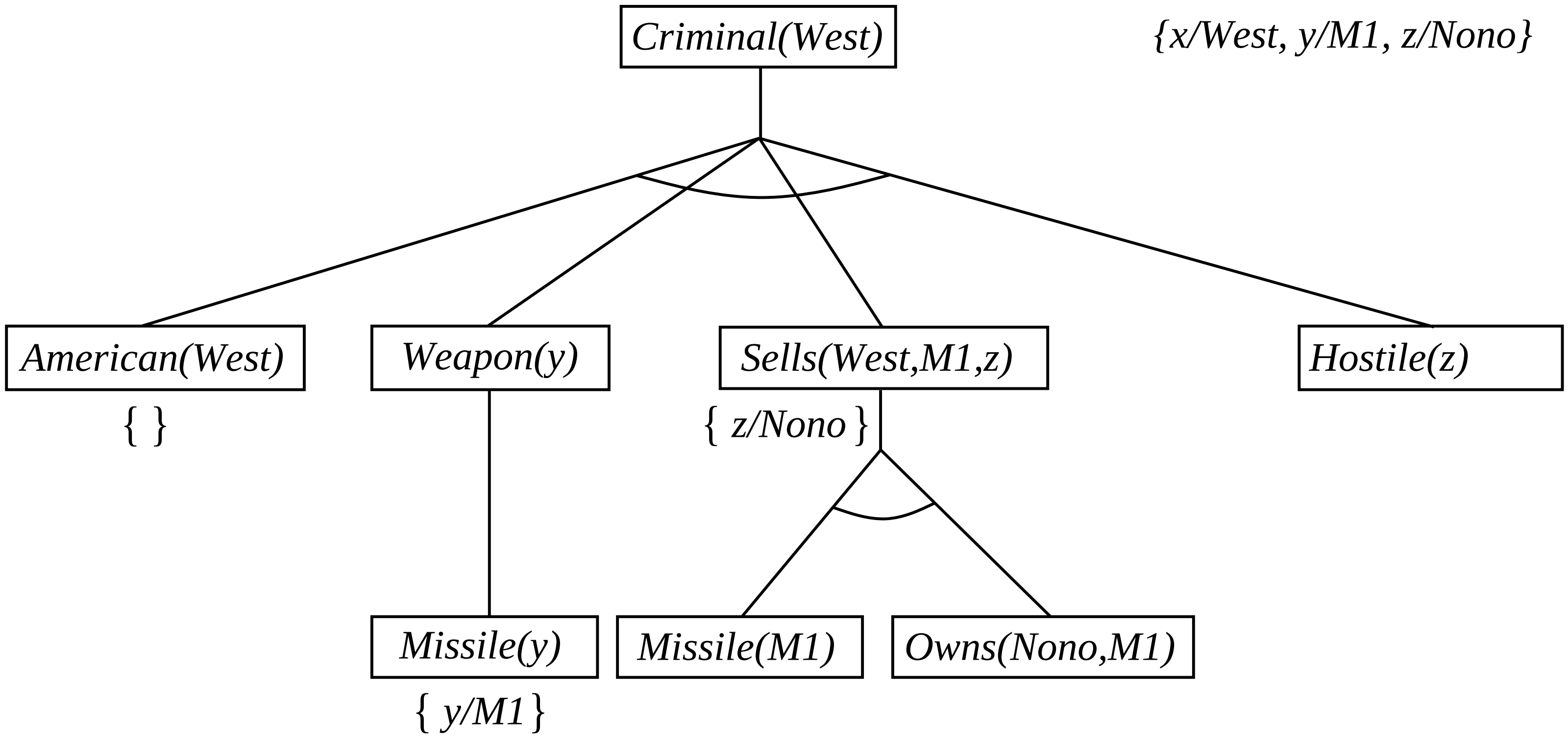
Backward chaining example



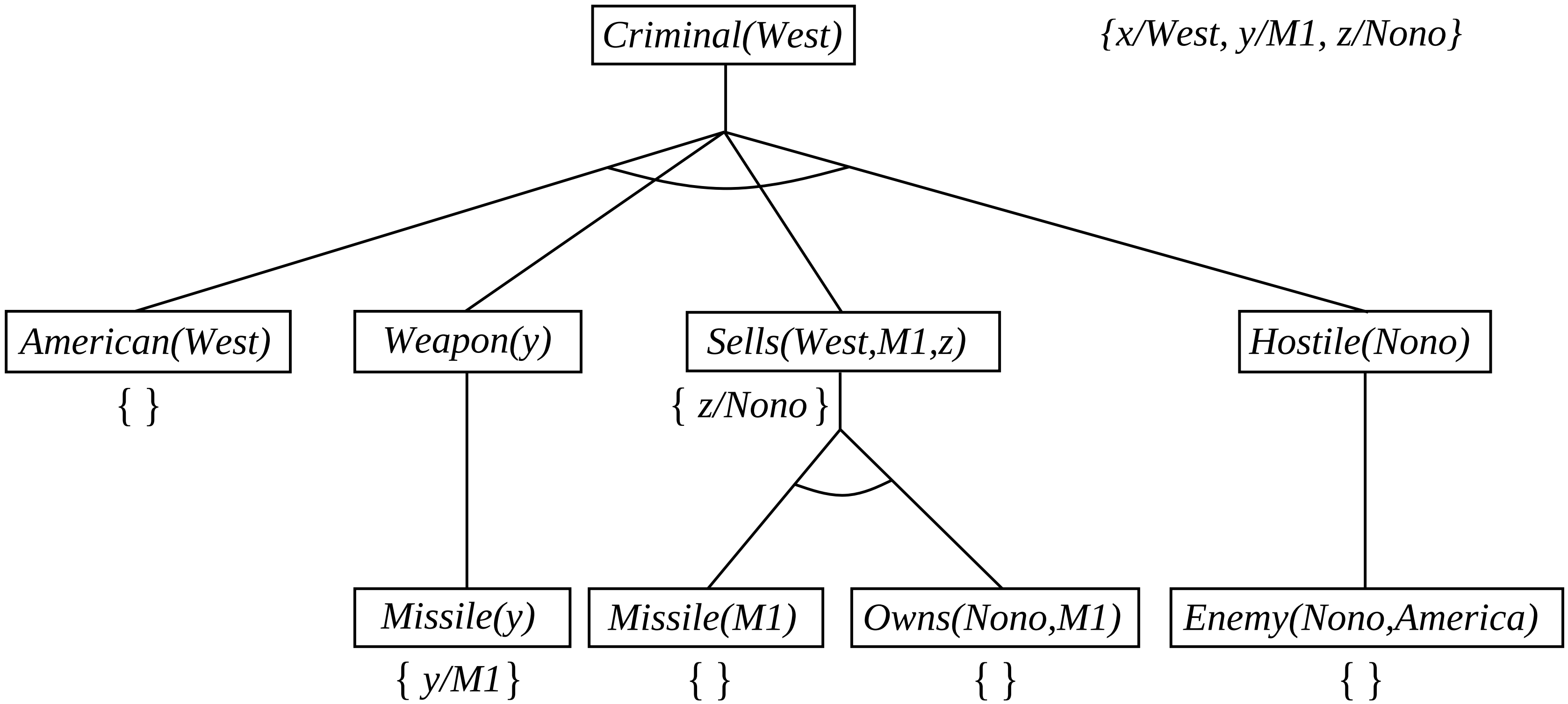
Backward chaining example



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

Resolution: brief summary

Full first-order version:

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

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

For example,

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

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:

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

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

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

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 $\vee$:

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

Resolution proof: definite clauses