# Inference in first-order logic
# 一阶逻辑中的推理

Chapter 9

# Last chapter

命题逻辑只是对事物的存在进行限定，而一阶逻辑对于对象和关系的存在进行限定，因而获得更强的表达能力。

First-order logic:

- objects and relations are semantic primitives（基本）
- syntax: constants, functions, predicates, equality, quantifiers
  - 语句的真值由一个模型和对句子符号的解释来判定。

Increased expressive power: sufficient to define wumpus world

在一阶逻辑中开发知识库是一个细致的过程，包括对域进行分析、选择词汇表、对支持所需推理必不可少的公理进行编码。

# Outline

- Reducing first-order inference to propositional inference
- Unification（合一）
- Generalized Modus Ponens（一般化分离规则）
- Forward and backward chaining
- Resolution

# Universal instantiation (UI) 全称实例化

Every instantiation of a universally quantified sentence is entailed by it:
全称量化语句蕴含它的所有实例

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

for any variable（变量） *v* and ground term（基项） *g*

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) 存在实例化

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

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

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（斯科伦常数）

Another example: from $\exists x \quad d(x^y)/dy = x^y$ 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

全称实例化可以多次应用从而获得许多不同的结果

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

存在实例化可以应用一次，然后取代存在量化语句；

新知识库逻辑上并不等价于旧知识库，但只有在原始知识库可满足时，新的知识库才是可满足的。

# Reduction to propositional inference
# 简化到命题逻辑推理

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（命题化）： proposition symbols are

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

# Reduction contd.

Claim: Every FOL KB can be propositionalized so as to preserve entailment

每一个一阶逻辑知识库都可以命题化使得蕴含关系得以保持

Claim: A ground sentence is entailed by new KB iff entailed by original KB

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

Problem: with function symbols, there are infinitely many（无限多个） ground terms（基项）,

– e.g., *Father*(*Father*(*Father*(*John*)))

# Reduction contd.

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

定理：如果某个语句被原始的一阶知识库蕴含，则存在一个只涉及命题化知识库的有限子集的证明

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 for FOL is semidecidable（半可判定的） (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.)

# Problems with propositionalization

Propositionalization seems to generate lots of irrelevant/不相关的 sentences.

E.g., from:

$\forall x$ King(x) $\wedge$ 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/谓词 and *n* constants, there are $p \cdot n^k$ instantiations.

With function symbols, it gets much much worse!

# Outline

- Reducing first-order inference to propositional inference
- Unification（合一）
- Generalized Modus Ponens（一般化分离规则）
- Forward and backward chaining
- Resolution

# Unification（合一）

如果存在某个置换θ使蕴涵的前提和KB中已有的语句完全相同，那么应用θ后，就可以断言蕴涵的结论

We can get the inference immediately if we can find a substitution（置换） θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

Unify(α,β) = θ if αθ = βθ

| 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（合一）

如果存在某个置换θ使蕴涵的前提和KB中已有的语句完全相同，那么应用θ后，就可以断言蕴涵的结论

We can get the inference immediately if we can find a substitution（置换）θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

Unify(α,β) = θ if αθ = βθ

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

# Unification（合一）

如果存在某个置换θ使蕴涵的前提和KB中已有的语句完全相同，那么应用θ后，就可以断言蕴涵的结论

We can get the inference immediately if we can find a substitution（置换） θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

Unify(α,β) = θ if αθ = βθ

| p | q | θ |
|---|---|---|
| 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)) | |
| Knows(John,x) | Knows(x,OJ) | |

# Unification（合一）

如果存在某个置换θ使蕴涵的前提和KB中已有的语句完全相同，那么应用θ后，就可以断言蕴涵的结论

We can get the inference immediately if we can find a substitution（置换） θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

Unify(α,β) = θ if αθ = βθ

| p | q | θ |
|---|---|---|
| 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) | |

# Unification（合一）

如果存在某个置换θ使蕴涵的前提和KB中已有的语句完全相同，那么应用θ后，就可以断言蕴涵的结论

We can get the inference immediately if we can find a substitution（置换） θ such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

θ = {x/John,y/John} works

Unify(α,β) = θ if αθ = βθ

| p | q | θ |
|---|---|---|
| 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（标准化分离） eliminates overlap of variables, e.g., Knows($z_{17}$,OJ)

# Unification（合一）

To unify *Knows(John,x)* and *Knows(y,z)*,

θ = {y/John, x/z } or θ = {y/John, x/John, z/John}

The first unifier is more general（更加一般） than the second.

– 对变量的取值限制比较少

There is a single most general unifier (MGU) that is unique up to renaming of variables.

对每个表达式的合一对，存在一个唯一的最一般合一者，不考虑变量的重新命名它是唯一的。

MGU = { y/John, x/z }

# Outline

- Reducing first-order inference to propositional inference
- Unification（合一）
- Generalized Modus Ponens（一般化分离规则）
- Forward and backward chaining
- Resolution

# Generalized Modus Ponens (GMP)

Modus Ponens（演绎推理，分离规则）(for Horn Form): complete for Horn KBs

$$\frac{\alpha_1,\ldots,\alpha_n, \quad \alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

GMP（一般化分离规则）：

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta}$$ 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 θ is *Evil*(*John*)

GMP used with KB of definite clauses确定子句 (exactly one positive literal)

All variables assumed universally quantified

# Semi-decidability （半可判定）

- First-order logic (even restricted to only Horn clauses) is semi-decidable.
  - If KB entails f, algorithms exist to prove f in finite time.
  - If KB does not entail f, no algorithm can show this in finite time.

# 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 sentence *p*, we have $p \models p\theta$

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

# Completeness of GMP

- GMP: incomplete for FOL
  - Not every sentence can be converted to Horn form
- GMP: complete for FOL KB of definite clauses

# 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（导弹）, and all of its missiles were sold to it by Colonel（上校）West, who is American.

Prove that Col. West is a criminal

# Example knowledge base contd.

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

# 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

# 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

# Example knowledge base contd.

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

*American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*

Nono … has some missiles, i.e., ∃x Owns(Nono,x) ∧ Missile(x):

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

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

*Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*

Missiles are weapons:

# Example knowledge base contd.

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

*American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*

Nono … has some missiles, i.e., ∃x Owns(Nono,x) ∧ Missile(x):

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

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

*Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*

Missiles are weapons:

*Missile(x) ⇒ Weapon(x)*

An enemy of America counts as "hostile“:

# 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

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

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 …

*$Enemy(Nono,America)$*

# Outline

- Reducing first-order inference to propositional inference
- Unification（合一）
- Generalized Modus Ponens（一般化分离规则）
- Forward and backward chaining
- Resolution

# Forward chaining algorithm

**function** FOL-FC-ASK($KB, \alpha$) **returns** a substitution or *false*

    **repeat until** *new* is empty
        $new \leftarrow \{\}$
        **for each** sentence $r$ **in** $KB$ **do**
            $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART($r$)
            **for each** $\theta$ such that $(p_1 \wedge \ldots \wedge p_n)\theta = (p'_1 \wedge \ldots \wedge p'_n)\theta$
                        for some $p'_1, \ldots, p'_n$ in $KB$
                $q' \leftarrow$ SUBST($\theta, q$)
                **if** $q'$ is not a renaming of a sentence already in $KB$ or *new* **then do**
                    add $q'$ to *new*
                    $\phi \leftarrow$ UNIFY($q', \alpha$)
                    **if** $\phi$ is not *fail* **then return** $\phi$
        add *new* to $KB$
    **return** *false*

# 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

*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)*

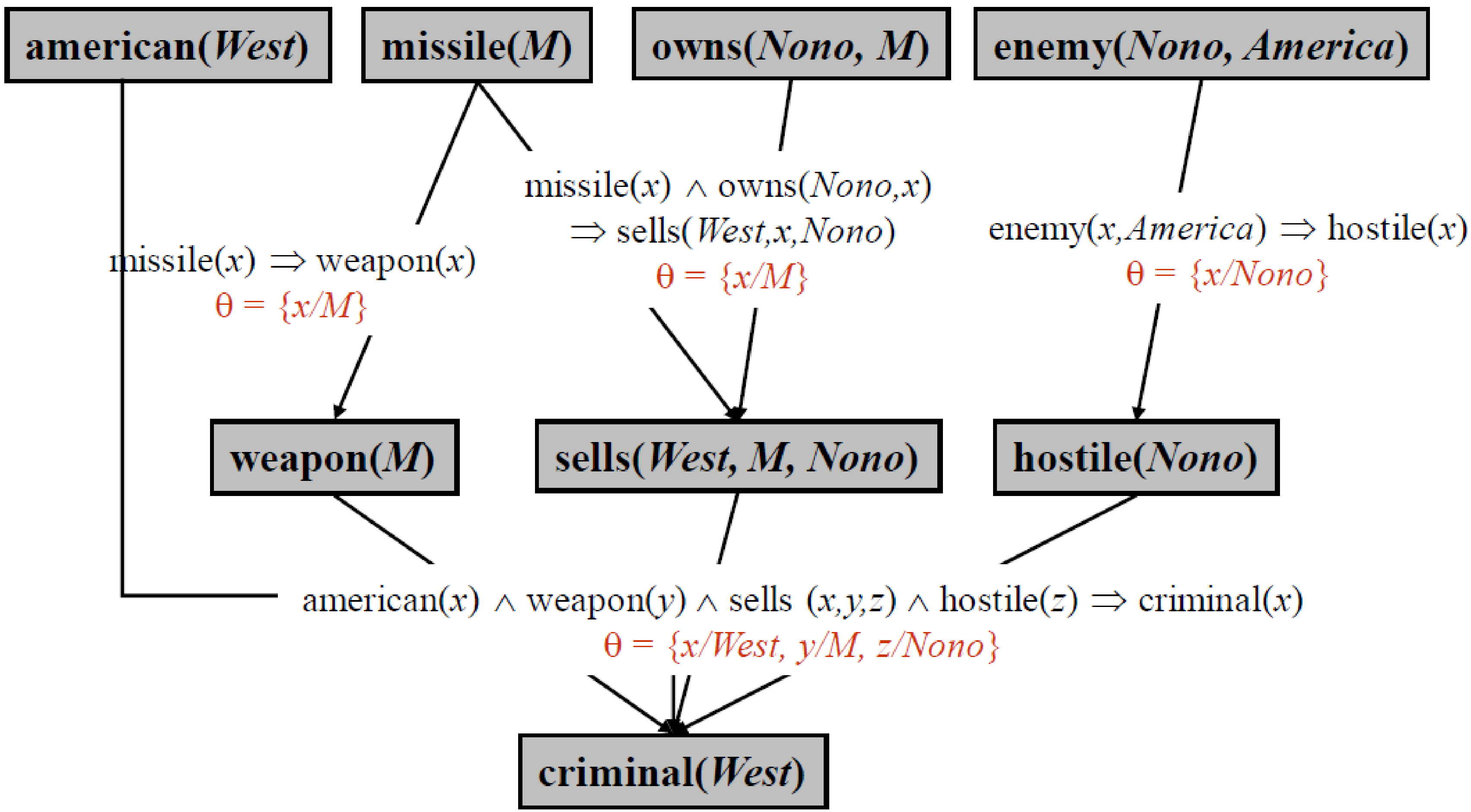
West, who is American …

*American(West)*

The country Nono, an enemy of America …

*Enemy(Nono,America)*

# Forward chaining proof



# Properties of forward chaining

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

Datalog （数据日志） = 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（半可判定的）

# Efficiency of forward chaining

Simple observation: no need to match（匹配） 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（索引） 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
把确定子句与事实集相匹配是一个NP难题

# Backward chaining algorithm

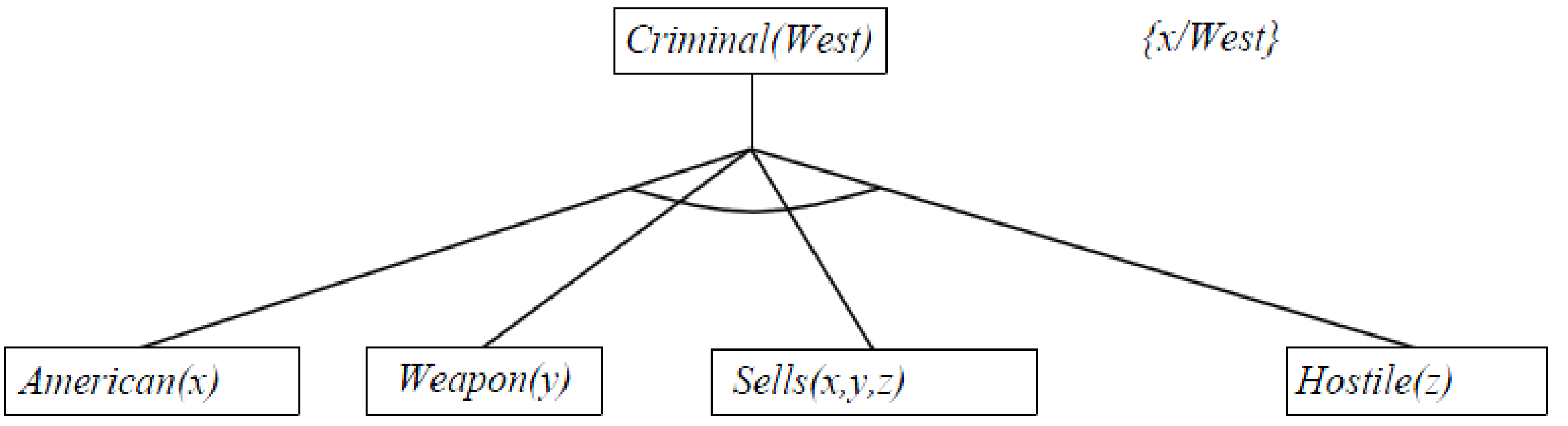
**function** FOL-BC-ASK($KB$, $goals$, $\theta$) **returns** a set of substitutions
  **inputs**: $KB$, a knowledge base
    $goals$, a list of conjuncts forming a query ($\theta$ already applied)
    $\theta$, the current substitution, initially the empty substitution $\{\ \}$
  **local variables**: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

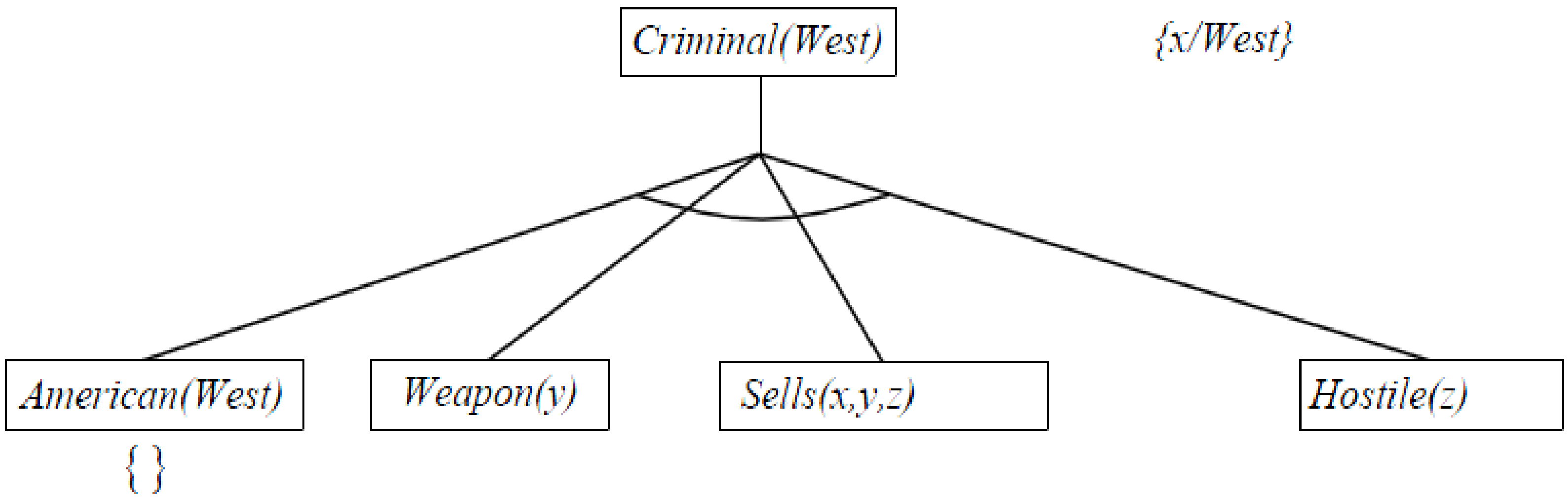
# Backward chaining example

*Criminal(West)*

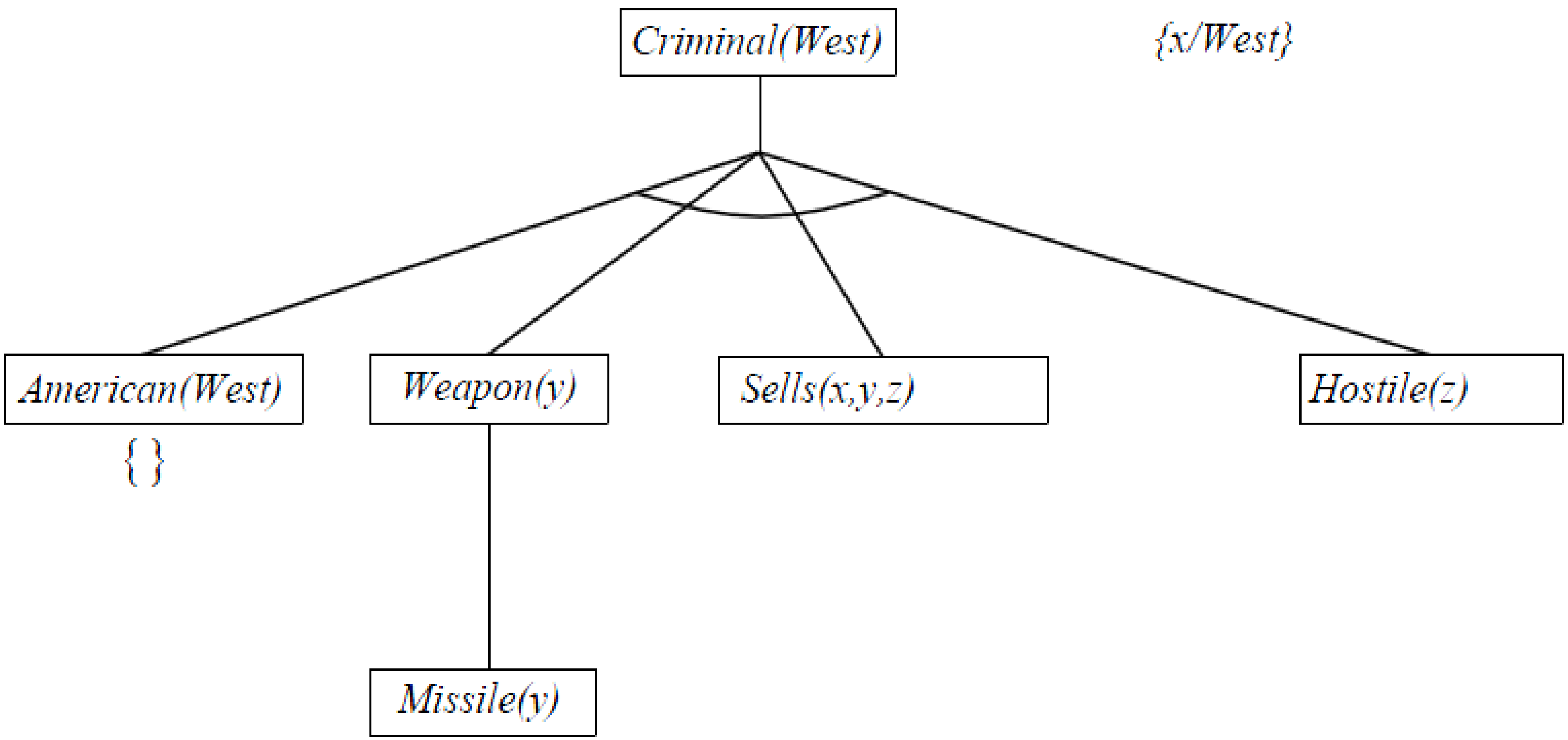
# Backward chaining example



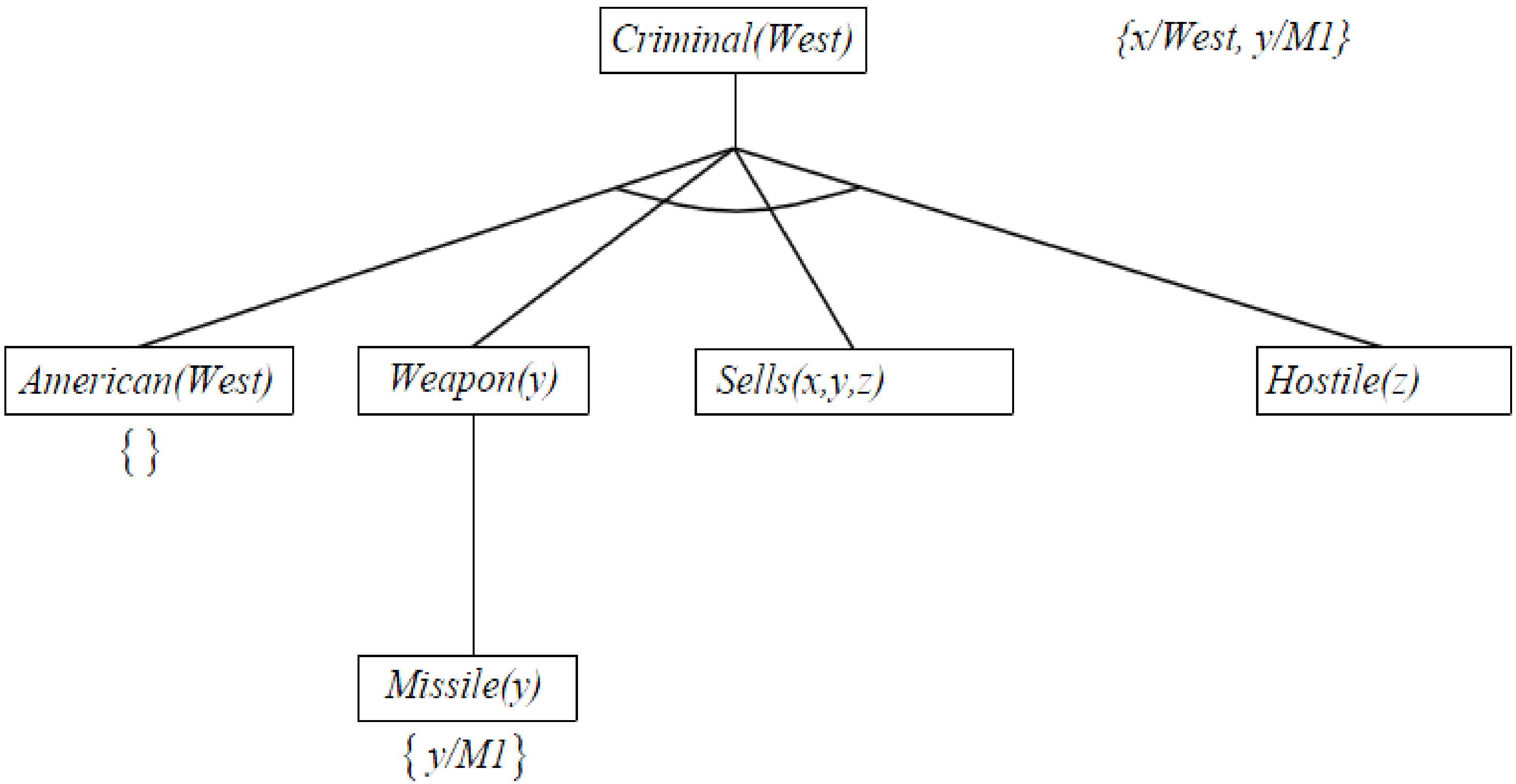
# Backward chaining example



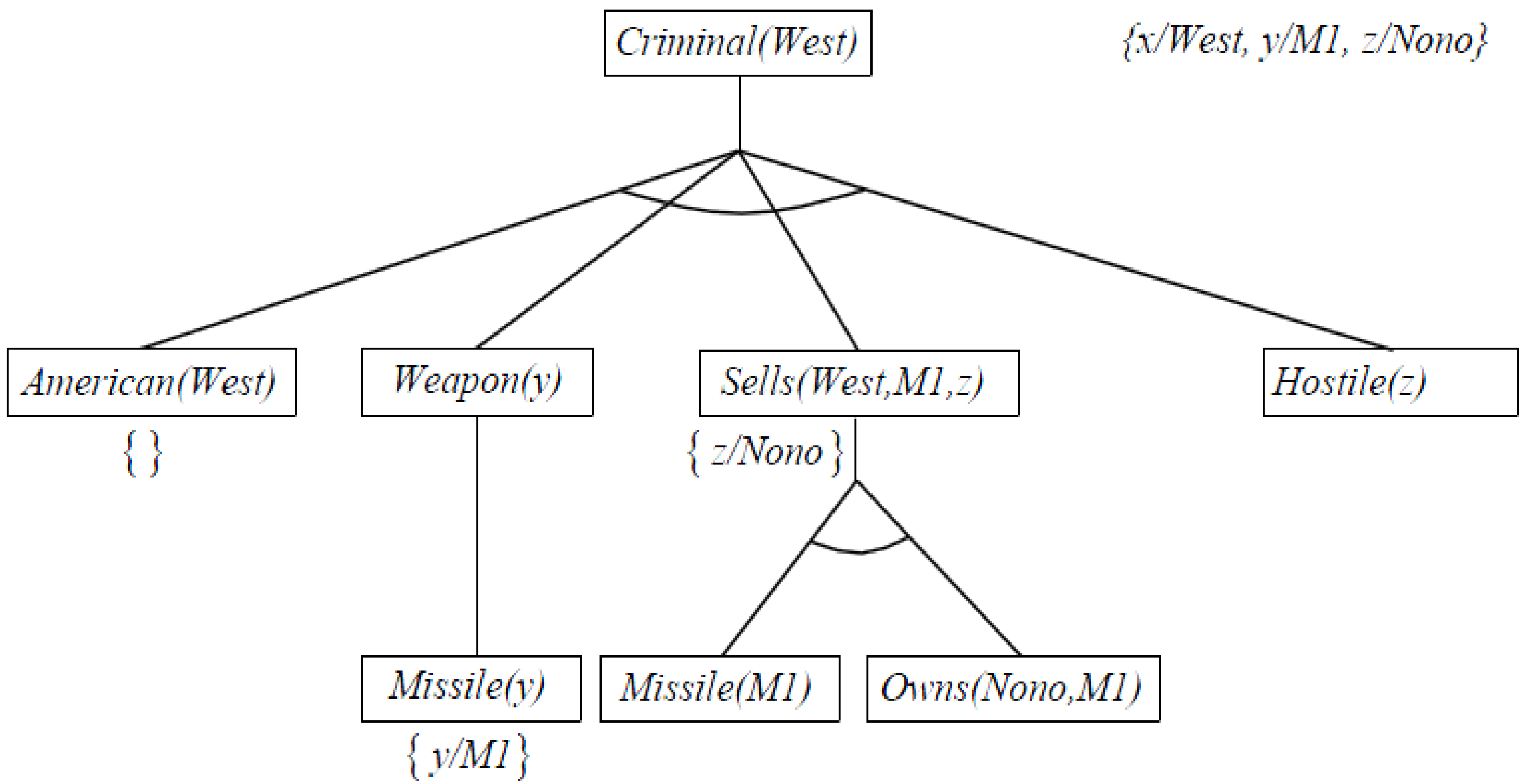
# Backward chaining example



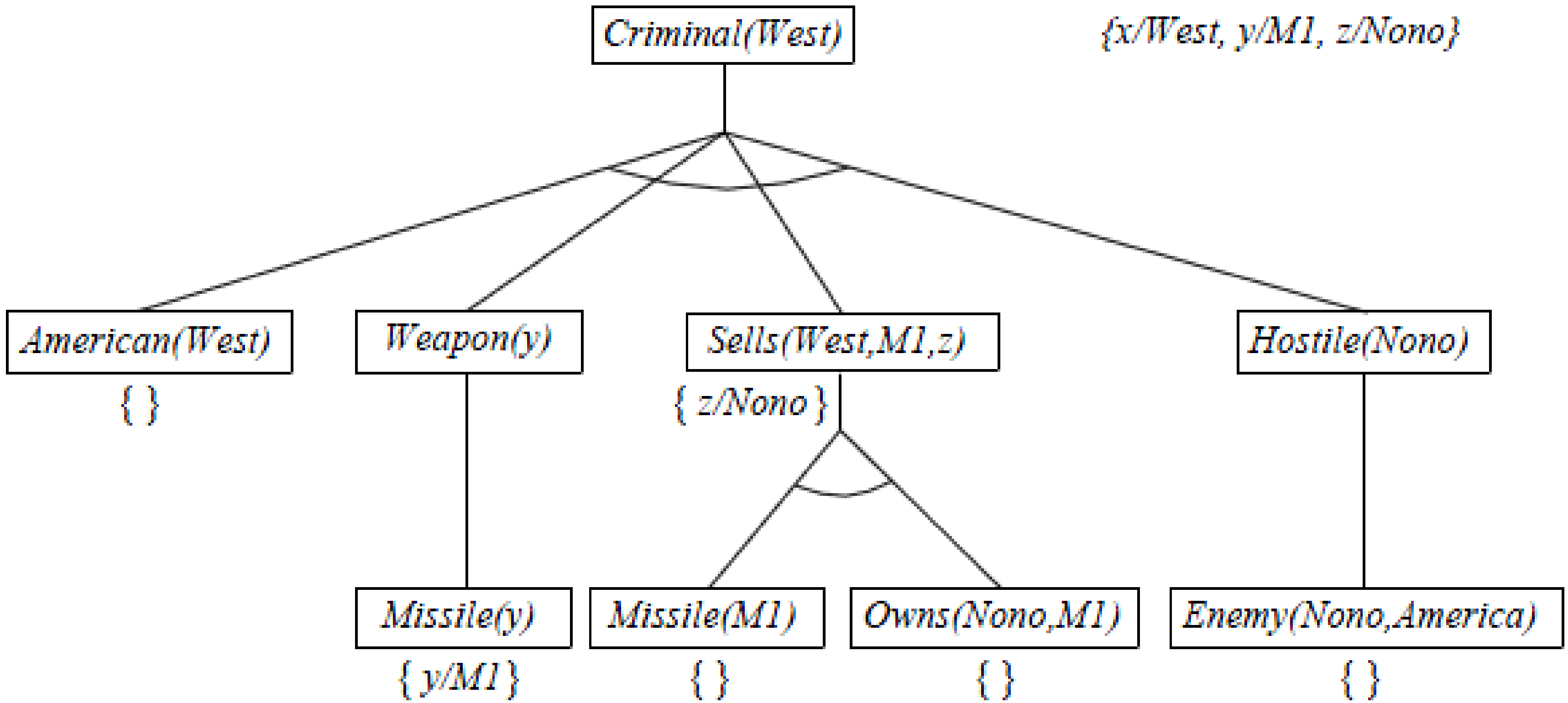
# Backward chaining example



# 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 for logic programming（逻辑程序设计）

# Completeness of FC/BC for General FOL

- FC and BC are complete for Horn KBs but are incomplete for general FOL KBs:

```
PhD(x)   ⇒ HighlyQualified(x)
¬PhD(x)  ⇒ EarlyEarnings(x)
HighlyQualified(x) ⇒ Rich(x)
EarlyEarnings(x)   ⇒ Rich(x)
Query: Rich(Me)
```

- Can't prove query with FC or BC. Why?
- Does a complete algorithm for FOL exist?

# Resolution algorithm

- Recall: KB operation boil down to satisfiability

  $KB \models \alpha$ if and only if ($KB \wedge \neg \alpha$) is unsatisfiable

- Algorithm: resolution-based inference
  - Convert all formulas to CNF
  - Repeatedly apply resolution rule
  - Return unsatisfaible iff derive false — empty clause

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

The two clauses are assumed to be standardized apart so that they share no variables.——假设两个子句已经标准化分离，没有共享变量。

For example,

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

with θ = {x/Ken}

Apply resolution steps to CNF(KB $\wedge$ ¬α); 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—消除蕴涵

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

2. Move ¬ inwards —将¬内移: $\neg\forall x\ p \equiv \exists x\ \neg p,\ \ \neg\exists x\ p \equiv \forall x\ \neg p$

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

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

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

# Conversion to CNF contd.

3. Standardize variables—变量标准化: 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（斯科伦函数）of the enclosing universally quantified variables:

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

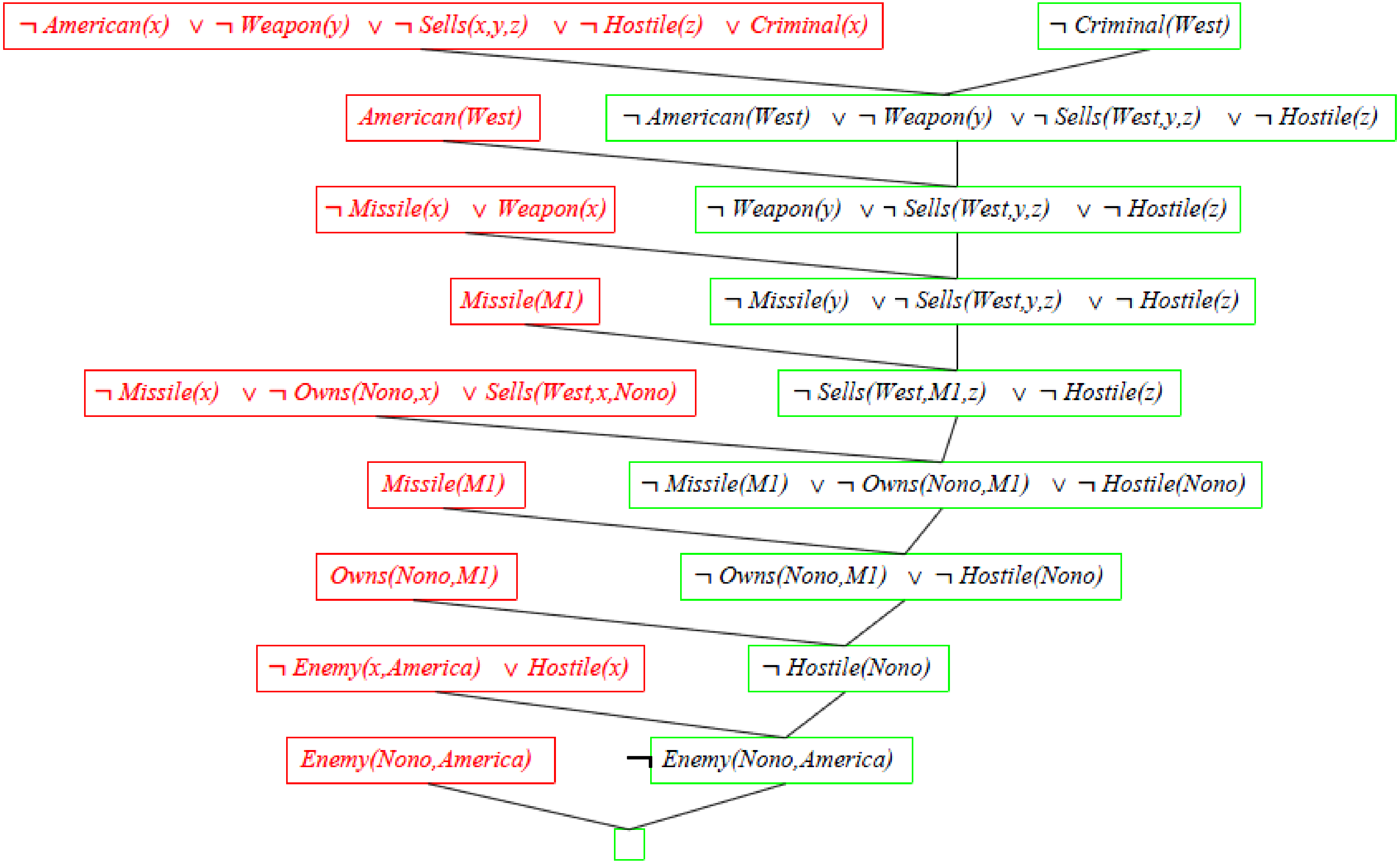
5. Drop universal quantifiers—去除全称量词:

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

6. Distribute $\vee$ over $\wedge$—将$\vee$分配到$\wedge$中:

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

# Resolution proof: definite clauses



# A brief history of reasoning

| | | |
|---|---|---|
| 450 B.C. | Stoics | propositional logic, inference (maybe) |
| 322 B.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 |

# Summary

一阶逻辑中的逻辑推理

命题化推理问题/Reducing first-order inference to propositional inference

  效率较低

合一/ Unification

  用于确定适当的变量置换

一般化分离规则/ Generalized Modus Ponens

  确定子句/definite clauses

  可靠的，完备的

  应用于前向链接和反向链接算法

前向链接，反向链接

归结推理/Resolution

# Summary

**Propositional logic**

- Model checking

← propositionalization

- Modus ponens
  (Horn clauses)

- Resolution (general)

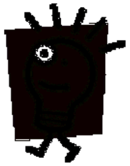
**First-order logic**

- n/a

- Modus ponens++
  (Horn clauses)

- Resolution++ (general)

++: unification and substitution



Key idea: variables in first-order logic

Variables yield compact knowledge representations.

# 作业

- 9.3，9.4，9.9，9.10(a,a,b)（第二版）=9.3，9.4，9.6，9.13(a,b,c)（第三版）