USTC

Chapter 9
Inference in First-order Logic

王子磊 (Zilei Wang)
Email: zlwang@ustc.edu.cn
http://vim.ustc.edu.cn

提纲

- 将一阶推理退化为命题推理
- 合一 (Unification)
- 一般化假言推理规则
- 前向链接
- 后向连接
- 归结

全称量词实例化 (Universal instantiation, UI)

- 全称量词语句的每个实例都是它蕴涵的：

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

对任一变量 v 和真实项 g 都成立

- E.g., $\forall x \;\; King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ 产生：

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

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

$King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$

.

.

.

存在量词实例化 (Existential instantiation, EI)

- 对任一语句 α, 变量 v 和常量符号 k (它们不会出现在 *KB* 的其他地方)

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

- E.g., $\exists x\ Crown(x) \land OnHead(x,John)$ 生成：

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

这里，C_1 是一个新的常量符号，称为 Skolem 常数

- 全称量词可以多次实例化获得不同的新语句（新旧 *KB* 是等价的）
- 存在量词只能实例化一次（新旧 *KB* 逻辑上不等价，但是推理上等价）

退化到命题推理

- 假设 KB 包含以下语句：

  $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

  $King(John)$

  $Greedy(John)$

  $Brother(Richard, John)$

- 用**所有的**方式实例化全称量词，则有：

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

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

  $King(John)$

  $Greedy(John)$

  $Brother(Richard, John)$

- 新 KB 是命题化的，命题符号是：

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

退化到命题推理

- 声明：每个 *FOL KB* 能够命题化并保留蕴涵
  - 一个真语句被新 *KB* 蕴涵 当且仅当 被原 *KB* 蕴涵
- 基本思想：命题化 *KB* 和查询，应用归结方法获得结果
- 问题：如果有函数符号，将会有无穷多的语句项
  - e.g., *Father*(*Father*(*Father*(*John*)))

退化到命题推理

- 定理：Herbrand (1930). 如果一个语句 α 被一个FOL *KB* 蕴涵，那么它被命题化 *KB* 的一个有限子集所蕴涵
- 基本思想：
  - For $n = 0$ to ∞ do
  - create a propositional *KB* by instantiating with depth-n terms
  - see if α is entailed by this *KB*
- 问题：如果 α 被蕴涵能够工作，但如果 α 不被蕴涵将会无限循环

- 定理：Turing (1936), Church (1936). FOL 的蕴涵是半可判定的 (semidecidable)
  - 存在能够证明蕴涵成立语句的算法，但不存在否定蕴涵不成立语句的算法

命题化方法的问题

- 命题化看起来产生了很多无关的语句
  - E.g., from:

    $$\forall x\, King(x) \wedge Greedy(x) \Rightarrow Evil(x)$$

    $$King(John)$$

    $$\forall y\, Greedy(y)$$

    $$Brother(Richard,John)$$

  - $Evil(John)$ 是明显的，但是命题化产生了很多无用的事实，比如：$Greedy(Richard)$ 就是无关的
- p 个 k 元的谓词和 n 个常量，将产生 $p \cdot n^k$ 个实例

合一 (Unification)

- 如果能够找到一个替换 θ， 使得 *King*(*x*) 和 *Greedy*(*x*) 匹配 *King*(*John*) 和 *Greedy*(*y*)，则我们能够直接推理
  - $\theta = \{x/John, y/John\}$ works

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

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

标准化分离(Standardizing apart) 来消除变量的名称冲突, e.g., *Knows*(z_{17},*OJ*)

合一 (Unification)

- 合一 $Knows(John,x)$ 和 $Knows(y,z)$

  $\theta = \{y/John,\ x/z\}$

  or $\theta = \{y/John,\ x/John,\ z/John\}$

 第一种比第二种更一般化

- 存在一个最一般合一置换 (most general unifier, MGU) ,不考虑变量重命名情况下它是唯一的

  MGU = { $y/John,\ x/z$ }

合一算法

function UNIFY(x, y, θ) **returns** a substitution to make x and y identical
  inputs: x, a variable, constant, list, or compound
    y, a variable, constant, list, or compound
    θ, the substitution built up so far

if θ = failure **then return** failure
else if $x = y$ **then return** θ
}失败与成功

else if VARIABLE?(x) **then return** UNIFY-VAR(x, y, θ)
else if VARIABLE?(y) **then return** UNIFY-VAR(y, x, θ)
}变量合一

else if COMPOUND?(x) **and** COMPOUND?(y) **then**
  return UNIFY(ARGS[x], ARGS[y], UNIFY(OP[x], OP[y], θ))
  首先操作合一
else if LIST?(x) **and** LIST?(y) **then**
  return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], θ))
  第一项
}依次转化为变量合一

else return failure

合一算法

function UNIFY-VAR(var, x, θ) **returns** a substitution
  inputs: var, a variable
    x, any expression
    θ, the substitution built up so far

  if $\{var/val\} \in \theta$ **then return** UNIFY(val, x, θ)
  else if $\{x/val\} \in \theta$ **then return** UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) **then return** failure
  else return add $\{var/x\}$ to θ

已经存在

发生检验：该变量是否在复合项中出现？

$S(x)$ 与 $S(S(x))$ 无法合一

一般化假言推理规则 (Generalized Modus Ponens, GMP)

$$\frac{p_1', p_2', \ldots, p_n', \quad (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta}$$ 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 采用确定子句的知识库 (*KB* of definite clauses)
  - exactly one positive literal
- 假设所有的变量是 全称量化的

GMP 的可靠性

- 需要证明

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

给定 $p_i'\theta = p_i\theta$ for all i

- 引理：对任一给定子句 p, 通过全称量化有 $p \models p\theta$

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. 根据前两条，假言推理就能够得到 $q\theta$

知识库样例

- 已知事实：
  - 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
  - All of its missiles were sold to it by Colonel West, who is American.
- 证明：Col. West is a criminal

知识库样例

... 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 …

$Enemy(Nono,America)$

前向链接算法

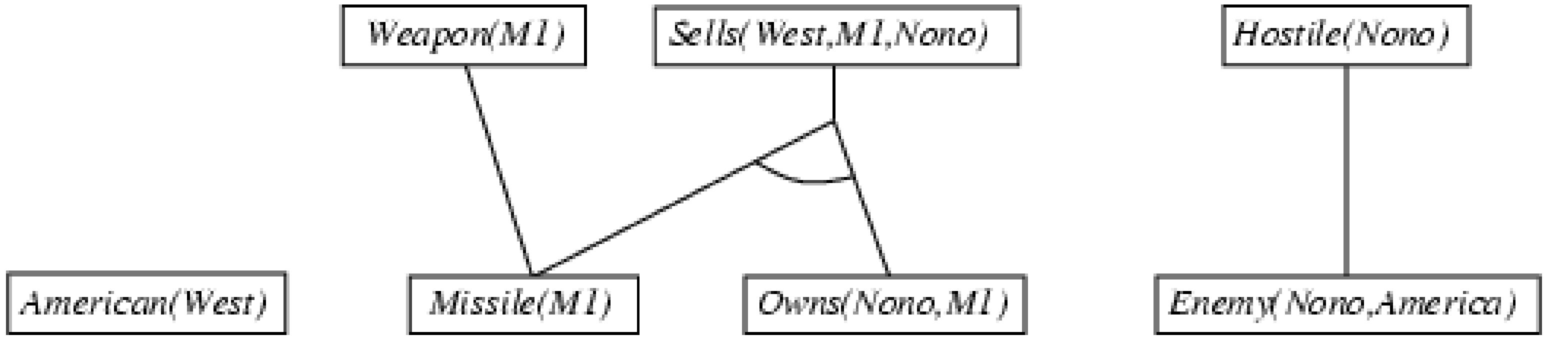
function FOL-FC-ASK(KB, α) **returns** a substitution or $false$

repeat until new is empty　直到无法产生新语句（不动点）
　$new \leftarrow \{\}$
　for each sentence r in KB **do**
　　$(p_1 \land \ldots \land p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART(r)　标准化分离
　　for each θ such that $(p_1 \land \ldots \land p_n)\theta = (p'_1 \land \ldots \land p'_n)\theta$
　　全部　for some $p'_1, \ldots, p'_n$ in KB　与前提合一
　　　$q' \leftarrow$ SUBST(θ, 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', α)
　　　　if ϕ is not $fail$ **then return** ϕ
　　　　与结论合一
　add new to KB
return $false$

前向链接证明

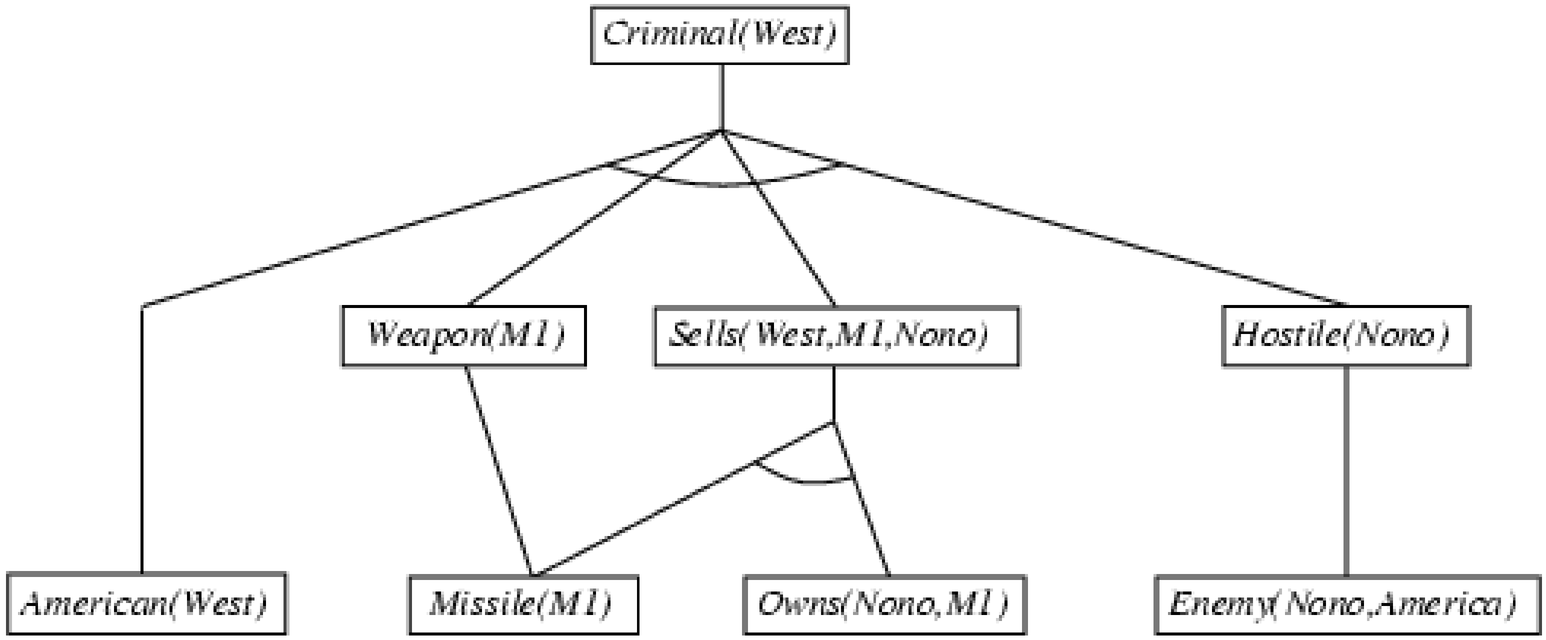
American(West) | Missile(M1) | Owns(Nono,M1) | Enemy(Nono,America)

前向链接证明



Weapon(M1)
Sells(West,M1,Nono)
Hostile(Nono)
American(West)
Missile(M1)
Owns(Nono,M1)
Enemy(Nono,America)

前向链接证明


Criminal(West)
Weapon(M1)
Sells(West,M1,Nono)
Hostile(Nono)
American(West)
Missile(M1)
Owns(Nono,M1)
Enemy(Nono,America)

前向链接的特点

- 对一阶确定子句是可靠的和完备的

- Datalog = first-order definite clauses + no functions
  - FC 能够在有限迭代次数内结束
  - 如果 α 不是蕴涵句，则通常不能结束

- 不可避免的：确定子句的蕴涵是半可判定的

前向链接的效率

递增的前向链接：在第 k 步迭代中，如果在 k-1 步没有增加一个前提，则该规则不必进行匹配

⇒ 只匹配前提中至少增加了一个新正文字的那些规则

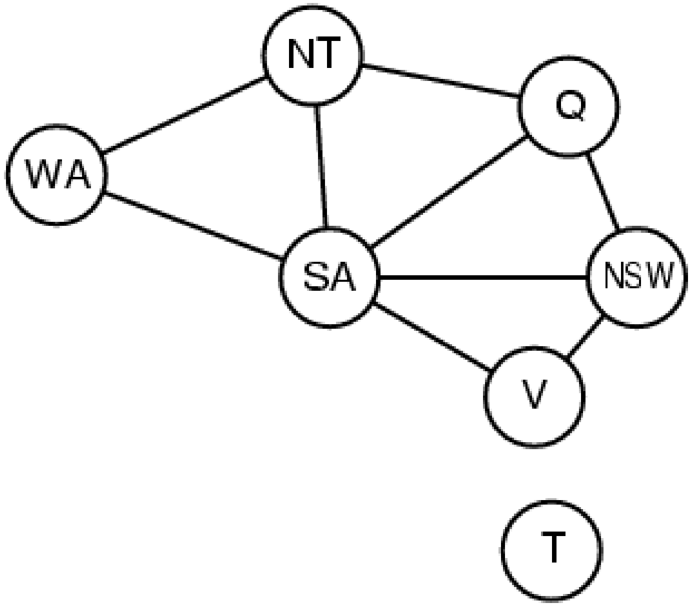
匹配本身是非常费时的

数据库索引提取已知事实的时间复杂性可以做到 $O(1)$

- e.g., query *Missile*(*x*) retrieves $Missile(M_1)$

前向链接在演绎数据库(deductive databases)中广泛使用

困难匹配示例 —— 着色问题 *



Diff(*wa*,*nt*) ∧ *Diff*(*wa*,*sa*) ∧ *Diff*(*nt*,*q*) ∧
Diff(*nt*,*sa*) ∧ *Diff*(*q*,*nsw*) ∧ *Diff*(*q*,*sa*) ∧
Diff(*nsw*,*v*) ∧ *Diff*(*nsw*,*sa*) ∧ *Diff*(*v*,*sa*)
⇒ *Colorable*()

Diff(*Red*,*Blue*)	*Diff* (*Red*,*Green*)
Diff(*Green*,*Red*)	*Diff*(*Green*,*Blue*)
Diff(*Blue*,*Red*)	*Diff*(*Blue*,*Green*)

- *Colorable*() 是可推理的，当且仅当 CSP 有解
- CSPs 包含 3SAT，因而匹配是 NP-hard 的

反向链接算法

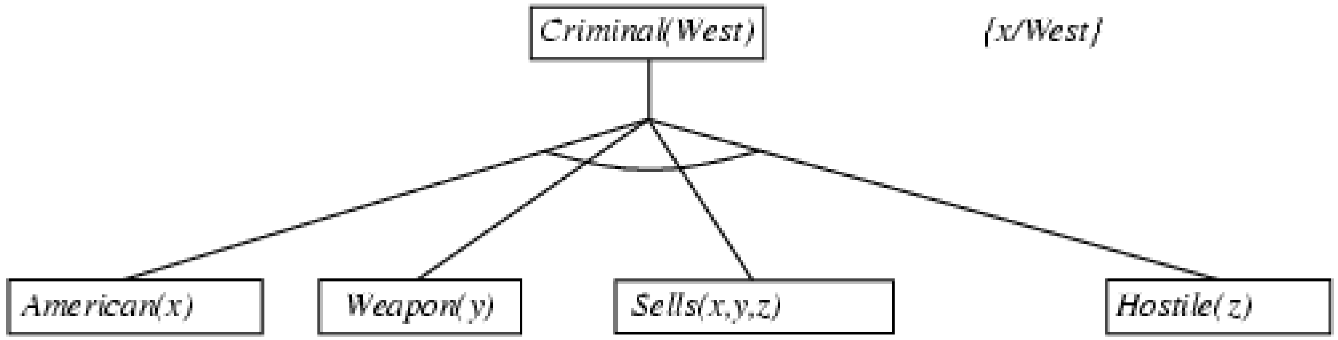
```
function FOL-BC-Ask(KB, goals, θ) returns a set of substitutions
    inputs: KB, a knowledge base
            goals, a list of conjuncts forming a query (θ already applied)
            θ, the current substitution, initially the empty substitution { }
    local variables: answers, a set of substitutions,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

    if goals is empty then return {θ}                      成功
    q′ ← SUBST(θ, FIRST(goals))
    for each sentence r in KB
            where STANDARDIZE-APART(r) = ( p1 ∧ ... ∧ pn ⇒ q)
            and θ′ ← UNIFY(q, q′) succeeds                与结论合一
        new_goals ← [ p1, ..., pn|REST(goals)]
        answers ← FOL-BC-Ask(KB, new_goals, COMPOSE(θ′, θ)) ∪ answers
    return answers                                        深度优先递归
```

$$SUBST(COMPOSE(\theta_1, \theta_2), p) = SUBST(\theta_2, SUBST(\theta_1, p))$$

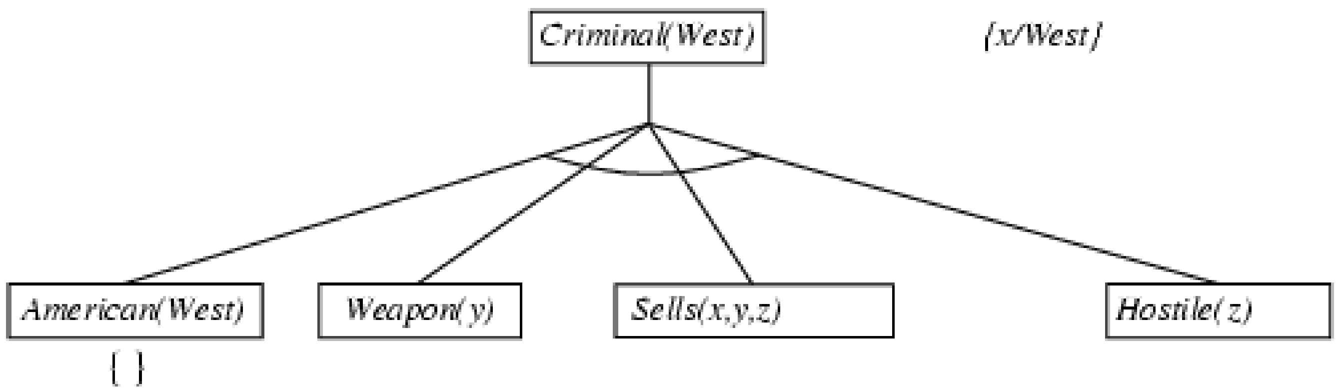
反向链接示例

$Criminal(West)$

反向链接示例

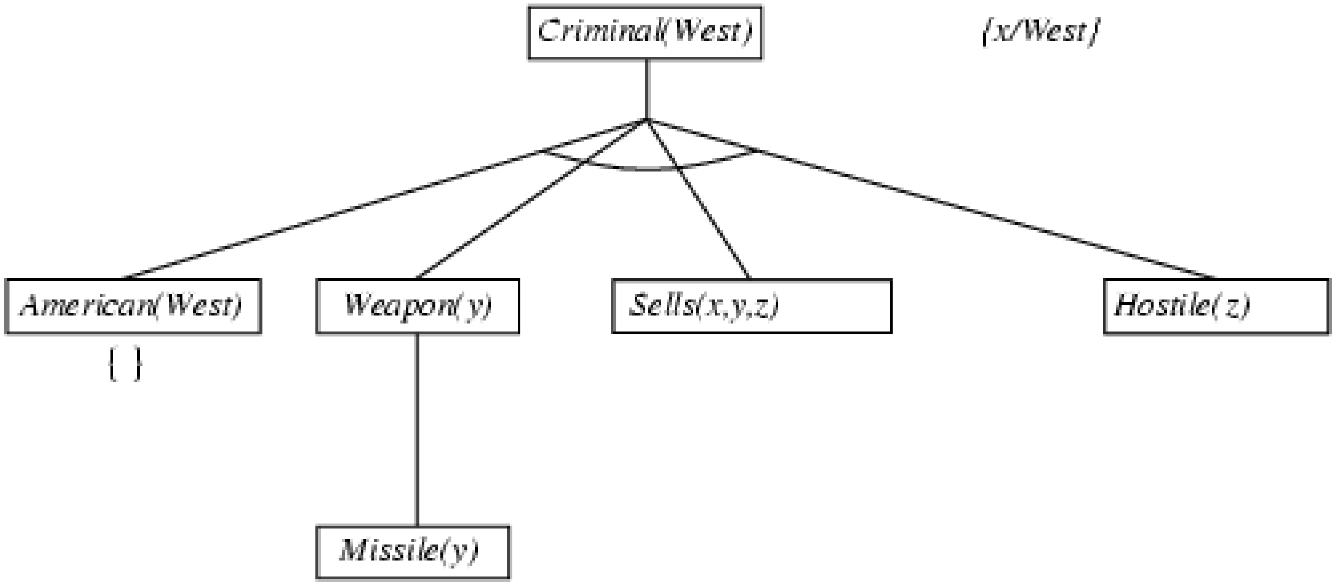


反向链接示例



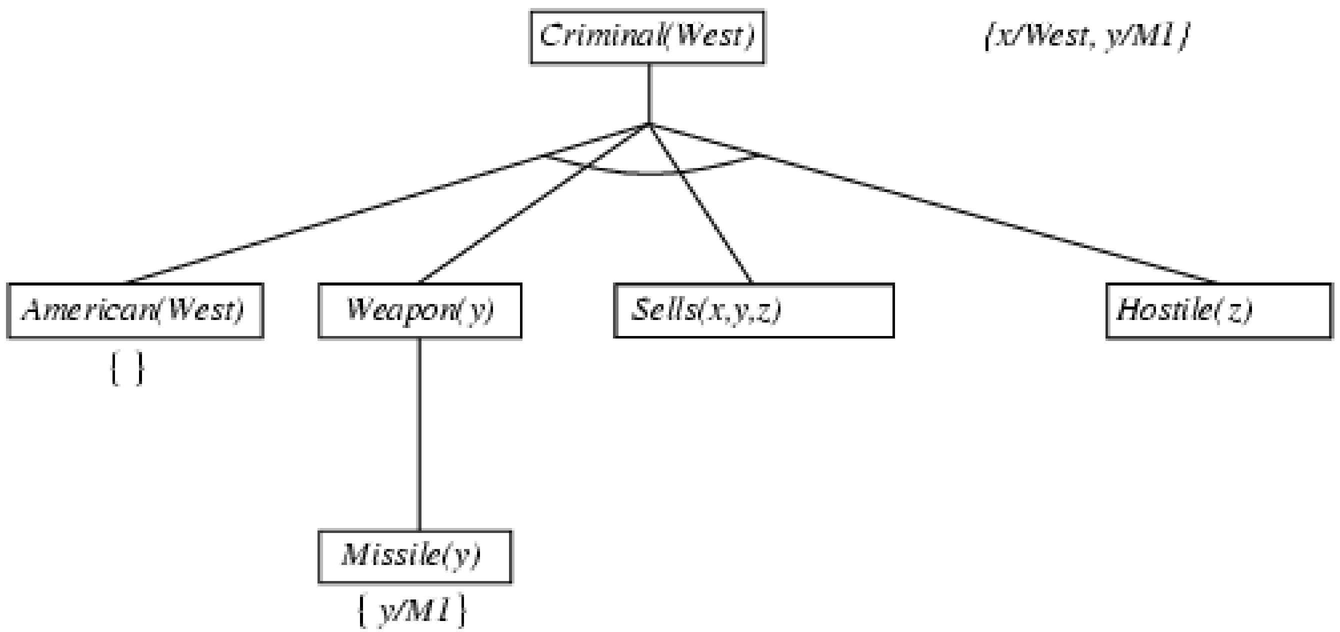
Criminal(West)
{x/West}
American(West)
{ }
Weapon(y)
Sells(x,y,z)
Hostile(z)

反向链接示例

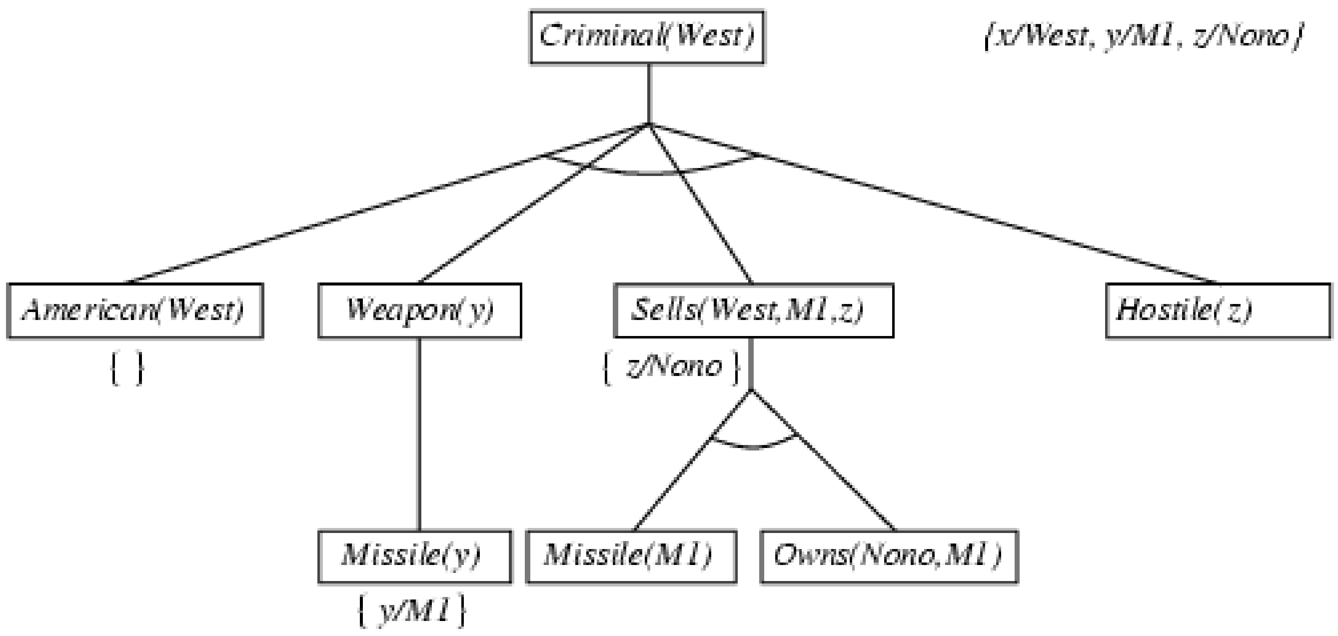


Criminal(West)
{x/West}
American(West)
{ }
Weapon(y)
Sells(x,y,z)
Hostile(z)
Missile(y)

反向链接示例

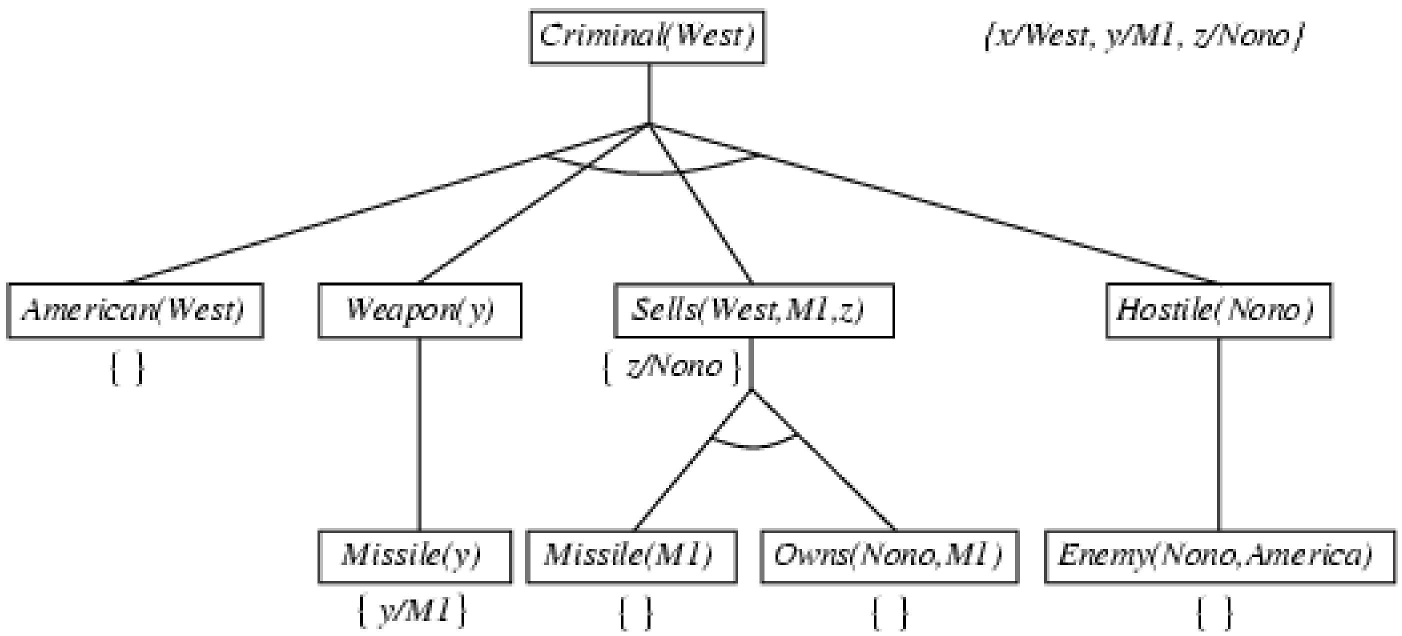

Criminal(West)
{x/West, y/M1}
American(West)
{ }
Weapon(y)
Sells(x,y,z)
Hostile(z)
Missile(y)
{ y/M1 }

反向链接示例



Criminal(West)
{x/West, y/M1, z/Nono}
American(West)
{ }
Weapon(y)
Sells(West,M1,z)
{ z/Nono }
Hostile(z)
Missile(y)
{ y/M1 }
Missile(M1)
Owns(Nono,M1)

反向链接示例


Criminal(West)
{x/West, y/M1, z/Nono}
American(West)
{ }
Weapon(y)
Sells(West,M1,z)
{ z/Nono }
Hostile(Nono)
Missile(y)
{ y/M1 }
Missile(M1)
{ }
Owns(Nono,M1)
{ }
Enemy(Nono,America)
{ }

反向链接的特点

- 深度优先的递归证明搜索
  - 空间与所需证明数呈线性关系
- 由于可能的无限循环，该算法是不完备的

  ⇒ 检查当前目标与堆栈中的每个目标是否相同
- 由于重复的子目标，对成功和失败两种情况都不高效

  ⇒ 使用推理结果缓存技术 (需要额外的存储空间）
- 广泛应用于逻辑编程 (logic programming)

逻辑编程：Prolog *

- 算法 = 逻辑 + 控制
- 基础：Horn子句的反向链接 + bells & whistles

  曾在 Europe, Japan (basis of 5th Generation project) 广泛使用

Program = set of clauses = head :- literal$_1$, … *literal*$_n$.

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

- 深度优先自左向右的反向链接
- 内建数学谓词等, e.g., X is Y*Z+3
- 封闭世界假说 ("negation as failure")
  - e.g., *alive*(*X*) :- not *dead*(*X*).
  - *alive*(*joe*) 成功，如果 *dead*(*joe*) 失败

Prolog *

- 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=[]
```

归结：简要总结

- 完全一阶逻辑的版本：

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

  这里 $\text{UNIFY}(\ell_i, \neg m_j) = \theta$.

  两个子句被假定已经标准化分离，因而它们不共享变量

- 举例：

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

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

- 在 $CNF(KB \wedge \neg\alpha)$ 上应用归结过程完成证明，对 FOL 是完备的

转换为 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. 消除双向和单向蕴含句

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

2. 将否定词 ¬ 内移: $\neg\forall x\ p \equiv \exists x\ \neg p,\ \neg\exists x\ p \equiv \forall x\ \neg p$

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

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

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

转换为 CNF

3. 标准化变量：每个量词应该使用不同的变量

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

4. Skolemize：消除存在量词

   每个存在变量用全称量词变量的 Skolem function 替代

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

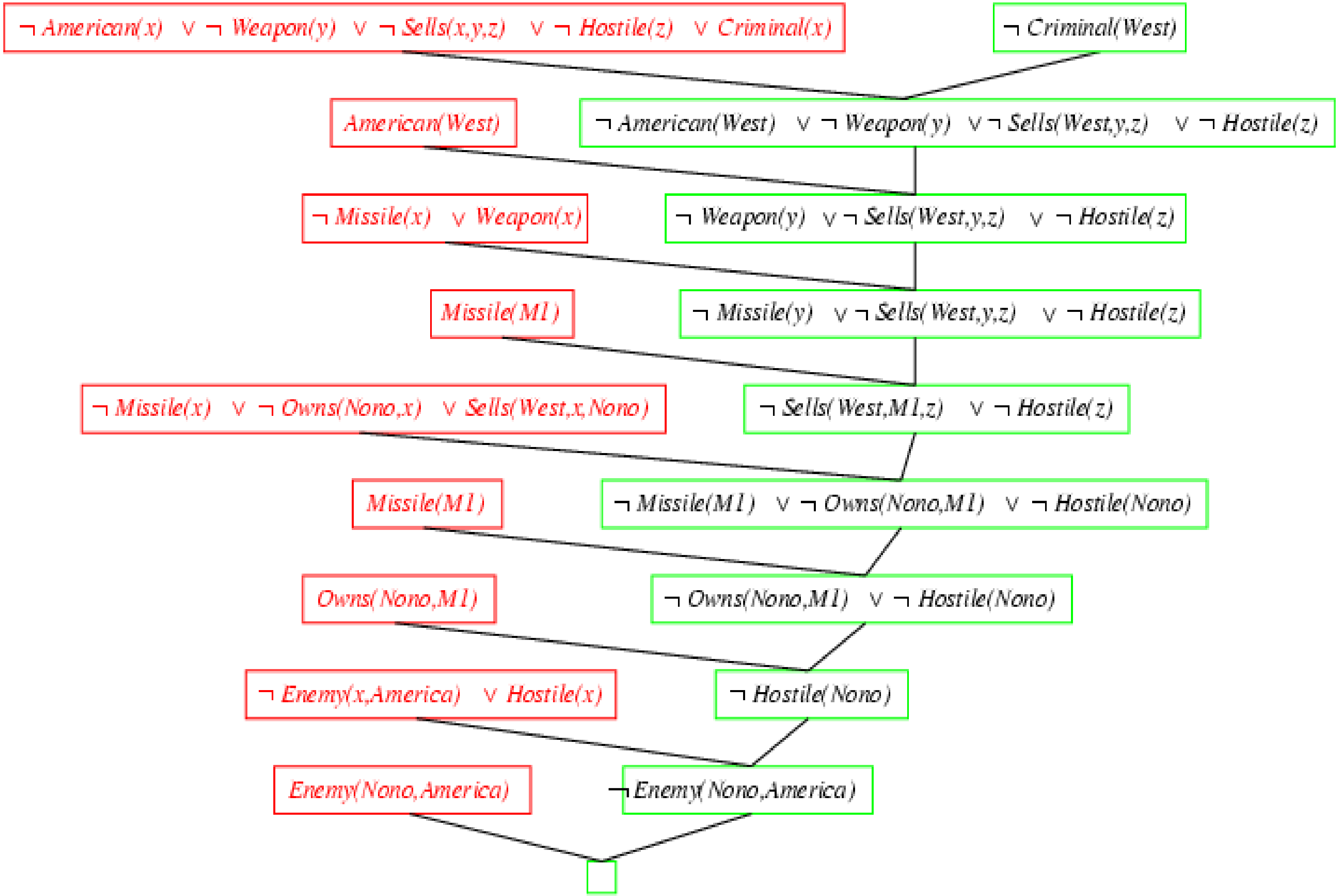
5. 删除全称量词

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

6. 将 ∨ 分配到 ∧ 中

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

归结证明：确定子句


¬ American(x) ∨ ¬ Weapon(y) ∨ ¬ Sells(x,y,z) ∨ ¬ Hostile(z) ∨ Criminal(x)
¬ Criminal(West)
American(West)
¬ American(West) ∨ ¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ Weapon(x)
¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ ¬ Owns(Nono,x) ∨ Sells(West,x,Nono)
¬ Sells(West,M1,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(M1) ∨ ¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
Owns(Nono,M1)
¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
¬ Enemy(x,America) ∨ Hostile(x)
¬ Hostile(Nono)
Enemy(Nono,America)
¬ Enemy(Nono,America)

谢谢聆听！