

# INFERENZA NELLA LOGICA DEL PRIMO ORDINE

## CAPITOLO 9

– *docente: Alessandro Sperduti* –

– *presentazione basata sui lucidi di S. Russell* –

# Outline

- ◇ Riduzione della inferenza di primo ordine alla inferenza proposizionale
- ◇ Unificazione
- ◇ Modus Ponens generalizzato
- ◇ Forward e backward chaining
- ◇ Programmazione Logica
- ◇ Risoluzione

## Istanziamento Universale (UI)

Ogni istanziazione di una sentenza quantificata universalmente è conseguenza logica di quest'ultima:

$$\frac{\forall v \alpha}{\text{SUBST}(\{v/g\}, \alpha)}$$

per ogni variabile  $v$  e termine ground  $g$

P.e.,  $\forall x \text{ Re}(x) \wedge \text{Ingordo}(x) \Rightarrow \text{Diavolo}(x)$  porta a

$$\text{Re}(\text{Giovanni}) \wedge \text{Ingordo}(\text{Giovanni}) \Rightarrow \text{Diavolo}(\text{Giovanni})$$

$$\text{Re}(\text{Riccardo}) \wedge \text{Ingordo}(\text{Riccardo}) \Rightarrow \text{Diavolo}(\text{Riccardo})$$

$$\text{Re}(\text{Padre}(\text{Giovanni})) \wedge \text{Ingordo}(\text{Padre}(\text{Giovanni})) \Rightarrow \text{Diavolo}(\text{Padre}(\text{Giovanni}))$$

⋮

## Istanziamento Esistenziale (EI)

Per ogni sentenza  $\alpha$ , variabile  $v$ , e simbolo costante  $k$   
*che non appare in nessuna parte della base di conoscenza:*

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

P.e.,  $\exists x \text{ Corona}(x) \wedge \text{SullaTesta}(x, \text{Giovanni})$  porta a

$$\text{Corona}(C_1) \wedge \text{SullaTesta}(C_1, \text{Giovanni})$$

a patto che  $C_1$  sia un nuovo simbolo di costante, chiamato **costante di Skolem**

Altro esempio: da  $\exists x d(x^y)/dy = x^y$  si ottiene

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

a patto che  $e$  sia un nuovo simbolo di costante

## Istanziamento Esistenziale

UI può essere applicata più volte per *aggiungere* nuove sentenze;  
la nuova KB è logicamente equivalente alla vecchia

EI può essere applicata solo una volta per *rimpiazzare* la sentenza esistenziale;  
la nuova KB *non* è equivalente alla vecchia,  
ma è soddisfacibile sse la vecchia KB era soddisfacibile

## Riduzione alla inferenza proposizionale

Supponiamo che KB contenga solo le seguenti sentenze:

$$\forall x \text{ Re}(x) \wedge \text{Ingordo}(x) \Rightarrow \text{Diavolo}(x)$$

$$\text{Re}(\text{Giovanni})$$

$$\text{Ingordo}(\text{Giovanni})$$

$$\text{Fratello}(\text{Riccardo}, \text{Giovanni})$$

Istanziando la sentenza universale in *tutti i possibili* modi, si ottiene

$$\text{Re}(\text{Giovanni}) \wedge \text{Ingordo}(\text{Giovanni}) \Rightarrow \text{Diavolo}(\text{Giovanni})$$

$$\text{Re}(\text{Riccardo}) \wedge \text{Ingordo}(\text{Riccardo}) \Rightarrow \text{Diavolo}(\text{Riccardo})$$

$$\text{Re}(\text{Giovanni})$$

$$\text{Ingordo}(\text{Giovanni})$$

$$\text{Fratello}(\text{Riccardo}, \text{Giovanni})$$

La nuova KB è **proposizionalizzata**: i simboli proposizionali sono

$\text{Re}(\text{Giovanni})$ ,  $\text{Ingordo}(\text{Giovanni})$ ,  $\text{Diavolo}(\text{Giovanni})$ ,  $\text{Re}(\text{Riccardo})$  etc.

# Riduzione

Affermazione: una sentenza ground è conseguenza logica della nuova KB sse è conseguenza logica della KB originaria

Affermazione: ogni FOL KB può essere proposizionalizzata in modo da preservarne le conseguenze logiche

Idea: proposizionalizzare sia KB che la query, applicare la risoluzione, restituire il risultato

Problema: con i simboli di funzione, c'è un numero infinito di termini ground,  
e.g.,  $Padre(Padre(Padre(Giovanni)))$

Teorema: Herbrand (1930). Se una sentenza  $\alpha$  è conseguenza logica di una FOL KB, essa è conseguenza logica di un sottoinsieme *finito* della KB proposizionale

Idea: For  $n = 0$  to  $\infty$  do  
    creare una KB proposizionale con termini di profondità  $n$   
    controllare se  $\alpha$  è conseguenza logica della KB considerata

Problema: funziona solo se  $\alpha$  è conseguenza logica, cicla se  $\alpha$  non è conseguenza logica

Teorema: Turing (1936), Church (1936), “entailment” in FOL è *semidecidibile*

## Problemi con la proposizionalizzazione

La proposizionalizzazione sembra generare tante sentenze irrilevanti

P.e.,

$$\forall x \text{ Re}(x) \wedge \text{Ingordo}(x) \Rightarrow \text{Diavolo}(x)$$

$$\text{Re}(\text{Giovanni})$$

$$\forall y \text{ Ingordo}(y)$$

$$\text{Fratello}(\text{Riccardo}, \text{Giovanni})$$

ok per  $\text{Diavolo}(\text{Giovanni})$ , ma la proposizionalizzazione produce tanti fatti come  $\text{Ingordo}(\text{Riccardo})$  che sono irrilevanti

Con  $p$  predicati  $k$ -ari e  $n$  costanti, ci sono  $p \cdot n^k$  istanziazioni!



## Unificazione

Si può ottenere l'inferenza immediatamente se possiamo trovare una sostituzione  $\theta$  tale che  $Re(x)$  e  $Ingordo(x)$  corrispondano a  $Re(Giovanni)$  e  $Ingordo(y)$

$\theta = \{x/Giovanni, y/Giovanni\}$  va bene

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

$p$	$q$	$\theta$
$Conosce(Giovanni, x)$	$Conosce(Giovanni, Giovanna)$	
$Conosce(Giovanni, x)$	$Conosce(y, OJ)$	
$Conosce(Giovanni, x)$	$Conosce(y, Madre(y))$	
$Conosce(Giovanni, x)$	$Conosce(x, OJ)$	

## Unificazione

Si può ottenere l'inferenza immediatamente se possiamo trovare una sostituzione  $\theta$  tale che  $Re(x)$  e  $Ingordo(x)$  corrispondano a  $Re(Giovanni)$  e  $Ingordo(y)$

$\theta = \{x/Giovanni, y/Giovanni\}$  va bene

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

$p$	$q$	$\theta$
$Conosce(Giovanni, x)$	$Conosce(Giovanni, Giovanna)$	$\{x/Giovanna\}$
$Conosce(Giovanni, x)$	$Conosce(y, OJ)$	
$Conosce(Giovanni, x)$	$Conosce(y, Madre(y))$	
$Conosce(Giovanni, x)$	$Conosce(x, OJ)$	

## Unificazione

Si può ottenere l'inferenza immediatamente se possiamo trovare una sostituzione  $\theta$  tale che  $Re(x)$  e  $Ingordo(x)$  corrispondano a  $Re(Giovanni)$  e  $Ingordo(y)$

$\theta = \{x/Giovanni, y/Giovanni\}$  va bene

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

$p$	$q$	$\theta$
$Conosce(Giovanni, x)$	$Conosce(Giovanni, Giovanna)$	$\{x/Giovanna\}$
$Conosce(Giovanni, x)$	$Conosce(y, OJ)$	$\{x/OJ, y/Giovanni\}$
$Conosce(Giovanni, x)$	$Conosce(y, Madre(y))$	
$Conosce(Giovanni, x)$	$Conosce(x, OJ)$	

# Unificazione

Si può ottenere l'inferenza immediatamente se possiamo trovare una sostituzione  $\theta$  tale che  $Re(x)$  e  $Ingordo(x)$  corrispondano a  $Re(Giovanni)$  e  $Ingordo(y)$

$\theta = \{x/Giovanni, y/Giovanni\}$  va bene

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

$p$	$q$	$\theta$
$Conosce(Giovanni, x)$	$Conosce(Giovanni, Giovanna)$	$\{x/Giovanna\}$
$Conosce(Giovanni, x)$	$Conosce(y, OJ)$	$\{x/OJ, y/Giovanni\}$
$Conosce(Giovanni, x)$	$Conosce(y, Madre(y))$	$\{y/Giovanni, x/Madre(Giovanni)\}$
$Conosce(Giovanni, x)$	$Conosce(x, OJ)$	

## Unificazione

Si può ottenere l'inferenza immediatamente se possiamo trovare una sostituzione  $\theta$  tale che  $Re(x)$  e  $Ingordo(x)$  corrispondano a  $Re(Giovanni)$  e  $Ingordo(y)$

$\theta = \{x/Giovanni, y/Giovanni\}$  va bene

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

$p$	$q$	$\theta$
$Conosce(Giovanni, x)$	$Conosce(Giovanni, Giovanna)$	$\{x/Giovanna\}$
$Conosce(Giovanni, x)$	$Conosce(y, OJ)$	$\{x/OJ, y/Giovanni\}$
$Conosce(Giovanni, x)$	$Conosce(y, Madre(y))$	$\{y/Giovanni, x/Madre(Giovanni)\}$
$Conosce(Giovanni, x)$	$Conosce(x, OJ)$	$fallimento$

**Standardizing apart** elimina sovrapposizioni di variabili, p.e.,  $Conosce(z_{17}, OJ)$

# Unificazione

Si può ottenere l'inferenza immediatamente se possiamo trovare una sostituzione  $\theta$  tale che  $Re(x)$  e  $Ingordo(x)$  corrispondano a  $Re(Giovanni)$  e  $Ingordo(y)$

$\theta = \{x/Giovanni, y/Giovanni\}$  va bene

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

$p$	$q$	$\theta$
$Conosce(Giovanni, x)$	$Conosce(Giovanni, Giovanna)$	$\{x/Giovanna\}$
$Conosce(Giovanni, x)$	$Conosce(y, OJ)$	$\{x/OJ, y/Giovanni\}$
$Conosce(Giovanni, x)$	$Conosce(y, Madre(y))$	$\{y/Giovanni, x/Madre(Giovanni)\}$
$Conosce(Giovanni, x)$	$Conosce(x, OJ)$	<i>fallimento</i>
<b>Standardizing apart</b> elimina sovrapposizioni di variabili, p.e., $Conosce(z_{17}, OJ)$		
$Conosce(Giovanni, x)$	$Conosce(z_{17}, OJ)$	$\{z_{17}/Giovanni, x/OJ\}$

# Algoritmo di Unificazione

**function** UNIFY( $x, y, \theta$ ) **returns** una sostituzione che rende  $x$  e  $y$  identici (dato  $\theta$ )

**inputs:**  $x$ , una variabile, costante, lista, o espressione composta

$y$ , una variabile, costante, lista, o espressione composta

$\theta$ , la sostituzione corrente, inizialmente vuota

**if**  $\theta = \text{fallimento}$  **then return** fallimento

**else if**  $x = y$  **then return**  $\theta$

**else if** **VARIABILE?**( $x$ ) **then return** UNIFY-VAR( $x, y, \theta$ )

**else if** **VARIABILE?**( $y$ ) **then return** UNIFY-VAR( $y, x, \theta$ )

**else if** **COMPOSTA?**( $x$ ) **and** **COMPOSTA?**( $y$ ) **then**

**return** UNIFY(ARGS[ $x$ ], ARGS[ $y$ ], UNIFY(OP[ $x$ ], OP[ $y$ ],  $\theta$ ))

**else if** **LISTA?**( $x$ ) **and** **LISTA?**( $y$ ) **then**

**return** UNIFY(RESTO[ $x$ ], RESTO[ $y$ ], UNIFY(PRIMO[ $x$ ], PRIMO[ $y$ ],  $\theta$ ))

**else return** fallimento

---

**function** UNIFY-VAR( $var, x, \theta$ ) **returns** una sostituzione

**inputs:**  $var$ , una variabile,  $x$  una qualsiasi espressione,  $\theta$  la sostituzione corrente

**if**  $\{var/val\} \in \theta$  **then return** UNIFY( $val, x, \theta$ )

**else if**  $\{x/val\} \in \theta$  **then return** UNIFY( $var, val, \theta$ )

**else if** **CONTROLLA-OCCORRENZA?**( $var, x$ ) **then return** fallimento

**else return** aggiungi  $\{var/x\}$  a  $\theta$

# Algoritmo di Unificazione **Modificato**

```
function UNIFY( $x, y, \theta$ ) returns una sostituzione che rende  $x$  e  $y$  identici (dato  $\theta$ )
  inputs:  $x$ , una variabile, costante, lista, o espressione composta
             $y$ , una variabile, costante, lista, o espressione composta
             $\theta$ , la sostituzione corrente, inizialmente vuota

  if  $\theta =$  fallimento then return fallimento
  else if  $x = y$  then return  $\theta$ 
  else if VARIABILE?( $x$ ) then return UNIFY-VAR( $x, y, \theta$ )
  else if VARIABILE?( $y$ ) then return UNIFY-VAR( $y, x, \theta$ )
  else if COMPOSTA?( $x$ ) and COMPOSTA?( $y$ ) then
    return UNIFY(ARGS[ $x$ ], ARGS[ $y$ ], UNIFY(OP[ $x$ ], OP[ $y$ ],  $\theta$ ))
  else if LISTA?( $x$ ) and LISTA?( $y$ ) then
    return UNIFY(SUBST( $\theta$ , RESTO[ $x$ ]), SUBST( $\theta$ , RESTO[ $y$ ]), UNIFY(SUBST( $\theta$ , PRIMO[ $x$ ]),
                                                                    SUBST( $\theta$ , PRIMO[ $y$ ]),  $\theta$ ))

  else return fallimento
```

---

```
function UNIFY-VAR( $var, x, \theta$ ) returns una sostituzione
  inputs:  $var$ , una variabile,  $x$  una qualsiasi espressione,  $\theta$  la sostituzione corrente

  if  $\{var/val\} \in \theta$  then return UNIFY( $val, x, \theta$ )
  else if  $\{x/val\} \in \theta$  then return UNIFY( $var, val, \theta$ )
  else if CONTROLLA-OCCORRENZA?( $var, x$ ) then return fallimento
  else return COMPOSE( $\{var/x\}, \theta$ )
```



## Modus Ponens Generalizzato (GMP)

$$\frac{p_1', p_2', \dots, p_n', (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)}{q\theta} \quad \text{dove } p_i'\theta = p_i\theta \text{ per tutte le } i$$

$p_1'$  è  $Re(Giovanni)$

$p_1$  è  $Re(x)$

$p_2'$  è  $Ingordo(y)$

$p_2$  è  $Ingordo(x)$

$\theta$  è  $\{x/Giovanni, y/Giovanni\}$   $q$  è  $Diavolo(x)$

$q\theta$  è  $Diavolo(Giovanni)$

GMP usato con KB composto di **clausole definite** (*esattamente* un letterale positivo)

Si assume che tutte le variabili siano quantificate universalmente

## Correttezza di GMP

Bisogna mostrare che

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

dato che  $p_i'\theta = p_i\theta$  per tutte le  $i$

Lemma: per ogni clausola definita  $p$ , abbiamo  $p \models p\theta$  per mezzo di UI

1.  $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \models (p_1 \wedge \dots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \dots \wedge p_n\theta \Rightarrow q\theta)$
2.  $p_1', \dots, p_n' \models p_1' \wedge \dots \wedge p_n' \models p_1'\theta \wedge \dots \wedge p_n'\theta$
3. da 1 e 2,  $q\theta$  segue da Modus Ponens ordinario

## Esempio di base di conoscenza

La legge in America dice che è un crimine per un americano vendere armi a nazioni ostili. La nazione Nono, nemico dell'America, possiede alcuni missili, venduti alla nazione dal Colonnello West, che è americano.

Provare che Col. West è un criminale

## Esempio di base di conoscenza

... è un crimine per un americano vendere armi a nazioni ostili:

## Esempio di base di conoscenza

... è un crimine per un americano vendere armi a nazioni ostili:

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

Nono ... possiede dei missili, cioè,  $\exists x \textit{Owns}(\textit{Nono}, x) \wedge \textit{Missile}(x)$ :

## Esempio di base di conoscenza

... è un crimine per un americano vendere armi a nazioni ostili:

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

Nono ... possiede dei missili, cioè,  $\exists x \textit{Owns}(\textit{Nono}, x) \wedge \textit{Missile}(x)$ :

$$\textit{Owns}(\textit{Nono}, M_1) \wedge \textit{Missile}(M_1)$$

... tutti i suoi missili gli sono stati venduti dal Colonnello West

## Esempio di base di conoscenza

... è un crimine per un americano vendere armi a nazioni ostili:

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

Nono ... possiede dei missili, cioè,  $\exists x \textit{Owns}(\textit{Nono}, x) \wedge \textit{Missile}(x)$ :

$$\textit{Owns}(\textit{Nono}, M_1) \wedge \textit{Missile}(M_1)$$

... tutti i suoi missili gli sono stati venduti dal Colonnello West

$$\textit{Missile}(x) \wedge \textit{Owns}(\textit{Nono}, x) \Rightarrow \textit{Sells}(\textit{West}, x, \textit{Nono})$$

I missili sono armi:

## Esempio di base di conoscenza

... è un crimine per un americano vendere armi a nazioni ostili:

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

Nono ... possiede dei missili, cioè,  $\exists x \textit{Owns}(\textit{Nono}, x) \wedge \textit{Missile}(x)$ :

$$\textit{Owns}(\textit{Nono}, M_1) \wedge \textit{Missile}(M_1)$$

... tutti i suoi missili gli sono stati venduti dal Colonnello West

$$\textit{Missile}(x) \wedge \textit{Owns}(\textit{Nono}, x) \Rightarrow \textit{Sells}(\textit{West}, x, \textit{Nono})$$

I missili sono armi:

$$\textit{Missile}(x) \Rightarrow \textit{Weapon}(x)$$

Un nemico dell'America è "ostile":



## Esempio di base di conoscenza

... è un crimine per un americano vendere armi a nazioni ostili:

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

Nono ... possiede dei missili, cioè,  $\exists x \textit{Owns}(\textit{Nono}, x) \wedge \textit{Missile}(x)$ :

$$\textit{Owns}(\textit{Nono}, M_1) \wedge \textit{Missile}(M_1)$$

... tutti i suoi missili gli sono stati venduti dal Colonnello West

$$\textit{Missile}(x) \wedge \textit{Owns}(\textit{Nono}, x) \Rightarrow \textit{Sells}(\textit{West}, x, \textit{Nono})$$

I missili sono armi:

$$\textit{Missile}(x) \Rightarrow \textit{Weapon}(x)$$

Un nemico dell'America è "ostile":

$$\textit{Enemy}(x, \textit{America}) \Rightarrow \textit{Hostile}(x)$$

West, che è americano ...

## Esempio di base di conoscenza

... è un crimine per un americano vendere armi a nazioni ostili:

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

Nono ... possiede dei missili, cioè,  $\exists x \textit{Owns}(\textit{Nono}, x) \wedge \textit{Missile}(x)$ :

$$\textit{Owns}(\textit{Nono}, M_1) \wedge \textit{Missile}(M_1)$$

... tutti i suoi missili gli sono stati venduti dal Colonnello West

$$\textit{Missile}(x) \wedge \textit{Owns}(\textit{Nono}, x) \Rightarrow \textit{Sells}(\textit{West}, x, \textit{Nono})$$

I missili sono armi:

$$\textit{Missile}(x) \Rightarrow \textit{Weapon}(x)$$

Un nemico dell'America è "ostile":

$$\textit{Enemy}(x, \textit{America}) \Rightarrow \textit{Hostile}(x)$$

West, che è americano ...

$$\textit{American}(\textit{West})$$

La nazione Nono, un nemico dell' America ...

## Esempio di base di conoscenza

... è un crimine per un americano vendere armi a nazioni ostili:

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

Nono ... possiede dei missili, cioè,  $\exists x \textit{Owns}(\textit{Nono}, x) \wedge \textit{Missile}(x)$ :

$$\textit{Owns}(\textit{Nono}, M_1) \wedge \textit{Missile}(M_1)$$

... tutti i suoi missili gli sono stati venduti dal Colonnello West

$$\textit{Missile}(x) \wedge \textit{Owns}(\textit{Nono}, x) \Rightarrow \textit{Sells}(\textit{West}, x, \textit{Nono})$$

I missili sono armi:

$$\textit{Missile}(x) \Rightarrow \textit{Weapon}(x)$$

Un nemico dell'America è "ostile":

$$\textit{Enemy}(x, \textit{America}) \Rightarrow \textit{Hostile}(x)$$

West, che è americano ...

$$\textit{American}(\textit{West})$$

La nazione Nono, un nemico dell' America ...

$$\textit{Enemy}(\textit{Nono}, \textit{America})$$

# Algoritmo di Forward chaining

```
function FOL-FC-ASK( $KB, \alpha$ ) returns una sostituzione oppure false
  repeat until new è vuoto
     $new \leftarrow \{ \}$ 
    for each formula  $r$  in  $KB$  do
       $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \leftarrow \text{STANDARDIZZA-SEPARATAMENTE}(r)$ 
      for each  $\theta$  tale che  $\text{SUBST}(\theta, p_1 \wedge \dots \wedge p_n) = \text{SUBST}(\theta, p'_1 \wedge \dots \wedge p'_n)$ 
        per qualche  $p'_1, \dots, p'_n$  nella  $KB$ 
           $q' \leftarrow \text{SUBST}(\theta, q)$ 
          if  $q'$  non è la ridenominazione di una formula già presente nella  $KB$ 
            o in new then do
              aggiungi  $q'$  a new
               $\phi \leftarrow \text{UNIFY}(q', \alpha)$ 
              if  $\phi$  non è fallimento then return  $\phi$ 
            aggiungi new alla  $KB$ 
  return false
```

# Prova tramite Forward chaining

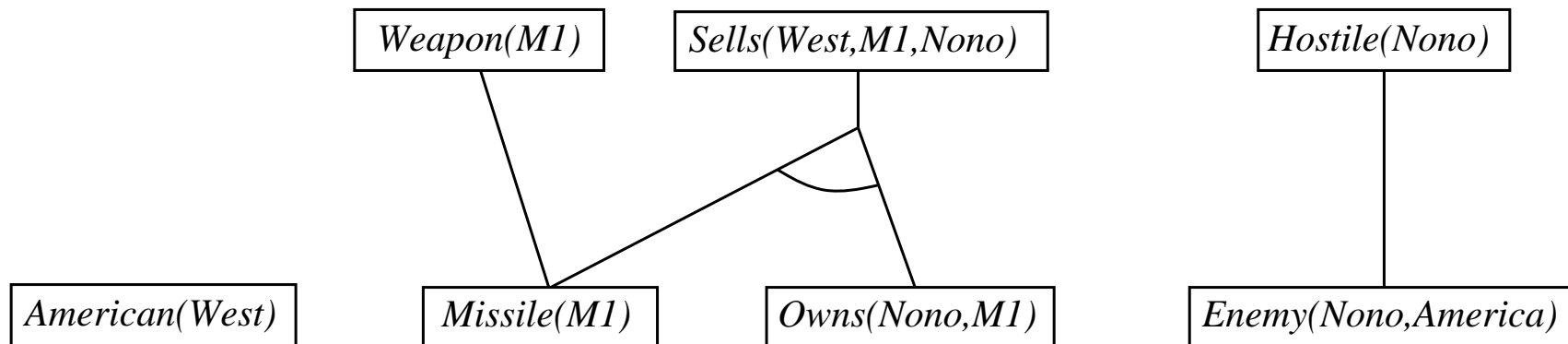
*American(West)*

*Missile(M1)*

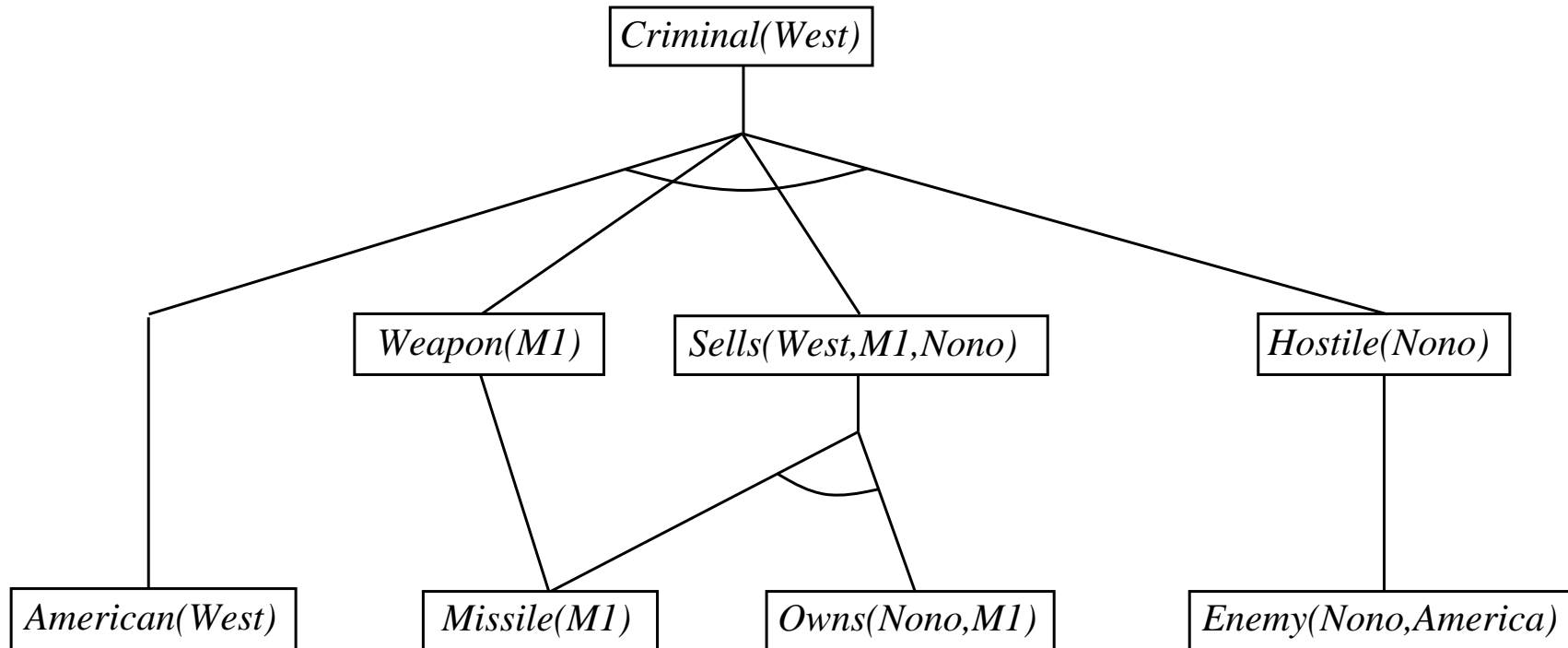
*Owns(Nono,M1)*

*Enemy(Nono,America)*

# Prova tramite Forward chaining



# Prova tramite Forward chaining



## Proprietà di Forward chaining

Corretta e completa per clausole definite di primo ordine  
(prova simile alla prova proposizionale)

Datalog = clausole definite del primo ordine + *nessuna funzione*

FC termina per Datalog in un numero di iterazioni polinomiale: al più  $p \cdot n^k$   
letterali

In generale può non terminare se  $\alpha$  non è conseguenza logica

Ciò è inevitabile: entailment con clausole definite è semidecidibile



## Efficienza di Forward chaining

Semplice osservazione: non c'è bisogno di “match-are” una regola alla iterazione  $k$  se una premessa non è stata aggiunta alla iterazione  $k - 1$   
⇒ “match-are” ogni regola le cui premesse contengono un letterale appena aggiunto

Il “matching” è costoso

Indicizzazione della Base di Dati permette il recupero di fatti conosciuti in  $O(1)$

p.e., la query  $Missile(x)$  recupera  $Missile(M_1)$

Matching di premesse congiuntive rispetto a fatti conosciuti è NP-hard

Forward chaining è largamente utilizzato in basi di dati deduttive

## Algoritmo Backward chaining

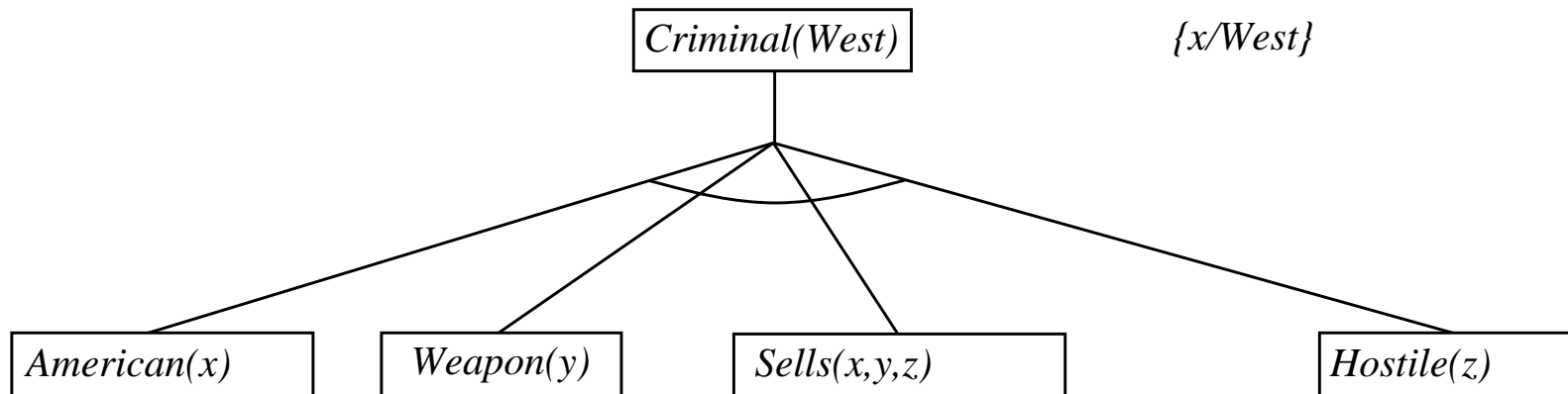
```
function FOL-BC-ASK(KB, obiettivi,  $\theta$ ) returns un insieme di sostituzioni
inputs: KB, una base di conoscenza
           goals, una lista di congiunti che formano una query ( $\theta$  già applicato)
            $\theta$ , la sostituzione corrente, inizialmente la sostituzione vuota  $\{\}$ 
local variables: risposte, un insieme di sostituzioni, inizialmente vuoto

if obiettivi è vuoto then return  $\{\theta\}$ 
 $q' \leftarrow$  SUBST( $\theta$ , PRIMO(obiettivi))
for each formula in KB dove
           STANDARDIZZA-SEPARATAMENTE( $r$ ) = ( $p_1 \wedge \dots \wedge p_n \Rightarrow q$ )
           e  $\theta' \leftarrow$  UNIFY( $q, q'$ ) ha successo
           nuovi_obiettivi  $\leftarrow$  [ $p_1, \dots, p_n$  | RESTO(obiettivi)]
           risposte  $\leftarrow$  FOL-BC-ASK(KB, nuovi_obiettivi, COMPOSE( $\theta', \theta$ ))  $\cup$  risposte
return risposte
```

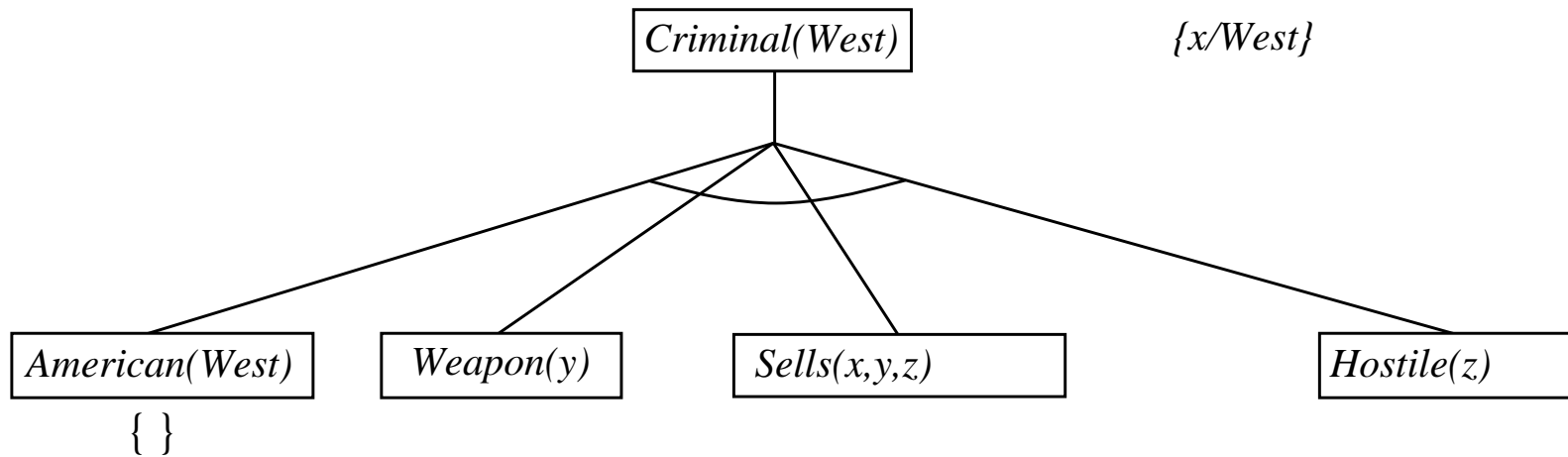
# Esempio Backward chaining

*Criminal(West)*

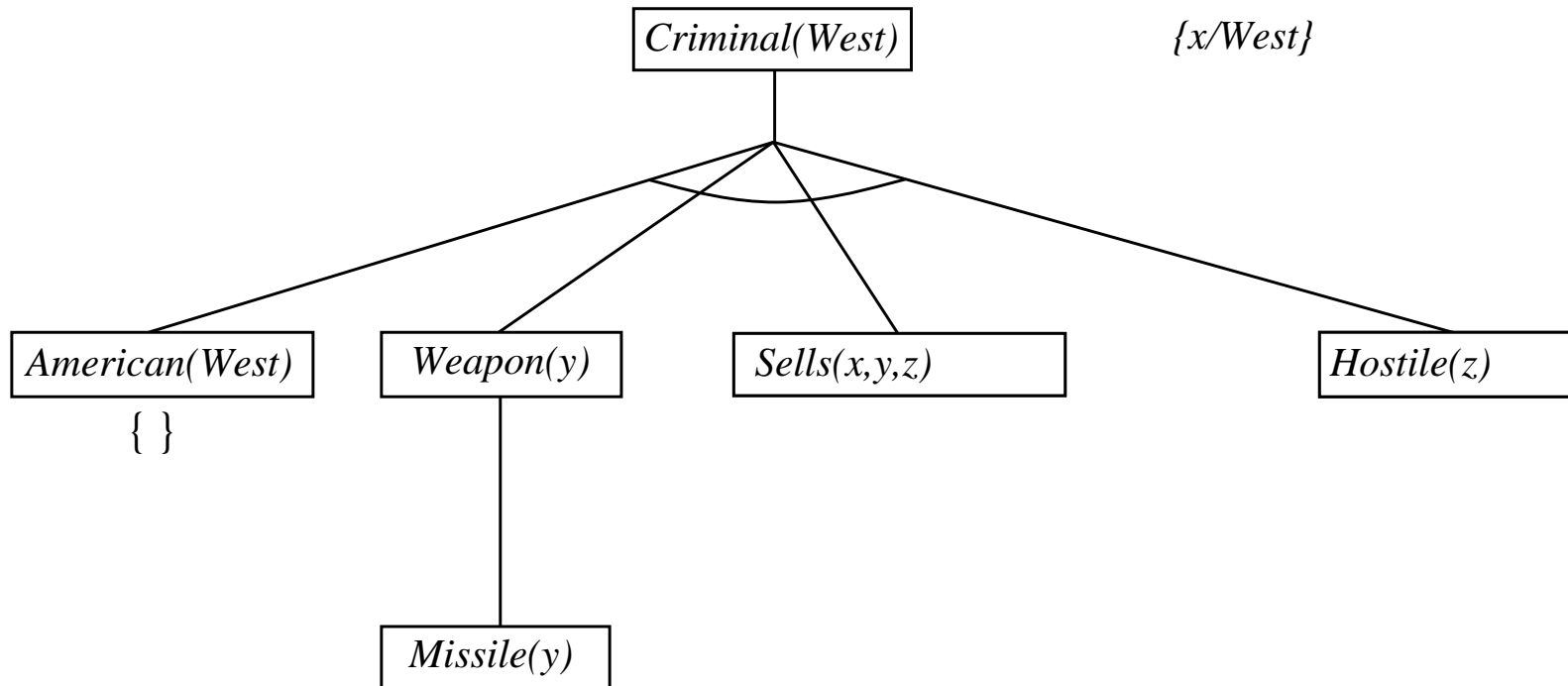
# Esempio Backward chaining



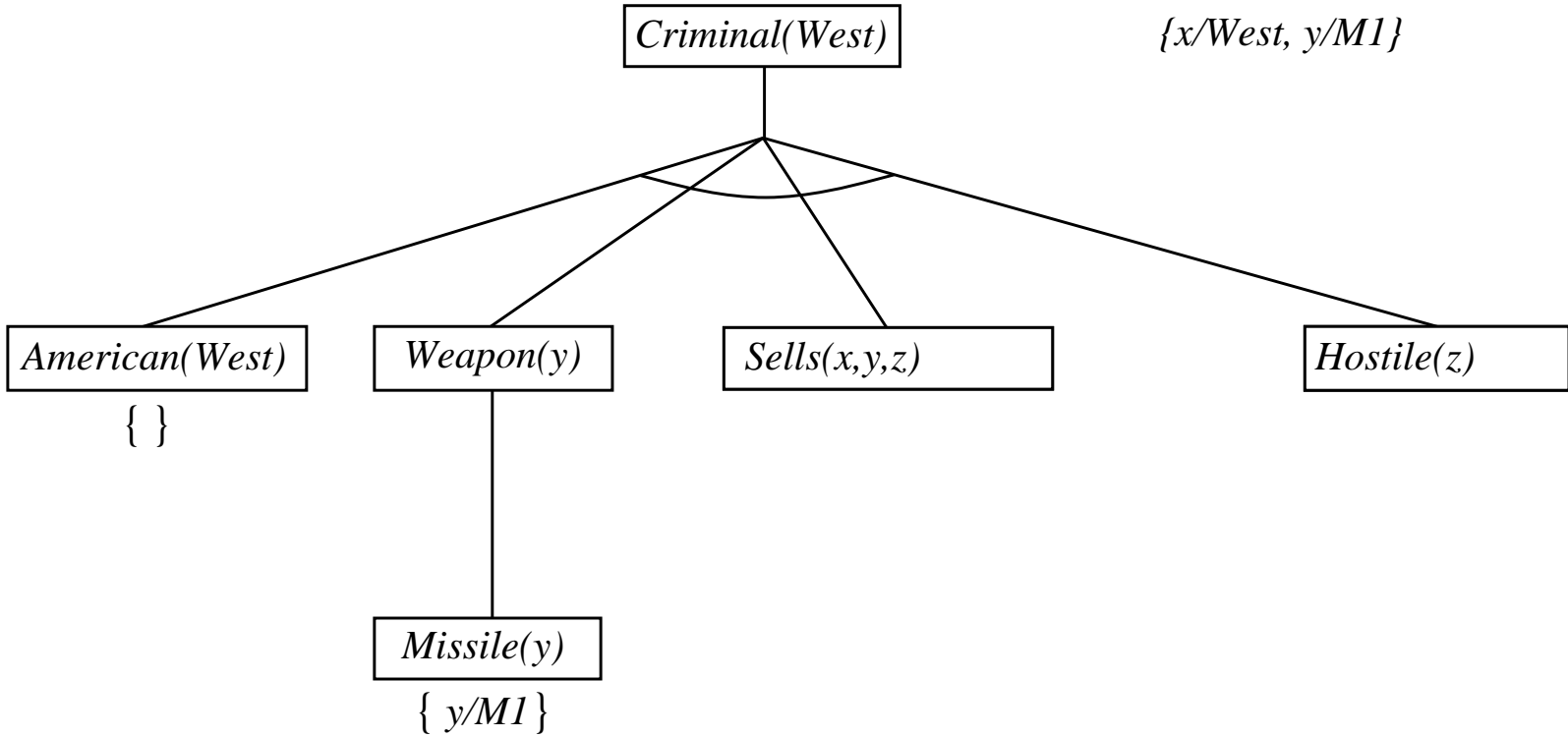
# Esempio Backward chaining



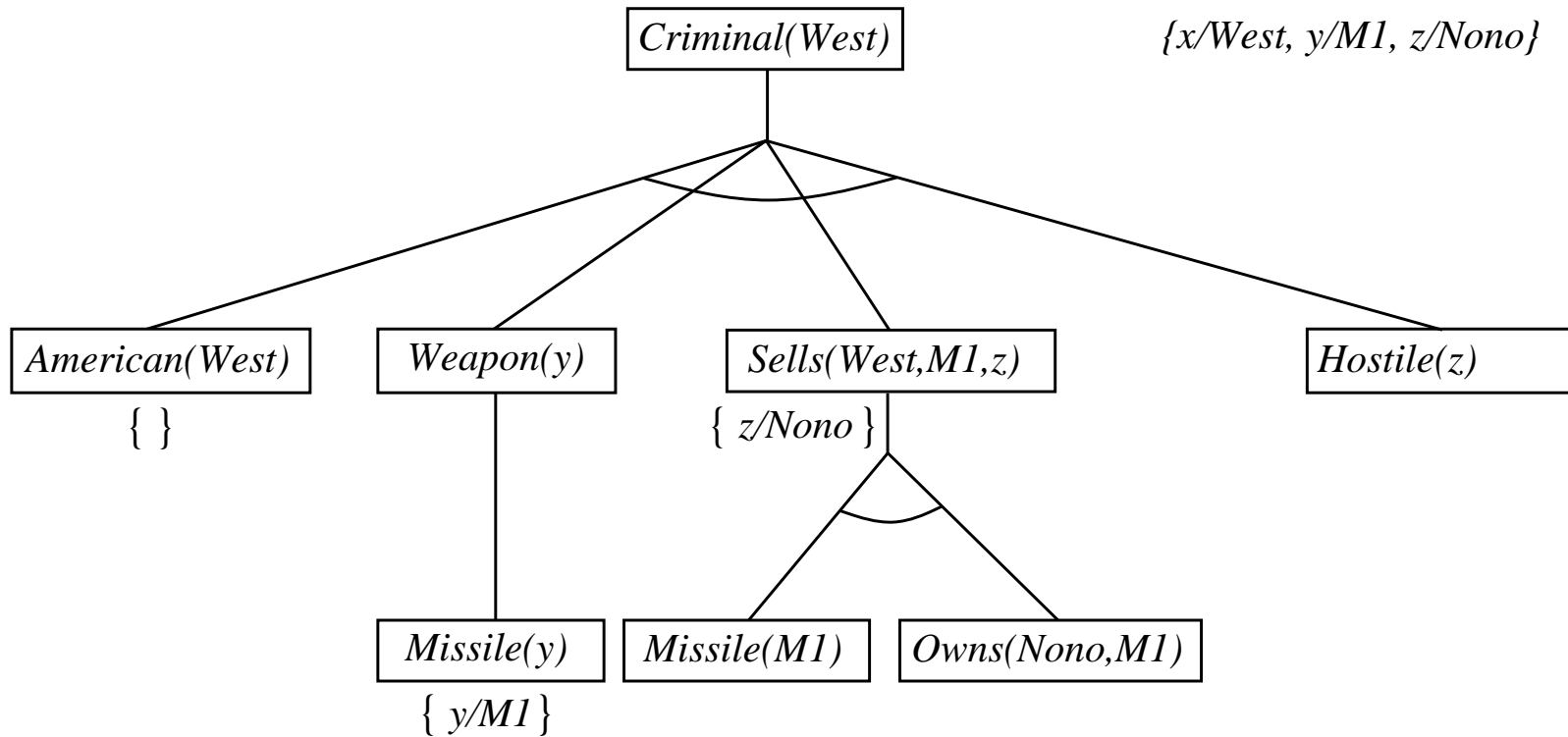
# Esempio Backward chaining



# Esempio Backward chaining

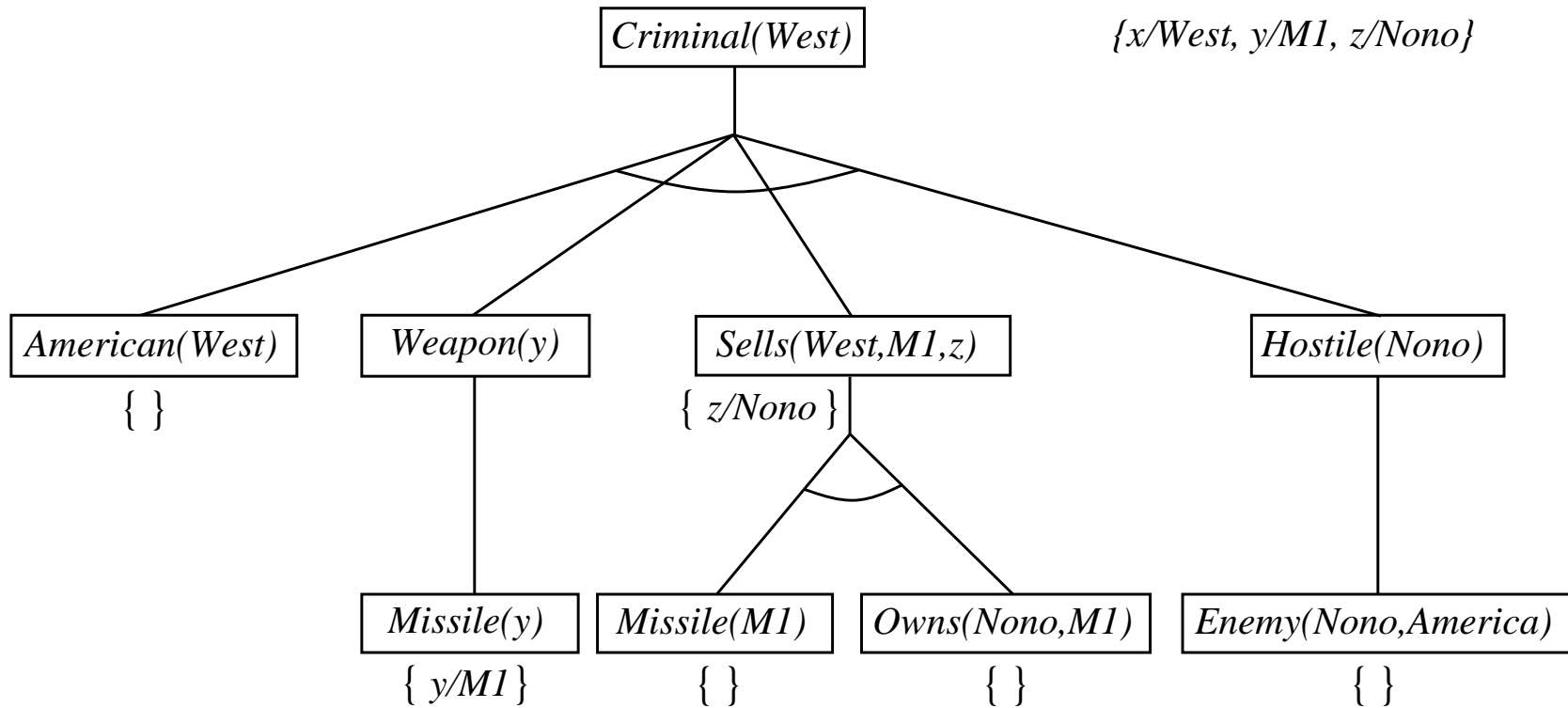


# Esempio Backward chaining





# Esempio Backward chaining



## Proprietà di backward chaining

Complessità in spazio lineare con la dimensione della prova

Incompleta a causa di cicli infiniti

⇒ bisogna controllare il goal corrente rispetto ad ogni goal sulla pila che implementa la ricerca in profondità

Inefficiente a causa di sottogoal ripetuti (sia successo che fallimento)

⇒ bisogna usare una cache che contiene i risultati già calcolati (spazio aggiuntivo!)

Largamente utilizzato (senza i miglioramenti!) per la [programmazione logica](#)

# Programmazione Logica: Prolog

Base: backward chaining con clausole Horn + altro

Tecniche di compilazione  $\Rightarrow$  60 milioni di LIPS

Programma = insieme di clausole = testa  $:-$  letterale<sub>1</sub>, ... letterale<sub>n</sub>.

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

Unificazione efficiente tramite [open coding](#)

Recupero efficiente di clausole “attivabili” per mezzo di direct linking

Depth-first, left-to-right backward chaining

Predicati built-in per l'aritmetica etc., e.g.,  $X$  is  $Y * Z + 3$

Assunzione del mondo chiuso (Closed-world assumption) (“negazione come fallimento”)

```
p.e., dato vivo(X) :- not morto(X).
```

```
vivo(joe) ha successo se morto(joe) fallisce
```

## Esempi di Prolog

Ricerca depth-first da uno stato iniziale X:

```
dfs(X) :- goal(X).
```

```
dfs(X) :- successor(X,S),dfs(S).
```

Append di due liste:

```
append([],Y,Y).
```

```
append([X|L],Y,[X|Z]) :- append(L,Y,Z).
```

```
query:    append(A,B,[1,2]) ?
```

```
risposte: A=[]      B=[1,2]
```

```
          A=[1]     B=[2]
```

```
          A=[1,2]   B=[]
```

## Risoluzione: breve sommario

Versione completa del primo ordine:

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

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

Per esempio,

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

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

Applica passi di risoluzione a  $CNF(KB \wedge \neg\alpha)$ ; completo per FOL

## Conversione a CNF

Ognuno che ama tutti gli animali è amato da qualcuno:

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

1. Eliminare doppie e singole implicazioni

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

2. Spostare  $\neg$  all'interno:  $\neg \forall x, p \equiv \exists x \neg p$ ,  $\neg \exists x, p \equiv \forall x \neg p$ :

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

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

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

## Conversione a CNF

3. Standardizzare variabili: ogni quantificatore deve usarne una differente

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

4. Skolemizzare:

ogni variabile esistenziale è rimpiazzata da una **funzione di Skolem** applicata a tutte quelle variabili quantificate universalmente nel cui scope compare il quantificatore esistenziale:

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

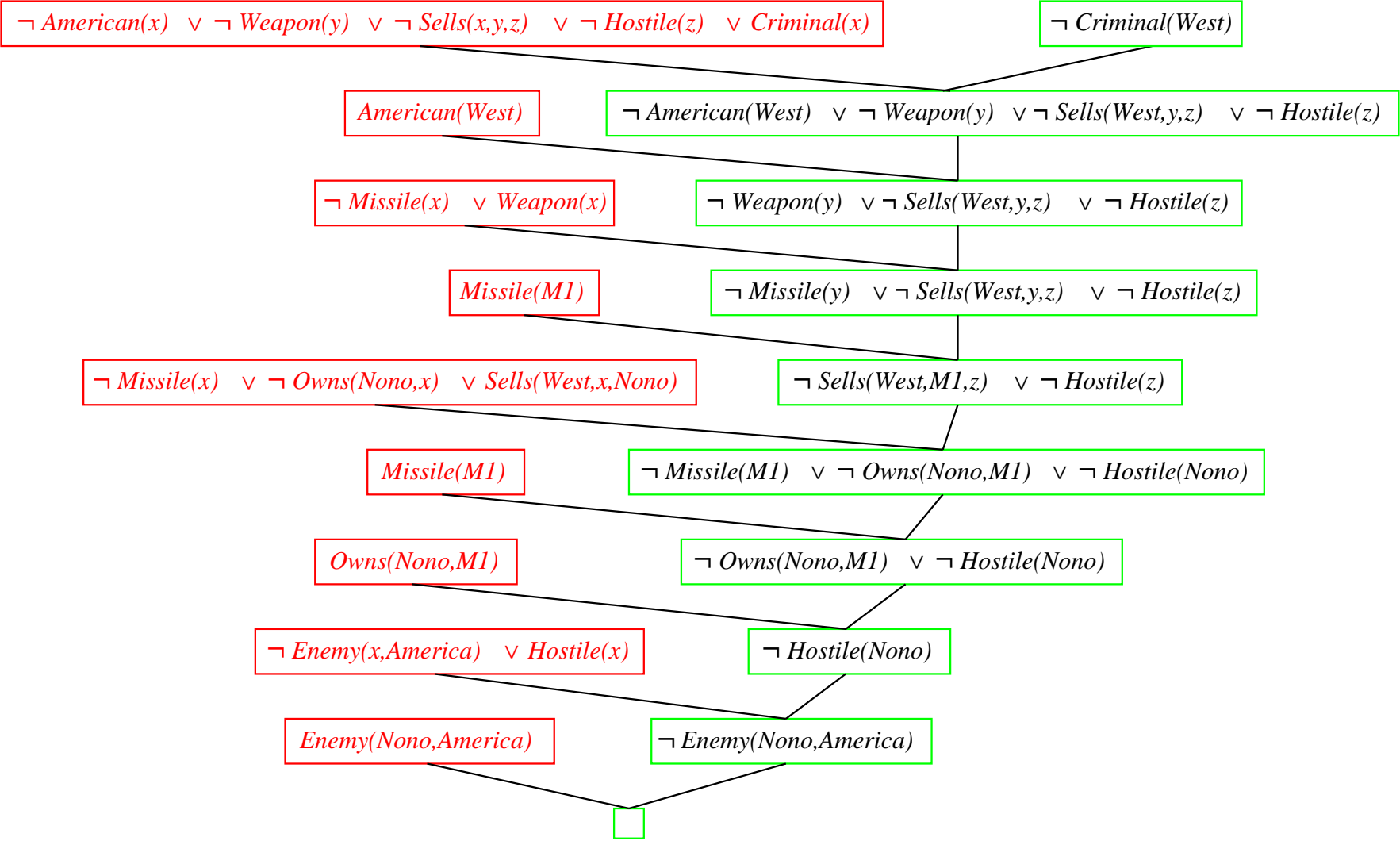
5. Rimuovere i quantificatori universali:

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

6. Distribuire  $\wedge$  su  $\vee$ :

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

# Prova di risoluzione per clausole definite





# Strategie di risoluzione

## ◇ Unit Clause

Si preferisce effettuare la risoluzione con una delle sentenze costituita da un singolo letterale (clausola unitaria)

## ◇ Unit Resolution

Forma ristretta di risoluzione in cui ogni passo di risoluzione deve coinvolgere una clausola unitaria (incompleta in generale, completa per clausole di Horn)

## ◇ Set of Support

Basata sull'insieme di supporto da cui si preleva una delle sentenze e dove si pone il risolvente. Esempio di insieme di supporto: il goal (e tutti i risolventi derivati da esso)

## ◇ Input Resolution

Combina una sentenza in input (KB e Query) con il risolvente corrente. Tipica struttura a spina di pesce.

## ◇ Linear Resolution

Come Input Resolution, ma ammette anche la combinazione del risolvente corrente con suoi avi.

◇ Subsumption

Elimina tutte le sentenze che sono “subsumed” (più specifiche di altre). Es.  $P(A)$  e  $P(A) \vee P(B)$  sono più specifiche di  $P(x)$ .

# Uguaglianza

Fino ad ora l'uguaglianza ( $=$ ) non è stata trattata.

Esistono fondamentalmente 3 approcci diversi:

## 1. Assiomatizzazione

Idea base: si aggiungono assiomi che descrivono le proprietà dell'uguaglianza ed assiomi opportuni per ogni predicato e funzione.

## 2. Aggiunta di una nuova regola di inferenza (p.e., Demodulation, Paramodulation)

Idea base: se  $x = y$  e  $UNIFY(x, z) = \theta$ ,  
rimpiazza  $z$  con  $SUBST(\theta, y)$ .

## 3. Estensione dell'algoritmo di unificazione

Idea base: unifica termini equivalenti,  
p.e.  $1 + 2$  e  $2 + 1$ .

# Completezza Risoluzione: schema prova

Ogni insieme di sentenze  $S$  è rappresentabile in forma normale congiuntiva



Si assuma  $S$  insoddisfacibile e in forma normale congiuntiva



← Teorema di Herbrand

Qualche insieme  $S'$  di istanze ground è insoddisfacibile



← Teorema di Risoluzione per termini ground

La risoluzione può trovare una contraddizione in  $S'$



← Lemma di Lifting

Prova tramite risoluzione dell'esistenza di una contraddizione per  $S$

## Riassunto

### Inferenza in FOL

- ◇ Proposizionalizzazione ed uso di unificazione per evitare la istanziamento di variabili coinvolte in una prova.
- ◇ Modus Ponens Generalizzato (GMP) usa l'unificazione e permette l'applicazione di forward e backward chaining su clausole definite.
- ◇ GMP è completo per clausole definite, anche se determinare le conseguenze logiche è un problema semidecidibile. Per Datalog (no funzioni e clausole definite) il problema è decidibile.
- ◇ Forward chaining è completo e polinomiale per Datalog.
- ◇ Backward chaining è usato in Programmazione Logica (p.e., Prolog) e rinforzato con opportune tecniche di compilazione.
- ◇ Risoluzione generalizzata è completa per KB in forma normale congiuntiva (CNF). Se però si usa il principio di induzione  $\Rightarrow$  non completa.