

Dimostrazioni in logica dei predicati

- La logica dei predicati e' un linguaggio piu' potente per esprimere KB (variabili e quantificatori)
- Vedremo la dimostrazione basata su
 - Risoluzione (corretta e completa per clausole generali)
 - Forward chaining (corretta e completa per clausole Horn) Usata nei Database dedurrivi (DATALOG)
 - Backward chaining (corretta e completa per clausole Horn) Usata nella Programmazione Logica (PROLOG).

TRASFORMAZIONE IN CLAUSOLE (1)

- Una qualunque fbf della logica dei predicati si può trasformare in un insieme di clausole generali.

- Passi

- 1) **Trasformazione in fbf chiusa**

Esempio la formula:

$$- \forall X (p(Y) \rightarrow \sim(\forall Y (q(X,Y) \rightarrow p(Y)))) \quad (1)$$

è trasformata in:

$$- \forall X \forall Y (p(Y) \rightarrow \sim(\forall Y (q(X,Y) \rightarrow p(Y)))) \quad (2)$$

- 2) **Applicazione delle equivalenze per i connettivi logici** (ad esempio $A \rightarrow B$ è sostituito da $\sim A \vee B$) e la si riduce in forma and-or.

La formula (2) diventa:

$$- \forall X \forall Y (\sim p(Y) \vee \sim(\forall Y (\sim q(X,Y) \vee p(Y)))) \quad (3)$$

TRASFORMAZIONE IN CLAUSOLE (2)

- 3) **Applicazione della negazione ad atomi e non a formule composte**, tenendo presente che:

$\forall X \sim A$ equivale a $\sim \exists X A$

$\exists X \sim A$ equivale a $\sim \forall X A$

$\sim(A_1 \vee A_2 \vee \dots \vee A_n)$ equivale a $\sim A_1 \wedge \sim A_2 \wedge \dots \wedge \sim A_n$

$\sim(A_1 \wedge A_2 \wedge \dots \wedge A_n)$ equivale a $\sim A_1 \vee \sim A_2 \vee \dots \vee \sim A_n$

(leggi di De Morgan).

(3) si modifica in:

$$\forall X \forall Y (\sim p(Y) \vee (\exists Y (q(X, Y) \wedge \sim p(Y)))) \quad (4)$$

- 4) **Cambiamento di nomi delle variabili**, nel caso di conflitti.

in (4) la seconda variabile Y viene rinominata Z:

$$\forall X \forall Y (\sim p(Y) \vee (\exists Z (q(X, Z) \wedge \sim p(Z)))) \quad (5)$$

TRASFORMAZIONE IN CLAUSOLE (3)

- 5) **Spostamento dei quantificatori** in testa alla formula (forma prenessa).

$$\forall X \forall Y \exists Z (\sim p(Y) \vee (q(X,Z) \wedge \sim p(Z))) \quad (6)$$

- 6) **Forma normale congiuntiva** cioè come congiunzione di disgiunzioni, con quantificazione in testa.

$$\forall X \forall Y \exists Z ((\sim p(Y) \vee q(X,Z)) \wedge (\sim p(Y) \vee \sim p(Z))) \quad (7)$$

- 7) **Skolemizzazione**: ogni variabile quantificata esistenzialmente viene sostituita da una funzione delle variabili quantificate universalmente che la precedono. Tale funzione è detta funzione di Skolem.

Ad esempio una formula del tipo: $\forall X \exists Y p(X,Y)$ può essere espressa in modo equivalente come: $\forall X p(X,g(X))$

In (7) Z è sostituita da $f(X,Y)$, perché Z si trova nel campo di azione delle quantificazioni $\forall X$ e $\forall Y$:

$$\forall X \forall Y ((\sim p(Y) \vee q(X,f(X,Y))) \wedge (\sim p(Y) \vee \sim p(f(X,Y)))) \quad (8)$$

TRASFORMAZIONE IN CLAUSOLE (4)

- **Perdita in espressività.** Non è la stessa cosa asserire: $F: \exists X p(X)$ oppure $F': p(f)$.
- Vale comunque la proprietà che F è inconsistente se e solo se F' è inconsistente.
- 8) **Eliminazione dei quantificatori universali:** si ottiene una formula detta universale (tutte le sue variabili sono quantificate universalmente) in forma normale congiuntiva.
- $((\sim p(Y) \vee q(X, f(X, Y))) \wedge (\sim p(Y) \vee \sim p(f(X, Y)))) \quad (9)$
- Una formula di questo tipo rappresenta **un insieme di clausole** (ciascuna data da un congiunto nella formula). La forma normale a clausole che si ottiene: $\{\sim p(Y) \vee q(X, f(X, Y)), \sim p(Y) \vee \sim p(f(X, Y))\} \quad (10)$
- La seconda clausola può essere riscritta rinominando le variabili (sostituendo cioè la formula con una sua variante).
- $\{\sim p(Y) \vee q(X, f(X, Y)), \sim p(Z) \vee \sim p(f(W, Z))\} \quad (11)$

TRASFORMAZIONE IN CLAUSOLE (5)

- Qualunque teoria del primo ordine T può essere trasformata in una teoria T' in forma a clausole.
- Anche se T non è logicamente equivalente a T' (a causa dell'introduzione delle funzioni di Skolem), vale comunque la seguente proprietà:

Proprietà

- Sia T una teoria del primo ordine e T' una sua trasformazione in clausole. Allora T è insoddisfacibile se e solo se T' è insoddisfacibile.
- Il principio di risoluzione è una procedura di dimostrazione che opera per contraddizione e si basa sul concetto di insoddisfacibilità.

UNIFICAZIONE

- Per applicare il principio di risoluzione alle clausole non “ground” è necessario introdurre il concetto di unificazione [Robinson 65].
- **Unificazione**: procedimento di manipolazione formale utilizzato per stabilire quando due espressioni possono coincidere procedendo a opportune sostituzioni.
- **Sostituzione**: σ insieme di legami di termini T_i a simboli di variabili X_i ($i=1,\dots,n$).

$$\sigma = \{X_1/T_1, X_2/T_2, \dots, X_n/T_n\}$$

dove X_1, X_2, \dots, X_n sono distinte.

- La sostituzione corrispondente all'insieme vuoto è detta **sostituzione identità** (ϵ).

SOSTITUZIONI E RENAMING

- **Applicazione della sostituzione σ a un'espressione E ,** $[E]\sigma$, produce una nuova espressione ottenuta sostituendo simultaneamente ciascuna variabile X_i dell'espressione con il corrispondente termine T_i . $[E]\sigma$ è detta **istanza** di E .
- **Renaming:** sostituzioni che cambiano semplicemente il nome ad alcune delle variabili di E . , $[E]\sigma$ è una **variante** di E .

SOSTITUZIONI ESEMPIO

- L'applicazione della sostituzione $\sigma = \{X/c, Y/a, Z/W\}$ all'espressione $p(X, f(Y), b, Z)$ produce l'istanza $p(c, f(a), b, W)$.

Analogamente:

$$[c(Y, Z)]\{Y/T, Z/neri\} = c(T, neri)$$

$$[c(T, neri)]\{Y/T, Z/neri\} = c(T, neri)$$

$$[c(Y, Z)]\{Y/bianchi, T/bianchi, Z/neri\} =$$

$$c(bianchi, neri)$$

$$[c(T, neri)]\{Y/bianchi, T/bianchi, Z/neri\} =$$

$$c(bianchi, neri)$$

$$[c(Y, Z)]\{Y/T, Z/bianchi\} = c(T, bianchi)$$

$$[c(T, neri)]\{Y/T, Z/bianchi\} = c(T, neri)$$

$$[t(l(a), t(l(b), l(H)))]\{H/l(a), Y/l(b), Z/l(l(a))\} =$$

$$t(l(a), t(l(b), l(l(a))))$$

$$[t(H, t(Y, Z))]\{H/T, Y/X, Z/W\} = t(T, t(X, W))$$

(e8) è una variante dell'espressione di partenza.

$$[c(T, neri)]\epsilon = c(T, neri)$$

$$[p(X, Y)]\{X/a, Y/X\} = p(a, X)$$

COMBINAZIONE DI SOSTITUZIONI

- **Combinazioni di sostituzioni:**

$$\sigma_1 = \{X_1/T_1, X_2/T_2, \dots, X_n/T_n\} \quad \sigma_2 = \{Y_1/Q_1, Y_2/Q_2, \dots, Y_m/Q_m\}$$

- **composizione** $\sigma_1\sigma_2$ di σ_1 e σ_2 è la sostituzione

$$\{X_1/[T_1]\sigma_2, \dots, X_n/[T_n]\sigma_2, Y_1/Q_1, Y_2/Q_2, \dots, Y_m/Q_m\}$$

cancellando le coppie $X_i/[T_i]\sigma_2$ per le quali si ha $X_i = [T_i]\sigma_2$ e le coppie Y_j/Q_j per le quali Y_j appartiene all'insieme $\{X_1, X_2, \dots, X_n\}$.

Esempio : $\sigma_1 = \{X/f(Z), W/R, S/c\}$ $\sigma_2 = \{Y/X, R/W, Z/b\}$

produce: $\sigma_3 = \sigma_1\sigma_2 = \{X/f(b), S/c, Y/X, R/W, Z/b\}$.

SOSTITUZIONI PIU' GENERALI

- **Sostituzioni più generali:** Una sostituzione θ è più generale di una sostituzione σ se esiste una sostituzione λ tale che $\sigma = \theta\lambda$.

Esempio : La sostituzione $\theta = \{Y/T, Z/neri\}$ è più generale della sostituzione $\sigma = \{Y/bianchi, T/bianchi, Z/neri\}$ in quanto σ si può ottenere attraverso la composizione $\{Y/T, Z/neri\}\{T/bianchi\}$ ($\sigma = \theta\lambda$, e $\lambda = \{T/bianchi\}$).

SOSTITUZIONE UNIFICATRICE

- L'**unificazione** rende identici due o più atomi (o termini) (o meglio le loro **istanze**) attraverso un'opportuna sostituzione.
- Se si considerano solo due atomi (o termini), uno dei quali senza alcuna variabile, si ricade in un caso particolare di unificazione, detto **pattern-matching**.
- Un insieme di atomi (o termini) A_1, A_2, \dots, A_n è **unificabile** se esiste una sostituzione σ tale che:
$$[A_1]\sigma = [A_2]\sigma = \dots = [A_n]\sigma.$$
- La sostituzione σ è detta **sostituzione unificatrice** (o unificatore)

ESEMPIO

Esempio Se si considerano gli atomi:

$$A1=c(Y,Z)$$

$$A2=c(T,neri)$$

- possibili sostituzioni unificatrici sono:

$$\theta = \{Y/T, Z/neri\}$$

$$\sigma = \{Y/bianchi, T/bianchi, Z/neri\}$$

- la loro applicazione produce la stessa istanza:

$$[c(Y,Z)]\theta = [c(T,neri)]\theta = c(T,neri)$$

$$[c(Y,Z)]\sigma = [c(T,neri)]\sigma = c(bianchi,neri)$$

- La sostituzione:

$\lambda=\{Y/T, Z/bianchi\}$ non è un unificatore per $A1$ e $A2$ perchè produce istanze diverse.

ESEMPIO

- Per gli atomi:

A3: $p(X, X, f(a, Z))$

A4: $p(Y, W, f(Y, J))$

- possibili sostituzioni unificatrici sono:

$\mu = \{X/a, Y/a, W/a, J/Z\}$

$\varphi = \{X/a, Y/a, W/a, J/c, Z/c\}$

- la loro applicazione ad A3 e A4 produce la stessa istanza:

$p(a, a, f(a, Z))$ nel caso della sostituzione μ

$p(a, a, f(a, c))$ nel caso della sostituzione φ .

- Possono esistere più sostituzioni unificatrici, si vuole individuare quella più generale (mgu, **most general unifier**).

μ mgu: φ si ottiene da μ componendola con: $\alpha = \{Z/c\}$.

ESEMPI

θ indica la sostituzione unificatrice più generale:

$$1) \quad p(X, f(a)) \quad p(b, Y)$$

$$\theta = \{X/b, Y/f(a)\}$$

$$2) \quad t(l(a), t(l(b), l(c))) \quad t(l(X), Z)$$

$$\theta = \{X/a, Z/t(l(b), l(c))\}$$

I termini $t(l(a), t(l(b), l(c)))$ e $t(l(X), Z)$ rappresentano alberi binari.

$$3) \quad t(l(a), t(l(b), l(H))) \quad t(H, t(Y, Z))$$

$$\theta = \{H/l(a), Y/l(b), Z/l(l(a))\}$$

$$4) \quad f(X, g(Y), T) \quad f(c(a, b), g(g(a, c)), Z).$$

$$\theta = \{X/c(a, b), Y/g(a, c), T/Z\}$$

$$5) \quad .(a, .(b, .(c, .(d, []))) \quad .(X, Y)$$

$$\theta = \{X/a, Y/.(b, .(c, .(d, [])))\}$$

rappresentano liste

$$6) \quad .(a, .(b, .(c, .(d, []))) \quad .(X, .(Y, Z))$$

$$\theta = \{X/a, Y/b, Z/.(c, .(d, []))\}$$

$$7) \quad .(a, .(b, .(c, .(d, []))) \quad .(c, Y)$$

Non sono unificabili.

$$8) \quad t(t(t(X, Y), l(c)), Y) \\ t(t(t(Y, Y), l(c)), l(X))$$

Non sono unificabili

ALGORITMO DI UNIFICAZIONE

- Algoritmo in grado di determinare se due atomi (o termini) sono unificabili o meno e restituire, nel primo caso, la sostituzione unificatrice più generale.
- Esistono vari algoritmi di unificazione di differente complessità.
- Regole alla base dell'algoritmo di unificazione fra due termini T1 e T2.

T1 \ T2	<costante> C2	<variabile> X2	<termine composto> S2
<costante> C1	ok se $C1 = C2$	ok {X2/C1}	NO
<variabile> X1	ok {X1/C2}	ok {X1/X2}	ok {X1/S2}
<termine composto> S1	NO	ok {X2/S1}	ok se S1 e S2 hanno stesso funtore e arietà e gli argomenti UNIFICANO

ALGORITMO DI UNIFICAZIONE

- Funzione UNIFY che ha come parametri di ingresso due atomi o termini da unificare A e B e la sostituzione eventualmente già applicata.
- La funzione termina sempre ed è in grado di fornire o la sostituzione più generale per unificare A e B o un fallimento (FALSE).
- Un termine composto (cioè diverso da costante o variabile) è rappresentato da un operatore OP (il simbolo funzionale del termine) e come altri elementi gli argomenti del termine ARGS (lista).
- Es: $f(a, g(b, c), X)$ rappresentato come: $[f, a, [g, b, c], X]$.
- La funzione head, applicata a una lista L, restituisce il primo elemento di L, mentre la funzione tail il resto della lista L.

L'algoritmo di unificazione

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

  if  $\theta = \text{failure}$  then return failure
  else if  $x = y$  then return  $\theta$ 
  else if VARIABLE?( $x$ ) then return UNIFY-VAR( $x, y, \theta$ )
  else if VARIABLE?( $y$ ) then return UNIFY-VAR( $y, x, \theta$ )
  else if COMPOUND?( $x$ ) and COMPOUND?( $y$ ) then
    return UNIFY(ARGS[ $x$ ], ARGS[ $y$ ], UNIFY(OP[ $x$ ], OP[ $y$ ],  $\theta$ ))
  else if LIST?( $x$ ) and LIST?( $y$ ) then
    return UNIFY(REST[ $x$ ], REST[ $y$ ], UNIFY(FIRST[ $x$ ], FIRST[ $y$ ],  $\theta$ ))
  else return failure
```

L'algoritmo di unificazione

```
function UNIFY-VAR( $var, x, \theta$ ) returns a substitution  
  inputs:  $var$ , a variable  
            $x$ , any expression  
            $\theta$ , the substitution built up so far  
  
  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 OCCUR-CHECK?( $var, x$ ) then return failure  
  else return add  $\{var/x\}$  to  $\theta$ 
```

OCCUR CHECK

- È il controllo che un termine variabile da unificare con un secondo termine non compaia in quest'ultimo. Necessario per assicurare la terminazione dell'algoritmo e la correttezza del procedimento di unificazione.
- I due termini "X" e "f(X)" non sono: non esiste una sostituzione per X che renda uguali i due termini.
- Se un termine t ha una struttura complessa, la verifica se X compare in t può essere anche molto inefficiente.
- Prolog NON utilizza l'occur-check: non corretto !!

OCCUR CHECK: ESEMPIO

- **Esempio:** Si considerino i termini:
 - $p(X, f(X))$.
 - $p(Y, Y)$.
 - Per Prolog (senza occur check) unificano e viene prodotta la sostituzione (contenente un termine infinito): $Y = f(f(f(\dots)))$
- **Quindi se p rappresentasse il predicato maggiore ed f la funzione successore deriverebbe che esiste un numero maggiore di se' stesso!**
- **Dimostrazione non corretta!!**
- .

IL PRINCIPIO DI RISOLUZIONE PER LE CLAUSOLE GENERALI

- Clausole nelle quali possono comparire variabili.
- Siano C_1 e C_2 due clausole del tipo:
$$C_1 = A_1 \vee \dots \vee A_n \qquad C_2 = B_1 \vee \dots \vee B_m$$
- dove A_i ($i=1..n$) e B_j ($j=1..m$) è un letterale positivo o negativo in cui possono comparire variabili.
- Se esiste A_i e B_j tali che $[A_i]\theta = [\sim B_j]\theta$, dove θ è la sostituzione unificatrice più generale, allora si può derivare una nuova clausola C_3 (il risolvente):
$$[A_1 \vee \dots \vee A_{i-1} \vee A_{i+1} \vee \dots \vee A_n \vee B_1 \vee \dots \vee B_{j-1} \vee B_{j+1} \vee \dots \vee B_m]\theta$$
- Date due clausole C_1 e C_2 , il loro risolvente C_3 è conseguenza logica di $C_1 \cup C_2$.

MGU PER RICAVARE IL RISOLVENTE

Regola di inferenza

$$\frac{A_1 \vee \dots \vee A_n B_1 \vee \dots \vee B_m \exists \theta : [A_i]\theta = [\sim B_j]\theta}{[A_1 \vee \dots \vee A_{i-1} \vee A_{i+1} \vee \dots \vee A_n \vee B_1 \vee \dots \vee B_{j-1} \vee B_{j+1} \vee \dots \vee B_m]\theta}$$

dove θ è la sostituzione più generale per A_i e $\sim B_j$.

- **Esempio**

Si considerino le seguenti clausole:

$$p(X, m(a)) \vee q(S, a) \vee c(X) \quad (C1)$$

$$\sim p(r, Z) \vee c(Z) \vee \sim b(W, U) \quad (C2)$$

I letterali $p(X, m(a))$ e $p(r, Z)$ sono unificabili attraverso la mgu $\theta = \{X/r, Z/m(a)\}$.

Risolvente:

$$q(S, a) \vee c(r) \vee c(m(a)) \vee \sim b(W, U) \quad (C3)$$

MGU PER RICAIVARE IL RISOLVENTE

Regola di inferenza

- **Esempio**

Dalle seguenti clausole:

$$p(a,b) \vee q(X,a) \vee c(X) \quad (C1)$$

$$\sim p(a,Y) \vee \sim c(f(Y)) \quad (C2)$$

si ottengono i seguenti risolventi:

$$q(X,a) \vee c(X) \vee \sim c(f(b)) \quad (C3)$$

$$p(a,b) \vee q(f(Y),a) \vee \sim p(a,Y) \quad (C4)$$

C3: ottenuto selezionando $p(a,b)$ da C1 e $\sim p(a,Y)$ da C2
e applicando la mgu $\theta = \{Y/b\}$.

C4: ottenuto selezionando $c(X)$ da C1 e $\sim c(f(Y))$ da C2
e applicando la mgu $\theta = \{X/f(Y)\}$.

ESEMPIO

- **Si considerino gli insiemi:**

$H = \{\forall X (\text{uomo}(X) \rightarrow \text{mortale}(X)), \text{uomo}(\text{socrate})\}$

$F = \{\exists X \text{mortale}(X)\}$

La trasformazione in clausole di H e $\sim F$ produce:

$H^C = \{\sim \text{uomo}(X) \vee \text{mortale}(X), \text{uomo}(\text{socrate})\}$

$F^C = \{\sim \text{mortale}(X)\}$

L'insieme $H^C \cup F^C$ è il seguente:

$\{\sim \text{uomo}(X) \vee \text{mortale}(X), (1)$

$\text{uomo}(\text{socrate}), (2)$

$\sim \text{mortale}(Y)\} (3)$

ESEMPIO

- La variabile X di F^C rinominata con Y che non appare in nessuna altra clausola dell'insieme. Questa operazione viene eseguita ogni volta che si aggiungono nuovi risolventi all'insieme delle clausole.
- Al passo 1, tutti i possibili risolventi, e le sostituzioni unificatrici più generali applicate per derivarli, sono:

{mortale(socrate), (4) $\theta=\{X/\text{socrate}\}$

$\sim\text{uomo}(Z)\}$ (5) $\theta=\{X/Y\}$

- Al passo 2, da (2) e (5) viene derivata la clausola vuota, applicando la sostituzione $\{Z/\text{socrate}\}$. La clausola vuota è anche derivabile dalle clausole (3) e (4) applicando la sostituzione $\{Y/\text{socrate}\}$.

ESEMPIO

1. cane(fido).
2. ~ abbaia(fido).
3. scodinzola(fido).
4. miagola(geo).
5. scodinzola(X) and cane(X) -> amichevole(X).
6. amichevole(X1) and ~ abbaia(X1) -> ~spaventato(Y1,X1).
7. cane(X2) -> animale(X2).
8. miagola(X3) -> gatto(X3).
- Goal : 9. esiste X4 Y cane(X4) and gatto(Y) and ~spaventato(Y,X4).

TRADUZIONE

1. cane(fido).
2. ~abbaia(fido).
3. scodinzola(fido).
4. miagola(geo).
5. ~scodinzola(X) or ~cane(X) or amichevole(X).
6. ~amichevole(X1) or abbaia(X1) or ~spaventato(Y1,X1).
7. ~cane(X2) or animale(X2).
8. ~miagola(X3) or gatto(X3).
- Goal negato: 9. ~cane(X4) or ~gatto(Y) or spaventato(Y,X4).

ESEMPIO

1. cane(fido).
2. ~abbaia(fido).
3. scodinzola(fido).
4. miagola(geo).
5. ~scodinzola(X) or ~cane(X) or amichevole(X).
6. ~amichevole(X1) or abbaia(X1) or ~spaventato(Y1,X1).
7. ~cane(X2) or animale(X2).
8. ~miagola(X3) or gatto(X3).
- Goal negato: 9. ~cane(X4) or ~gatto(Y) or spaventato(Y,X4).
- Da 9 e 1: 10. ~gatto(Y) or spaventato(Y,fido).
- Da 10. e 6: 11. ~amichevole(fido) or abbaia(fido) or ~gatto(Y).
- Da 11. e 2: 12. ~amichevole(fido) or ~gatto(Y).
- Da 8. e 12: 13. ~amichevole(fido) or ~miagola(Y).
- Da 13. e 4: 14. ~amichevole(fido).
- Da 5. e 14: 15. ~scodinzola(fido) or ~cane(fido).
- Da 3. e 15: 16. ~cane(fido)
- Da 1. e 16. CONTRADDIZIONE !!

CORRETTEZZA e COMPLETEZZA

- Si può dimostrare che sotto opportune strategie, la risoluzione è **corretta e completa**.
- Se viene generata la clausola vuota la teoria $H^C \cup F^C$ è insoddisfacibile e se la teoria $H^C \cup F^C$ è insoddisfacibile la derivazione genera la clausola vuota in un numero finito di passi.
- **Teorema**
(Correttezza e completezza della risoluzione)
 - Un insieme di clausole è insoddisfacibile se e solo se l'algoritmo di risoluzione termina con successo in un numero finito di passi, generando la clausola vuota.
 - Il metodo di risoluzione procede esaustivamente generando tutti i possibili risolventi ad ogni passo.

STRATEGIE

- Si definiscono strategie che scelgono opportunamente le clausole da cui derivare un risolvente.
- I metodi di prova che si ottengono risultano più efficienti anche se in alcuni casi possono introdurre incompletezza.
- La dimostrazione attraverso il principio di risoluzione può essere rappresentata con un grafo, detto **grafo di refutazione**.
- Le clausole dell'insieme base $H^C \cup F^C$ sono nodi del grafo dai quali possono solo uscire archi.
- Un risolvente corrisponde a un nodo nel quale entrano almeno due archi (ciascuno da una delle due clausole “parent”).
- **Strategia in ampiezza (“breadth-first”)**. Al passo i ($i \geq 0$), genera tutti i possibili risolventi a livello $i+1$ -esimo utilizzando come clausole “parent” una clausola di C_i (cioè una clausola a livello i) e una di C_j ($j \leq i$), cioè una clausola appartenente a un livello uguale o minore di i .

ESEMPIO

$$H = H^C = \{\text{on}(\text{b1}, \text{tavola}), \quad (1)$$

$$\text{on}(\text{b2}, \text{tavola}), \quad (2)$$

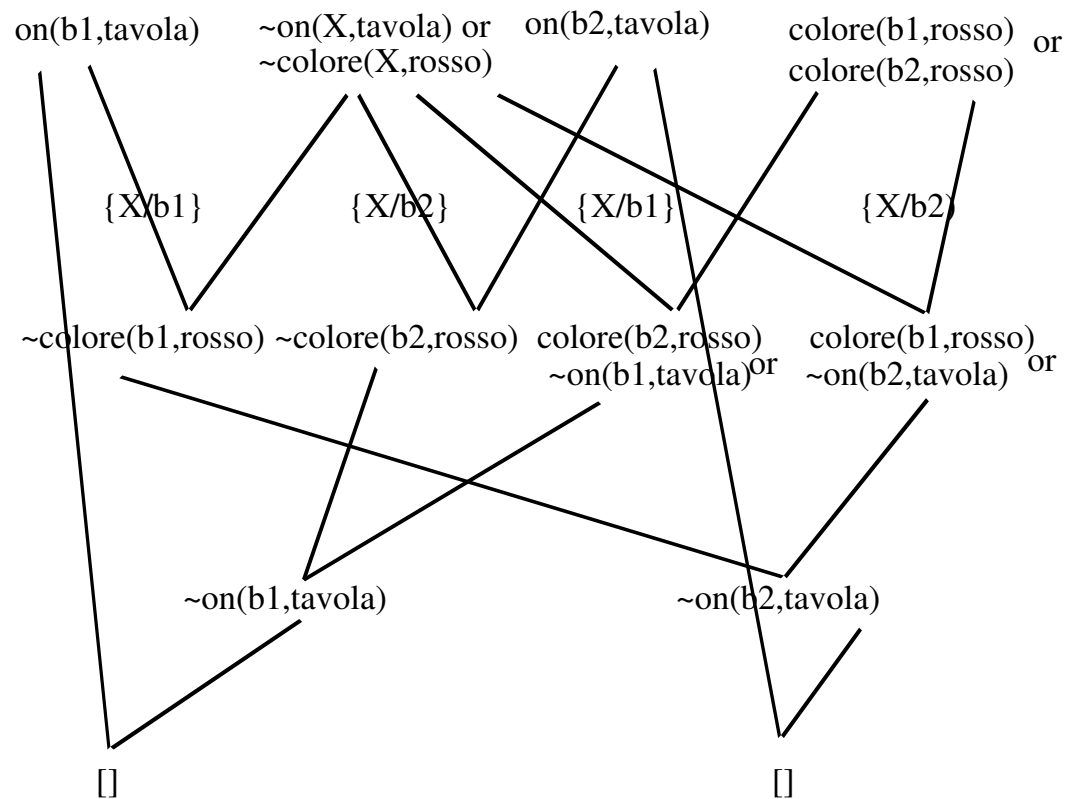
$$\text{colore}(\text{b1}, \text{rosso}) \vee \text{colore}(\text{b2}, \text{rosso}) \quad (3)$$

$$F = \{\exists X (\text{on}(X, \text{tavola}) \wedge \text{colore}(X, \text{rosso}))\}$$

$$F^C = \{\sim \text{on}(X, \text{tavola}) \vee \sim \text{colore}(X, \text{rosso})\}$$

- L'applicazione della strategia in ampiezza produce il **grafo di refutazione** riportato nel seguito:

ESEMPIO

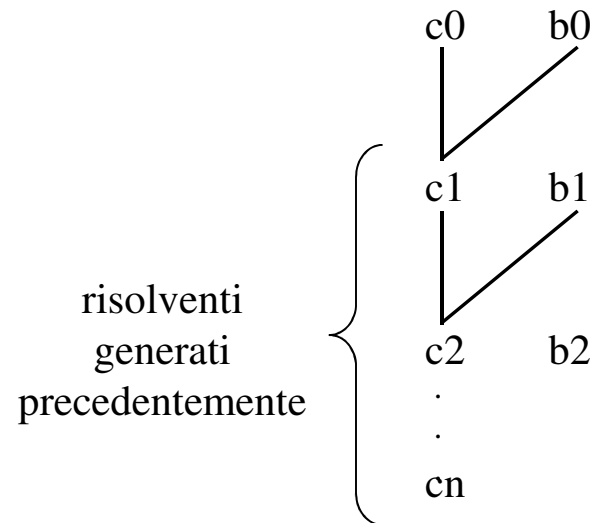


STRATEGIE

- Strategia **set-of-support** (completa) sceglie almeno una clausola “parent” fra quelle derivate dalla negazione della formula che si vuole dimostrare o dai loro discendenti.
- Strategia **lineare** (completa) sceglie almeno una clausola “parent” dall'insieme base C_0 oppure tra i risolventi generati precedentemente. La seconda clausola parent è sempre il risolvente ottenuto al passo precedente.
- Nel caso di risoluzione lineare, il grafo di refutazione diventa un albero, detto albero di refutazione.

STRATEGIE

Albero lineare : c_0 appartiene a C_0 (cioè all'insieme base) e b_i appartiene a C_0 oppure è uguale a c_j con $j < i$.

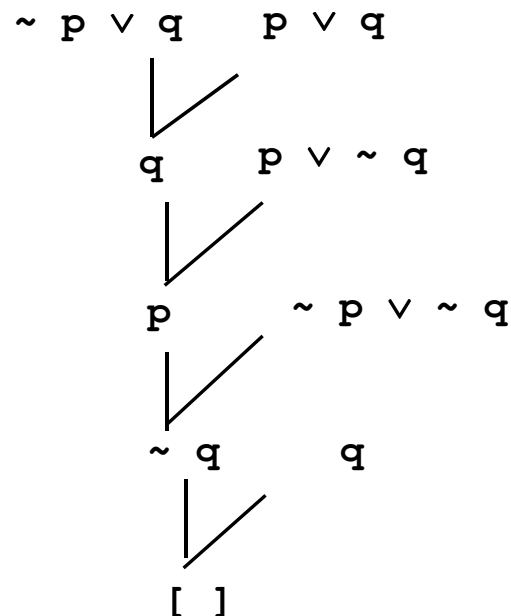


ESEMPIO

- $C0 = \{p \vee q, \sim p \vee q, p \vee \sim q, \sim p \vee \sim q\}$, ottenuto da:

$H = \{q \rightarrow p, \sim q \rightarrow p, p \rightarrow q\}$; $F = q \wedge p$ negando $\sim (q \wedge p)$ cioè $\sim p \vee \sim q$

- Le clausole "parent" rappresentate sulla destra dell'albero sono formule di $C0$ (le prime tre) e un risolvente generato in precedenza (la formula " q ").



ESEMPIO

$$H^C = \{\text{sum}(0, X1, X1), \text{sum}(s(X), Y, s(Z)) \vee \sim \text{sum}(X, Y, Z)\}$$

$$F^C = \{\sim \text{sum}(s(0), s(s(0)), Y1)\}$$

- dove F^C è stato ottenuto negando la formula:

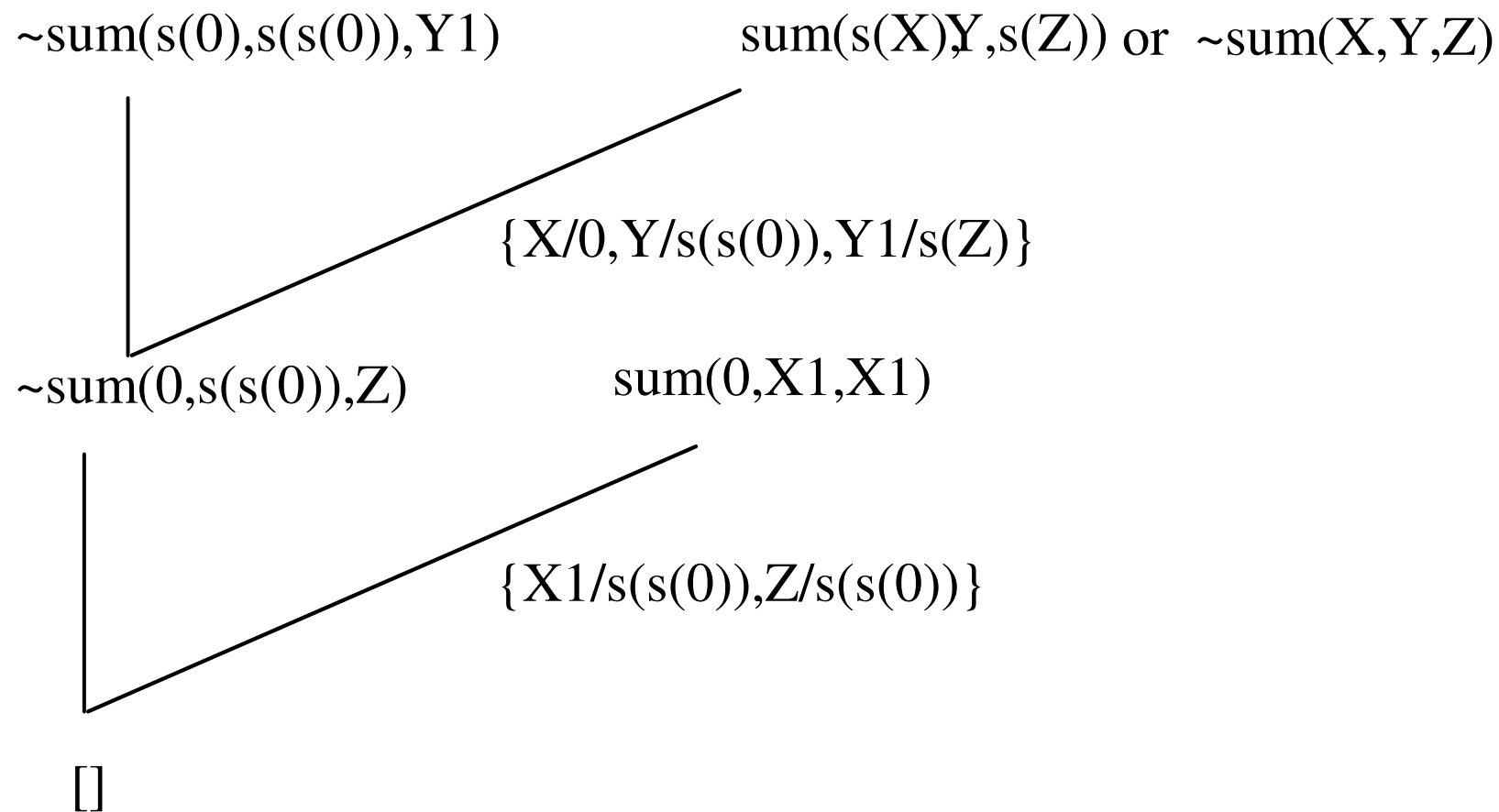
$$F = \exists Y \text{sum}(s(0), s(s(0)), Y)$$

- L'insieme base $C_0 = H^C \cup F^C$ risulta:

$$C_0 = \{\text{sum}(0, X1, X1), \text{sum}(s(X), Y, s(Z)) \vee \sim \text{sum}(X, Y, Z), \\ \sim \text{sum}(s(0), s(s(0)), Y1)\}$$

- Strategie complete producono comunque grafi di refutazione molto grandi \rightarrow inefficienti \rightarrow strategie non complete.

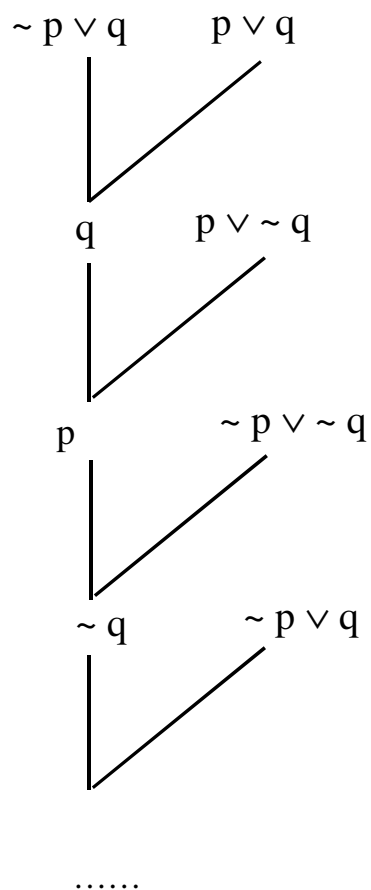
ESEMPIO



STRATEGIE

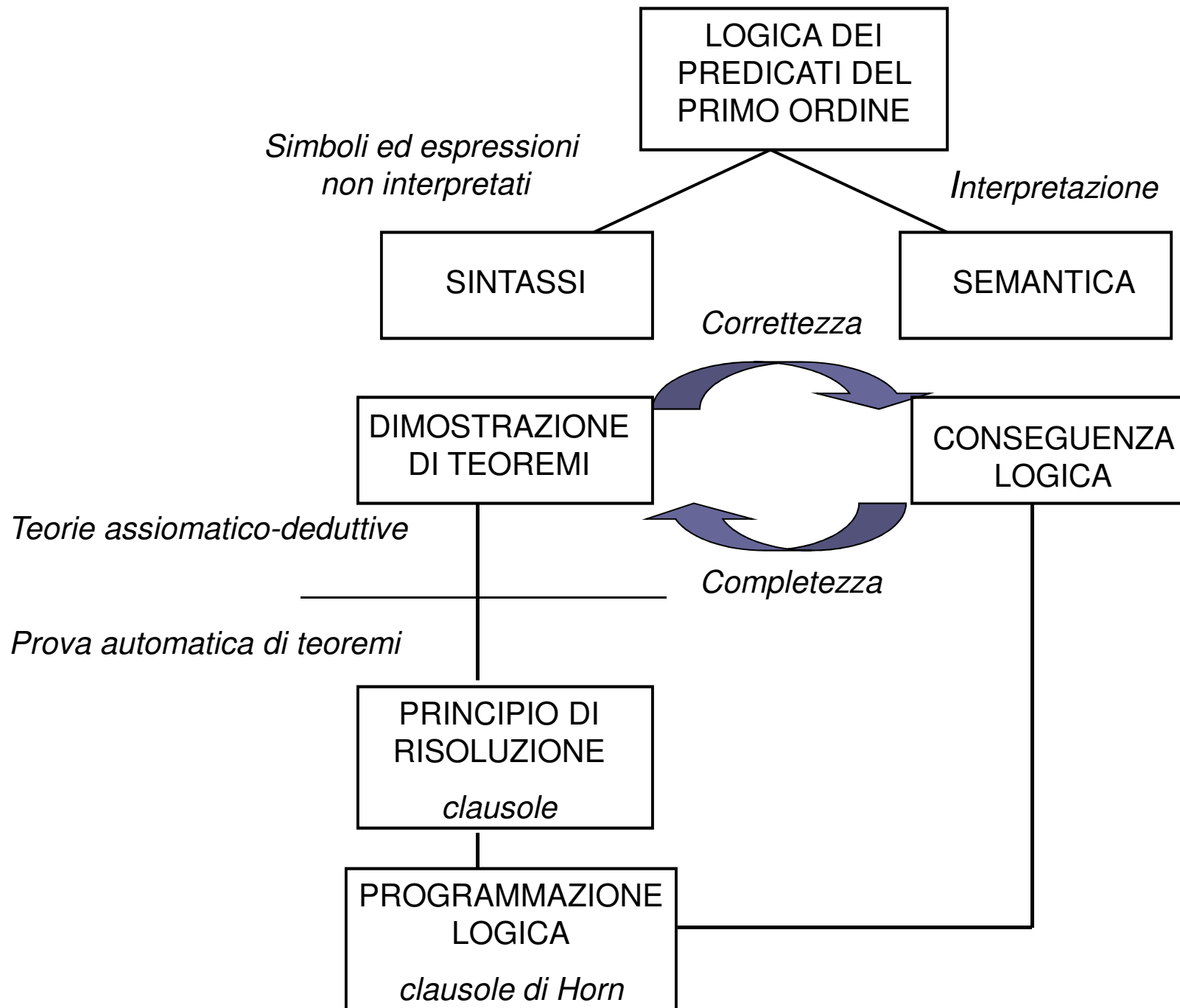
- Strategia **linear-input** non completa sceglie che ha sempre una clausola “parent” nell'insieme base C_0 , mentre la seconda clausola “parent” è il risolvente derivato al passo precedente.
- Caso particolare della risoluzione lineare:
 - vantaggio: memorizzare solo l'ultimo risolvente
 - svantaggio: non completa

ESEMPIO



$C0 = \{p \vee q, \sim p \vee q, p \vee \sim q, \sim p \vee \sim q\}$
 insoddisfacibile, ma la risoluzione con
 strategia “linear-input” produce sempre
 clausole che hanno almeno un letterale, e
 quindi non è in grado di derivare la clausola
 vuota.

Se ci si limita, però, a un
 sottoinsieme delle clausole, in
 particolare alle clausole di Horn,
 allora la strategia “*linear-input*” è
completa.



LE CLAUSOLE DI HORN

- *La logica a clausole di Horn è un sottoinsieme della logica a clausole*
- **Le clausole di Horn hanno al più un letterale positivo.**
- *Le clausole possono essere scritte in una forma equivalente sostituendo al simbolo di disgiunzione \vee e negazione \sim il simbolo di implicazione logica (\rightarrow) ricordando che:*

$$\sim B \vee A \quad \text{equivale a} \quad B \rightarrow A$$

- *Nel seguito si indicherà l'implicazione logica $B \rightarrow A$ con:*

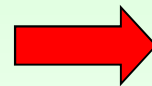
$$A \leftarrow B \quad \text{("B implica A", oppure "A se B").}$$

- *La clausola: $A_1 \vee \dots \vee A_n \vee \sim B_1 \vee \dots \vee \sim B_m$*
- *può essere riscritta come: $A_1, \dots, A_n \leftarrow B_1, \dots, B_m$*
- *dove i simboli "," che separano gli atomi A_i sono da interpretare come disgiunzioni, mentre quelli che separano gli atomi B_j sono congiunzioni.*
- *Clausole di Horn: $A \leftarrow B_1, \dots, B_m \quad \leftarrow B_1, \dots, B_m \text{ Goal}$*

Potenza Espressiva delle Clausole di Horn

- Molte formule della logica dei predicati possono essere scritte come clausole di Horn.
- Esempi:

$\forall x \text{ cat}(x) \vee \text{dog}(x) \rightarrow \text{pet}(x)$



$\text{pet}(x) \leftarrow \text{cat}(x)$
 $\text{pet}(x) \leftarrow \text{dog}(x)$

$\forall x \text{ poodle}(x) \rightarrow \text{dog}(x) \wedge \text{small}(x)$



$\text{dog}(x) \leftarrow \text{poodle}(x)$
 $\text{small}(x) \leftarrow \text{poodle}(x)$

- MA:

$\forall x \text{ human}(x) \rightarrow \text{male}(x) \vee \text{female}(x)$



????

$\forall x \text{ dog}(x) \wedge \sim \text{abnormal}(x) \rightarrow \text{has_4_legs}(x)$



????

ESEMPIO

$H = \{on(b1, tavolo) \wedge on(b2, tavolo) \wedge ((colore(b1, rosso) \vee colore(b2, rosso)))\}$

è esprimibile in clausole, ma non in clausole di Horn.

- La sua trasformazione in clausole risulterà infatti:

$H^C = \{on(b1, tavolo), on(b2, tavolo), colore(b1, rosso) \vee colore(b2, rosso)\}$

dove la clausola: $colore(b1, rosso) \vee colore(b2, rosso)$
non è una clausola di Horn.

- Risoluzione per le clausole di Horn: **risoluzione SLD** *resolution with Selection rule, Linear input strategy for Definite clauses*
- Risoluzione SLD opera per contraddizione e quindi si procede negando la formula F da dimostrare.
- Poiché F è una congiunzione di formule atomiche quantificate esistenzialmente, la sua negazione produrrà una disgiunzione di formule atomiche negate quantificata universalmente, cioè una clausola di Horn goal.

Esempio KB

- 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.
- Dimostra che “Col. West is a criminal”
- Nella traduzione useremo una diversa (inversa) notazione(Russel-Norvig): Variabili x,y,z ecc; Costanti con una maiuscola.

Esempio KB (cont)

... 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":

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

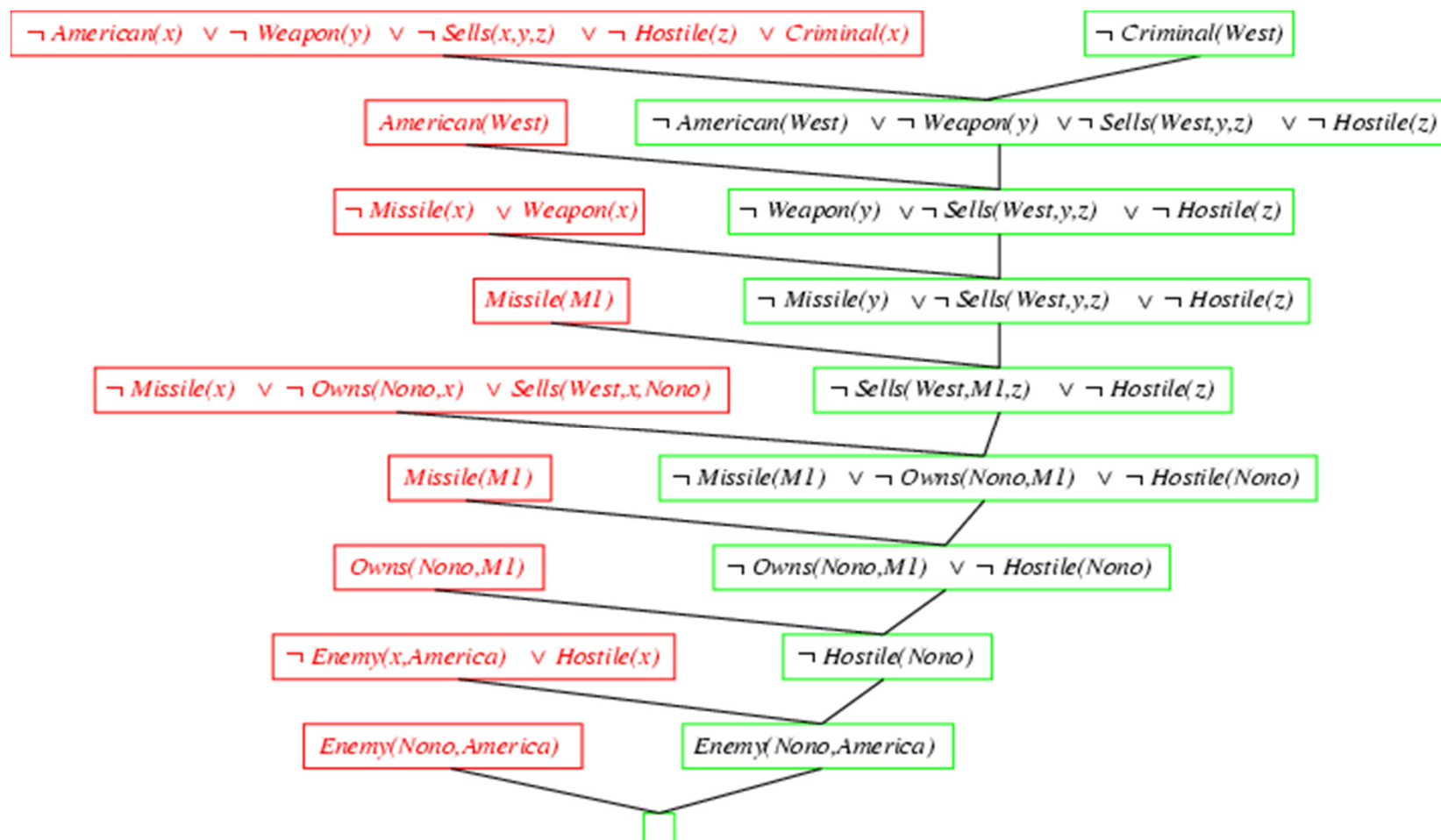
West, who is American ...

$American(West)$

The country Nono, an enemy of America ...

$Enemy(Nono,America)$

Risoluzione linear-input: clausole definite



Forward e backward chaining: Modus Ponens Generalizzato (GMP)

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

- Utilizzato con KB di clausole **definite** (**esattamente un letterale positivo**)
- Tutte le variabili sono universalmente quantificate.
- Corrisponde alla risoluzione con strategia linear input

Forward chaining

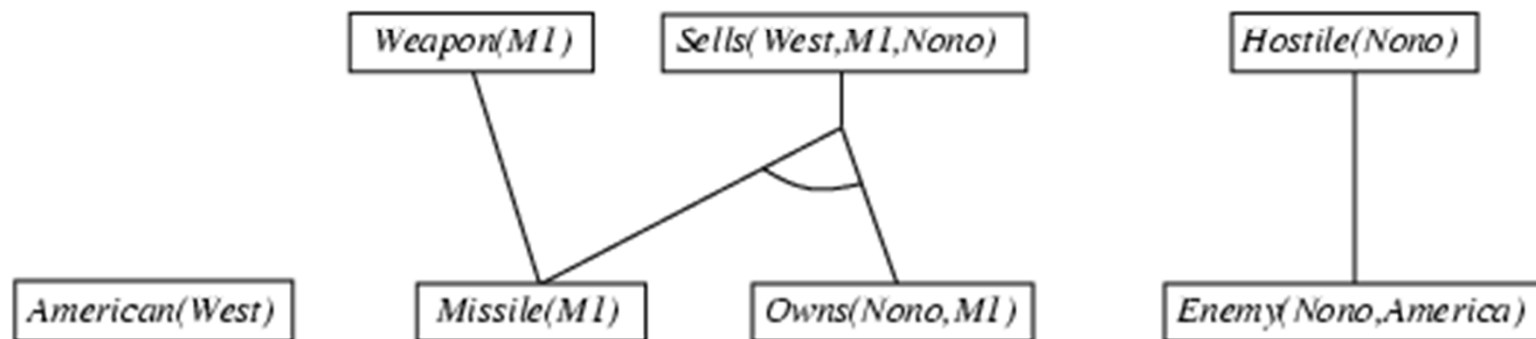
American(West)

Missile(M1)

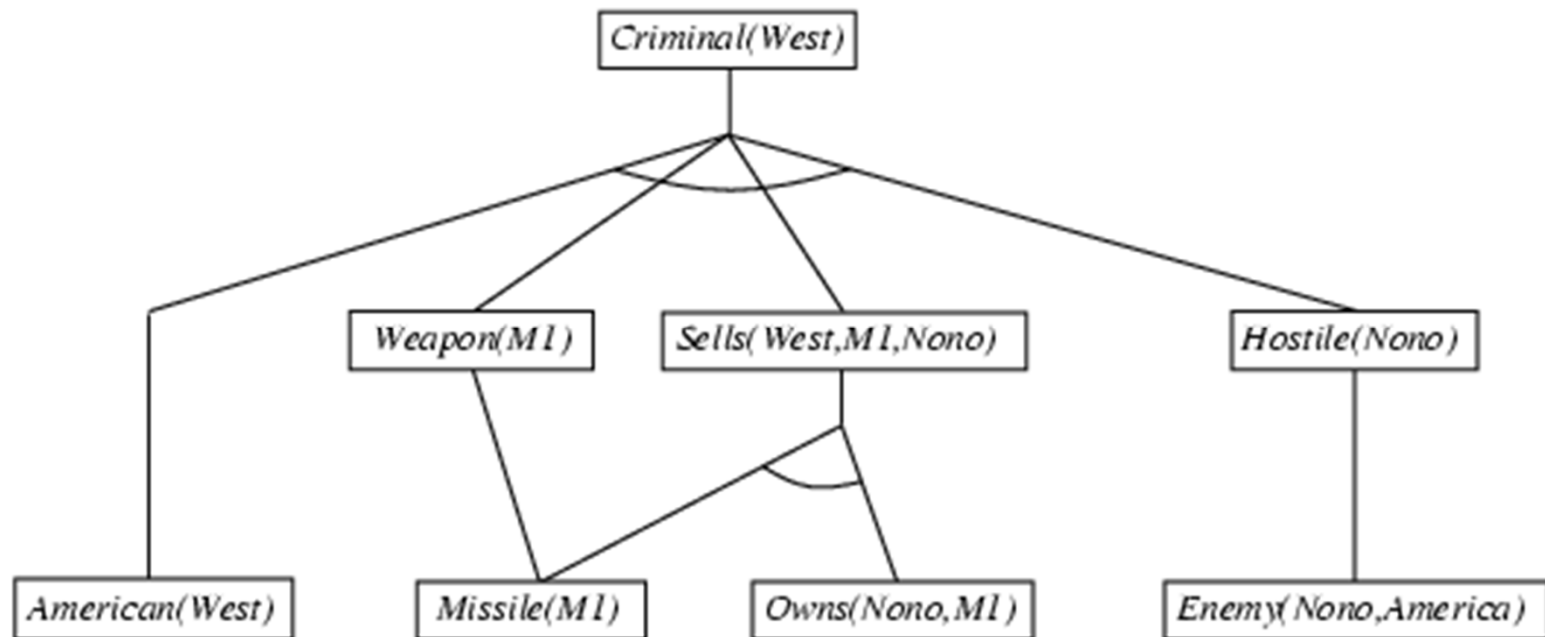
Owns(Nono,M1)

Enemy(Nono,America)

Forward chaining



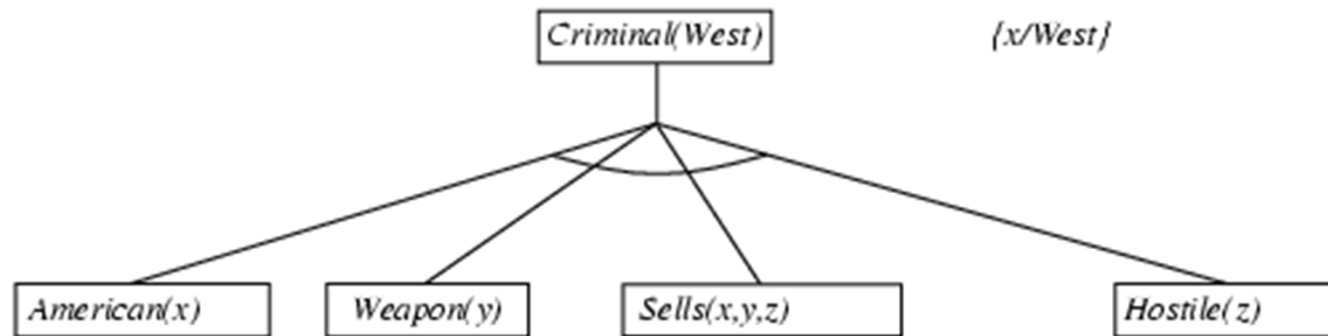
Forward chaining



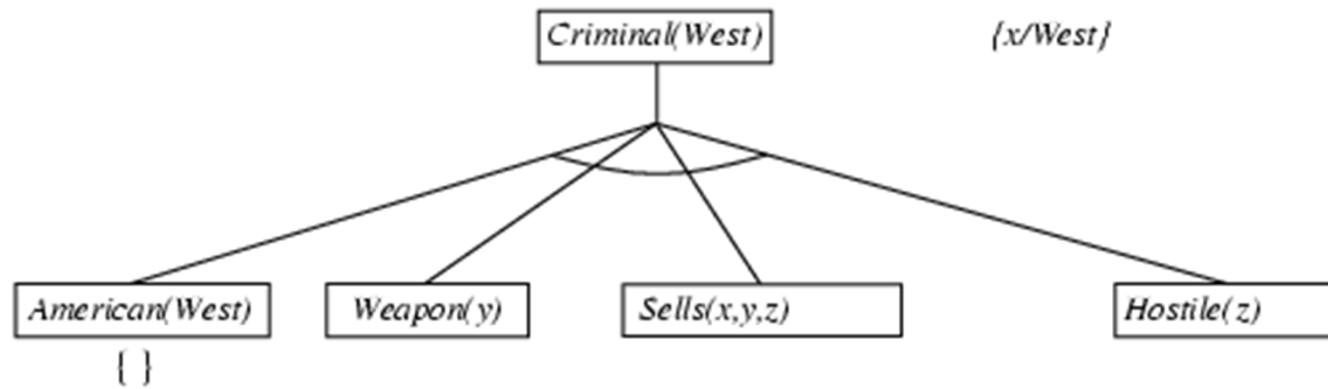
Backward chaining

Criminal(West)

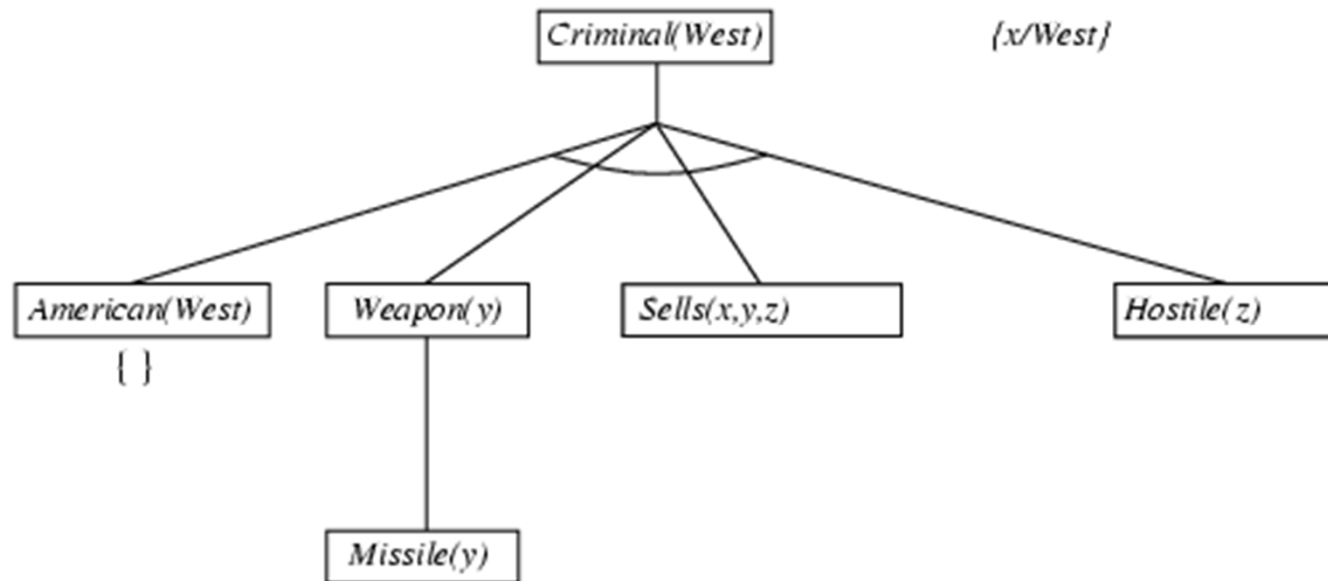
Backward chaining



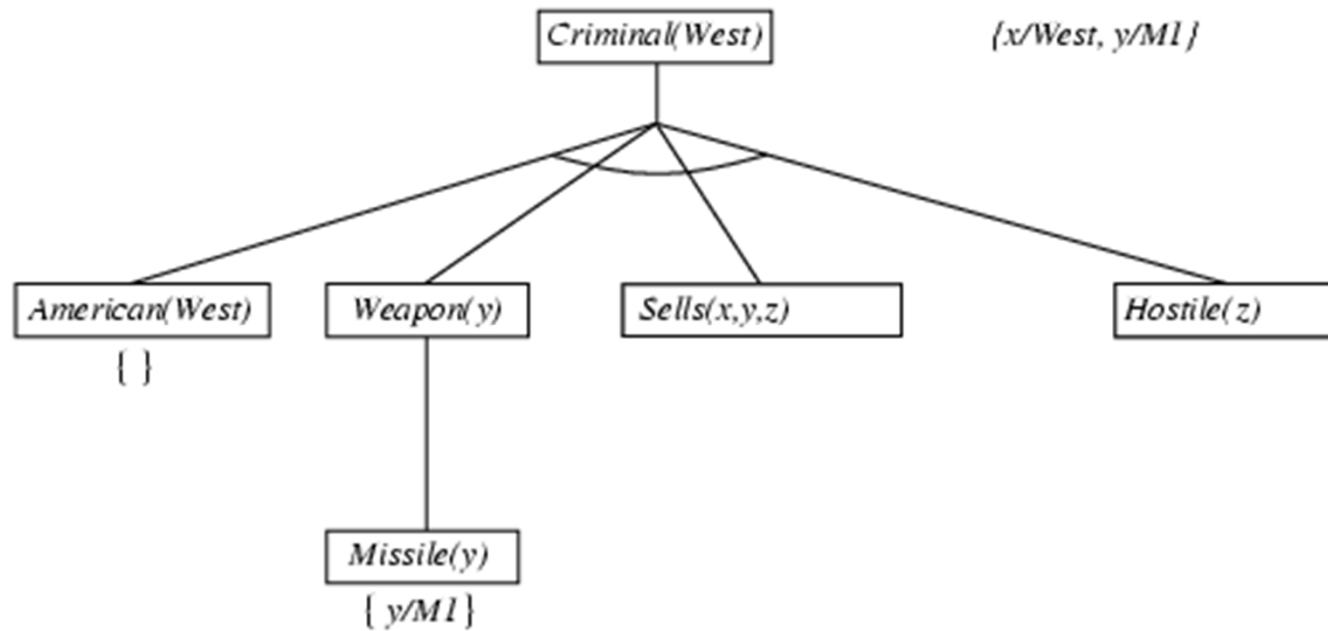
Backward chaining



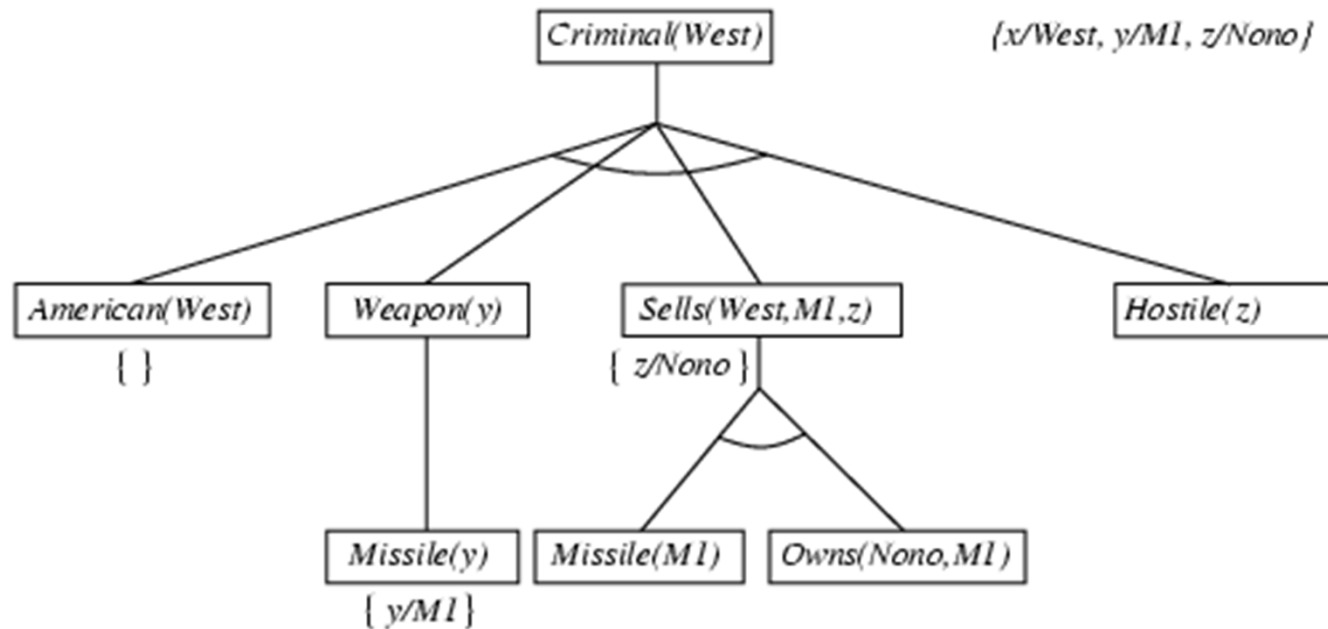
Backward chaining



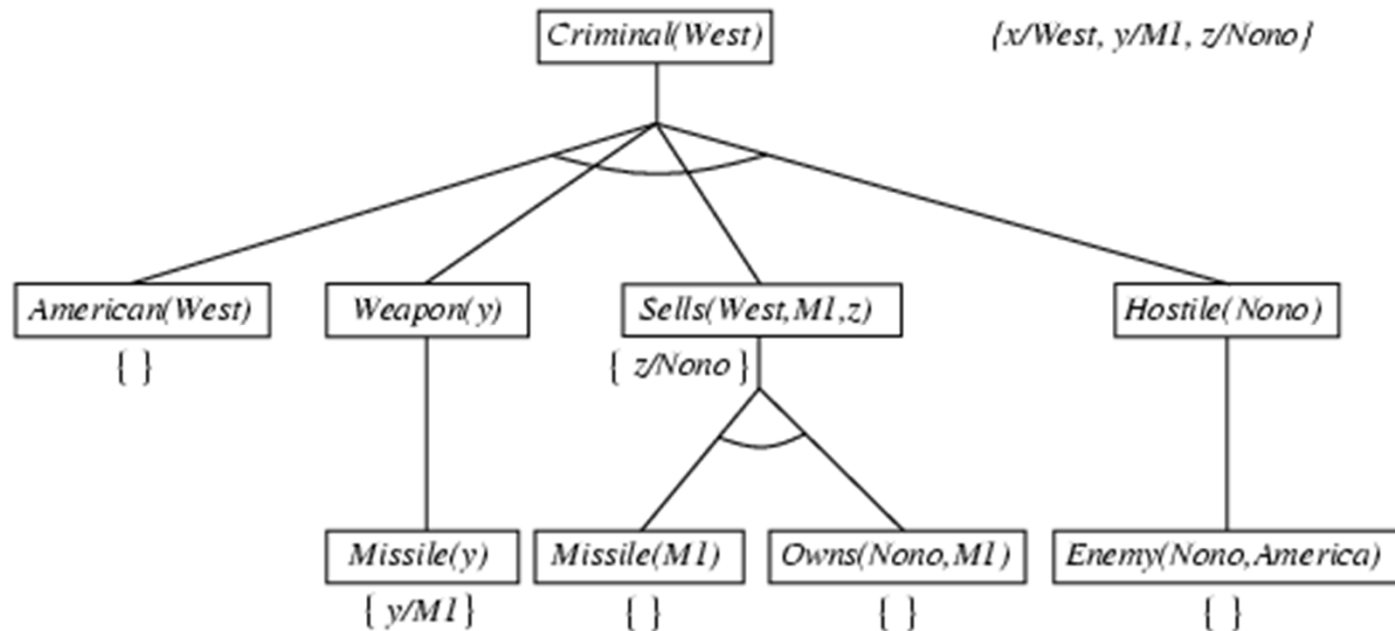
Backward chaining



Backward chaining



Backward chaining



Backward chaining

