Università di Bergamo
Facoltà di Ingegneria

Intelligenza Artificiale

Paolo Salvaneschi

A7_6 V1.5

Inferenza nella logica del primo ordine

Nota: è utilizzato in parte il materiale didattico associato al testo di Stuart J. Russell, Peter Norvig

INDICE

- Inferenza in PL e in FOL
- Riduzione a inferenza proposizionale
- Modus ponens generalizzato
- Unificazione
- Forward chaining
- Backward chaining
- Risoluzione

Inferenza in PL e in FOL

- Tre famiglie di procedure di inferenza:
- Riduzione a inferenza proposizionale
  - Convertire una KB scritta in FOL in una KB (inefficiente) scritta in PL e applicare i metodi di inferenza di PL
- Modus ponens generalizzato, Unificazione, Forward chaining, Backward chaining
  - Estendere le regole di inferenza di PL a FOL
  - Sistemi a produzioni e Prolog (inferenza in KB di definite Horn clauses – restrizione di FOL)
- Risoluzione
  - Estendere le regole di inferenza di PL a FOL
  - Theorem provers (inferenza in full First Order Logic)

Inferenza in PL e in FOL

- Inferenza in PL– estendere i risultati all'inferenza in FOL
- Primo approccio: Convertire una KB scritta in FOL in una KB scritta in PL e applicare i metodi di inferenza di PL
  - Regole di inferenza per trasformare frasi con quantificatori in frasi senza quantificatori
  - Proposizionalizzazione: trasformare frasi in FOL in frasi in PL

Inferenza in PL e in FOL

- Termini ground e sostituzioni
  - Ground term: termine senza variabili (chiuso)
  - Sostituzione dei singoli individui alle variabili.

    Una sostituzione è una funzione dall'insieme delle variabili Var all'insieme dei termini Term. (indica la sostituzione delle variabili con i termini corrispondenti senza variabili)

Inferenza in PL e in FOL

- *SUBST* (θ, α) denota il risultato che si ottiene applicando la sostituzione θ alla frase α

- La sostituzione θ di un termine t al posto di un simbolo di variabile x è indicata da $\theta = \{t/x\}$ oppure $\theta = \{t=x\}$

$$SUBST(\{t/x\}, \alpha)$$

$$SUBST(\{x/Richard; y/John\}, Brother(x,y))$$
$$= Brother(Richard, John)$$

Riduzione a inferenza proposizionale

- Regola di inferenza che trasforma frasi con quantificatore universale in frasi senza quantificatori (Istanziazione universale)

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

Per ogni variabile *v* e ground term *g*

Riduzione a inferenza proposizionale

- Esempio

  La frase:

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

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

Tutti i re avidi sono malvagi

Può portare alle istanziazioni:

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

…….

Ottenute con le sostituzioni {x/John}, {x/Richard}, {x/Father(John)}

Riduzione a inferenza proposizionale

- Regola di inferenza che trasforma frasi con quantificatore esistenziale in frasi senza quantificatori (Istanziazione esistenziale)

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

Per ogni variabile v e k

k simbolo costante
che non appare altrove nella base di conoscenza

Riduzione a inferenza proposizionale

- Esempio

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

La frase:

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

Porta alla istanziazione:

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

C1 è un simbolo che non appare altrove nella base di conoscenza

“C’è un oggetto che soddisfa la frase ed il suo nome è C1.
Questo nome non deve appartenere a nessun altro oggetto”

Riduzione a inferenza proposizionale

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

- Altro esempio

  Il numero 2,71828…

  soddisfa per *x* la frase $\exists x \;\; d(x^y)/dy = x^y$

  Al numero diamo il nome *e* e otteniamo

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

e deve essere il nome di un unico oggetto

Il nuovo nome è chiamato Costante di Skolem

Il processo è chiamato skolemizzazione

Riduzione a inferenza proposizionale

- Nota:

– L'istanziazione universale può essere applicata più volte per aggiungere nuove frasi

– La nuova KB è logicamente equivalente alla precedente

– L'istanziazione esistenziale può essere applicata una sola volta per sostituire la frase quantificata esistenzialmente

– La nuova KB non è logicamente equivalente alla precedente (vere nello stesso insieme di modelli). Si dimostra che è inferenzialmente equivalente: la nuova KB è soddisfacibile (vera in almeno un modello) se e solo se la precedente KB era soddisfacibile (è soddisfacibile esattamente tutte le volte che è KB originale è soddisfacibile)

Riduzione a inferenza proposizionale

- Riduzione a inferenza proposizionale
  Proposizionalizzazione: trasformare frasi in FOL in frasi in PL
- Idea:
  - Sostituire ogni frase quantificata esistenzialmente con una istanza
  - Sostituire ogni frase quantificata universalmente con un insieme di istanze costituite da tutte le possibili istanziazioni
  - Considerare ogni frase atomica ground (senza più variabili) come un simbolo proposizionale
  - Applicare gli algoritmi di inferenza per PL

Riduzione a inferenza proposizionale

- Esempio
- KB contiene le frasi:

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

- L'istanziazione della frase quantificata universalmente in tutti i possibili modi produce:

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

- KB è proposizionalizzata. I simboli proposizionali sono:

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

Riduzione a inferenza proposizionale

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

disgiunzione di letterali di cui esattamente uno è positivo

$\neg A \vee \neg B \vee C$ definite Horn Clause

$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ de Morgan

$\neg (A \wedge B) \vee C$

$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ implication elimination

$(A \wedge B) \Rightarrow C$

$A \wedge B \Rightarrow C$
$D \wedge E \Rightarrow F$
A
B
G

→ Base di conoscenza in PL in Clausole di Horn (definite clauses) Es. Algoritmo di forward chaining (modus ponens) → C $Evil(John)$

Riduzione a inferenza proposizionale

- La tecnica di Proposizionalizzazione può essere resa generale (ogni KB in FOL e ogni query può essere trasformata in PL conservando l'implicazione logica)
- Problema:

  Se ci sono simboli di funzione il numero dei termini che rappresentano individui del dominio è infinito: $\forall x\ King\ (Father(x))$ per ogni x il padre di x è re

  John; Father(John); Father(Father(John));……..

  equivalente: $\forall x\ \ King(x) \Rightarrow King(Father(x))$

Riduzione a inferenza proposizionale

- Teorema di Herbrand (1930): se una frase α è implicata da una KB in FOL, allora α è implicata da un sottoinsieme finito della KB proposizionalizzata
- Idea: per n da 0 a ∞
  - generare una KB proposizionalizzata, istanziando i termini di profondità n
  - Verificare se α è implicata dalla KB generata

KB ottenuta istanziando con i simboli costanti (es. Richard, John)

…….Father (Richard), Father (John)

…….Father (Father (Richard)), Father (Father (John))

…….

Riduzione a inferenza proposizionale

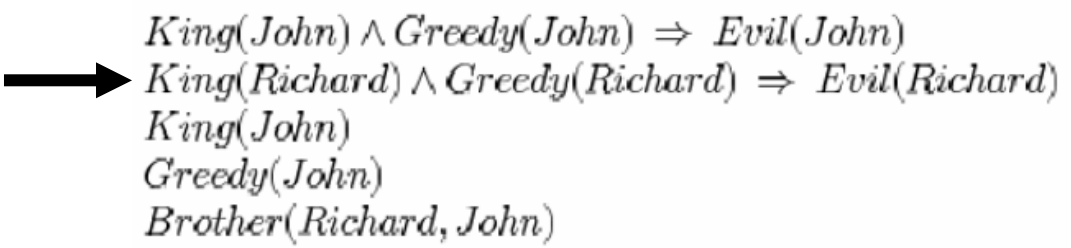
- Completezza (ogni frase implicata può essere provata)
  - Se la frase α è implicata da KB, la prova termina dimostrando α.
  - E se α non è implicata?
- L'implicazione logica in FOL è semidecidibile
  - Se la frase α non è implicata da KB, la procedura di prova può non terminare (la procedura non è in grado di decidere)

- Riduzione a inferenza proposizionale
- Questo era lo stato dell'arte fino ai primi anni 60

450B.C.	Stoics	propositional logic, inference (maybe)
322B.C.	Aristotle	"syllogisms" (inference rules), quantifiers
1565	Cardano	probability theory (propositional logic + uncertainty)
1847	Boole	propositional logic (again)
1879	Frege	first-order logic
1922	Wittgenstein	proof by truth tables
1930	Gödel	$\exists$ complete algorithm for FOL
1930	Herbrand	complete algorithm for FOL (reduce to propositional)
1931	Gödel	$\neg\exists$ complete algorithm for arithmetic
1960	Davis/Putnam	"practical" algorithm for propositional logic
1965	Robinson	"practical" algorithm for FOL—resolution

Riduzione a inferenza proposizionale

- Problema: la proposizionalizzazione è inefficiente



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

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

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

Modus ponens generalizzato

- Non proposizionalizzare; ragionare direttamente in FOL. Modo di ragionare: (Modus ponens generalizzato)

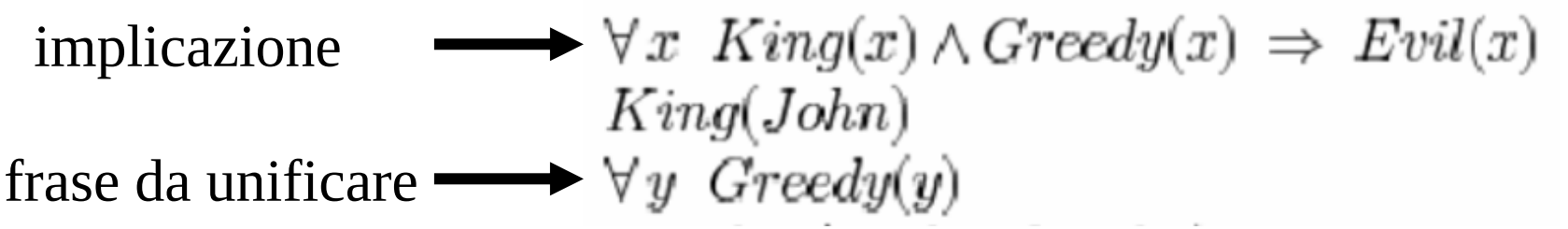
$$\forall x \; King(x) \wedge Greedy(x) \Rightarrow Evil(x)$$
$$King(John)$$
$$Greedy(John)$$

Modus Ponens

$$\frac{\alpha \Rightarrow \beta \quad \alpha}{\beta}$$

- Modo di inferire che $Evil(John)$
  - Trovare una x tale che *x* è *King* e *x* è *Greedy*
  - Inferire che quella *x* è *Evil*
- In generale: se esiste una sostituzione θ -- $\{x/John\}$ che rende la premessa dell'implicazione identica a frasi già in KB, allora asserisci la conclusione, dopo aver applicato θ

Modus ponens generalizzato



- Complicazione: Trovare una sostituzione sia per le variabili dell'implicazione che per le variabili della frase da unificare

  La sostituzione *{x/John, y/John}* rende la premessa dell'implicazione identica a frasi già in KB

Modus ponens generalizzato

- Modus ponens generalizzato

p_i', p_i , q frasi atomiche

$$\frac{p_1',\ p_2',\ \ldots,\ p_n',\ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{SUBST\ (\theta\ ,\ q)}$$

Dove la sostituzione θ è tale che $SUBST\ (\theta\ ,\ p_i') = SUBST\ (\theta\ ,\ p_i)$ per tutti gli i
(Applicare la sostituzione rende la premessa dell'implicazione identica a frasi già in KB)

Esempio

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

$SUBST\ (\theta\ ,\ q)$ is $Evil(John)$

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

- Si dimostra che GMP è sound

Modus ponens generalizzato

- Nota: significato di “generalizzato”

$$\frac{p_1',\ p_2',\ \ldots,\ p_n',\ \ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{SUBST\ (\theta\ ,\ q)}$$

Dove $SUBST\ (\theta\ ,\ p_i')$ = $SUBST\ (\theta\ ,\ p_i)$ per tutti gli i

Modus ponens (PL)

$$\frac{\alpha \Rightarrow \beta \qquad \alpha}{\beta}$$

- E’ meno generale del modus ponens: α non è qualsiasi ma ha una forma speciale (KB di definite clauses- Horn clauses con esattamente un letterale positivo)
- E’ generale nel senso che permette un numero qualsiasi di frasi atomiche p_i'

Unificazione

- Estensione da PL a FOL di regole di inferenza: modus ponens, forward chaining, backward chaining
- Richiedono tutte di trovare sostituzioni che rendono identiche differenti espressioni logiche:

↓

Algoritmo di Unificazione

$$UNIFY(p, q) = \theta \text{ dove } SUBST(\theta, p) = SUBST(\theta, q)$$

Considera due frasi e restituisce, se esiste, un unificatore θ che, applicato alle due frasi, le rende identiche

Unificazione

Esempio: query $Knows(John, x)$

Query Frasi esistenti nella KB Unificatore

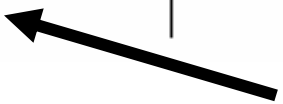
p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	
$Knows(John, x)$	$Knows(y, Mother(y))$	
$Knows(John, x)$	$Knows(x, OJ)$	

Unificazione

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

Unificazione

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



- Fallisce perché *x* non non può prendere i due valori *John* e *OJ*
- Dovremmo essere in grado di inferire: *tutti conoscono OJ* quindi anche *John conosce OJ*
- Soluzione: "standardizing apart" (standardizzazione separata). Rinominare la variabile che collide. Es. $Knows(z_{17}, OJ)$

  Unificatore: $\{ x/OJ, z_{17}/John \}$

Unificazione

- Nota: l'algoritmo di unificazione può restituire più di un unificatore

UNIFY (Knows(John, x), (Knows(y, z)) =
{ y/John, x/z } oppure *{ y/John, x/John, z/John }*

Il primo unificatore genera *Knows (John, z)*
Il secondo genera *Knows (John, John)*

{ y/John, x/z } most general unifier MGU (impone meno restrizioni sul valore delle variabili)
Il secondo risultato *Knows (John, John)* si ricava dal primo *Knows (John, z)*, ottenuto applicando MGU, con una sostituzione aggiuntiva *{ z /John }*

- Algoritmo di inferenza forward chaining
  - Si applica a KB costituite da definite Horn clauses
    disgiunzione di letterali di cui esattamente uno è positivo (implicazione il cui antecedente è una congiunzione di letterali positivi ed il cui conseguente è un unico letterale positivo)
  - Es. KB di definite Horn clauses:

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

- First Order definite Horn clauses

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

  - Come in PL sono disgiunzione di letterali di cui esattamente uno è positivo
  - I letterali in FOL possono includere variabili
  - Si assume che ogni variabili sia quantificata universalmente (si omette la scrittura del quantificatore)
  - Non tutte le KB possono essere scritte in definite clauses (restrizione dell'unico letterale positivo implicato)

Forward chaining

- Idea (come in PL)
  - Partire da frasi atomiche e applicare forward il modus ponens, aggiungendo nuove frasi atomiche fino a quando non sono possibili ulteriori inferenze
- Punti chiave dell'utilizzo del modus ponens generalizzato nell'algoritmo di forward chaining (e backward chaining)
  - Combina piccole inferenze per passi successivi
  - Utilizza l'unificazione per eliminare i quantificatori universali
  - Richiede una KB di definite Horn clauses

Forward chaining

- Esempio: KB e query

The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

Prove that Col. West is a criminal

- Rappresentazione in FOL definite clauses

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

$$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x) .$$

"Nono ... has some missiles." The sentence $\exists x\ Owns(Nono, x) \land Missile(x)$ is transformed into two definite clauses by Existential Elimination, introducing a new constant M_1:

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

Skolemizzazione

"All of its missiles were sold to it by Colonel West":

$$Missile(x) \land Owns(Nono, x) \Rightarrow Sells(West, x, Nono) .$$

Forward chaining

We will also need to know that missiles are weapons:

$$Missile(x) \Rightarrow Weapon(x)$$

and we must know that 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 . . .":

$$Enemy(Nono, America) .$$

Backgroud Knowledge

Vedi: il problema delle assunzioni nella specifica dei requisiti

Nota: KB non contiene simboli di funzioni –es *King (Father(x))*
(Datalog KB)

Forward chaining

- forward chaining
  - Parte dai fatti conosciuti
  - Seleziona tutte le regole le cui premesse sono soddisfatte
  - Aggiunge le conclusioni ai fatti
  - Ripete fino a quando la query è soddisfatta (assumiamo l'esistenza di una query) o non sono aggiunti nuovi fatti

    Nuovo fatto: un fatto non è nuovo se è soltanto una rinominazione di un fatto esistente (fatti identici salvo i nomi delle variabili)

    Es. *Likes (x, IceCream)* e *Likes (y, IceCream)* sono identici. Il significato è lo stesso: a tutti piace il gelato

Forward chaining

Ad ogni iterazione aggiunge alla KB tutte le formule atomiche che possono essere inferite in un passo dalle formule di implicazione e dalle formule atomiche già presenti in KB

function FOL-FC-ASK(KB, α) **returns** a substitution or *false*
  inputs: KB, the knowledge base, a set of first-order definite clauses
    α, the query, an atomic sentence
  local variables: *new*, the new sentences inferred on each iteration

  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 θ such that SUBST($\theta, p_1 \wedge \ldots \wedge p_n$) = SUBST($\theta, p'_1 \wedge \ldots \wedge p'_n$)
          for some $p'_1, \ldots, p'_n$ in KB
        $q' \leftarrow$ SUBST(θ, q)
        if q' is not a renaming of some 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
    add *new* to KB
  return *false*

Se la sostituzione unifica la query restituisce la query unificata

Applica modus ponens generalizzato

Forward chaining

$$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$$
$$Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$$
$$Missile(x) \Rightarrow Weapon(x)$$
$$Enemy(x, America) \Rightarrow Hostile(x)$$

regole

$$American(West)$$
$$Enemy(Nono, America)$$
$$Owns(Nono, M_1)$$
$$Missile(M_1)$$

fatti

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

Forward chaining

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

$$Missile(x) \Rightarrow Weapon(x)$$

$$Owns(Nono, M_1)$$

$$Missile(M_1)$$

$$Missile(M_1)$$

$\{x/M1\}$

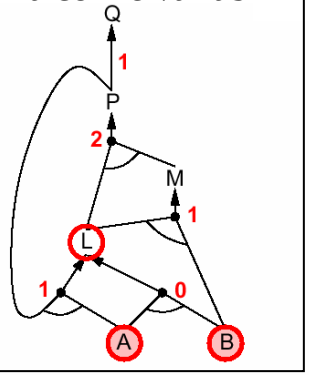
$$Enemy(x, America) \Rightarrow Hostile(x)$$

$$Enemy(Nono, America)$$

$\{x/Nono\}$

Applico il modus ponens generalizzato con l'unificatore $\{x/M1\}$ e aggiungo *Weapon(M1)*

"Come in PL ma con le variabili"



Forward chaining

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

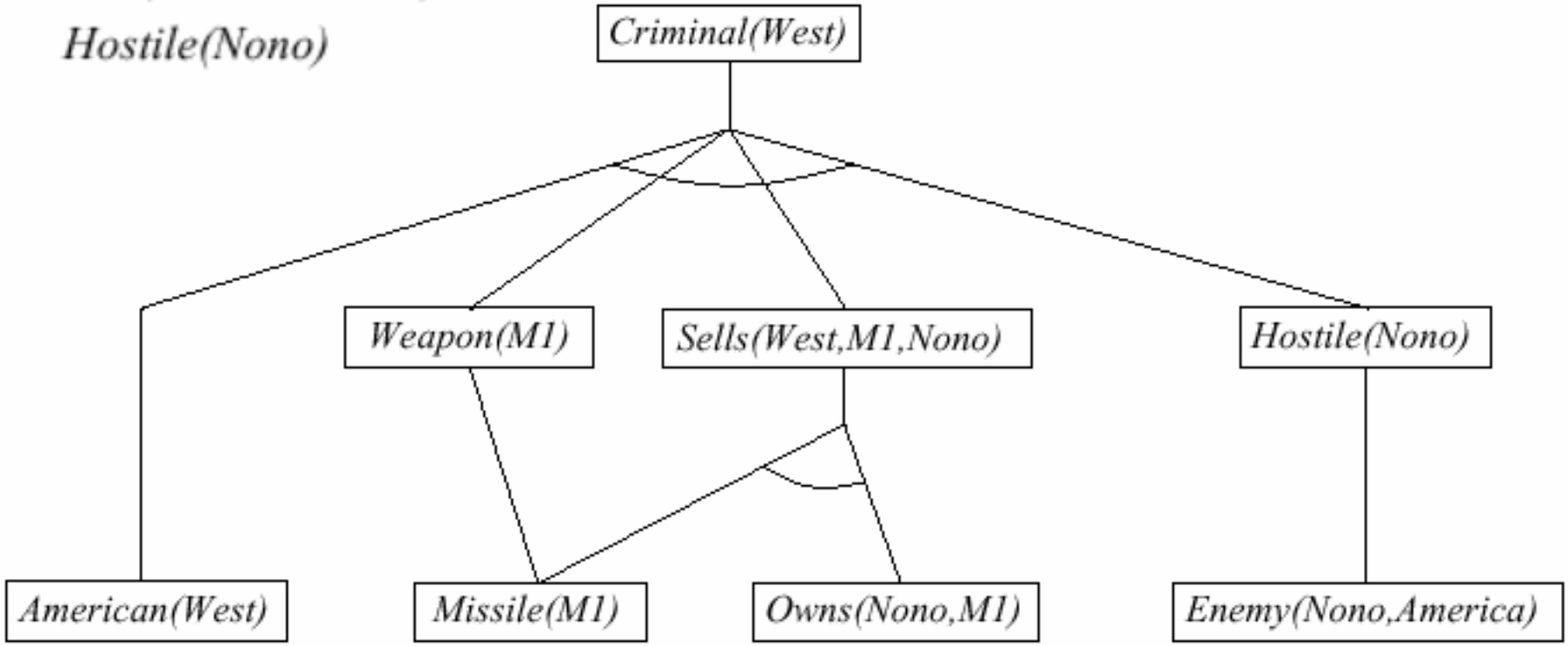
American(West)

Weapon(M1)

Sells(West,M1,Nono)

Hostile(Nono)

④ Applico il modus ponens generalizzato con l'unificatore *{ x/West, y/M1, z/Nono }* e aggiungo *Criminal(West)*



Forward chaining

- Proprietà dell'algoritmo di forward chaining
- Correttezza
  - Ogni inferenza è l'applicazione della regola di inferenza corretta "generalized modus ponens". L'algoritmo e quindi corretto
- Completezza
  - E' completo per definite clause KB

    (completo per Datalog KB- senza simboli di funzione; termina)

    (completo per definite clause KB con funzioni – teorema di Herbrand – se la risposta per la query q è implicata logicamente. L'algoritmo può non terminare se la risposta non è implicata)

    L'implicazione logica in FOL definite clauses è semidecidibile (come per FOL in generale)

Forward chaining

- Forward chaining efficiente
- Problemi dell'algoritmo precedente FOL-FC
  - Trovare tutti i possibili unificatori che unificano le premesse di una regola con i fatti in KB– Pattern matching
    - Il match di una definite clause verso un insieme di fatti è NP-Hard
    - Molte KB reali sono semplici

Forward chaining

– Riverificare ogni volta tutte le regole di KB anche se sono state effettuate poche aggiunte a KB

- Nell'esempio precedente alla seconda iterazione, la regola $Missile(x) \Rightarrow Weapon(x)$ può essere unificata con il fatto $Missile(M_1)$

  La conclusione è già nota dall'iterazione precedente ed è inutile aggiungerla a KB
- Ogni nuovo fatto inferito all'iterazione t deve essere derivato da almeno un fatto nuovo inferito all'iterazione t-1 (Forward chaining incrementale)
- Algoritmo RETE

Forward chaining

- Generare fatti irrilevanti per l’obiettivo
  - Selezionare le regole a cui applicare l’algoritmo di forward chaining (gestione del conflict set)
  - Utilizzare l’algoritmo di backward chaining

- Backward chaining
  - Parte da una lista di goal (uno stack)
  - Quando tutti i goal nello stack sono soddisfatti la prova ha successo
  - Seleziona il goal in testa allo stack e trova ogni clausola in KB il cui letterale positivo (head) (conseguenza della regola) unifica con il goal.
  - La premessa della clausola (body) è aggiunta allo stack (chiamando ricorsivamente la procedura)
  - Nota: un fatto è una clausola con solo l'head: quando un goal unifica con un fatto, non si aggiunge nulla allo stack ed il goal è risolto

Backward chaining

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

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

$Missile(x) \Rightarrow Weapon(x)$

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

$American(West)$

$Enemy(Nono, America)$

$Owns(Nono, M_1)$

$Missile(M_1)$

regole

fatti

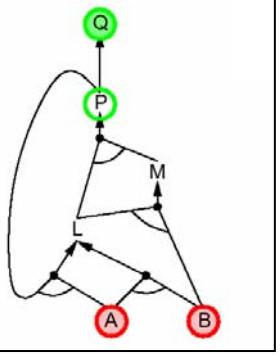
head

unifica con
unificatore
{ x/West }

Criminal(West) goal

Backward chaining

"Come in PL
ma con le variabili"



unificatore

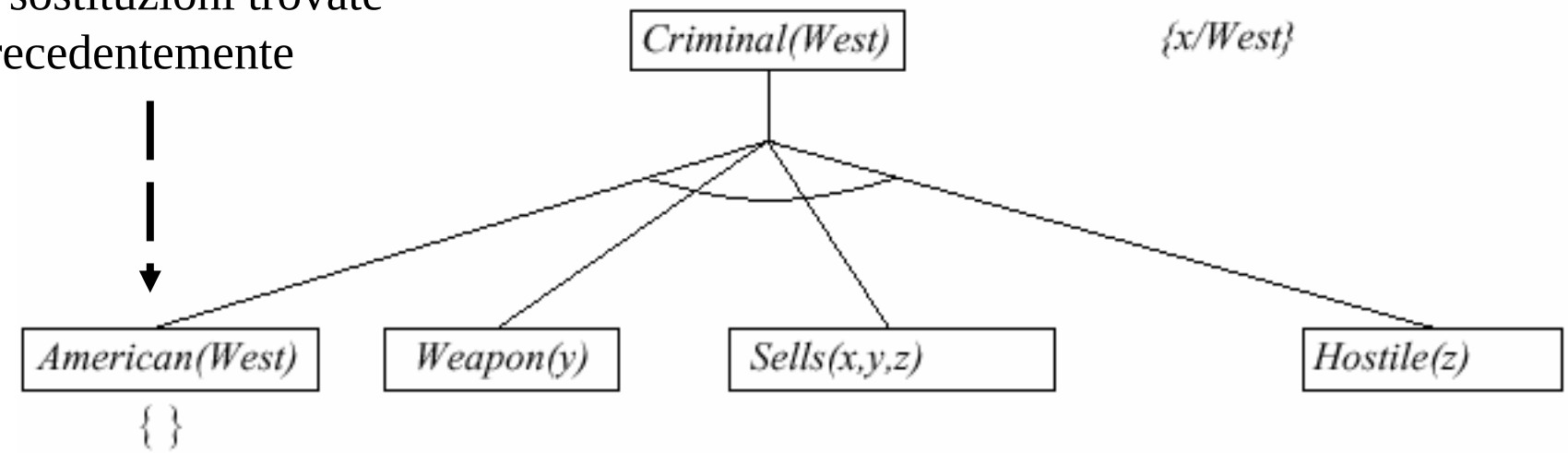
{x/West}

La premessa della clausola (body) è aggiunta allo stack
(tutte le premesse diventano goal)

Albero costruito depth first left to right

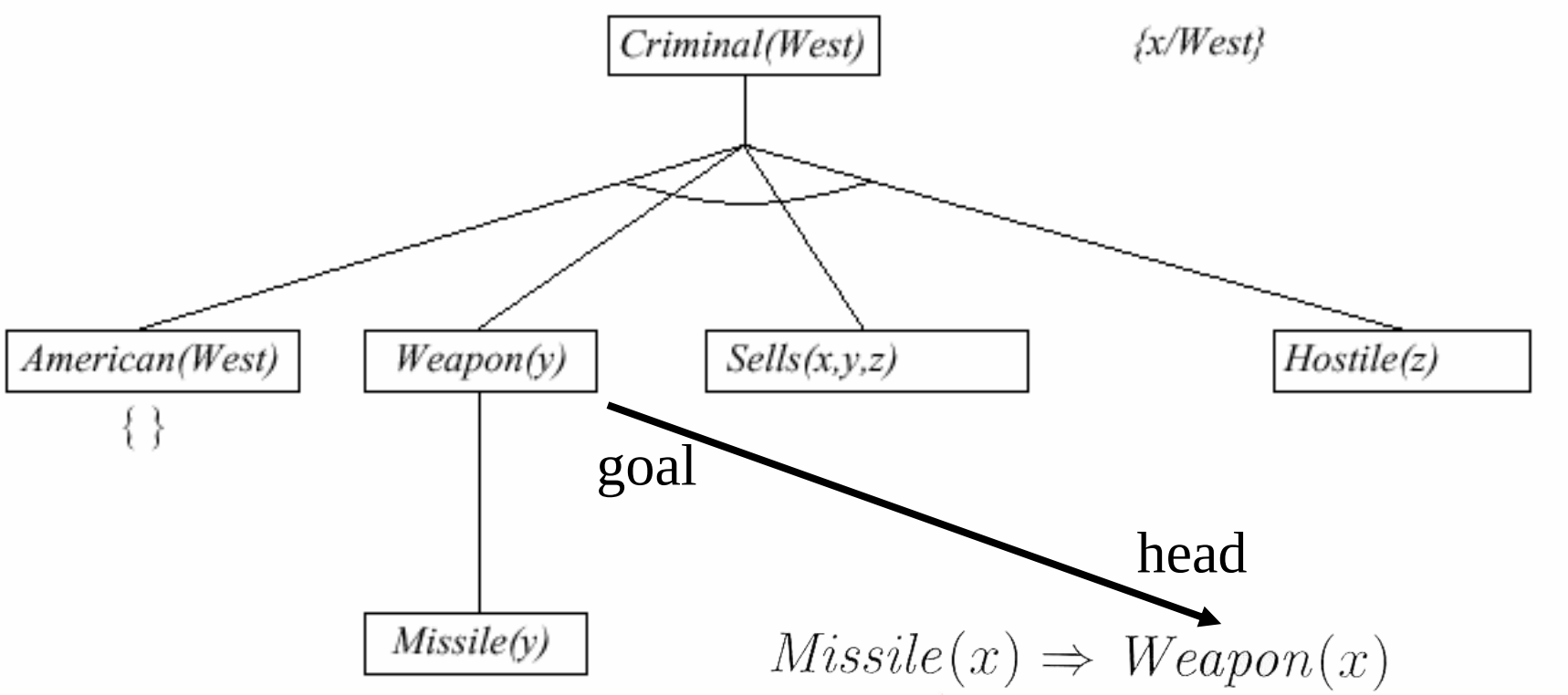
Backward chaining

Nota: al subgoal da provare
sono applicate tutte
le sostituzioni trovate
precedentemente



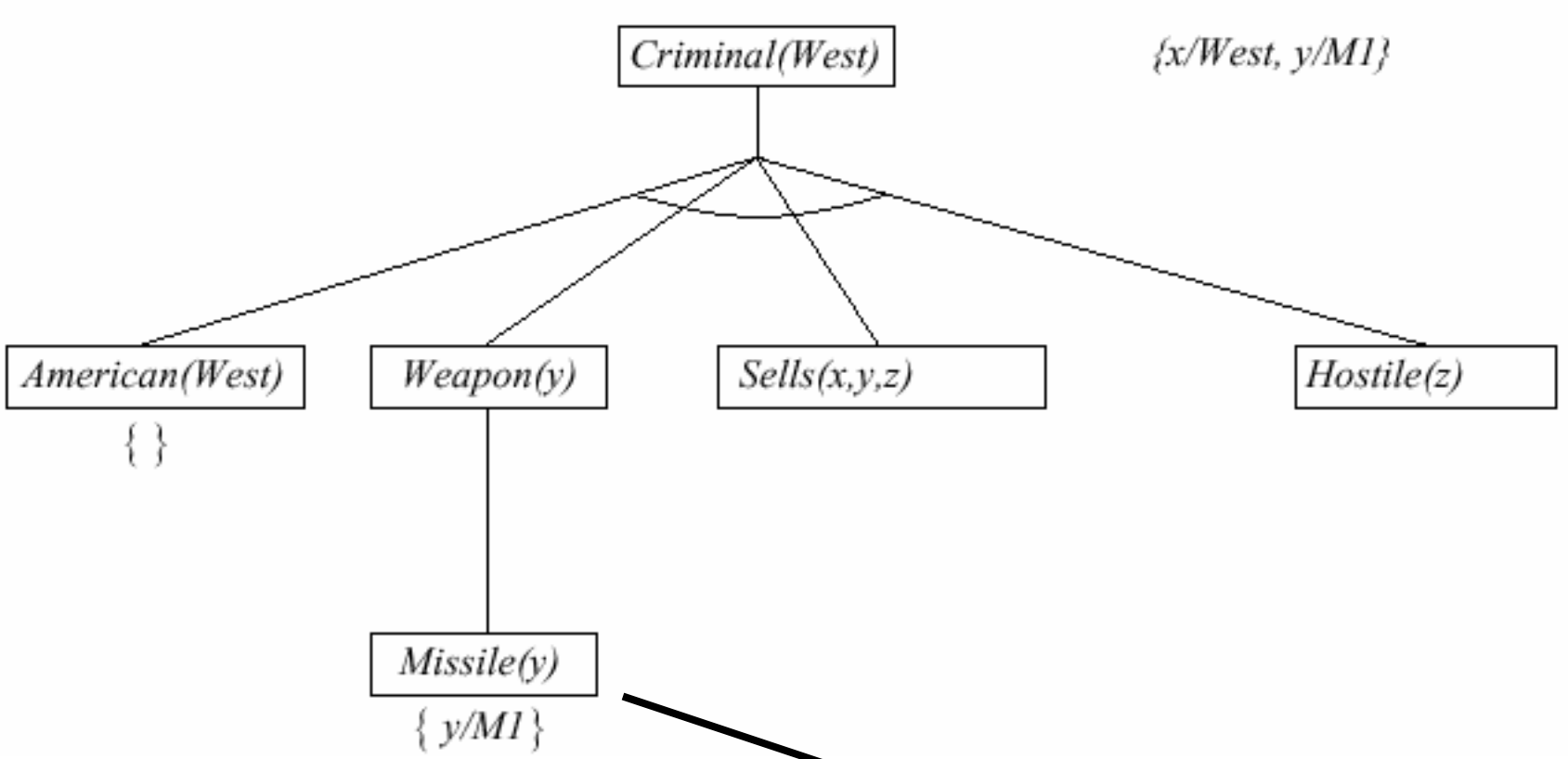
il goal $American(West)$ unifica con il fatto $American(West)$ (head)
Non si aggiunge nulla allo stack ed il goal è risolto

Backward chaining



x rinominato con *y (il significato è lo stesso)*
La premessa della clausola (body) è aggiunta allo stack (diventa un goal)

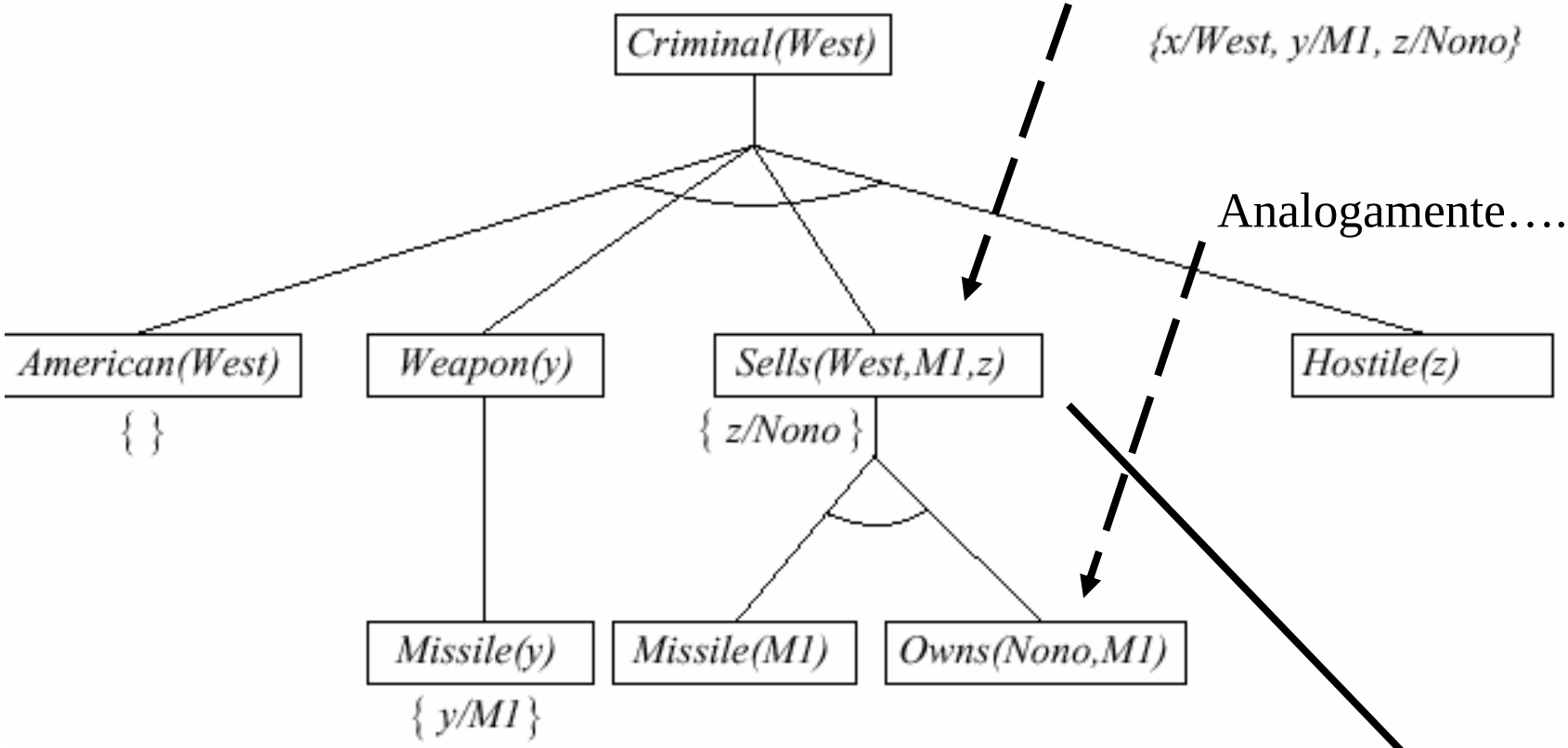
Backward chaining



il goal unifica con il fatto $Missile(M_1)$
con unificatore *{ y/M1 }*
Non si aggiunge nulla allo stack ed il goal è risolto

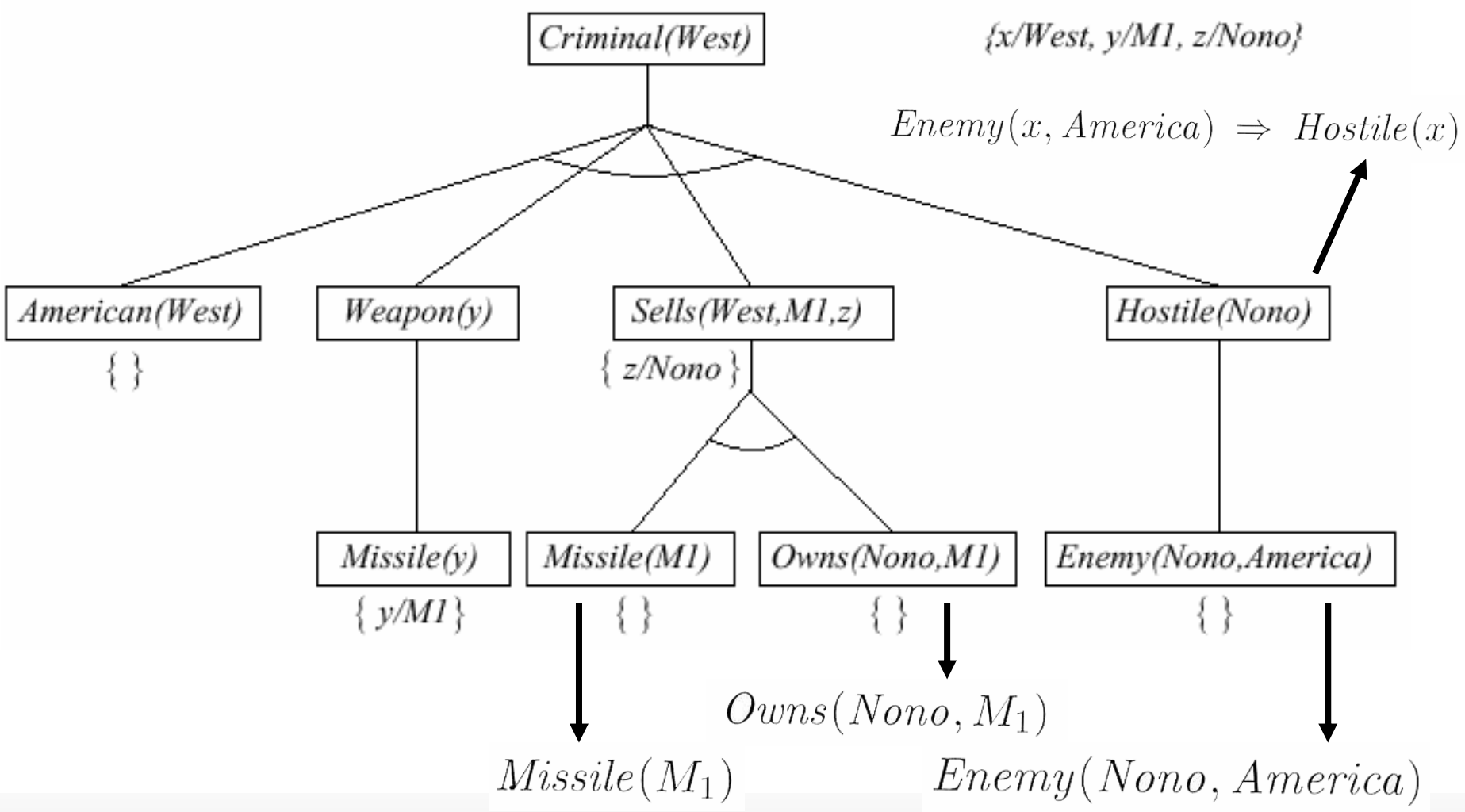
Backward chaining

Nota: al subgoal da provare sono applicate tutte le sostituzioni trovate precedentemente *{ x/West, y/M1 }*



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

Backward chaining



Backward chaining

- Proprietà dell'algoritmo di backward chaining
  - Corretto
  - Incompleto (loop infinito- dept first search con stati ripetuti e percorsi infiniti)

Backward chaining

- Programmazione logica – Prolog
  - Algorithm= logic + control
  - Insieme di definite Horn clauses + depth first backward chaining.
  - Scelte realizzative (extra-logiche):
    - Clausole esaminate sequenzialmente nell'ordine di scrittura nella KB.
    - Ordine di selezione dei goal nel risolvente (sinistra destra)

```
head :- literal1, ... literaln.

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

- Programmazione logica – Prolog
  - Aspetti aggiuntivi (oltre la logica)
    - Funzioni di calcolo (“provate” eseguendo codice)
    - Predicati di input/output e di modifica della KB (assert, retract). Generano side effects quando eseguiti.
    - Negation as failure (¬ P è provato se il sistema fallisce nel provare P)

```
e.g., given alive(X) :- not dead(X).
alive(joe) succeeds if dead(joe) fails
```

- Programmazione logica -- Prolog

	Logic programming	Ordinary programming
1.	Identify problem	Identify problem
2.	Assemble information	Assemble information
3.	Tea break	Figure out solution
4.	Encode information in KB	Program solution
5.	Encode problem instance as facts	Encode problem instance as data
6.	Ask queries	Apply program to data
7.	Find false facts	Debug procedural errors

Risoluzione

- Risoluzione in PL: procedura di inferenza corretta e completa (refutazione - non per enumerare le frasi vere ma solo per confermare o rifiutare una frase)
- Corrispondente corretto e completo in FOL?

Risoluzione

- Kurt Gödel
  - (1930) Teorema di completezza per FOL: ogni frase implicata logicamente ha una prova finita.
  - Modo pratico di scrivere la prova: risoluzione; Robinson (1965)
  - (1931) Teorema di incompletezza: ogni sistema logico che include il principio dell'induzione (base per la costruzione della matematica) è necessariamente incompleto; contiene frasi che sono implicate logicamente ma per le quali non esiste prova finita
- Applicazioni dei dimostratori di teoremi basati sulla risoluzione: derivazione di teoremi matematici, costruzione di programmi, verifica di progetti hardware

Risoluzione

- Risoluzione. Due passi:
  - Conversione delle frasi in CNF (forma normale congiuntiva) con eliminazione dei quantificatori esistenziali (Skolemizzazione)
    - Ogni frase in FOL può essere convertita in una frase CNF inferenzialmente equivalente -- la nuova KB è non soddisfacibile se e solo se la precedente KB era non soddisfacibile; base per la prova per contraddizione)
  - Applicazione della regola di risoluzione per FOL

- Conversione delle frasi in CNF

Esempio di conversione della frase:

Everyone who loves all animals is loved by someone:

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

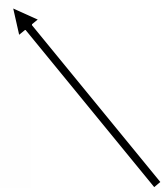
---

1. Eliminate biconditionals and implications $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ implication elimination

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

2. Move $\neg$ inwards: $\neg\forall x \ p \equiv \exists x \ \neg p$, $\neg\exists x,p \equiv \forall x \ \neg p$:

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



$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ de Morgan

Risoluzione

3. **Standardize variables:** each quantifier should use a different one

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

Persona x
Animale y
Persona z

4. **Skolemize:** a more general form of existential instantiation.
   Each existential variable is replaced by a Skolem function
   of the enclosing universally quantified variables:

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

5. **Drop universal quantifiers:** Tutte le variabili sono quantificate universalmente

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

6. **Distribute $\wedge$ over $\vee$:** $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$

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

La frase è composta dalla congiunzione di due clausole (disgiunzioni di letterali)

- Nota sulla skolemizzazione

  Se si applica la skolemizzazione sostituendo alle variabili da istanziare esistenzialmente i simboli costanti unici in KB “A” e “B”, si ottiene:

$$\forall x \ [Animal(\ A\ ) \wedge \neg Loves(x,\ A\ )] \vee Loves(\ B\ , x)$$

  significato modificato: ognuno o non ama uno specifico animale o è amato da un’entità B

  Significato originale: una persona x o non ama un animale o è amato da un’altra persona: L’entità di Skolem deve dipendere da x. Si sostituisce ad ogni variabile una funzione di x – variabile quantificata universalmente nel cui campo di azione appaiono i quantificatori esistenziali (Skolem function)

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

Risoluzione

Promemoria: risoluzione in PL

- Clausole
- Una clausola è una disgiunzione di letterali ℓ
  Letterale: frase atomica (letterale positivo) o frase atomica negata (letterale negativo)

$$\ell_1 \vee \ell_2 \qquad \neg\ell_2 \vee \ell_3$$

- Un letterale è considerato come una disgiunzione di un letterale (clausola unitaria)
- Letterali complementari: uno la negazione dell'altro

Risoluzione

- Unit resolution inference rule

  ℓ letterale ℓ_i m letterali complementari

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k}$$

m eliminato

ℓ_i eliminato

Promemoria: risoluzione in PL

- Full resolution inference rule

  ℓ_i m_j letterali complementari

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

ℓ_i eliminato

m_j eliminato

Risoluzione

- Applicazione della regola di risoluzione per FOL

- Letterali complementari in PL:
  uno la negazione dell'altro

- Letterali complementari in FOL:
  uno unifica con la negazione dell'altro

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

Considera due frasi e restituisce,se esiste,
un unificatore θ che, applicato alle due frasi, le rende identiche

- Applicazione della regola di risoluzione per FOL

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

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

ℓ_i eliminato

m_j eliminato

– Come per PL
– Clausole standardized apart (non condividono variabili) per evitare collisioni durante l'unificazione
– Sono risolte due clausole che contengono letterali complementari
– Due letterali sono complementari in FOL se uno si unifica con la negazione dell'altro
– Si applica l'unificatore per produrre la clausola risolvente
– Fattorizzazione (eliminazione letterali ridondanti)

Risoluzione

Esempio:

$$[Animal(F(x)) \vee Loves(G(x), x)] \quad \text{and} \quad [\neg Loves(u, v) \vee \neg Kills(u, v)]$$

Si eliminano i letterali complementari $Loves(G(x), x)$ e $\neg Loves(u, v)$

con l'unificatore $\theta = \{u/G(x), v/x\}$.

Risultato(con l'unificatore applicato) :

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

Risoluzione

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

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

$Missile(x) \Rightarrow Weapon(x)$

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

$American(West)$

$Enemy(Nono, America)$

$Owns(Nono, M_1)$

$Missile(M_1)$

$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ implication elimination

$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ de Morgan

Convertita in CNF:

$\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$

$\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$.

$\neg Enemy(x, America) \vee Hostile(x)$.

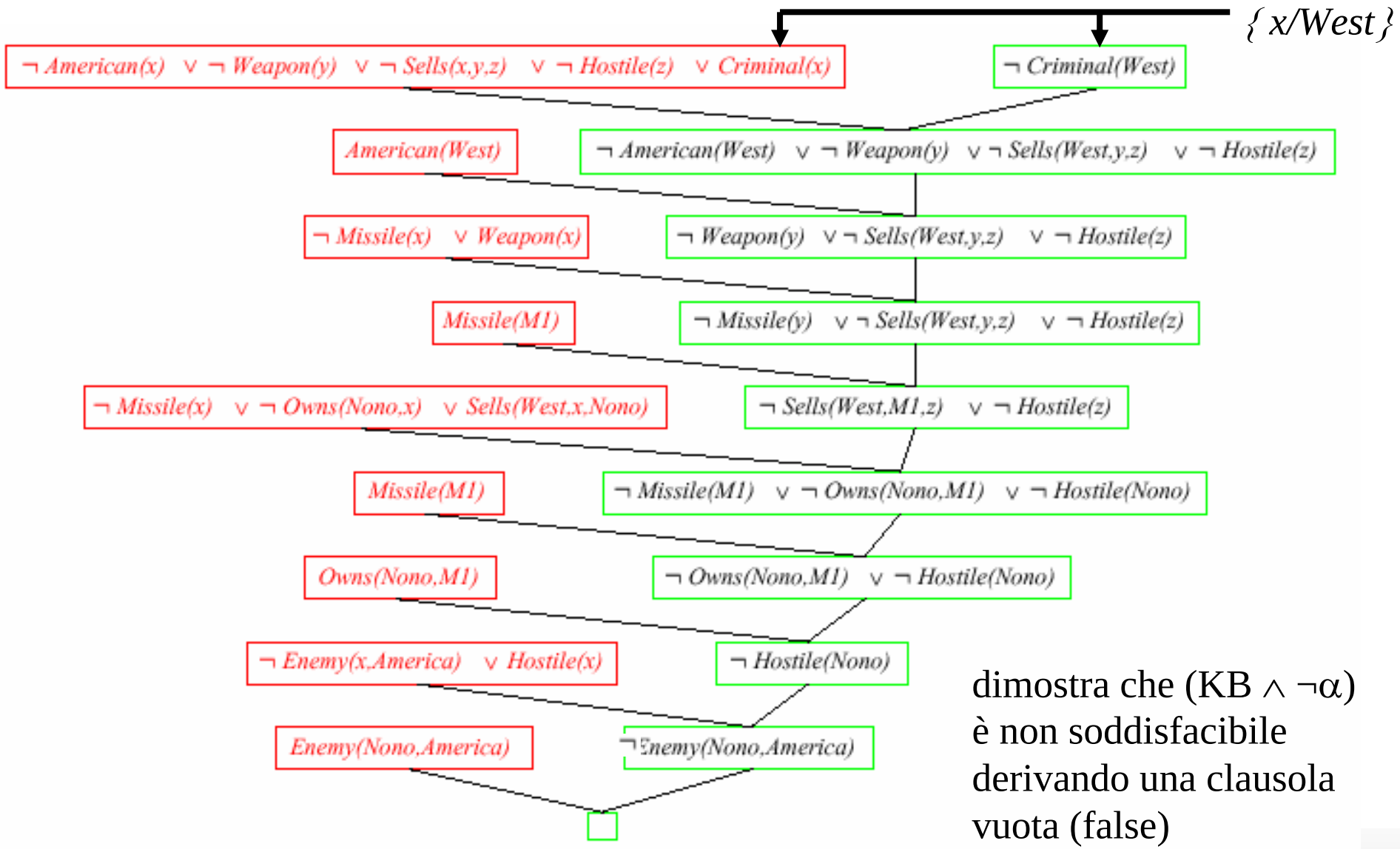
$\neg Missile(x) \vee Weapon(x)$.

$Owns(Nono, M_1)$. $Missile(M_1)$.

$American(West)$. $Enemy(Nono, America)$.

$\neg Criminal(West)$ ⟵ Goal negato

Risoluzione



dimostra che (KB $\wedge \neg\alpha$) è non soddisfacibile derivando una clausola vuota (false)

Risoluzione

- Completezza della risoluzione
  - Refutation complete: se un insieme di frasi è non soddisfacibile, la risoluzione è sempre capace di derivare la contraddizione
  - Non può essere usata per generare tutte le conseguenze logiche ma per dire se una frase è implicata logicamente da un insieme di frasi