Regole Inferenza FOL

- Tutte quelle della logica proposizionale….
- Più tre regole per variabili quantificate

1. Eliminazione variabile universale:

$$\frac{\forall x\ Piace(x, GELATO)}{?}$$

2. Eliminazione variabile esistenziale:

$$\frac{\exists x\ Piace(x,\ GELATO)}{?} \qquad \frac{\forall x \exists y\ Ama(x, y)}{?} \qquad \frac{\forall x \forall y \exists z\ Connesso(x, y, z)}{?}$$

Regola generale (*skolemizzazione*)

Regole Inferenza FOL

3. Introduzione Variabile Esistenziale:

$$\frac{Piace(MARIO, GELATO)}{\exists x\ Piace(MARIO, x)}$$

$$\exists x \exists y\ Piace(y, x)$$

4. E Introduzione Variabile Universale??

Generalized Modus Ponens (GMP)

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

dove $p_i'\theta = p_i\,\theta$ for all *i*

p_1' è *King*(*John*)
p_2' è *Avido*(*John*)
θ è {x/John}
qθ è *Malvagio*(*John*)

p_1 è *King*(*x*)
p_2 è *Avido*(*x*)
q è *Malvagiol*(*x*)

- GMP usato con KB di definite clauses (esattamente 1 letterale +)
- Tutte le variabili sono assunte quantificate universalmente

Example knowledge base (Russel & Norvig)

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.

- Formalize as first order definite clauses
- Prove that Col. West is a criminal using GMP

Example knowledge base
(all vars assumed universally quantified)

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

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

Nono … has some missiles, i.e., $\exists x$ Owns(Nono,x) $\wedge$ Missile(x):

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

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

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

Missiles are weapons:

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

An enemy of America counts as "hostile“:

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

West, who is American …

$American(West)$

The country Nono, an enemy of America …

$Enemy(Nono,America)$

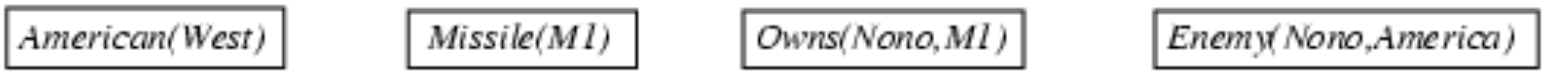
Forward chaining algorithm

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

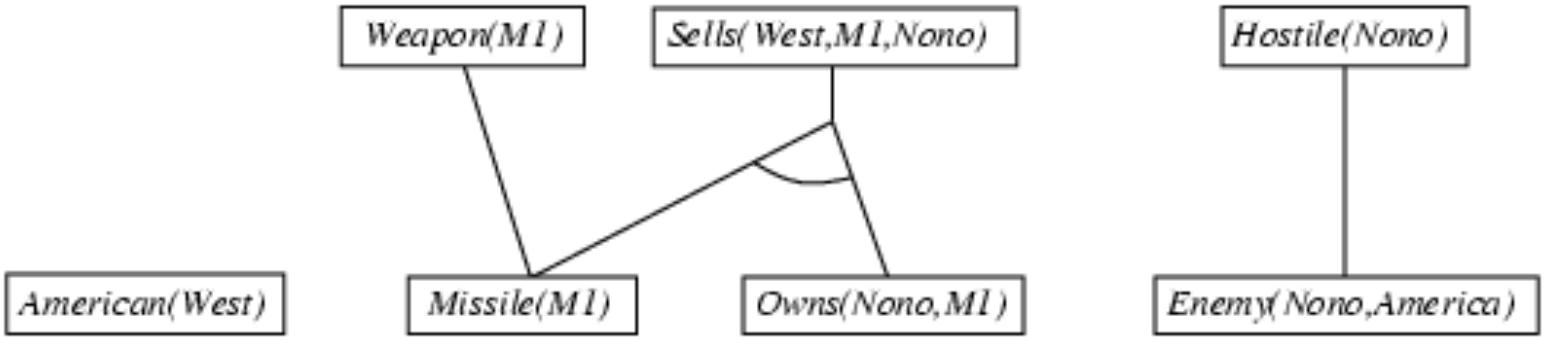
repeat until *new* is empty
- $new \leftarrow \{\}$
- **for each** sentence r **in** KB **do**
  - $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART(r)
  - **for each** θ such that $(p_1 \wedge \ldots \wedge p_n)\theta = (p'_1 \wedge \ldots \wedge p'_n)\theta$ for some $p'_1,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

return *false*

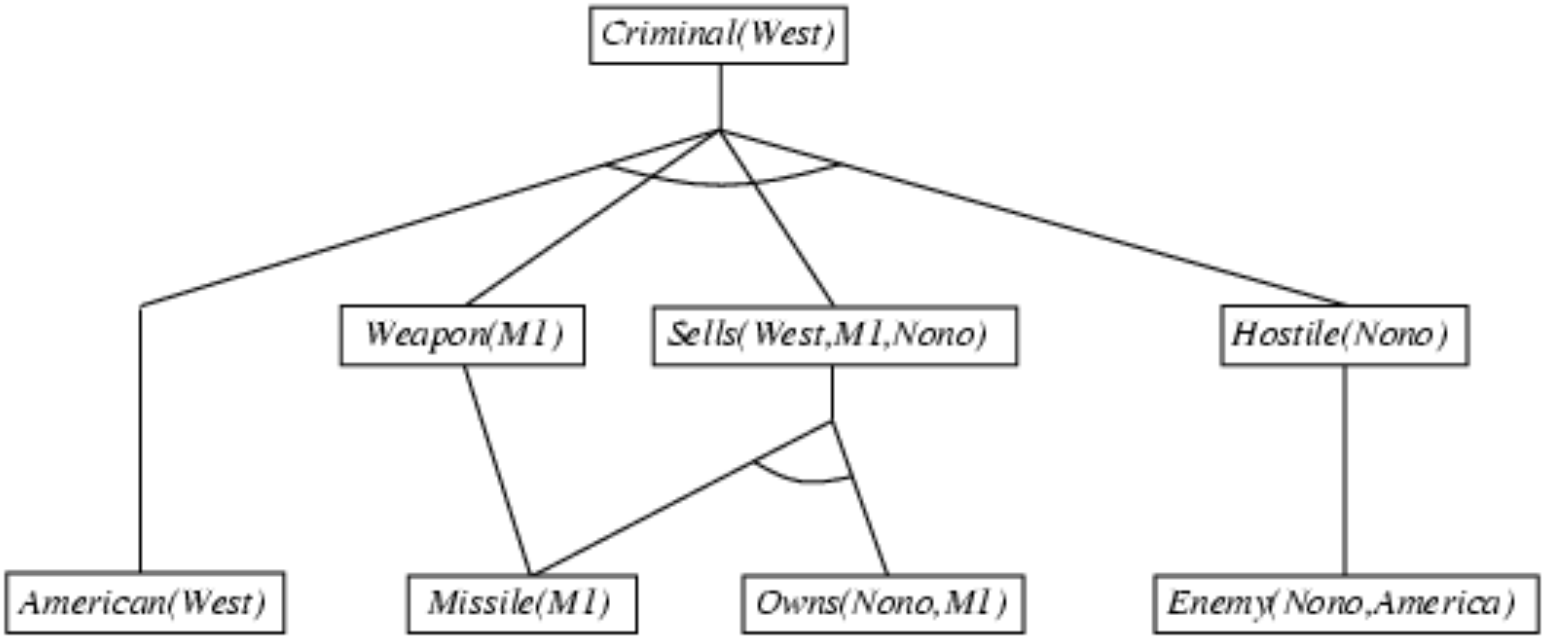
Forward chaining proof



Forward chaining proof



Forward chaining proof



Properties of forward chaining

- Sound and complete for first-order definite clauses
- Datalog = first-order definite clauses + no functions
- FC terminates for Datalog in finite number of iterations
- May not terminate in general if α is not entailed
- This is unavoidable: entailment with definite clauses is semidecidable

Unification

Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	
Knows(John,x)	Knows(y,Oliver)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,Oliver)	

- Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},Oliver)

Unification

Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,Oliver)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,Oliver)	

Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},Oliver)

Unification

Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,Oliver)	{x/Oliver,y/John}}
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,Oliver)	

Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},Oliver)

Unification

Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,Oliver)	{x/Oliver,y/John}}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x,Oliver)	

Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},Oliver)

Unification

Unify(α,β) = θ if αθ = βθ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}}
Knows(John,x)	Knows(y,Oliver)	{x/Oliver,y/John}}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x,Oliver)	{fail} se non standardizzo le var

Standardizing apart eliminates overlap of variables, e.g., Knows(z_{17},Oliver)

Unification

- *Knows(John,x)* e *Knows(y,z)* unificano con:
  θ = {y/John, x/z } or θ = {y/John, x/John, z/John}

- Il primo unificatore è più generale (informalmente, vincola di meno le variabili)

- Esiste un solo most general unifier (MGU) che è unico a meno di rinominazione di variabili
  MGU = { y/John, x/z }

Unification: Occur Check

Esempio necessità del controllo di occorrenza:

Ama(x,x) e Ama(y,Padre(y)) unificano con x/y e y/Padre(y) che non ha senso!

Occur check: *una variabile unifica con un termine composto (funzione) solo se il termine composto non coinvolge la variabile stessa.*

Una variabile universal non può essere sostituita da una funzione nella variabile stessa

The unification algorithm

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

  if θ = failure **then return** failure
  else if $x = y$ **then return** θ
  else if VARIABLE?(x) **then return** UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) **then return** UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) **and** COMPOUND?(y) **then**
    return UNIFY(x.ARGS, y.ARGS, UNIFY(x.OP, y.OP, θ))
  else if LIST?(x) **and** LIST?(y) **then**
    return UNIFY(x.REST, y.REST, UNIFY(x.FIRST, y.FIRST, θ))
  else return failure

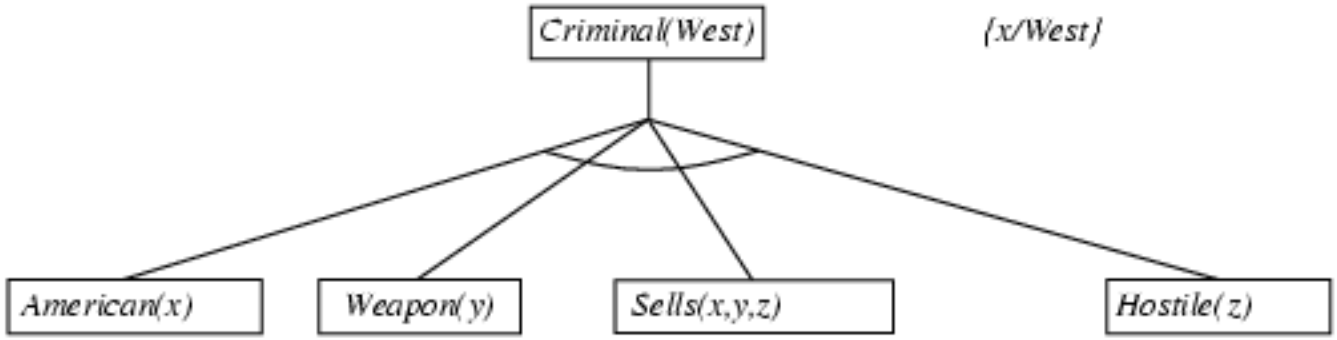
---

function UNIFY-VAR(var, x, θ) **returns** a substitution

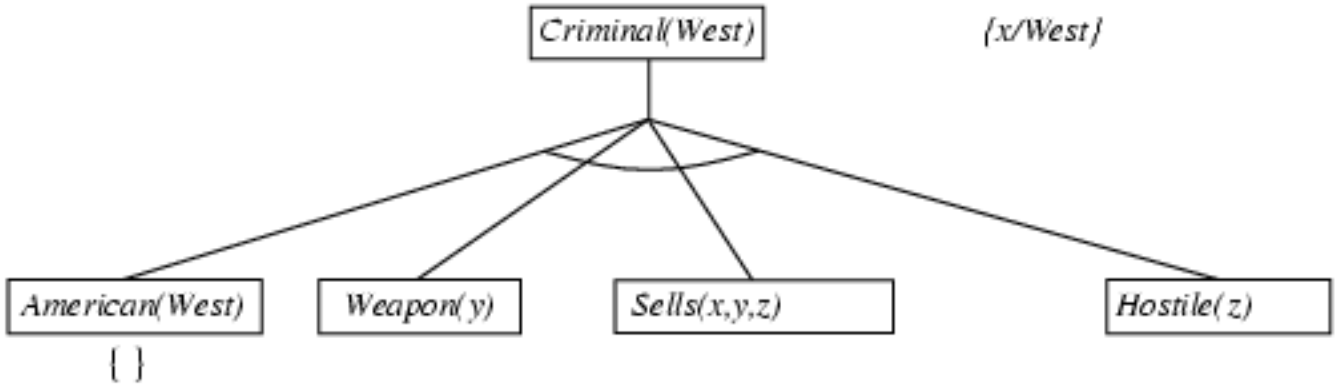
  if $\{var/val\} \in \theta$ **then return** UNIFY(val, x, θ)
  else if $\{x/val\} \in \theta$ **then return** UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) **then return** failure
  else return add $\{var/x\}$ to θ

Backward chaining example

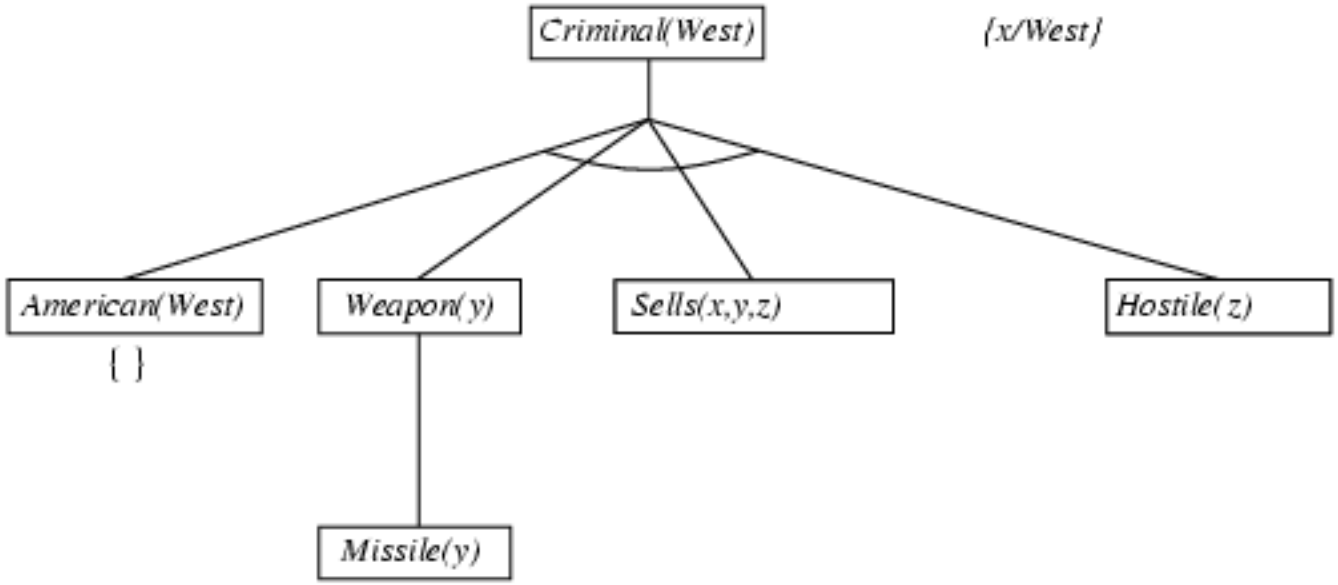
Backward chaining example



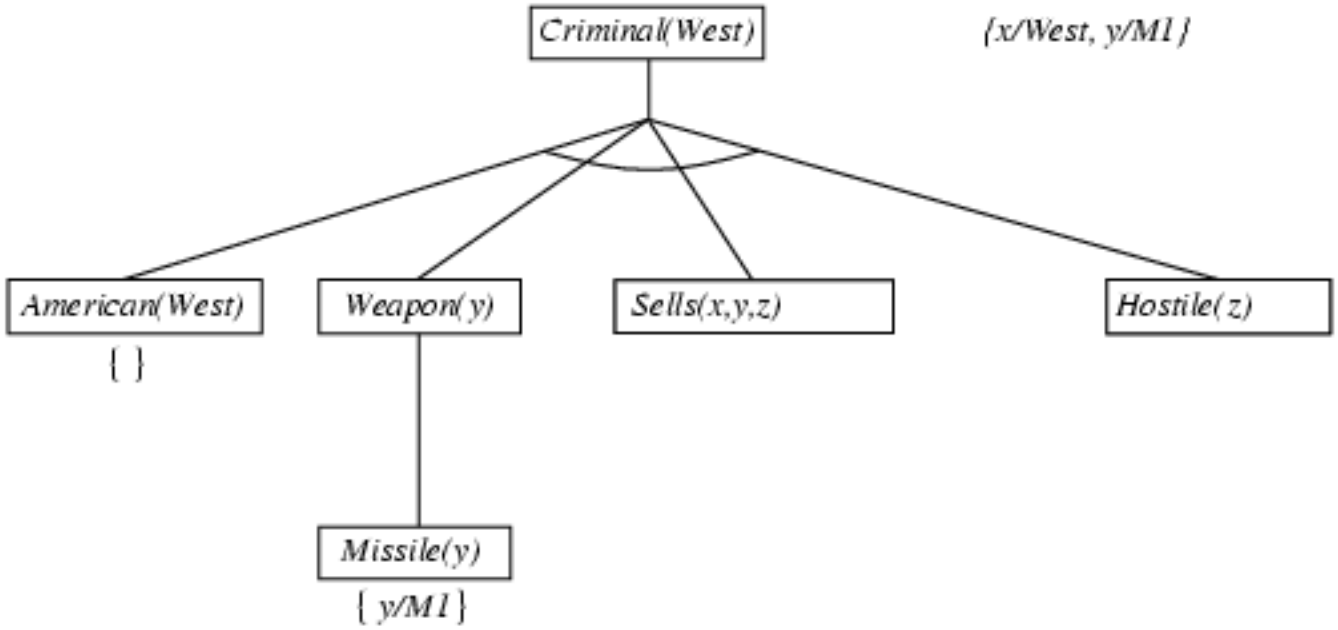
Backward chaining example



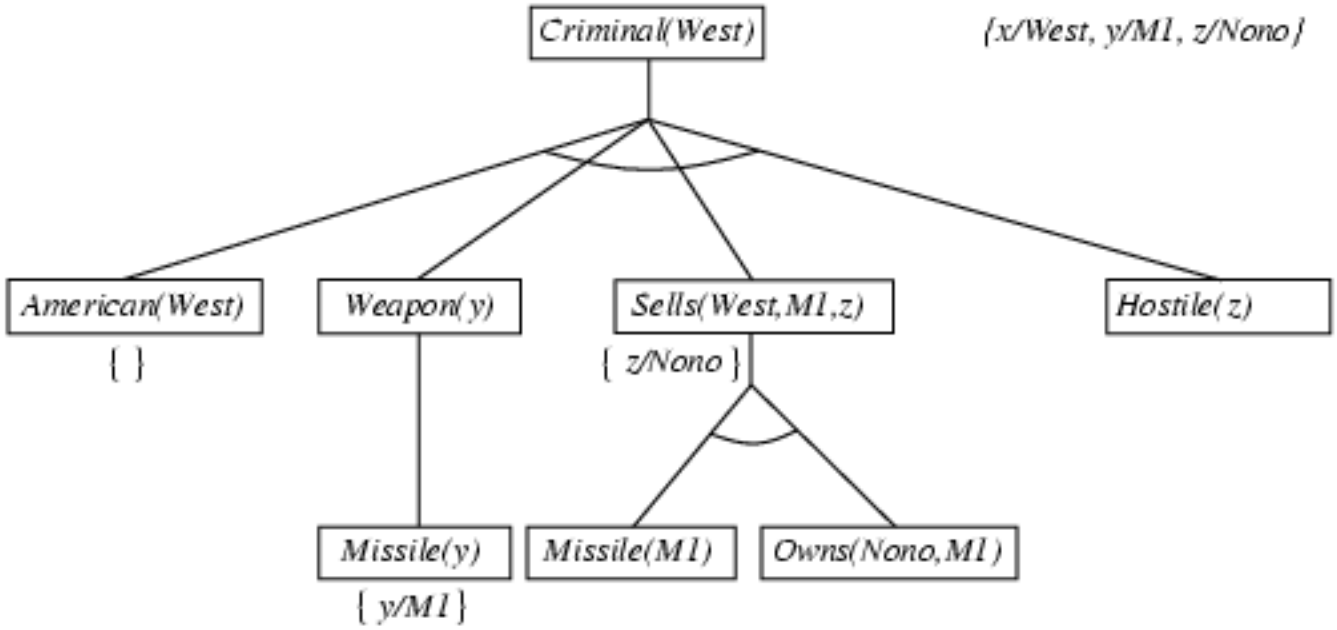
Backward chaining example



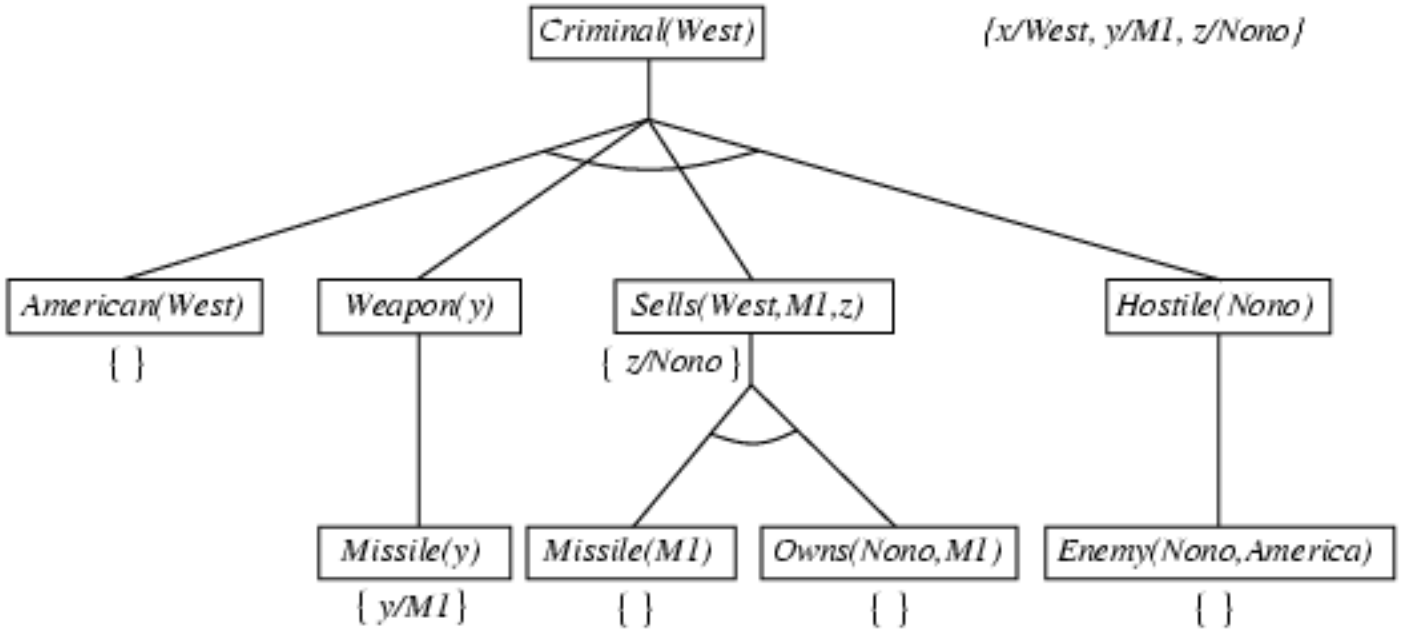
Backward chaining example



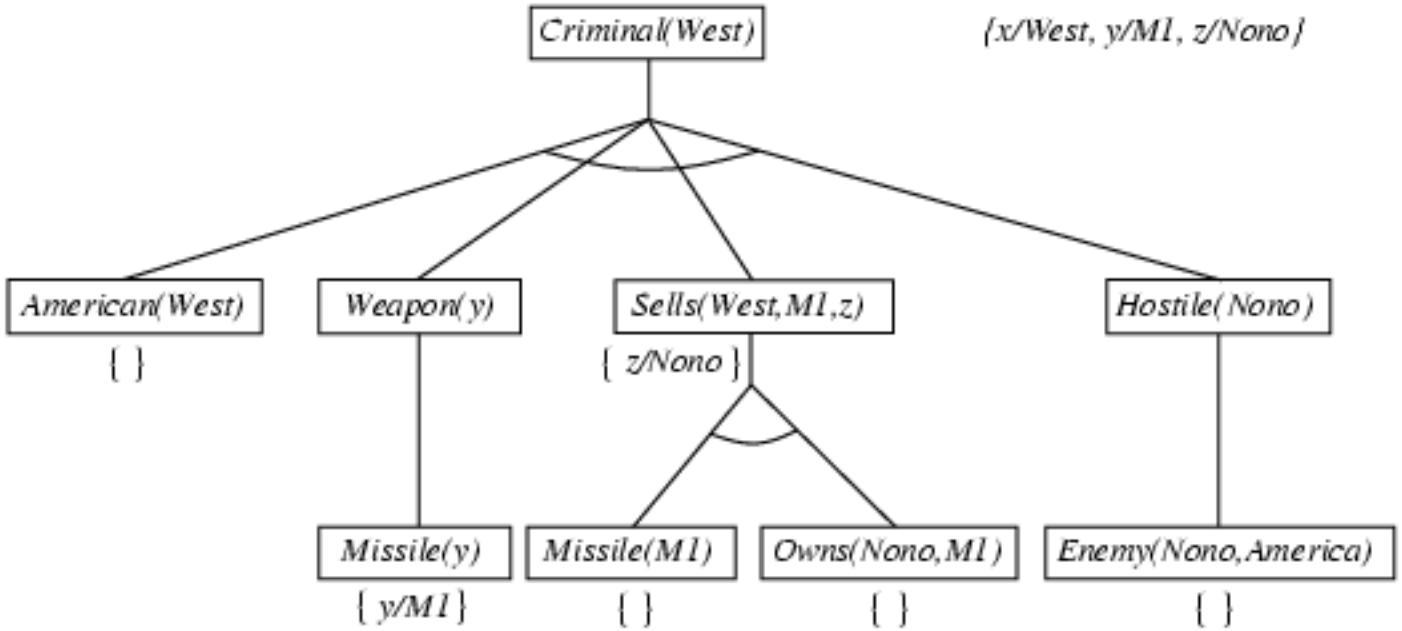
Backward chaining example



Backward chaining example



Backward chaining example



Backward chaining algorithm

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

  if $goals$ is empty **then return** $\{\theta\}$
  $q' \leftarrow$ SUBST(θ, FIRST($goals$))
  for each r **in** KB where STANDARDIZE-APART(r) $= (p_1 \wedge \ldots \wedge p_n \Rightarrow q)$
    and $\theta' \leftarrow$ UNIFY(q, q') succeeds
    $ans \leftarrow$ FOL-BC-ASK(KB, $[p_1, \ldots, p_n | \text{REST}(goals)]$, COMPOSE($\theta, \theta'$)) $\cup$ ans
  return ans

SUBST(COMPOSE(θ_1, θ_2), p) = SUBST(θ_2, SUBST(θ_1, p))

Properties of backward chaining

- Depth-first recursive proof search: space is linear in size of proof
- Incomplete due to infinite loops
  - $\Rightarrow$ fix by checking current goal against every goal on stack
- Inefficient due to repeated subgoals (both success and failure)
  - $\Rightarrow$ fix using caching of previous results (extra space)
- Widely used for logic programming

Resolution

- Full first-order version:

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

  where $\mathtt{Unify}(\ell_i, \neg m_j) = \theta$.

- The two clauses are assumed to be **standardized apart** so that they share no variables. For example,

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

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

- Apply resolution steps to CNF(KB $\wedge \neg\alpha$); complete for FOL

Teorema correttezza e completezza risoluzione

*Un insieme di clausole è **insoddisfacibile** se e solo se l'algoritmo di risoluzione (con una appropriate strategia) **termina** con successo in un **insieme finito** di passi generando la **clausula vuota***

J.A. Robinson, 1965

Conversion to CNF

- Everyone who loves all animals is loved by someone:

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

1. Eliminate biconditionals and implications

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

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

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

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

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

Conversion to CNF cont.

3. Standardize variables: each quantifier should use a different variable

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

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

$$\forall x\ [Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$$

5. Drop universal quantifiers:
$$[Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$$

6. Distribute $\lor$ over $\land$:
$$[Animal(F(x)) \lor Loves(G(x),x)] \land [\neg Loves(x,F(x)) \lor Loves(G(x),x)]$$

Example knowledge base
(all vars assumed universally quantified)

... 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 …

$Enemy(Nono,America)$

Esempio Conversione in Clusole

$$\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$$
$$\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$$
$$\neg Enemy(x, America) \vee Hostile(x)$$
$$\neg Missile(x) \vee Weapon(x)$$
$$Owns(Nono, M_1) \qquad Missile(M_1)$$
$$American(West) \qquad Enemy(Nono, America)\ .$$

Sono Clausole Definite; forma implicativa ottenuta reintroducendo il simbolo di implicazione

Un altro esempio: il gatto curioso

Everyone who loves all animals is loved by someone.
Anyone who kills an animal is loved by no one.
Jack loves all animals.
Either Jack or Curiosity killed the cat, who is named Tuna.
Did Curiosity kill the cat?

In logica del I ordine…

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

B. $\forall x \ [\exists z \ \ Animal(z) \wedge Kills(x, z)] \Rightarrow [\forall y \ \neg Loves(y, x)]$

C. $\forall x \ \ Animal(x) \Rightarrow Loves(Jack, x)$

D. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

E. $Cat(Tuna)$

F. $\forall x \ \ Cat(x) \Rightarrow Animal(x)$

¬G. $\neg Kills(Curiosity, Tuna)$

In clausole del I ordine…

A1. $Animal(F(x)) \vee Loves(G(x), x)$

A2. $\neg Loves(x, F(x)) \vee Loves(G(x), x)$

B. $\neg Loves(y, x) \vee \neg Animal(z) \vee \neg Kills(x, z)$

C. $\neg Animal(x) \vee Loves(Jack, x)$

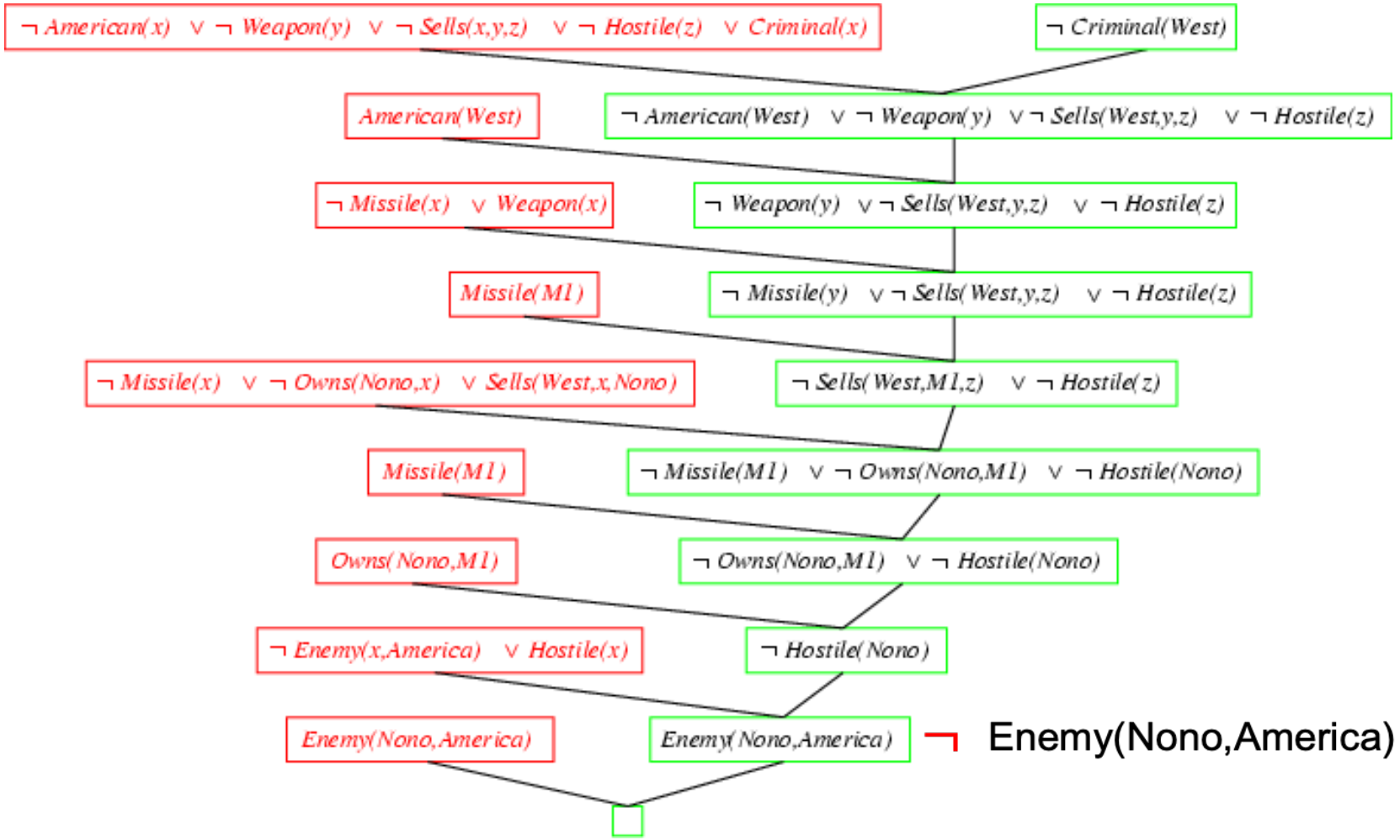
D. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

E. $Cat(Tuna)$

F. $\neg Cat(x) \vee Animal(x)$

¬G. $\neg Kills(Curiosity, Tuna)$

Suppose Curiosity did not kill Tuna. We know that either Jack or Curiosity did; thus Jack must have. Now, Tuna is a cat and cats are animals, so Tuna is an animal. Because anyone who kills an animal is loved by no one, we know that no one loves Jack. On the other hand, Jack loves all animals, so someone loves him; so we have a contradiction. Therefore, Curiosity killed the cat.



Strategie di risoluzione

- In ampiezza
- Lineare
- Unitaria
- Input
- Linear-input
- Insieme di supporto
- E altre….

Strategie di risoluzione

- In ampiezza …
- Lineare …
- Unitaria …
- Input …
- Linear-input …
- Insieme di support …

Esercizi Vari

- Esempio di non terminazione strategia linear-input:

  1) $p \vee q$ 2) $\neg p \vee q$ 3) $p \vee \neg q$ 4) $\neg p \vee \neg q$

  1+2 = q, q+3 = p, p+4=$\neg$q, $\neg$q+1=p …..

  Dove “+” indica la risoluzione

- Tabella dei casi per unificare due formule atomiche o temini complessi (next slide)
- Trasformazione in CNF di formule complesse (slide successive)

Tabella dei casi per l'unificazione

	Costante c2	**Var x2**	**Termine composto TC2**
Costante c1	Si, se c1 = C2	Si, {x2/c2}	no
Var x1	Si, {x1/c2}	Si. {x1/x2}	Si, {x1/TC2}
Termine compsto TC1	No	Si, {x2/TC1}	Si, se TC1 e TC2 hanno stesso simbolo funzione/predica e gli argomenti unificano

$\forall x (\forall y P(x,y)) \Rightarrow \neg (\forall y Q(x,y) \Rightarrow R(x,y))$

DA INTERPRETARE:

$\forall x [(\forall y P(x,y)) \Rightarrow \neg (\forall y (Q(x,y) \Rightarrow R(x,y)))]$

1. $\forall x \neg (\forall y P(x,y)) \vee \neg (\forall y \neg Q(x,y) \vee R(x,y))$
2. $\forall x (\exists y \neg P(x,y)) \vee (\exists y Q(x,y) \wedge \neg R(x,y))$
3. $\forall x (\exists y \neg P(x,y)) \vee (\exists z Q(x,z) \wedge \neg R(x,z))$
4. $\forall x \exists y \exists z (\neg P(x,y)) \vee (Q(x,z) \wedge \neg R(x,z))$
5. $\forall x (\neg P(x, F_1(x))) \vee (Q(x, F_2(x)) \wedge \neg R(x, F_2(x)))$
6. $(\neg P(x, F_1(x))) \vee (Q(x, F_2(x)) \wedge \neg R(x, F_2(x)))$
7. $(\neg P(x, F_1(x))) \vee Q(x, F_2(x)) \wedge$
   $(\neg P(x, F_1(x))) \vee \neg R(x, F_2(x))$
8. $(\neg P(x, F_1(x)) \vee Q(x, F_2(x))) \wedge$
   $(\neg P(x, F_1(x)) \vee \neg R(x, F_2(x))$
9. $(\neg P(x, F_1(x)) \vee Q(x, F_2(x))),$
   $(\neg P(x, F_1(x)) \vee \neg R(x, F_2(x)))$
10. $(\neg P(x_1, F_1(x_1)) \vee Q(x_1, F_2(x_1))),$
    $(\neg P(x_2, F_1(x_2)) \vee Q(x_2, F_2(x_2)))$

FORMA NORMALE CLAUSOLARE

11. } ESERCIZIO PER CASA
12. FORMA NORMALE IMPLICATIVA

TRASFORMARE IN FORMA NORMALE IMPLICATIVA

$(\forall x\, P(x)) \Rightarrow (\exists x\, Q(x))$

NB:
$\exists \neq \exists A$
$\forall x\,(P(x) \Rightarrow \exists x\, Q(x))$

$\neg(\forall x\, P(x)) \vee (\exists x\, Q(x))$

$(\exists x \neg P(x)) \vee (\exists x\, Q(x))$

$(\exists x \neg P(x)) \vee (\exists y\, Q(y))$

$\exists x \exists y\, (\neg P(x)) \vee (Q(y))$

$(\neg P(A)) \vee (Q(B))$

$(\neg P(A) \vee Q(B)) \approx \neg P(A) \vee Q(B)$

$(P(A) \Rightarrow Q(B)) \approx P(A) \Rightarrow Q(B)$