Unificazione

E' un procedimento preliminare che si applica alle clausole qualora in esse vi siano delle variabili prima di applicare la risoluzione.

Sostituzione- consiste in un insieme di assegnamenti di *termini* a simboli di *variabile*

$$\sigma = \{ x_1/T_1, x_2/T_2 , ...,x_n/T_n \}$$

L'applicazione della sostituzione σ ad una espressione **E** produce una nuova espressione [**E**] σ detta **istanza** di **E.**

1. [c(y,z)] { y/t, z/Neri } = c(t, Neri)
2. [c(t,Neri)] { y/t, z/Neri } = c(t, Neri)

unificano

3. [c(y,z)] { y/Bianchi, t/Bianchi, z/Neri } = c(Bianchi, Neri)
4. [c(t,Neri)] { y/Bianchi, t/Bianchi, z/Neri } = c(Bianchi, Neri)

unificano

5. [c(y,z)] { y/t, z/Bianchi} = c(t,Bianchi)
6. [c(t,Neri)] { y/t, z/Bianchi} = c(t,Neri)

non unificano

7. [c(t,Neri)] ε = c(t,Neri)

L'unificazione rende identici le istanze di due o più atomi tramite una opportuna sostituzione

Composizione di sostituzioni

Date le due sostituzioni

$\sigma_1 = \{ x_1/T_1, x_2/T_2 , ...,x_n/T_n \}$

$\sigma_2 = \{ y_1/Q_1, y_2/Q_2 , ...,y_m/Q_m \}$

si definisce la sostituzione composta $\boldsymbol{\sigma_1 \sigma_2}$ ottenuta da

$$\{x_1/[T_1]\ \sigma_2, x_2/[T_2]\ \sigma_2,....,x_n/[T_n]\sigma_2, y_1/Q_1,....,y_m/Q_m \}$$

togliendo 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 $y_j \in \{ x_1, x_2,.....,x_n\}$

Ex:

$\sigma_1 = \{ x/f(z), w/r, s/c\}$ $\qquad$ $\sigma_3 = \sigma_1\ \sigma_2 =$?

$\sigma_2 = \{ y/x, r/w, z/b\}$

$x/[f(z)]\ \sigma_2 = x/[f(z)]\ \{ y/x, r/w, z/b\} = x/[f(b)]$

$w/[r]\ \sigma_2 = w/[r]\ \{ y/x, r/w, z/b\} = w/w$ $\Leftarrow$ si esclude

$s/[c]\ \sigma_2 = s/[c]\ \{ y/x, r/w, z/b\} = s/c$

Si ha pertanto:

$\sigma_3 = \sigma_1\ \sigma_2 = x/f(b),\ s/c, y/x, r/w, z/b\}$

Teorema di unificazione

Esiste un algoritmo (di unificazione) che, dati due atomi, produce la loro sostituzione unificatrice più generale *MGU* (Most General Unifier), se essi sono unificabili, altrimenti termina in tempo finito restituendo la non esistenza di tale sostituzione.

Una sostituzione θ è più generale di una sostituzione σ se esiste una sostituzione λ tale che $\sigma = \theta\,\lambda$

Ex: date le due clausole C(y,z) C(t, Neri) e le due sostituzioni
θ = {y/t, z/Neri} σ= {y/Bianchi, t/Bianchi, z/Neri} si ha:

[C(y,z)]θ = [C(y,z)] {y/t, z/Neri} = C(t,Neri)
[C(t,Neri)] θ = [C(t,Neri)] {y/t, z/Neri}= C(t,Neri)

[C(y,z)]σ = [C(y,z)] {y/Bianchi, t/Bianchi, z/Neri} = C(Bianchi,Neri)
[C(t,Neri)] σ = [C(t,Neri)] {y/Bianchi, t/Bianchi, z/Neri} = C(Bianchi,Neri)

Siccome σ = {y/Bianchi, t/Bianchi, z/Neri} è esprimibile come {y/t, z/Neri} {t/Bianchi} vuol dire che θ= {y/t, z/Neri} è più generale di σ.

Regole di unificazione

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 lo stesso funtore e arietà e gli argomenti UNIFICANO

Risoluzione nel caso generale

Date le clausole

$C_1 = A_1 \vee, \ldots, \vee A_n$

$C_2 = B_1 \vee, \ldots, \vee B_m$

Dove nei letterali A_i (i=1,...n) e B_j (j=1,...m) possono comparire variabili. Se si ha

$A_i = p(t_i \ldots\ldots t_k)$ e $B_j = \neg\, p\,(s_i \ldots\ldots s_k)$

oppure

$A_i = \neg\, p(t_i \ldots\ldots t_k)$ e $B_j = p\,(s_i \ldots\ldots s_k)$

e si verifica che i due atomi unificano con una qualche sostituzione MGU θ, cioé

$[p(t_i \ldots\ldots t_k)]\,\theta = [p(s_i \ldots\ldots s_k)]\,\theta$

allora si può scrivere una terza clausola C_3 detta risolvente data da:

$C_3 = [A_1 \vee, \ldots, \vee A_{i-1},\ A_{i+1} \vee, \ldots, \vee A_n \vee B_1 \vee, \ldots, \vee B_{j-1},\ B_{j+1} \vee, \ldots, \vee B_m\,]\,\theta$

Savana

1. Gli animali superano nella corsa gli animali che mangiano

$$\forall x\ \forall y\ \mathbf{Mangia}(x,y) \rightarrow \mathbf{Supera}(x,y)$$

2. I Carnivori mangiano ogni animale non carnivoro

$$\forall x\ \forall y\ (\mathbf{Carnivoro}(x) \wedge \neg\ \mathbf{Carnivoro}(y)) \rightarrow \mathbf{Mangia}(x,y)$$

3. La proprietà di superare è transitiva

$$\forall x\ \forall y\ \forall z\ (\mathbf{Supera}(x,y) \wedge \mathbf{Supera}(y,z)) \rightarrow \mathbf{Supera}(x,z)$$

4. I Leoni mangiano le zebre

Mangia(Leone, Zebra)

5. Le zebre superano nella corsa i cani

Supera (Zebra, cane)

6. I Leoni sono carnivori

Carnivoro(Leone)

Quali sono gli animali che possono essere superati dai leoni ?

$$\exists x\ \mathbf{Supera}(\mathbf{Leone},x)$$

Savana

1. ¬ Mangia(x,y) ∨ Supera(x,y)

2. ¬ Carnivoro(x) ∨ Carnivoro(y) ∨ Mangia(x,y)

3. ¬ Supera(x,y) ∨ ¬ Supera(y,z) ∨ Supera(x,z)

4. Mangia(Leone, Zebra)

5. Supera (Zebra, cane)

6. Carnivoro(Leone)

7. ¬ Supera(Leone,x)

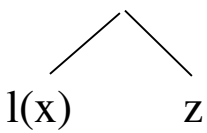
Da 1 e 7 si ricava	**8. ¬ Mangia(Leone, x)**	x/Leone	y/x
Da 8 e 4 si ricava	**Nil**	x/Zebra	
In alternativa da 1 e 4 si ricava	**9. Supera(Leone, Zebra)**	x/Leone	y/Zebra
da 3 e 9 si ricava	**10. ¬ Supera(Zebra,z) ∨ Supera(Leone,z)**		
da 10 e 5 si ricava	**11. Supera(Leone, cane)**	z/Cane	
da 11 e 7 si ricava	**Nil**	x/Cane	

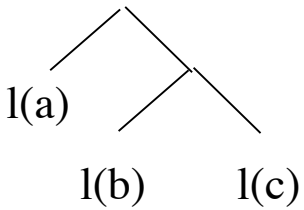
Ulteriori Esempi

1. P(x,f(a)) Unificano in virtù della sostituzione
P(b,y) $\theta = \{x/b,\ y/f(a)\}$

2. T(l(a), T(l(b), l(c))) Unificano in virtù della sostituzione
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 strutture ad albero





L'unificazione consente di accedere all'unica foglia del sottoalbero di sinistra tramite la variabile x e al sottoalbero destro tramite la variabile z

3. T(T(T(X, Y), f(c)),Y)
T(T(T(Y, Y), f(c)), f(X))

Non unificano poiché X non può mai essere identico a f(X)

Strategie di Ricerca

E' mandatorio per motivi di efficienza, ridurre il numero di risolventi generati ad ogni passo.

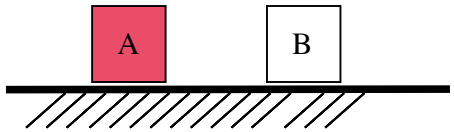
Strategia
E' un criterio di scelta delle clausole da cui derivare il risolvente

Una Strategia si dice *completa* se trova sempre la contraddizione logica quando è applicata a un insieme di clausole insoddisfacibile.

Spesse volte per motivi di efficienza si fa uso di strategie incomplete

Breadth-First
Al passo i ($i \geq 0$) genera tutti i possibili risolventi a livello i+1 utilizzando come "parent" una clausola del livello i e una clausola di un livello j ($j \leq i$). Questa strategia è *completa*

Mondo a blocchi



H= H^c = {On (A, Tavola),
ON (B, Tavola),
Colore(A, Rosso) ∨Colore (B,Rosso)}

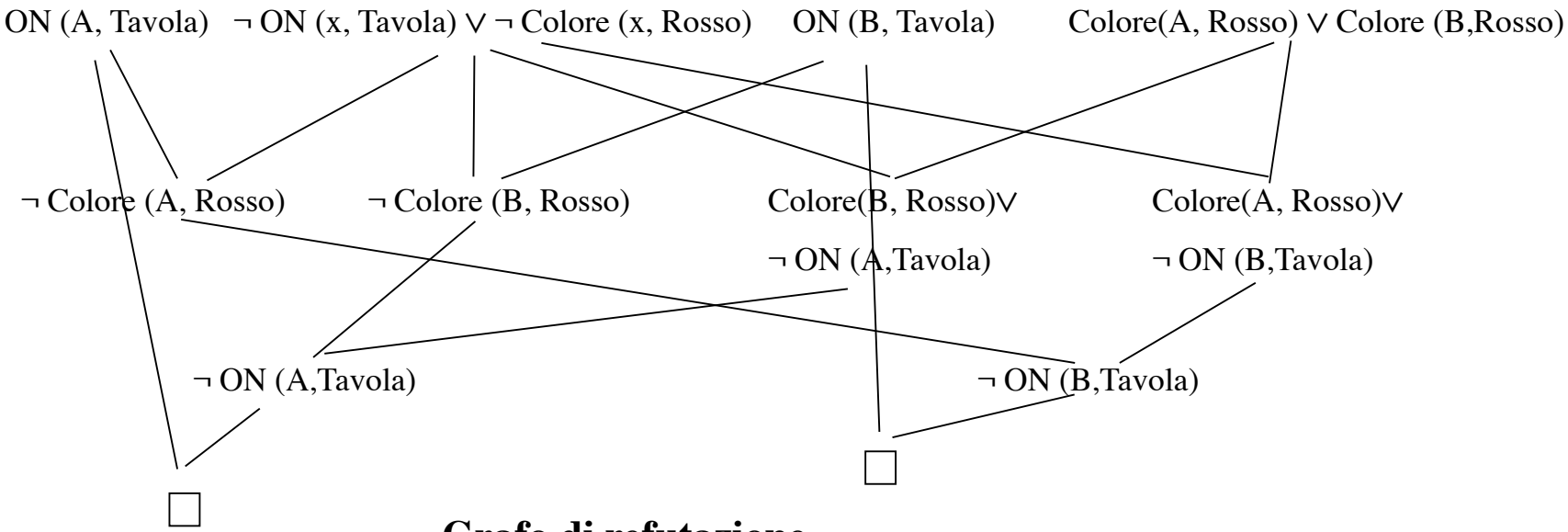
Tenuto conto che

$\neg \exists x A \equiv \forall x \neg A$

F= {∃ x On (x, Tavola) ∧Colore (x, Rosso))}

e per le leggi di De Morgan

F^c = {¬ On (x, Tavola) ∨ ¬ Colore (x, Rosso))}



Grafo di refutazione

Altre Strategie

Set of Support

E' una variante della ricerca in Breadth-First. Si sceglie almeno una clausola "parent" tra quelle che derivano dalla $\mathbf{F}^c$. L'intuizione che sta dietro questa strategia è che è inutile produrre un risolvente che proviene da $\mathbf{H}^c$ [che per definizione è consistente]. La strategia è *completa*.

Lineare

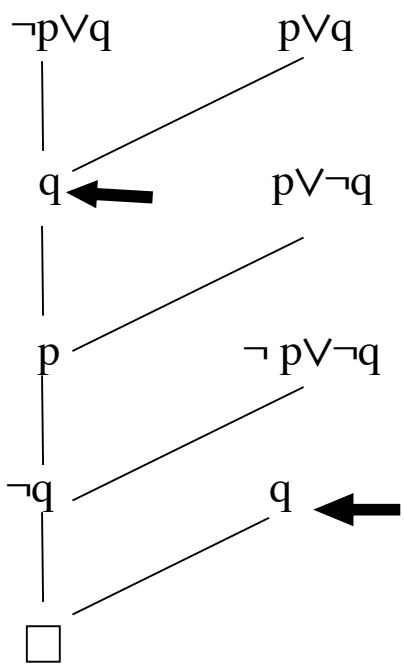
Viene scelta almeno una clausola "parent" che proviene dall'insieme base C_0 oppure tra i risolventi generati in precedenza. Mentre la seconda clausola "parent" è sempre il risolvente del passo precedente. Il grafo di refutazione diventa un **albero di refutazione.**
La strategia è *completa*.

Linear-Input

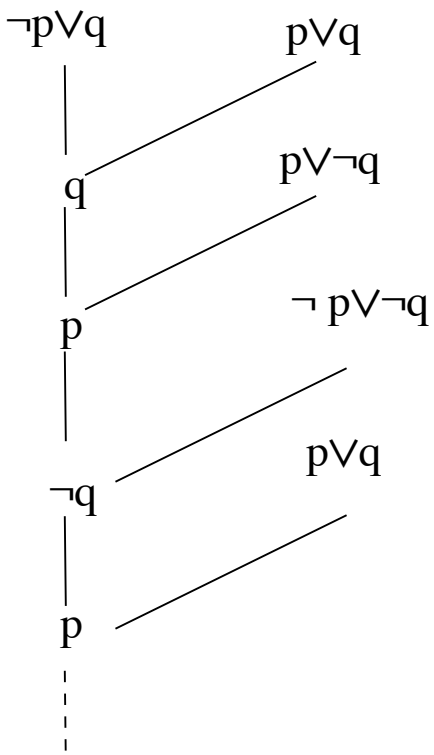
Viene scelta una clausola "parent" dall'insieme base C_0 . Mentre la seconda clausola "parent" è sempre il risolvente del passo precedente.
La strategia *non è completa*.

Alberi di refutazione

C_0 = $\{p \vee q,\ \neg p \vee q,\ p \vee \neg q,\ \neg p \vee \neg q\}$



Albero di refutazione lineare



Albero di refutazione linear-input

Ancora sulle Clausole

Clausola definita - E' una clausola che contiene un solo letterale positivo

$A_1 \vee \neg B_1 \vee \neg B_2 \ldots \vee \neg B_m$ ovvero $B_1 \wedge B_2 \ldots \wedge B_m \rightarrow A$

Clausola Negativa o goal - E' una clausola che contiene solo letterali negativi

$B_1 \wedge B_2 \ldots \wedge B_m \rightarrow$

Clausole di Horn -

$B_1 \wedge B_2 \ldots \wedge B_m \rightarrow A$ (Regola)

$\rightarrow A$ (Fatto o asserzione)

$B_1 \wedge B_2 \ldots \wedge B_m \rightarrow$ (Goal)

$\rightarrow$ (Clausola vuota)

Clausole di Horn scritte in altro modo *[la virgola sottointende l'operatore $\wedge$]*

$A \leftarrow B_1, B, \ldots, B_m$ (Regola)

$A \leftarrow$ (Fatto o asserzione)

$\leftarrow B_1, B, \ldots, B_m$ (Goal)

$\leftarrow$ (Clausola vuota)

Si osservi come $\leftarrow B_1, B, \ldots, B_m$ equivale alla formula

$\neg (B_1 \wedge B_2 \ldots \wedge B_m)$

Cenni Sul PROLOG

femmina (mary).
padre (john, mary).
ama((john, vino).
da (john, mary, libro).
ama (john, mary).
ama (mary, libro).

Database di fatti (atomi positivi)

Ad un Database è possibile formulare Domande

? - ama(john, soldi).

No

? - ama(mary, john).

No

? - ama(mary, libro).

Yes

la query viene considerata una clausola negativa e quindi candidata per essere unificata con qualche fatto presente nel database

Congiunzioni

ama (mary, cibo).
ama (mary, vino).
ama (john, vino).
ama (john, mary).

? - ama(john, mary), ama (mary, john).

No

? - ama (mary, X), ama (john, X).

C'è qualche oggetto che è amato congiuntamente sia da John che da Mary?

? - ama (mary, X), ama(john, X).

X/cibo

X/vino

? - ama (mary, cibo), ama (john, cibo).

? - ama (mary, vino), ama (john, vino).

? - ama (john, cibo).

? - ama (john, vino).

Fallimento

Successo

Regole

```
maschio (alberto).
maschio (edoardo).
femmina (alice).
femmina (vittoria).
genitori (edoardo, vittoria , alberto).
genitori (alice, vittoria , alberto).
sorella_di (X, Y) : -
        femmina( X),
        genitori (X, M, P),
        genitori (Y, M,P).
```

testa della regola

corpo della regola

```
?- sorella_di (alice, edoardo).

 yes
```

primo passo (sostituzione di X e Y)

```
sorella_di (alice, edoardo) : -
        femmina( alice),
        genitori (alice, M, P),
        genitori (edoardo, M,P).
```

secondo passo (sostituzione di M e P)

```
sorella_di (alice, edoardo) : -
        femmina( alice),
        genitori (alice, vittoria, alberto),
        genitori (edoardo, vittoria, alberto).
```

Regole

```
maschio (alberto).
maschio (edoardo).
femmina (alice).
femmina (vittoria).
genitori (edoardo, vittoria , alberto).
genitori (alice, vittoria , alberto).
sorella_di (X, Y) : -
        femmina( X),
        genitori (X, M, P),
        genitori (Y, M,P).

?- sorella_di (alice, X).

X= edoardo
```

primo passo

```
sorella_di (alice, Y) : -
        femmina (alice),
        genitori (alice, M, P),
        genitori (Y, M,P).
```

X/alice

secondo passo

```
sorella_di (alice, Y) : -
        femmina( alice),
        genitori (alice, vittoria, alberto),
        genitori (Y, vittoria, alberto).
```

M/vittoria
P/alberto

terzo passo

```
sorella_di (alice, edoardo) : -
        femmina( alice),
        genitori (alice, vittoria, alberto),
        genitori (edoardo, vittoria, alberto).
```

Y/edoardo