LOGICA

Lezione 11-12: Metodo di risoluzione

Laura Nenzi

DIA – Università degli Studi di Trieste

Contenuti della lezione

- Teoria di Herbrand
  - Algoritmo di Herbrand
- Risoluzione nella logica proposizionale
  - Algoritmo di risoluzione
  - Algoritmo di refutazione
- Unificazione
- Risoluzione nella logica dei predicati

Metodi di risoluzione

- Metodi per provare la validità di una formula:
  - teorema di Herbrand (semantico)
  - metodo di risoluzione (sintattico)

- Sono metodi per **refutazione**:
  - Per dimostrare che P è valida si dimostra che la negazione di P è insoddisfacibile.

Teoria di Herbrand

- Logica proposizionale:
  - il numero di interpretazioni è finito.
- Logica dei predicati:
  - il numero di possibili interpretazioni è infinito, infatti possiamo avere infiniti domini!
- Soluzione: *dominio canonico* H (*Universo di Herbrand*)
  - P è insoddisfacibile se è falsa in tutte le interpretazioni di H.

Notazione

- Useremo solo formule chiuse e in forma di Skolem.
- Trasformazione di una fbf P:
  - Chiusura esistenziale, se FV(P)=$\{x_1, \ \ldots, x_n\}$ è: $\exists x_1, \ldots, x_n P$.
  - si trasforma $\exists x_1, \ldots, x_n P$ in forma di Skolem.

- Sia P una fbf. È possibile trasformare P in una formula P' chiusa e in forma di Skolem tale che:
  - P è soddisfacibile sse P' è soddisfacibile.

Universo di Herbrand

- L'universo di Herbrand, *H(P)*, di una formula chiusa del primo ordine P è l'insieme dei termini contenente:
  - tutte le costanti che compaiono in P. Se P non ha costanti allora scelgo un simbolo di costante a, e pongo a∈H(P)
  - tutti i termini costruiti applicando i simboli di funzione che occorrono in P a tutti i termini di H(P)

Esempio

- Sia P = ∀xA(x,c) ∨ B(g(c,f(c)),c,a), P contiene:
  - i simboli di costante a e c,
  - i simboli di funzione f unaria e g binaria
  - i simboli di predicato A binario e B ternario
- Universo di Herbrand H(P):
  - $\{a, c, f(a), f(c), g(a,a), g(c,c), g(a,c), g(c,a), f(f(a)), f(g(a,a)), g(a,f(a)), \ldots.\}$

Formule ground

- **Formule ground**: formule che non contengono variabili.
- **Sostituzione ground**: sostituzione che elimina le variabili.
- **Istanze ground**: formula ottenuta mediante una sostituzione ground delle formula di partenza.

Base di Herbrand

- La **base di Herbrand** di una fbf P, B(P), è l'insieme di tutte le formule atomiche ground che si possono costruire:
  - utilizzando i simboli di predicato che compaiono in P
  - con argomento gli elementi di H(P).

Esempio

- Sia $P = \forall x A(x, c) \vee B(g(c, f(c)), c, a)$
- Universo di Herbrand H(P):
  - $\{a, c, f(a), f(c), g(a, a), g(c, c), g(a, c), g(c, a), f(f(a)), f(g(a, a)), g(a, f(a)), \ldots\}$
- B(P):
  - {A(a,a), A(a, f(a)), A(g(a,a), c), B(a,a,a),...}

Interpretazione di Herbrand

- Sia P una fbf chiusa, $H = (D_H , I_H)$ è un'**interpretazione di Herbrand** per P sse valgono le seguenti condizioni:
- $D_H = H(P)$
- $I_H(c) = c$, c simbolo di costante
- $I_H(f(t_1, \dots, t_n)) = f(t_1, \dots, t_n)$, f simbolo di funzione di arietà n.
- *Osservazione*: non ci sono restrizioni per i simboli di predicato.

Esempio

- Sia $P = \forall x A(x, c) \vee B(g(c, f(c)), c, a)$
- Un'interpretazione di Herbrand di P:
  - $D_H = H(P) = \{a, c, f(a), f(c), g(a, a), g(c, c), g(a, c), g(c, a), f(f(a)), f(g(a, a)), g(a, f(a)), \ldots.\}$
  - $I_H(a) = a, I_H(c) = c$
  - $I_H\big(f(a)\big) = f(a), I_H\big(g(a, c)\big) = g(a, c), \ldots$
  - $I_H(A(x, y)) = \{(x, y) \mid x, y \in D_H \; e \; x = y\}$
  - $I_H(B(x, y, z)) = \{(x, y, z) \mid x, y, z \in D_H \; e \; x = g(y, z)\}$

Modello di Herbrand

- Data una fbf P, un **modello di Herbrand** di P è un'interpretazione di Herbrand che la soddisfa.

Esempio

- Sia $P = \forall x A(x, c) \vee B(g(c, f(c)), c, a)$
- Un'interpretazione di Herbrand di P:
  - $D_H = H(P) = \{a, c, f(a), f(c), g(a, a), g(c, c), g(a, c), g(c, a), f(f(a)), f(g(a, a)), g(a, f(a)), \ldots.\}$
  - $I_H(a) = a, I_H(c) = c$
  - $I_H\big(f(a)\big) = f(a), I_H\big(g(a, c)\big) = g(a, c), \ldots$
  - $I_H(A(x, y)) = \{(x, y) \mid x, y \in D_H \; e \; x = y\}$
  - $I_H(B(x, y, z)) = \{(x, y, z) \mid x, y, z \in D_H \; e \; x = g(y, z)\}$
- H **non** è un modello per P, infatti:
  - $A(x, c)$ è falso per $x = a$
  - $B(g(c, f(c)), c, a)$ è falso poiché $g\big(c, f(c)\big) \neq g(c, a)$

Esempio

- Sia $P = \forall x A(x, c) \vee B(g(c, f(c)), c, a)$
- L'interpretazione di Herbrand H' di P:
  - $D_{H'} = H'(P) = \{a, c, f(a), f(c), g(a, a), g(c, c), g(a, c), g(c, a), f(f(a)), f(g(a, a)), g(a, f(a)), \ldots.\}$
  - $I_{H'}(a) = a, I_{H'}(c) = c$
  - $I_{H'}\big(f(a)\big) = f(a), I_H\big(g(a, c)\big) = g(a, c), \ldots$
  - $\boldsymbol{I_{H'}(A(x, y)) = \{(x, y) \mid x, y \in D_{H'} \; e \; y = c\}}$
  - $I_{H'}(B(x, y, z)) = \{(x, y, z) \mid x, y, z \in D_{H'} \; e \; x = g(y, z)\}$
- H' è un modello per P, infatti:
  - $A(x, c)$ è vero per tutti gli $x \in D_{H'}$
  - $B(g(c, f(c)), c, a)$ è falso poiché $g\big(c, f(c)\big) \neq g(c, a)$

Teoremi

- Sia P una fbf chiusa e in forma di Skolem:
  - P è soddisfacibile sse ha un modello di Herbrand

Equivale a dire:

- Sia P una fbf chiusa e in forma di Skolem:
  - P è insoddisfacibile sse non ha un modello di Herbrand

Esempio

- È necessario che la fbf sia in forma di Skolem.
- Sia $P = R(c) \wedge \exists x \neg R(x)$
- P è soddisfacibile,infatti ha il seguente modello $M = (D, I)$:
  - $D = \{a, c\}$
  - $I(c) = c, I(a) = a$
  - $I(R(x)) = \{x | x = c\}$
- Quindi in $M, R(c)$ è vero e $R(a)$ è falso
- Ma $P = R(c) \wedge \exists x \neg R(x)$ non ha un modello di Herbrand!

Esempio cont.

- Tutte le possibili interpretazioni di Herbrand sono formate da:
  - $D_H = H(P) = \{c\}$
  - $I_H(c) = c$
- ho due possibili interpretazioni di R(x):
  - $I_H(R(c)) = 1\ (vero\ R(c)\ ma\ falso\ \exists x \neg R(x))$
  - $I_H(R(c)) = 0\ (vero\ \exists x \neg R(x)\ ma\ falso\ R(c))$
- Entrambi i casi non sono modelli per P.
- Ciò è possibile perché P **non** è in forma di Skolem chiusa!
- Oss: $P = R(c) \wedge \exists x \neg R(x) \equiv \exists x(\neg R(x) \wedge R(c))$
- $P' = \neg R(a) \wedge R(c)$

Espansione di Herbrand

- Sia $P = \forall x_1, \ldots, x_n P'$ una fbf chiusa in forma di Skolem. **L'espansione di Herbrand** di P, E(P), è l'insieme delle formule ground ottenute sostituendo i termini dell'universo di Herbrand di P alle variabili di P' in tutti i possibili modi.
- Ovvero:
  - $E(P) = \{P'[t_1, \ldots, t_n / x_1, \ldots, x_n] \mid t_1, \ldots, t_n \in H(P)\}$.

Esempio

- $Sia\ P\ =\ \forall x \forall y((A(a)\ \vee\ A(f(x)))\wedge \neg A(y))$
- $H(P)\ =\ \{a, f(a), f(f(a)), \dots\}$
- $E(P) = \{$
  - $x = a, y = a$: $(A(a)\ \vee\ A(f(a)))\wedge \neg A(a)$
  - $x = a, y = f(a)$: $(A(a)\ \vee\ A(f(a)))\wedge \neg A(f(a))$
  - $x = f(a), y = a$: $(A(a)\ \vee\ A(f(f(a))))\wedge \neg A(a)$
  - $x = f(a), y = f(a)$: $\left(A(a)\vee\ A\left(f\left(f(a)\right)\right)\right)\wedge \neg A\left(f(a)\right)$
  - $x = a, y = f\left(f(a)\right)$: $\left(A(a)\vee\ A\left(f(a)\right)\right)\wedge \neg A\left(f\left(f(a)\right)\right)$
  - $\dots\ \}$

Teoremi

- Una formula chiusa P in forma di Skolem è soddisfacibile sse E(P) è soddisfacibile.

- **Teorema di Herbrand:**
  - Una formula chiusa P in forma di Skolem è insoddisfacibile sse esiste un sottoinsieme finito di E(P) che è insoddisfacibile.

Algoritmo di Herbrand

- Sia P una fbf chiusa e in forma di Skolem e sia $E(P) = \{P_1, P_2, \dots\}$ l'espansione di Herbrand di P.
- Algoritmo di Herbrand per P:
  - n=0
  - repeat
    - n++

    until $(P_1 \wedge P_2 \wedge \cdots \wedge P_n)$ è insoddisfacibile
  - output "P è insoddisfacibile"
- L'algoritmo permette di semidecidere l'insoddisfacibilità di una formula
- Se la formula non è insoddisfacibile ripete all'infinito il ciclo

Esempio

- $E(P) = \{$
  - $x = a, y = a$: $\big(A(a) \vee A\big(f(a)\big)\big) \wedge \neg A(a)$
  - $x = a, y = f(a)$: $\big(A(a) \vee A\big(f(a)\big)\big) \wedge \neg A\big(f(a)\big)$
  - $\ldots\}$
- $n = 1$: $\big(A(a) \vee A\big(f(a)\big)\big) \wedge \neg A(a)$ è soddisfacibile

$A(a)$	$A(f(a))$	$\big(A(a) \vee A\big(f(a)\big)\big) \wedge \neg A(a)$
0	0	0
0	1	1
1	0	0
1	1	0

Esempio

- $n = 2$
  - $\big(A(a) \vee A\big(f(a)\big)\big) \wedge \neg A(a) \wedge \big(A(a) \vee A\big(f(a)\big)\big) \wedge \neg A(f(a))$ è insoddisfacibile
- Per il teorema di Herbrand $\forall xy((A(a) \vee A(f(x))) \wedge \neg A(y))$ è insoddisfacibile

$A(a)$	$A(f(a))$	$\big(A(a) \vee A\big(f(a)\big)\big) \wedge \neg A(a) \wedge \big(A(a) \vee A\big(f(a)\big)\big) \wedge \neg A(f(a))$
0	0	0
0	1	0
1	0	0
1	1	0

Riassumendo 1:

- Il metodo di Herbrand può essere usato per dimostrare che **P è insoddisfacibile**:
  1. Calcolo la forma chiusa di Skolem, P^S, di P
  2. Calcolo H(P^S)
  3. Calcolo E(P^S)
  4. Applico l'algoritmo di Herbrand a P^S

Riassumendo 2:

- Il metodo di Herbrand può essere usato per dimostrare che **P è valida**:
  - P è valida sse $\neg \mathrm{P}$ è insoddisfacibile
  1. Calcolo $\neg \mathrm{P}$
  2. Calcolo la forma chiusa di Skolem, $(\neg \mathrm{P})^S$, di $\neg \mathrm{P}$
  3. Calcolo $H((\neg \mathrm{P})^S)$
  4. Calcolo $E((\neg \mathrm{P})^S)$
  5. Applico l'algoritmo di Herbrand a $(\neg \mathrm{P})^S$

Riassumendo 3:

- Il metodo di Herbrand può essere usato per dimostrare che **P è conseguenza logica di Γ**:
  - **$\Gamma \models P$ sse $\Gamma \cup \{\neg P\}$ è insoddisfacibile**

1. Calcolo $\Gamma \cup \{\neg P\}$
2. Sia $\Gamma = \{P_1\ P_2, \dots, P_n\}$ allora calcolo $Q = \neg P \wedge P_1 \wedge P_2 \wedge \cdots \wedge P_n$
3. Calcolo la forma chiusa di Skolem, Q^S, di Q
4. Calcolo $H(Q^S)$
5. Calcolo $E(Q^S)$
6. Applico l'algoritmo di Herbrand a Q^S

Risoluzione nella logica proposizionale

- L’algoritmo di Herbrand è molto inefficiente.
- Esaminiamo ora una procedura molto più efficiente che è alla base della programmazione logica.
- Anche in questo caso usiamo la refutazione.
- Prima vediamo la versione **proposizionale**

Clausole

- Una **clausola** è una disgiunzione finita di zero o più letterali.
- Una clausola che non contiene letterali è detta **clausola vuota**, □.
- Una clausola è **soddisfatta** se ha ameno un letterale vero.
- La clausola vuota è sempre falsa.
- Una clausola è una **tautologia** se contiene un letterale e il suo negato.

Rappresentazione

- La clausole sono rappresentate come insiemi.
- Esempio:
  - la clausola $A \vee B \vee \neg C \vee B \vee \neg D$
  - viene rappresentata con $\{A, B, \neg C, \neg D\}$

Trasformazione

- Ogni fbf **proposizionale** P può essere rappresentata in forma a clausole.
- Trasformazione di P in forma a clausole:
  - P è trasformata in FNC
  - ogni fattore (clausola) è rappresentato con un insieme.
- Esempio: $P = (A \vee B) \wedge (A \vee \neg C \vee \neg D) \wedge (\neg B \vee D)$
  - diventa $P^c = \{\{A, B\}, \{A, \neg C, \neg D\}, \{\neg B, D\}\}$
- □=falso,
- l'insieme vuoto di clausole {} = vero

Risolvente

- Siano C_1 e C_2 e R clausole. R è una **risolvente** di C_1 e C_2 se esiste un letterale tale che:
  - $l \in C_1$
  - $\neg l \in C_2$
  - $R = (C_1 - \{l\}) \cup (C_2 - \{\neg l\})$
- Esempio
  - $C_1 = \{A, \neg B\}$ e $C_2 = \{B, C\}$
  - $R = \{A, C\}$
- Esempio
  - $C_1 = \{A\}$ e $C_2 = \{\neg A\}$
  - $R = \square$

Derivazione

- Sia S un insieme di clausole, **una derivazione** (o prova) **per risoluzione** di S_1 da S, $S \vdash_R S_1$, è una sequenza $C_1, \dots, C_m$ di clausole tali che:
  - $C_m = S_1$
  - $C_i, \forall i \in \{1,2, \dots, m\}$, è
    - una clausola in S oppure
    - un risolvente di due clausole C_j e C_k con $j, k < i$.

Refutazione proposizionale

- Una derivazione di □ da S si dice **refutazione**.
- Esempio:
  - $S = \{\{A, \neg B\}, \{B, C\}, \{\neg A, C\}, \{\neg C\}\}$
    - $C_1 = \{A, \neg B\}$ clausola di S
    - $C_2 = \{B, C\}$ clausola di S
    - $C_3 = \{A, C\}$ risolvente di C_1 e C_2
    - $C_4 = \{\neg A, C\}$ clausola di S
    - $C_5 = \{C\}$ risolvente di C_3 e C_4
    - $C_6 = \{\neg C\}$ clausola di S
    - $C_7 = \square$ risolvente di C_5 e C_6

Lemma di Risoluzione

- **Lemma di risoluzione:**
  - Siano C_1 e C_2 clausole, e R un loro risolvente, allora R è conseguenza logica di C_1 e C_2 .
- **Dim.** Sia v modello per C_1 e C_2 , basta dim che $v(R) = 1$. Sia $R = (C1 - \{l\}) \cup (C2 - \{\neg l\})$ ci sono due casi:
- $v(l) = 1$: allora dato che $v(C_2) = 1$ e $v(\neg l) = 0$ (ovvero non è $\neg l$ a rendere vera C_2) segue che $v(C2 - \{\neg l\}) = 1$ e quindi $v(R) = 1$.
- $v(l) = 0$:allora dato che $v(C_1) = 1$ e $v(l) = 0$ (ovvero non è l a rendere vera C_1) segue che $v(C_1 - \{l\}) = 1$ e quindi $v(R) = 1$.

Teorema di Risoluzione

- **Teorema di Risoluzione**
  - Un insieme di clausole S è insoddisfacibile sse $S \vdash_R \square$.
  - La dimostrazione deriva dal lemma di risoluzione.

- **Corollario**
  - $S \vDash P \; sse \; S \cup \{\neg P\} \vdash_R \square$

Refutazione per $S \models P$

- Per verificare che $S \models P$ dobbiamo dim che $S^C \cup \{\neg P\}^C \vdash_R \square$

- **Algoritmo di refutazione:** dati S e P
  - scriviamo S e $\neg P$ in FNC
  - trasformiamo le FNC negli insiemi di clausole S^c e $(\neg P)^c$
  - troviamo una refutazione di $S^c \cup (\neg P)^c$

Refutazione per $\vDash P$

- Per calcolare la validità di P ($\vDash P$) basta refutare $\neg P$ (ovvero $(\neg P)^C \vdash_R \square$)

- **Algoritmo di refutazione:** dato P
  - scriviamo $\neg P$ in FNC
  - trasformiamo le FNC negli insieme di clausole $(\neg P)^c$
  - troviamo una refutazione di $(\neg P)^c$

Esempio: provare $S \models P$

- Siano $S = \{(\neg A \rightarrow B) \wedge (A \rightarrow B), \neg A \rightarrow \neg B\}$ e $P = \{A \wedge B\}$ provare che $S \models P$
- FNC di S e $\neg P$
  - $FNC(S) = (A \vee B) \wedge (\neg A \vee B) \wedge (A \vee \neg B)$
  - $FNC(\neg P) = \neg A \vee \neg B$
- Insiemi di clausole:
  - $S^c = \{\{A, B\}, \{\neg A, B\}, \{A, \neg B\}\}$
  - $(\neg \mathrm{P})^c = \{\{\neg A, \neg B\}\}$

Esempio cont.

- Insieme di clausole:
  - $S^C = \{\{A, B\}, \{\neg A, B\}, \{A, \neg B\}\}$
  - $(\neg P)^C = \{\{\neg A, \neg B\}\}$
- Risoluzione per refutazione :
- Esempio:
  - $C_1 = \{A, B\}$ clausola di S^C
  - $C_2 = \{\neg A, B\}$ clausola di S^C
  - $C_3 = \{B\}$ risolvente di C_1 e C_2
  - $C_4 = \{A, \neg B\}$ clausola di S^C
  - $C_5 = \{\neg A, \neg B\}$ clausola di $(\neg P)^C$
  - $C_6 = \{\neg B\}$ risolvente di C_4 e C_5
  - $C_7 = \square$ risolvente di C_3 e C_6

Insieme dei risolventi

- Sia S un insieme di clausole, l'**insieme dei risolventi** di S, Ris(S), è definito come:
  - $Ris(S) = S \cup \{C_{\{i,j\}} | C_{\{i,j\}}$ è la risolvente di $C_i, C_j \in S\}$
- Inoltre
- $Ris^0(S) = S$
- $Ris^{\{n+1\}}(S) = Ris(Ris^n(S))$ per $n \geq 0$
- $\boldsymbol{Ris^*(S)} = Ris^0(S) \cup Ris^1(S) \cup Ris^2(S) \cup \ldots$
- Teorema
  - Un insieme di clausole S è insoddisfacibile sse $\square \in Ris^*(S)$

Algoritmo di risoluzione

- Sia P da refutare:
- calcolare la FNC di $\neg P$
- calcolare la forma in clausole $S = (\neg P)^c$
- **repeat**
  - $R = S$
  - S = Ris(S) // calcola tutti i risolventi di S

  until $(\square \in S \; or \; S == R)$
- **if** ($\square \in S$) output “P è valida”
  else output “P non è valida”

Osservazione

- L’algoritmo di risoluzione nel caso proposizionale ha una strategia che termina e che calcola la validità di P.
- È possibile dimostrare P anche dando una sola derivazione che da $\neg P$ porta a □ (algoritmo di refutazione).
- L’algoritmo di risoluzione le prova invece tutte.

Es. algoritmo di risoluzione

- $P = \neg A \vee \neg(A \rightarrow B) \vee (A \wedge B)$
- Calcolare la FNC di $\neg P$
  - $\neg P = A \wedge (A \rightarrow B) \wedge (\neg A \vee \neg B)$
  - $FNC(\neg P) = A \wedge (\neg A \vee B) \wedge (\neg A \vee \neg B)$
- Insieme di clausole:
  - $S = (\neg P)^c = \{\{A\}, \{\neg A, B\}, \{\neg A, \neg B\}\}$

Es. algoritmo di risoluzione

- repeat (primo ciclo):
  - $R = \{\{A\}, \{\neg A, B\}, \{\neg A, \neg B\}\}$
  - $S = Ris(R) = \{\{A\}, \{\neg A, B\}, \{\neg A, \neg B\}\} \cup \{\{B\}, \{\neg B\}, \{\neg A\}\}$

  $S \neq R$ e □ non è in S: devo andare avanti.
- repeat (secondo ciclo):
  - $R = \{\{A\}, \{\neg A, B\}, \{\neg A, \neg B\}, \{B\}, \{\neg B\}, \{\neg A\}\}$
  - $S = Ris(R) = \{\{A\}, \{\neg A, B\}, \{\neg A, \neg B\}, \{B\}, \{\neg B\}, \{\neg A\}\} \cup \{\square\}$

  □ è in S: ho finito
- output “P è valida”

Es. algoritmo di refutazione

- Invece di calcolare tutti i risolventi diamo direttamente una refutazione per $\neg P$
  - $P = \neg A \vee \neg(A \rightarrow B) \vee (A \wedge B)$
  - $\neg P = A \wedge (A \rightarrow B) \wedge (\neg A \vee \neg B)$
  - $FNC(\neg P) = A \wedge (\neg A \vee B) \wedge (\neg A \vee \neg B)$
  - $(\neg P)^{\mathcal{C}} = \{\{A\}, \{\neg A, B\}, \{\neg A, \neg B\}\}$

Es. algoritmo di refutazione

- $(\neg P)^c = \{\{A\}, \{\neg A, B\}, \{\neg A, \neg B\}\}$
- Refutazione per $\neg P$:
- $C_1 = \{A\}$ clausola di $(\neg P)^c$
- $C_2 = \{\neg A, B\}$ clausola di $(\neg P)^c$
- $C_3 = \{B\}$ risolvente di C_1 e C_2
- $C_4 = \{\neg A, \neg B\}$ clausola di $(\neg P)^c$
- $C_5 = \{\neg A\}$ risolvente di C_3 e C_4
- $C_6 = \square$ risolvente di C_1 e C_5

Osservazioni

- Nell'algoritmo di Herbrand per controllare che $(P_1 \wedge P_2 \wedge \ldots \wedge P_n)$ sia insoddisfacibile si può usare il metodo di risoluzione invece delle tabelle di verità.
- Spesso il metodo di risoluzione è più efficiente delle tavole di verità.
- A volte però è necessario un numero di passi esponenziale.

Confronto

- Metodo di Herbrand:
  - costruisce tutte le formule ground
  - applica la risoluzione alle formule proposizionali ottenute.

- Metodo di risoluzione:
  - generalizza l'algoritmo di risoluzione alla logica predicativa facendo il minimo numero di sostituzioni (senza generare tutte le possibili clausole ground).

Quali sostituzioni fare?

- Useremo la tecnica di unificazione
  - usata anche in altri contesti
- Una **sostituzione** σ è un insieme finito, eventualmente vuoto, delle forma $\sigma = \{t_1/x_1, \ldots, t_n/x_n\}$ dove
  - Per ogni $i \neq j$, $x_i \neq x_j$.
  - Per ogni i, $t_i \neq x_i$.
- La **sostituzione vuota** è indicata con ε.

Sostituzione

- Sia $\sigma = \{t_1/x_1, \ldots, t_n/x_n\}$ una sostituzione ed E un'espressione. Eσ è l'applicazione di σ a E ottenuta cambiando ogni occorrenza della **variabile** x_i con il **termine** t_i .
- Esempio:
  - σ = {c/x, g(b)/y, a/z}
  - E = A(x, f(y), z)
  - Eσ = A(c, f(g(b)), a)

Composizione

- Siano $\sigma = \{v_1/x_1, \ldots, v_n/x_n\}$ e $\theta = \{u_1/y_1, \ldots, u_m/y_m\}$ due sostituzioni. La composizione di σ e θ, $\sigma \circ \theta$, si ottiene:
  - cancellando dall'insieme $S = \{v_1\theta/x_1, \ldots, v_n\theta/x_n\}, i\ v_i\theta/x_i$ per i quali $v_i\theta = x_i$.
  - cancellando dall'insieme $\theta = \{u_1/y_1, \ldots, u_m/y_m\}$, i u_i/y_i per i quali $y_i \in \{x_1, \ldots, x_n\}$.
  - unendo i due insiemi risultanti.
- Esempio:
  - $\sigma = \{f(u)/x, b/y, y/z\}$
  - $\theta = \{c/u, z/y, b/z\}$
  - $S = \{f(u)\theta/x, b\theta/y, y\theta/z\} = \{f(c)/x, b/y, z/z\}$ e ottengo $\{f(c)/x, b/y\}$ cancellando z/z.
  - Cancello z/y e b/z da {c/u, z/y, b/z} e ottengo {c/u}
  - $\sigma \circ \theta$ = {f(c)/x, b/y, c/u}

Proprietà

- Siano $\sigma_1, \sigma_2, \sigma_3$ tre sostituzioni ed E un'espressione, allora vale che:
  - $\sigma 1 \circ \varepsilon = \varepsilon \circ \sigma 1 = \sigma 1$
  - $(E\sigma 1)\sigma 2 = E(\sigma 1 \circ \sigma 2)$
  - $(\sigma 1 \circ \sigma 2) \circ \sigma 3 = \sigma 1 \circ (\sigma 2 \circ \sigma 3)$
- La composizione di sostituzioni non è commutativa:
  - In generale: $\sigma 1 \circ \sigma 2 \neq \sigma 2 \circ \sigma 1$

Unificatore

- Una sostituzione σ si dice **unificatore** di un insieme $E = \{E_1, E_2, \dots, E_n\}$ di espressioni se $E_1\sigma = E_2\ \sigma = \cdots = E_n\sigma$. Se esiste una unificazione σ di E, E è detto **unificabile**.
- Esempio:
  - $E = \{A(x, a), A(y, a)\}$
  - $\theta = \{b/x, b/y\}, A(x, a)\theta = A(y, a)\theta = A(b, a)$
  - $\sigma = \{y/x\}, A(x, a)\sigma = A(y, a)\sigma = A(y, a)$
  - σ più generale di θ $(\theta = \sigma \circ \{b/y\})$

Mgu (Most General Unifier)

- Un unificatore σ per un insieme $E = \{E_1, E_2, \dots, E_n\}$ di espressioni è detto **unificatore più generale** (**mgu**) sse per ogni altro unificatore θ dello stesso insieme esiste una sostituzione ρ tale che $\theta = \sigma \circ \rho$.

- l'mgu se esiste è unico (a meno di ridenominazione delle variabili)
- Esempio di prima: $\sigma = \{y/x\}$ e $\sigma' = \{x/y\}$ sono mgu.

Insieme di disaccordo

- Sia E un insieme finito di espressioni, l'**insieme di disaccordo** di E, D(E), è così definito:
- localizzare la posizione P più a sinistra in cui non tutte le espressioni di E hanno lo stesso simbolo
- estrarre da ogni espressione di E la sottoespressione che comincia con P
- D(E) è l'insieme di tali sottoespressioni
- Esempio:

$$E = \{A(x, f(u)), A(g(y), f(h(z)))\} \quad D(E) = \{x, g(y)\}$$
$$E = \{A(g(y), f(u)), A(g(y), f(h(z)))\} \quad D(E) = \{u, h(z)\}$$

Osservazioni

- Qualunque unificatore di E unifica D(E).

- |Eσ| = numero di espressioni differenti nell'E al quale abbiamo applicato la sostituzione σ.

Algoritmo di Unificazione

Dato un insieme di espressioni E, l'Algoritmo di unificazione calcola l'unificatore più generale (mgu) di E.

- $k = 0; \sigma_k = \varepsilon;$
- repeat
  - if (|Eσk| == 1)
    - then output σ_k; stop;
    - else trova $D(E\sigma_k)$;
  - if (esistono x e t in $D(E\sigma_k)$ t.c. x è una variabile non in t)
    - then $\sigma_{\{k+1\}} = \sigma_k \circ \{t/x\}; \; k++;$
    - else output "E non è unificabile"; stop;
- until ();

Osservazioni

- L'algoritmo può fare diverse scelte per x e t; questo porta alla formazione di mgu diversi (per nomi di variabili).
- La sostituzioe si può fare solo se x non occorre in t (si controlla con l'*occur check*):
- È necessario per garantire la terminazione dell'algoritmo
- Es: E = {A(x), A(f(x))} non è unificabile poiché x occorre in f(x).

Quando posso unificare?

	costante c_2	variabile x_2	funzione/predicato S_2
costante c_1	se $c_1 = c_2$	$\{c_1/x_2\}$	no
variabile x_1	$\{c_2/x_1\}$	$\{x_2/x_1\}\ o\ \{x_1/x_2\}$	$\{S_2/x_1\}\ dopo\ o.c.$
funzione/predicato S_1	no	$\{S_1/x_2\}$. dopo o. c	se $S_1 = S_2$ si va ricorsivamente sugli argomenti

Esempi

- a e f(x,y) non sono unificabili
- a e b non sono unificabili
- f(x) e f(g(x)) non sono unificabili (o.c.)
- x e y sono unificabili mgu = {x/y}
- f(h(x)) e f(h(P(y))) sono unificabili mgu= {P(y)/x}

Esempio

$$E = \{A(f(y, g(v)), h(b)), A(f(h(w), g(a)), t), A(f(h(b), g(v)), t)\}$$

Passo 1 : $|E\sigma_0| > 1$; $D(E\sigma_0) = \{y, h(w), h(b)\}$; $\sigma_1 = \{^{h(w)}/_{y}\}$ allora $E\sigma_1 = \{A(f(h(w), g(v)), h(b)), A(f(h(w), g(a)), t), A(f(h(b), g(v)), t)\}$.

Passo 2 : $|E\sigma_1| > 1$; $D(E\sigma_1) = \{w, b\}$; $\sigma_2 = \sigma_1 \circ \{^{b}/_{w}\}$, allora $E\sigma_2 = \{A(f(h(b), g(v)), h(b)), A(f(h(b), g(a)), t), A(f(h(b), g(v)), t)\}$.

Passo 3 : $|E\sigma_2| > 1$; $D(E\sigma_2) = \{v, a\}$; $\sigma_3 = \sigma_2 \circ \{^{a}/_{v}\}$, allora $E\sigma_3 = \{A(f(h(b), g(a)), h(b)), A(f(h(b), g(a)), t), A(f(h(b), g(a)), t)\}$.

Passo 4 : $|E\sigma_3| > 1$; $D(E\sigma_3) = \{h(b), t\}$; $\sigma_4 = \sigma_3 \circ \{^{h(b)}/_{t}\}$, allora $E\sigma_4 = \{A(f(h(b), g(a)), h(b)), A(f(h(b), g(a)), h(b)), A(f(h(b), g(a)), h(b))\}$.

Passo 5 : $|E\sigma_4| = 1$; $\sigma_4 = \{^{h(w)}/_{y}, ^{b}/_{w}, ^{a}/_{v}, ^{h(b)}/_{t}\}$ è un unificatore più generale per E.

Risoluzione nella logica del primo ordine

- Ogni formula del primo ordine può essere trasformata in una formula $\forall x_1 x_2 \ldots x_n P$:
  - chiusa
  - senza quantificatori esistenziali
  - con P in FNC
- $\forall x_1 x_2 \ldots x_n (C_1 \ \wedge \ C_2 \ \wedge \ \ldots \wedge C_n)$

Osservazioni

- $\forall x_1 x_2 \dots x_n (C_1 \ \wedge \ \dots \ \wedge C_n)$
- è equivalente a
  - $\forall x_1 x_2 \dots x_n C_1 \ \wedge \ \dots \ \wedge \forall x_1 x_2 \dots x_n C_n$
- Possiamo eliminare i quantificatori e otteniamo $(C_1 \ \wedge \ \dots \ \wedge C_n)$ che in forma a clausole diventa:
  - $\{C_1 , \dots , C_n\}$

Risolvente

- Siamo C_1, C_2 e R clausole.R è una **risolvente** di C_1 e C_2 se sono verificate le seguenti condizioni:
  - esistono due sostituzioni s_1 e s_2 che ridenominano le variabili in modo che $C_1 s_1$ e $C_2 s_2$ non abbiano variabili in comune.
  - esistono due insiemi di letterali $\{l_1, \dots, l_m\}$ in $C_1 s_1$ $(m > 0)$ e $\{l'_1, \dots, l'_n\}$ in $C_2 s_2$ $(n > 0)$ tali che $L = \{\neg l_1, \dots, \neg l_m, l'_1, \dots, l'_n\}$ è unificabile con l'unificatore mgu σ.
  - R ha la forma
    - $((C_1 s_1 - \{l_1, \dots, l_m\}) \cup (C_2 s_2 - \{l'_1, \dots, l'_n\}))\sigma$.

Esempio

- $C_1 = \{A(z), \neg B(y), A(g(x))\}$
- $C_2 = \{\neg A(x), \neg C(x)\}$
- $s_1 = \{\}$; $C_1 s_1 = \{A(z), \neg B(y), A(g(x))\}$
- $s_2 = \{u/x\}$; $C_2 s_2 = \{\neg A(u), \neg C(u)\}$
- $L = \{\neg A(z), \neg A(g(x)), \neg A(u)\}$
- $\sigma = \{g(x)/z, g(x)/u\}$
- $R = \{\neg B(y), \neg C(g(x))\}$

Osservazione

- La ridenominazione è necessaria:
- Es.$C_1 = \{A(x)\}, C_2 = \{\neg A(g(x))\}$.
- senza ridenominazione non si può eseguire l'unificazione (occur check)
- $s_2 = \{y/x\}$; $C_2 s_2 = \{\neg A(g(y))\}$.
- $\sigma = \{g(y)/x\}$
- $R = \square$.

Osservazione

- Nella logica dei predicati si cancellano più letterali alla volta, ma devono essere tutti unificabili tra loro!
- Non sempre è possible eliminare un letterale alla volta da C_1 e C_2.
- Es. $C1 = \{A(x), A(y)\}, C2 = \{\neg A(x), \neg A(y)\}$.
- $R = \square$.
- Se si elimina solo un letterale per volta non si ottiene mai $\square$.

Derivazione

- Sia S un insieme di clausole, **una derivazione** (o prova) **per risoluzione** di S_1 da S, $S \vdash_R S_1$, è una sequenza $C_1, \dots, C_m$ di clausole tali che:
  - $C_m = S_1$
  - $C_i, \forall i \in \{1,2, \dots, m\}$, è
    - una clausola in S oppure
    - un risolvente di due clausole C_j e C_k con $j, k < i$.

Insieme dei risolventi

- Sia S un insieme di clausole, l'**insieme dei risolventi** di S, **Ris(S)**, è definito come:
  - $Ris(S) = S \cup \{C_{\{i,j\}} | C_{\{i,j\}}$ è la risolvente di $C_i, C_j \in S\}$
- Inoltre
  - $Ris^0(S) = S$
  - $Ris^{\{n+1\}}(S) = Ris(Ris^n(S))$ per $n \geq 0$
  - $\boldsymbol{Ris^*(S)} = Ris^0(S) \cup Ris^1(S) \cup Ris^2(S) \cup \ldots$

Teorema di risoluzione

- **Teorema di risoluzione**:
  - Un insieme di clausole S è insoddisfacibile sse $S \vdash_R \square$.
- Oppure
  - Un insieme di clausole S è insoddisfacibile sse $\square \in$Ris*(S)

Algoritmo di risoluzione

- Sia P da refutare:
- 1. negare P
- 2. trasformare $\neg P$ in clausole
- 3. $S = \{(\neg P)^c\}$
- 4. repeat
  - S = Ris(S)
- 5. $until\ (\square \in S)$
- 6. output “P è insoddisfacibile”

Esempio

- L'algoritmo può non terminare.
- $S = \{\{A(0)\}, \{\neg A(x), A(s(x))\}\}$

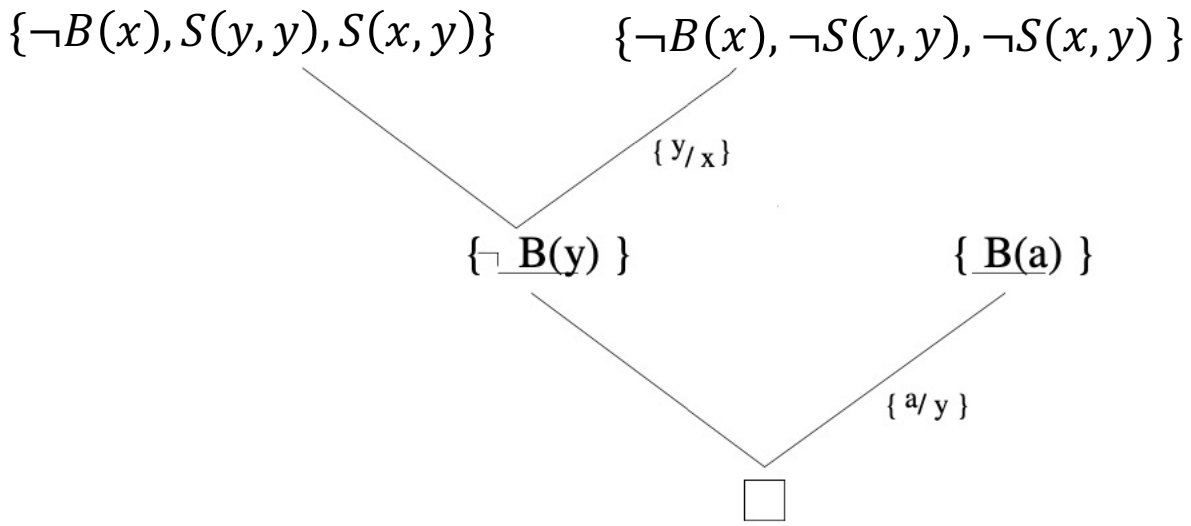
- $Ris^1(S) = S \cup \{\{A(s(0))\}\}$
- $Ris^2(S) = Ris^1(S) \cup \{\{A(s(s(0)))\}\}$
- $Ris^3(S) = Ris^2(S) \cup \{\{A(s(s(s(0))))\}\}$
- …

Esempio del Barbiere

- $B(x)$ = x è un barbiere
- S(x, y) = x rade y
- Ogni barbiere rade tutti coloro che non si radono da soli.
  - a) $\big(B(x) \wedge \neg S(y, y)\big) \rightarrow S(x, y)$
- Nessun barbiere rade chi si rade da solo.
  - b) $(B(x) \wedge S(y, y)) \rightarrow \neg S(x, y)$
- Non esistono Barbieri
  - c) $\neg B(a)$
- Dimostriamo che c è conseguenza logica di {a,b}

Esempio del Barbiere

- Dimostriamo che c è conseguenza logica di {a,b}
- In clausole:
  a) $\{\neg B(x), S(y,y), S(x,y)\}$
  b) $\{\neg B(x), \neg S(y,y), \neg S(x,y)\ \}$
  c) $\{B(a)\}$



Raffinamenti

- Il metodo di risoluzione
  - è più efficiente dell'algoritmo di Herbrand,
  - ma può generare clausole ridondanti e irrilevanti.
- Si possono effettuare delle semplificazioni:
  - Eliminazione delle tautologie, $\{A, \neg A\}$.
  - Eliminazione delle clausole già generate.
  - Eliminazione delle clausole comprese in altre

Strategie di risoluzione

- pur effettuando tali semplificazioni, in molti casi, il numero di risolventi generati pu`o essere comunque troppo grande.
- per migliorare le prestazioni dei dimostratori automatici dei teoremi, si utilizzano delle strate-gie che scelgono in modo opportuno le clausole da cui generare una risolvente.

Risoluzione lineare

- Una prova per **risoluzione lineare** di C da un insieme di clausole S è una sequenza $C_1, \dots, C_n$ tale che
  - $C_1 \in S$
  - $C_n = C$
  - $\forall i = 2, \dots, n$ risulta:
    - C_i è la risolvente di $C_{\{i-1\}}$ e $B_{\{i-1\}}$ (clausola laterale), con $B_{\{i-1\}} \in S$ o $B_{i-1} = C_j$ con $j < i$.

- Osservazione: a ogni passo si utilizza la risolvente del passo precedente.

Esempio

- $S = \{\{A, B\}, \{A, \neg B\}, \{\neg A, B\}, \{\neg A, \neg B\}\}$
- $C_1 = \{A, B\}, \quad B_1 = \{A, \neg B\}$
- $C_2 = \{A\}, \quad B_2 = \{\neg A, B\}$
- $C_3 = \{B\}, \quad B_3 = \{\neg A, \neg B\}$
- $C_4 = \{\neg A\}, \quad B_4 = \{A\}$
- $C_5 = \square$

Completezza

- Teorema. Se un insieme di clausole S è insoddisfacibile, allora la clausola vuota è derivabile dalla risoluzione lineare
- La risoluzione lineare è complete per refutazione
- Dobbiamo però “ricordarci” tutti i risolventi calcolati fino ad ora
- C’è un’altra risoluzione che risolve questo problema: risoluzione da input

Risoluzione di input

- Una prova per **risoluzione di input** di C da un insieme di clausole S, è una sequenza $C_1, \ldots, C_n$ tale che $C_n = C$ e, ad ogni passo, una delle clausole risolventi è un'istanza di un elemento di S. Ovvero:
- $C_1 \in S$
- $C_n = C$
- C_i è la risolvente di $C_{\{i-1\}}$ e $B_{\{i-1\}}$ (clausola laterale), con $B_{\{i-1\}} \in S$

Programmazione logica

- Per la dimostrazione automatica dei teoremi sono necessarie strategie efficienti.
- La refutazione lineare da input è
  - particolarmente efficiente
  - ma non completa per refutazione.
- Se ci restringiamo a particolari clausole (clausole di Horn) la risoluzione da input diventa completa per refutazione.

Clausole di Horn

- Una **clausola di Horn** è una clausola che contiene al più una formula atomica positiva
- Esempio sono clausole di Horn:
  - $\{A, \neg B\}$,
  - $\{\neg A, \neg B, \neg C, D\}$,
  - $\{A\}$,
  - $\{\neg A, \neg B\}$

Programmi logici

- $\{A_0, \neg A_1, \neg A_2, \ldots, \neg A_n\}$ è detta **clausola di Horn definita.**
- $\{\neg A_0, \neg A_1, \ldots, \neg A_n\}$ è detta **clausola goal**
- **Notazione:**
  - clausola definita: A0:-A1, A2, ..., An
  - clausola goal: ?-A1, A2, ..., An
- Un **programma logico** è un insieme finito di clausole definite.

Risoluzione SLD

- Una **prova per risoluzione SLD** (Linear resolution for Definite clauses with Selection function) è una derivazione per risoluzione lineare di input tale che:
  - ogni risolvente laterale è una clausola di Horn definita,
  - ogni altro risolvente è una clausola goal.
- La risoluzione SLD è completa per clausole di Horn.
- Una **refutazione SLD** è una prova per risoluzione SLD nella quale l'ultimo risolvente è la clausola vuota.