Il metodo di risoluzione per la logica proposizionale

- Sistema di inferenza con una sola regola: la regola di risoluzione
- La regola di risoluzione si applica a **clausole** (disgiunzioni di **letterali**)

$$L_1 \vee L_2 \vee .... \vee L_k$$

dove $L_i = p_i$ oppure $L_i = \neg p$

Un **letterale** è un atomo o la negazione di un atomo

Una **clausola** è una disgiunzione di letterali

- Una clausola si può rappresentare mediante l'insieme dei suoi letterali

$$L_1 \vee L_2 \vee .... \vee L_k \implies \{L_1, L_2, ..., L_k\}$$

Regola di risoluzione proposizionale

$$\frac{C_1 \cup \{p\} \ ; \quad C_2 \cup \{\neg p\}}{C_1 \cup C_2}$$

$C_1 \cup C_2$ è il **risolvente**
p e $\neg p$ sono **letterali complementari**

Esempio

$$\neg p \vee q, p \vee r, \neg q \vee s \vdash_{RES} r \vee s$$

$$\dfrac{\dfrac{\{\neg p, q\} \quad \{p, r\}}{\{q, r\}} \qquad \{\neg q, s\}}{\{r, s\}}$$

Oppure:

$$\dfrac{\dfrac{\neg p \vee q \quad p \vee r}{q \vee r} \qquad \neg q \vee s}{r \vee s}$$

$$\neg p \vee q, \neg q, p \vdash_{RES} \bot$$

$$\dfrac{\dfrac{\{\neg p, q\} \quad \{\neg q\}}{\{\neg p\}} \qquad \{p\}}{\emptyset}$$

$$\dfrac{\dfrac{\neg p \vee q \quad \neg q}{\neg p} \qquad p}{\Box}$$

$\Box$ è la **clausola vuota** (il falso)

Risoluzione proposizionale

Teorema . La regola di risoluzione proposizionale è **corretta**:
se C è un risolvente di C_1 e C_2, allora $C_1, C_2 \models C$

Corollario . Il sistema di risoluzione proposizionale è corretto:

$$S \vdash_{RES} C \implies S \models C$$

Ma il sistema di risoluzione proposizionale non è completo rispetto alla derivabilità:

$$\not\vdash_{RES} p \vee \neg p$$

Il sistema di risoluzione come metodo di refutazione

$$S \models A \text{ sse } S \cup \{\neg A\} \text{ è insoddisfacibile}$$

Per dimostrare $S \models A$ si **refuta** $S \cup \{\neg A\}$, cioè si dimostra che $S \cup \{\neg A\} \vdash \bot$

Per "dimostrare" per risoluzione $S \models A$:

1. trasformare ogni formula C in $S \cup \{\neg A\}$ in **forma a clausole**:

$$\begin{aligned} C &\implies (L_{1,1} \vee ... \vee L_{1,n_1}) \wedge ... \wedge (L_{k,1} \vee ... \vee L_{k,n_k}) \\ &\implies \{L_{1,1} \vee ... \vee L_{1,n_1}, ..., L_{k,1} \vee ... \vee L_{k,n_k}\} \\ &\implies \{\{L_{1,1}, ..., L_{1,n_1}\}, ..., \{L_{k,1}, ..., L_{k,n_k}\}\} \end{aligned}$$

2. Se S è l'insieme di clausole ottenute, dimostrare che:

$$S \vdash_{RES} \square$$

Esempio

$$p \wedge q \rightarrow r, p \rightarrow q \models p \rightarrow r$$

1. Trasformazione in forma a clausole

$$
\begin{array}{lll}
a) & p \wedge q \rightarrow r & \Longrightarrow \neg(p \wedge q) \vee r \\
 & & \Longrightarrow \neg p \vee \neg q \vee r \\
 & & \Longrightarrow \{\neg p \vee \neg q \vee r\} \\
b) & p \rightarrow q & \Longrightarrow \neg p \vee q \\
 & & \Longrightarrow \{\neg p \vee q\} \\
c) & \neg(p \rightarrow r) & \Longrightarrow p \wedge \neg r \\
 & & \Longrightarrow \{p, \neg r\}
\end{array}
$$

$$S = \{\neg p \vee \neg q \vee r, \neg p \vee q, p, \neg r\}$$

2. Derivazione della clausola vuota:

$$
\dfrac{\dfrac{\dfrac{\neg p \vee \neg q \vee r \quad \neg r}{\neg p \vee \neg q} \quad \dfrac{\neg p \vee q \quad p}{q}}{\neg p} \quad p}{\Box}
$$

COMPLETEZZA della risoluzione proposizionale come metodo di refutazione

$$\text{Se } S \models A \text{ allora } S \cup \{\neg A\} \vdash_{RES} \Box$$

cioè: se $S \models A$ allora esiste una derivazione per risoluzione di $\Box$ dalla forma a clausole di $S \cup \{\neg A\}$.

Quindi: se S è un insieme di clausole:

$$S \text{ è insoddisfacibile sse } S \vdash_{RES} \Box$$

Il metodo di risoluzione per la logica del primo ordine

FORME NORMALI PRENESSE

$$\underbrace{Q_1 x_1 Q_2 x_2 ... Q_n x_n}_{prefisso} \ \underbrace{A}_{matrice}$$

la matrice è senza quantificatori.

Ogni formula è logicamente equivalente a una formula in forma normale prenessa.

FORME NORMALI DI SKOLEM

$$\forall x_1 ... \forall x_n A \qquad A \text{ senza quantificatori}$$

Trasformazione di una formula in forma di Skolem (**"skolemizzazione"**):

1. Trasformazione in forma prenessa: $Q_1 x_1 ... Q_n x_n A$
2. Eliminazione dei quantificatori esistenziali:

$$\forall x_1 ... \forall x_n \exists y A \Longrightarrow \forall x_1 ... \forall x_n A[f(x_1, ..., x_n)/y]$$

   dove $\forall x_1, ..., \forall x_n$ sono tutti i quantificatori universali che precedono $\exists y$ e f è un simbolo funzionale **nuovo** (***funzione di skolem***)

Esempio

$$\exists x\forall y\forall z\exists u\forall w\exists v\ p(x,y,z,u,w,v)$$

$$\Rightarrow\ \forall y\forall z\exists u\forall w\exists v\ p(c,y,z,u,w,v)$$

$$\Rightarrow\ \forall y\forall z\forall w\exists v\ p(c,y,z,f(y,z),w,v)$$

$$\Rightarrow\ \forall y\forall z\forall w\ p(c,y,z,f(y,z),w,g(y,z,w))$$

dove c, f, g sono simboli *nuovi*

$$\forall x\exists y\, p(x,y) \Longrightarrow \forall x\, p(x,f(x))$$

y "dipende" da x

$$\exists y\forall x\, p(x,y) \Longrightarrow \forall x\, p(x,c)$$

y non dipende da x

Forma clausale

1. Trasformare A in forma normale di Skolem
2. Eliminare i quantificatori universali
3. Trasformare la matrice in forma normale congiuntiva
4. Trasformare in insieme di clausole

Esempio

$$\forall x \exists y \exists z (\neg p(x,y) \vee \neg(r(x,y,z) \rightarrow q(x,z)))$$

$$\Rightarrow \;\; \neg p(x, f(x)) \vee \neg(r(x, f(x), g(x)) \rightarrow q(x, g(x)))$$

$$\Rightarrow \;\; \neg p(x, f(x)) \vee (r(x, f(x), g(x)) \wedge \neg q(x, g(x)))$$

$$\Rightarrow \;\; (\neg p(x, f(x)) \vee r(x, f(x), g(x))) \wedge (\neg p(x, f(x)) \vee \neg q(x, g(x)))$$

$$\Rightarrow \;\; \{\neg p(x, f(x)) \vee r(x, f(x), g(x)), \neg p(x, f(x)) \vee \neg q(x, g(x))\}$$

N.B: Le variabili libere si intendono quantificate universalmente; se A è una formula con le variabili libere $x_1, ..., x_n$, A sta per la sua "chiusura universale" $\forall x_1, ..., \forall x_n A$

Denotiamo con $\forall A$ la chiusura universale di A

Esercizio 16 del paragrafo 2.3.9

Alcuni botanici sono eccentrici. Alcuni botanici non amano cose eccentriche. Quindi alcuni botanici non sono amati da tutti i botanici.

$$\exists x(b(x) \land e(x)),\ \exists x(b(x) \land \forall y(e(y) \to \neg a(x,y))) \ \models\ \exists x(b(x) \land \neg \forall y(b(y) \to a(y,x)))$$

Trasformazione delle formule di S e della negazione della conclusione in forma clausale:

$$\begin{aligned} \exists x(b(x) \land e(x)) &\Rightarrow b(c_0) \land e(c_0) \\ &\Rightarrow \{b(c_0),\ e(c_0)\} \end{aligned}$$

$$\begin{aligned} \exists x(b(x) \land \forall y(e(y) \to \neg a(x,y))) &\Rightarrow \exists x \forall y(b(x) \land (\neg e(y) \lor \neg a(x,y))) \\ &\Rightarrow b(c_1) \land (\neg e(y) \lor \neg a(c_1,y)) \\ &\Rightarrow \{b(c_1),\ \neg e(y) \lor \neg a(c_1,y)\} \end{aligned}$$

$$\begin{aligned} \neg \exists x(b(x) \land \neg \forall y(b(y) \to a(y,x))) &\Rightarrow \forall x \neg(b(x) \land \exists y \neg(b(y) \to a(y,x))) \\ &\Rightarrow \forall x \neg \exists y(b(x) \land \neg(b(y) \to a(y,x))) \\ &\Rightarrow \forall x \forall y \neg(b(x) \land \neg(b(y) \to a(y,x))) \\ &\Rightarrow \neg b(x) \lor (b(y) \to a(y,x)) \\ &\Rightarrow \neg b(x) \lor \neg b(y) \lor a(y,x) \\ &\Rightarrow \{\neg b(x) \lor \neg b(y) \lor a(y,x)\} \end{aligned}$$

L'insieme di clausole che si ottiene è:

$$\{b(c_0),\ e(c_0),\ b(c_1),\ \neg e(y) \lor \neg a(c_1,y),\ \neg b(x) \lor \neg b(y) \lor a(y,x)\}$$

Che relazione c'è tra A e la sua forma a clausole?

$$
\begin{aligned}
A &\Longrightarrow \text{forma prenessa (1)}\\
&\Longrightarrow \text{forma di Skolem}\\
&\Longrightarrow \text{forma di Skolem con matrice in FNC (2)}\\
&\Longrightarrow \text{eliminazione dei } \forall \text{ (3)}\\
&\Longrightarrow \text{insieme di clausole (4)}
\end{aligned}
$$

1. Se $pre(A)$ è una forma prenessa di A, allora $A \ \leftrightarrow \ pre(A)$
2. Se $FNC(A)$ è una forma normale congiuntiva di A, allora $\forall x_1 ... \forall x_n A \ \leftrightarrow \ \forall x_1 ... \forall x_n FNC(A)$
3. Eliminazione dei quantificatori universali: A sta per $\forall x_1 ... \forall x_n A$
4. $\forall x_1 ... \forall x_n (C_1 \wedge ... \wedge C_k) \ \leftrightarrow \ \forall x_1 ... \forall x_n C_1 \wedge ... \wedge \forall x_1 ... \forall x_n C_k$
   Un insieme di formule S sta per la congiunzione delle formule in S, quindi
   $\forall x_1 ... \forall x_n (C_1 \wedge ... \wedge C_k)$ equivale a $\{\forall x_1 ... \forall x_n C_1, ..., \forall x_1 ... \forall x_n C_k\}$, cioè $\{C_1, ..., C_k\}$

Relazione tra A e $sk(A)$: non sono equivalenti

Se $sk(A)$ è una forma di Skolem di A

$$\not\models A \equiv sk(A)$$

Più precisamente:

$$\boxed{\models sk(A) \rightarrow A}$$

Sia $A = \forall x_1 ... \forall x_n \exists y A$ e $sk(A) = \forall x_1 ... \forall x_n A[f(x_1, ..., x_n)/y]$

$\mathcal{M} \models \forall x_1 ... \forall x_n A[x_1, ..., x_n, f(x_1, ..., x_n)]$
$\Rightarrow$ per ogni s e $d_1, ..., d_n \in D$: $(\mathcal{M}, s[d_1/x_1, ..., d_n/x_n]) \models A[x_1, ..., x_n, f(x_1, ..., x_n)]$
$\Rightarrow$ per ogni s e $d_1, ..., d_n \in D$: $(\mathcal{M}, s[d_1/x_1, ..., d_n/x_n]) \models \exists y A[x_1, ..., x_n, y]$
$\Rightarrow$ $\mathcal{M} \models \forall x_1 ... \forall x_n \exists y A[x_1, ..., x_n, y]$

Ma $\boxed{\not\models A \rightarrow sk(A)}$

Sia $A = \exists x\, p(x)$, $sk(A) = p(c)$
Consideriamo $\mathcal{M}$ con $D = \{1, 2\}$, $\mathcal{M}(c) = 1$, $\mathcal{M}(p) = \{2\}$
Chiaramente: $\mathcal{M} \models \exists x\, p(x)$, ma $\mathcal{M} \not\models p(c)$

Relazione tra A e $sk(A)$

Tuttavia,

$$A \text{ è soddisfacibile} \Longrightarrow sk(A) \text{ è soddisfacibile}$$

Quindi **la skolemizzazione conserva la *soddisfacibilità***

$$\boxed{A \text{ è soddisfacibile} \Longleftrightarrow sk(A) \text{ è soddisfacibile}}$$

(poichè $\models sk(A) \rightarrow A$: $sk(A)$ soddisfacibile $\Longrightarrow$ A soddisfacibile)

Questo è quel che interessa per i metodi di prova per refutazione:

$$\begin{aligned} S \models A \ &\Leftrightarrow S \cup \{\neg A\} \text{ è insoddisfacibile} \\ &\Leftrightarrow \text{ la forma a clausole di } S \cup \{\neg A\} \text{ è insoddisfacibile} \end{aligned}$$

Teorema

- A è insoddisfacibile sse la sua forma a clausole è insoddisfacibile
- se S è un insieme di formule e S' è ottenuto trasformando ogni formula in S in forma clausale, allora S è insoddisfacibile sse S' è insoddisfacibile.

Il metodo di risoluzione controlla l'insoddisfacibilità di insiemi di clausole

Regola di risoluzione

$$\frac{C_1 \cup \{P\}\ ;\quad C_2 \cup \{\neg Q\}}{C_1\theta \cup C_2\theta} \qquad \text{se } \theta = mgu(P,Q)$$

$C_1\theta \cup C_2\theta$ è un **risolvente binario** di $C_1 \cup \{P\}$ e $C_2 \cup \{\neg Q\}$.

Esempio:

$$\frac{p(x) \lor q(x)\ ;\quad \neg p(f(y)) \lor r(y)}{q(f(y)) \lor r(y)} \qquad \theta = [f(y)/x]$$

La regola di risoluzione al primo ordine combina l'istanziazione di variabili (universali) con la regola di risoluzione proposizionale:

$$\frac{\dfrac{p(x) \lor q(x)}{p(f(y)) \lor q(f(y))}\,IST \qquad \neg p(f(y)) \lor r(y)}{q(f(y)) \lor r(y)}$$

Si può assumere che le due **clausole "genitrici"** non abbiano variabili in comune: quando una clausola viene usata, si rinominano le sue variabili (***standardizing apart***), ottenendo una **variante** della clausola.

Rinominare le variabili in una clausola = rinominare variabili quantificate universalmente

Esercizio 16 del paragrafo 2.3.9: dimostrazione

$$\exists x(b(x) \land e(x)),\ \exists x(b(x) \land \forall y(e(y) \to \neg a(x,y))) \ \models\ \exists x(b(x) \land \neg \forall y(b(y) \to a(y,x)))$$

se e solo se

$$b(c_0),\ e(c_0),\ b(c_1),\ \neg e(y) \lor \neg a(c_1,y),\ \neg b(x) \lor \neg b(y) \lor a(y,x)\ \vdash_{RES} \square$$

1. $b(c_0)$ *in S*
2. $e(c_0)$ *in S*
3. $b(c_1)$ *in S*
4. $\neg e(y) \lor \neg a(c_1,y)$ *in S*
5. $\neg b(x) \lor \neg b(y) \lor a(y,x)$ *in S*

6. $\neg b(y) \lor a(y,c_0)$ $RES(1,5), \{c_0/x\}$
7. $a(c_1,c_0)$ $RES(3,6), \{c_1/y\}$
8. $\neg e(c_0)$ $RES(4,7), \{c_0/y\}$
9. $\square$ $RES(2,8)$

Esempio

1. Alcuni funzionari di dogana hanno perquisito tutti coloro che sono entrati nel paese, ad eccezione dei VIP.
2. Alcuni spacciatori di droga sono entrati nel paese e sono stati perquisiti solo da spacciatori di droga.
3. Nessuno spacciatore è un VIP.
4. Quindi alcuni funzionari sono spacciatori di droga.

Rappresentazione

$\mathcal{L}$:	$E(x)$	x è entrato nel paese
	$V(x)$	x è un VIP
	$P(x,y)$	y ha perquisito x
	$F(x)$	x è un funzionario di dogana
	$S(x)$	x è uno spacciatore di droga

1. $\forall x(E(x) \land \neg V(x) \rightarrow \exists y(F(y) \land P(x,y)))$
2. $\exists x(S(x) \land E(x) \land \forall y(P(x,y) \rightarrow S(y)))$
3. $\forall x(S(x) \rightarrow \neg V(x))$
4. $\exists x(F(x) \land S(x))$

Trasformazione in clausole

1. $\forall x(E(x) \land \neg V(x) \rightarrow \exists y(F(y) \land P(x,y)))$
$\Rightarrow\ \forall x \exists y(\neg(E(x) \land \neg V(x)) \lor (F(y) \land P(x,y)))$
$\Rightarrow\ \forall x(\neg E(x) \lor V(x) \lor (F(f(x)) \land P(x,f(x))))$
$\Rightarrow\ (\neg E(x) \lor V(x) \lor F(f(x))) \land (\neg E(x) \lor V(x) \lor P(x,f(x)))$
$\Rightarrow\ \{\neg E(x) \lor V(x) \lor F(f(x)), \neg E(x) \lor V(x) \lor P(x,f(x))\}$

2. $\exists x(S(x) \land E(x) \land \forall y(P(x,y) \rightarrow S(y)))$
$\Rightarrow\ \exists x \forall y(S(x) \land E(x) \land (\neg P(x,y) \lor S(y)))$
$\Rightarrow\ S(a) \land E(a) \land (\neg P(a,y) \lor S(y))$
$\Rightarrow\ \{S(a), E(a), \neg P(a,y) \lor S(y)\}$

3. $\forall x(S(x) \rightarrow \neg V(x))$
$\Rightarrow\ \forall x(S(x) \rightarrow \neg V(x))$
$\Rightarrow\ \{\neg S(x) \lor \neg V(x)\}$

$\neg$4. $\neg \exists x(F(x) \land S(x))$
$\Rightarrow\ \forall x \neg(F(x) \land S(x))$
$\Rightarrow\ \{\neg F(x) \lor \neg S(x)\}$

$$S = \{\ \neg E(x) \lor V(x) \lor F(f(x)), \neg E(x) \lor V(x) \lor P(x,f(x)), S(a), E(a), \neg P(a,y) \lor S(y), \neg S(x) \lor \neg V(x), \neg F(x) \lor \neg S(x)\}$$

Esempio (continua)

Dimostrazione per risoluzione da S:

$$\begin{array}{rll}
1. & \neg E(x) \vee V(x) \vee F(f(x)) & \mathit{in}\ S \\
2. & \neg E(x) \vee V(x) \vee P(x, f(x)) & \mathit{in}\ S \\
3. & S(a) & \mathit{in}\ S \\
4. & E(a) & \mathit{in}\ S \\
5. & \neg P(a, y) \vee S(y) & \mathit{in}\ S \\
6. & \neg S(x) \vee \neg V(x) & \mathit{in}\ S \\
7. & \neg F(x) \vee \neg S(x) & \mathit{in}\ S \\
 & & \\
8. & V(a) \vee P(a, f(a)) & Res(2, 4), \{a/x\} \\
9. & \neg V(a) & Res(3, 6), \{a/x\} \\
10. & P(a, f(a)) & Res(8, 9) \\
11. & S(f(a)) & Res(5, 10), \{f(a)/y\} \\
12. & \neg F(f(a)) & res(7, 11), \{f(a)/x\} \\
13. & V(a) \vee F((f(a)) & Res(1, 4), \{a/x\} \\
14. & F(f(a)) & Res(9, 13) \\
15. & \Box & Res(12, 14)
\end{array}$$

$$
\dfrac{
  \dfrac{(7) \qquad \dfrac{(5) \qquad \dfrac{\dfrac{(2) \quad (4)}{V(a) \lor P(a, f(a))} \qquad \dfrac{(3) \quad (6)}{\neg V(a)}}{P(a, f(a))}}{S(f(a))}}{\neg F(f(a))}
  \qquad
  \dfrac{\dfrac{(3) \quad (6)}{\neg V(a)} \qquad \dfrac{(1) \quad (4)}{V(a) \lor F(f(a))}}{F(f(a))}
}{\square}
$$

Fattorizzazione

$$S = \{p(x) \vee p(y),\ \neg p(c) \vee \neg p(z)\}$$

è insoddisfacibile, ma ogni clausola derivabile mediante la regola di risoluzione, come è stata formulata fin qui, contiene due letterali. Quindi la clausola vuota non è derivabile.

Definizione. Se $C = C' \cup D$ e esiste un mgu θ per D, allora $C\theta$ è un **fattore** di C.

Esempio: $p(f(y)) \vee r(f(y), y)$ è un fattore di $p(x) \vee p(f(y)) \vee r(x, y)$.

Una clausola è sempre un fattore (banale) di se stessa.

Definizione: Un **risolvente** di C_1 e C_2 è un risolvente binario di un fattore di C_1 e di un fattore di C_2.

Regola di risoluzione: se

$C_1' \cup \{P\}$ è un fattore di C_1
$C_2' \cup \{\neg Q\}$ è un fattore di C_2
θ è un $mgu(P, Q)$

allora:

$$\frac{C_1\ ;\quad C_2}{C_1'\theta \cup C_2'\theta}$$

Esempio:

$$\frac{p(x) \vee p(y),\ \neg p(c) \vee \neg p(z)}{\square}$$

$p(x)$ è un fattore di $p(x) \vee p(y)$
$\neg p(c)$ è un fattore di $\neg p(c) \vee \neg p(z)$

Esercizi

Risovere gli esercizi seguenti, utilizzando, dove appropriato, il metodo di risoluzione:

- 4 pag. 27 (usare la risoluzione per le formule valide)
- 6 pag. 29 (usare la risoluzione per i ragionamenti corretti)
- 7 pag. 29-30
- 1, 3, 4, 5, 6 pag. 59-60 (sostituendo NK con RES)
- I, K, L pag. 92-94
- 1 pag. 147-148
- Dimostrare per risoluzione la validità del seguente ragionamento: Tutti gli stabilimenti hanno un cane da guardia. La notte scorsa qualcuno ha compiuto un furto allo stabilimento della Turbo & Co. di Bellegra, ma i cani da guardia non hanno abbaiato. Un cane da guardia non abbaia a un ladro, soltanto se è il proprio padrone. Quindi il padrone dei cani da guardia dello stabilimento della Turbo & Co. di Bellegra è colpevole.

  Utilizzare un linguaggio con i predicati seguenti:

$cane(x, y)$	x è un cane da guardia del posto y
$stab(x)$	x è uno stabilimento
$ladro(x, y)$	x ha compiuto un furto nel posto y
$abbaia(x, y)$	x abbaia a y
$padrone(x, y)$	y è un padrone di x

Raffinamenti della risoluzione

Il metodo di risoluzione fornisce vantaggi drastici rispetto ad altri sistemi di inferenza.

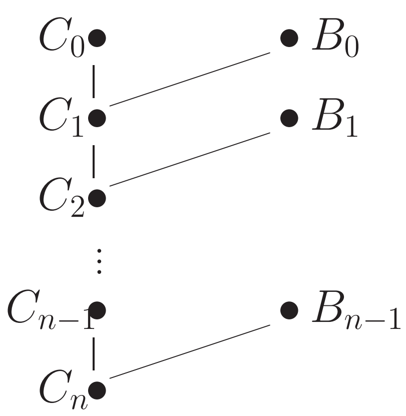
Ma l'applicazione non ristretta della risoluzione genera molte clausole inutili

Esempio: dimostrazione per saturazione di livelli

1.	$P \vee Q$	S
2.	$\neg P \vee R$	S
3.	$\neg Q \vee R$	S
4.	$\neg R$	S
5.	$Q \vee R$	$1, 2$
6.	$P \vee R$	$1, 3$
7.	$\neg P$	$2, 4$
8.	$\neg Q$	$3, 4$
9.	R	$3, 5$
10.	Q	$4, 5$
11.	R	$3, 6$
12	P	$4, 6$
13.	Q	$1, 7$
14.	R	$6, 7$
15.	P	$1, 8$
16.	R	$5, 8$

17.	$\square$	$4, 9$
18.	R	$3, 10$
19.	$\square$	$8, 10$
20.	$\square$	$4, 11$
21.	R	$2, 12$
22.	$\square$	$7, 12$
23.	R	$3, 13$
24.	$\square$	$8, 13$
25.	$\square$	$4, 14$
26.	R	$2, 15$
27.	$\square$	$7, 15$
28.	$\square$	$4, 16$
29.	$\square$	$4, 18$
30.	$\square$	$4, 21$
31.	$\square$	$4, 23$
32.	$\square$	$4, 26$

Risoluzione Lineare



C_0 : clausola iniziale
C_i : clausole centrali
B_i : clausole laterali

Per ogni $i = 0, ..., n-1$
$B_i \in S$, oppure
$B_i = C_j$ per qualche $j < i$

Esempio: $S = \{P \lor Q, \neg P \lor Q, P \lor \neg Q, \neg P \lor \neg Q\}$

$$\dfrac{\dfrac{\dfrac{\dfrac{P \lor Q \qquad \neg P \lor Q}{Q} \qquad P \lor \neg Q}{P} \qquad \neg P \lor \neg Q}{\neg Q} \qquad Q}{\square}$$

Si evita la ridondanza generata dalla risoluzione di conclusioni intermedie con altre conclusioni intermedie: il focus è su S e sugli antenati della clausola centrale.

La risoluzione lineare è completa (rispetto alla refutazione).

Scelta della clausola iniziale: se $S = S' \cup \{C\}$ con S' soddisfacibile, allora S è insoddisfacibile sse esiste una refutazione lineare di S con C come clausola iniziale.

Risoluzione con clausole unitarie (unit resolution)

Ogni risolvente è UNITARIO: almeno una delle due clausole genitrici è una clausola unitaria.

Esempio: $S = \{P \vee Q, \neg P \vee R, \neg Q \vee R, \neg R\}$

1.	$P \vee Q$	S
2.	$\neg P \vee R$	S
3.	$\neg Q \vee R$	S
4.	$\neg R$	S
5.	$\neg P$	$2, 4$
6.	$\neg Q$	$3, 4$

7.	Q	$1, 5$
8.	P	$1, 6$
9.	R	$3, 7$
10.	$\Box$	$6, 7$
11.	R	$2, 8$
12.	$\Box$	$5, 8$

Si ottengono clausole con un numero sempre minore di letterali.

È efficiente ma non completa:

$$\{P \vee Q, \neg P \vee Q, P \vee \neg Q, \neg P \vee \neg Q\}$$

è insoddisfacibile ma non ne esiste una unit-refutation

È completa per clausole Horn (con al massimo un letterale positivo)

Programmazione Logica

Clausole Horn: al massimo un letterale positivo

$$\left.\begin{array}{ll} A & A \\ A \vee \neg B_1 \vee \ldots \vee \neg B_n & A \text{ :- } B_1, \ldots, B_n \end{array}\right\} \quad \textbf{clausole definite}$$

$$\left.\begin{array}{ll} \neg B_1 \vee \ldots \vee \neg B_n & ?\text{- } B_1, \ldots, B_n \\ \square & \end{array}\right\} \quad \textbf{clausole negative}$$

Un **programma logico** è un insieme finito di clausole definite.

Ogni programma logico è soddisfacibile.

risoluzione SLD

risoluzione Lineare con funzione di Selezione per clausole Definite

Regola di calcolo: funzione R che, applicata a un goal G (clausola negativa), riporta un atomo di G (l'atomo selezionato in G)

$$R(?\text{- } B_1, \ldots, B_n) = B_i$$

Risoluzione SLD con regola di calcolo R:

$$\frac{?\text{- } A_1, \ldots, A_j, \ldots, A_m \quad A \text{ :- } B_1, \ldots, B_k}{(?\text{- } A_1, \ldots, A_{j-1}, B_1, \ldots, B_k, A_{j+1}, \ldots, A_m)\theta}$$

dove $R(?\text{- } A_1, \ldots, A_j, \ldots, A_m) = A_j$ e θ è un mgu di A_j e A

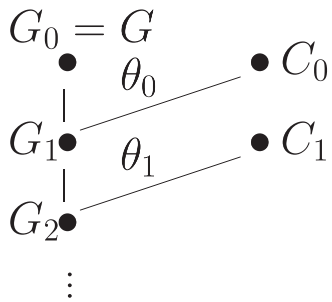
Derivazioni SLD

Sia P un programma logico, G un goal e R una regola di calcolo.

Una SLD-derivazione da $P \cup \{G\}$ con R è costituita da:

- una sequenza di goal $G_0 = G, G_1, G_2, \ldots$
- una sequenza $C_0, C_1, C_2, \ldots$ di varianti di clausole in P (clausole, con variabili rinominate)
- una sequenza $\theta_0, \theta_1, \theta_2, \ldots$ di sostituizioni

tali che ogni G_{i+1} deriva da G_i e C_i con un'applicazione della regola di SLD-risoluzione, con R come regola di calcolo e θ_i come mgu.



$$\cfrac{\cfrac{G_0 = G \quad C_0}{G_1}\,\theta_0 \qquad C_1}{G_2}\,\theta_1 \qquad \vdots$$

$\boxed{\textbf{SLD-refutazione}}$: SLD-derivazione finita con $\square$ come ultimo goal.

Esempio

$P = \{q, p \text{ :- } q\}, \quad G = \text{?- } q, p$

con $R(\text{?- } A_1, ..., A_n) = A_1$ con $R(\text{?- } A_1, ..., A_n) = A_n$

$$\cfrac{\cfrac{\cfrac{\text{?- } q, p \quad q}{\text{?- } p \qquad p \text{ :- } q}}{\text{?- } q \qquad q}}{\square} \qquad\qquad \cfrac{\cfrac{\text{?- } q, p \quad p \text{ :- } q}{\text{?- } q \qquad q}}{\square}$$

Completezza della risoluzione SLD

Se P è un programma logico, G un goal e R una regola di calcolo, e se $P \cup \{G\}$ è insoddisfacibile, allora esiste una SLD-refutazione di $P \cup \{G\}$ con R.

Indipendenza dalla regola di calcolo

Programmazione logica

Supponiamo di avere la seguente KB:

$$\{padre(aldo, piero), \quad padre(piero, ugo),$$
$$\forall x \forall y(\exists z(padre(x, z) \wedge padre(z, y)) \rightarrow nonno(x, y))\}$$

La forma a clausole di questa teoria è il seguente programma logico P:

```
padre(aldo,piero).
padre(piero,ugo).
nonno(X,Y) :- padre(X,Z), padre(Z,Y).
```

Se vogliamo sapere se da KB è derivabile $\exists x \; nonno(x, ugo)$, possiamo cercare una derivazione di $\Box$ da $P \cup \{?\text{-}\; nonno(X, ugo)\}$.

Utilizziamo la regola di calcolo "scelta del primo atomo".

$$\cfrac{\cfrac{\cfrac{?\text{-}\; n(X, ugo) \quad n(X_1, Y) \;\text{:-}\; p(X_1, Z), p(Z, Y)}{?\text{-}\; p(X, Z), p(Z, ugo)} \qquad p(aldo, piero)}{?\text{-}\; p(piero, ugo)} \qquad p(piero, ugo)}{\Box}$$

Quindi $KB \models \exists x \, nonno(x, ugo)$

Domande ASK hanno spesso la forma "quale x è tale che ...?"

Un programma serve per calcolare dei valori e non solo per rispondere SI o NO.

Programmazione logica per rispondere a domande ASK

Informazioni utili si possono estrarre dalle sostituzioni applicate nella SLD-refutazione.

Le sostituzioni applicate nella derivazione data sopra sono:
$\theta_0 = \{X/X_1, ugo/Y\}$
$\theta_1 = \{aldo/X, piero/Z\}$
$\theta_2 = \emptyset$

La loro composizione è:

$$\theta = \theta_0 \circ \theta_1 \circ \theta_2 = \{aldo/X, aldo/X_1, ugo/Y, piero/Z\}$$

Non solo $P \models \exists x\, nonno(x, ugo)$, ma anche

$$P \models nonno(X, ugo)\theta \quad \text{cioè} \quad P \models nonno(aldo, ugo)$$

Quindi la composizione delle sostituzioni utilizzate nella SLD-refutazione fornisce una risposta alla domanda: “chi è un nonno di ugo?”

Sostituzione di risposta (answer substitution)

Sia P un programma logico, $G =$?- $A_1, ..., A_n$ e siano $X = \{x_1, ..., x_k\}$ tutte le variabili in G.

Una sostituzione θ è una *answer substitution corretta* per $P \cup \{G\}$ se:

- $P \models (A_1 \wedge ... \wedge A_n)\theta$,
  cioè $\forall P \models \forall((A_1 \wedge ... \wedge A_n)\theta)$
- per ogni $t/y \in \theta$, $y \in X$

Supponiamo ora che $P \cup \{G\}$ sia insoddisfacibile.

Sia R una regola di calcolo e sia $\theta_0, \theta_1, ..., \theta_n$ la sequenza di mgu usata in una SLD-refutazione di $P \cup \{G\}$.

Una answer substitution calcolata da R è la restrizione di $\theta_0 \circ \theta_1 \circ ... \circ \theta_n$ alle variabili di G.

Correttezza della risoluzione SLD

Siano P un programma logico, G un goal e R una regola di calcolo. Ogni answer substitution calcolata da R per $P \cup \{G\}$ è una answer substitution corretta.

Esempio

```
(1) colore(X,Y) :- vicino(X,Z), colore(Z,Y)
(2) colore(c1,blu)
(3) colore(c2,rosso)
(4) vicino(c,c1)
(5) vicino(c,c2)
```

$$
\cfrac{\cfrac{\cfrac{\texttt{?- colore}(\texttt{c},\texttt{Y}) \qquad \begin{array}{l}\texttt{colore}(\texttt{X},\texttt{Y1}) :- \\ \qquad \texttt{vicino}(\texttt{X},\texttt{Z}),\texttt{colore}(\texttt{Z},\texttt{Y1})\end{array}}{\texttt{?- vicino}(\texttt{c},\texttt{Z}),\texttt{colore}(\texttt{Z},\texttt{Y})}\,\theta_0 \qquad \texttt{vicino}(\texttt{c},\texttt{c1})}{\texttt{?- colore}(\texttt{c1},\texttt{Y})}\,\theta_1 \qquad \texttt{colore}(\texttt{c1},\texttt{blu})}{\square}\,\theta_2
$$

$\theta_0 = \{\texttt{c}/\texttt{X}, \texttt{Y}/\texttt{Y1}\}$, $\theta_1 = \{\texttt{c1}/\texttt{Z}\}$ $\theta_2 = \{\texttt{blu}/\texttt{Y}\}$

$$\theta_0 \circ \theta_1 \circ \theta_2 = \{\texttt{c}/\texttt{X}, \texttt{blu}/\texttt{Y1}, \texttt{c1}/\texttt{Z}, \texttt{blu}/\texttt{Y}\}$$

Restrizione alle variabili in ?- $\texttt{colore}(\texttt{c}, \texttt{Y})$:

$$\{\texttt{blu}/\texttt{Y}\}$$

(una derivazione diversa darebbe $\{\texttt{rosso}/\texttt{Y}\}$ come answer substitution)

Correttezza: ogni answer substitution calcolata da R per $P \cup \{G\}$ è una answer substitution corretta.

Per clausole generali questo non vale: non sempre esiste una answer substitution corretta

$$P = \{colore(c1, blu) \vee colore(c1, rosso)\}$$
$$G = \text{?- } colore(c1, X)$$

$P \models \exists X\, colore(c1, X)$,
ma non esiste alcuna sostituzione θ tale che $P \models colore(c1, X)\theta$.

$$\dfrac{\dfrac{\neg colore(c1, X) \quad colore(c1, blu) \vee colore(c1, rosso)}{colore(c1, rosso)}\,\theta_0 \qquad \neg colore(c1, X)}{\square}\,\theta_1$$

$$\theta_0 = \{blu/X\} \qquad \theta_1 = \{rosso/X\}$$

Sistemi di programmazione logica basati sulla risoluzione SLD

- scelta di una regola di calcolo
- scelta di una strategia di ricerca della refutazione

La regola di calcolo può influenzare drasticamente la dimensione e la struttura dell'albero di derivazione, ma non la completezza.

La strategia di ricerca può compromettere la completezza.

Prolog

- $R(?\text{-}\ A_1, ..., A_n) = A_1$
- strategia di ricerca in profondità:
  INCOMPLETEZZA

Esercizi

1. Dimostrare per risoluzione che se una relazione R è simmetrica, transitiva e totale, allora è anche riflessiva. (Una relazione R è totale se per ogni x e y, xRy oppure yRx).

2. Dimostrare che

$$\exists x(p(x) \wedge \forall y(q(y) \rightarrow r(x,y)) \vdash_{RES} \forall y(q(y) \rightarrow \exists x(p(x) \wedge r(x,y))$$

3. Dimostrare che

$$\begin{aligned} &\forall x \forall y \forall z\, (p(x,y) \wedge p(y,z) \rightarrow p(x,z)), \\ &\qquad \forall x_1 \forall y_1 \exists z_1\, (p(x_1,z_1) \wedge p(z_1,y_1)) \;\vdash_{RES} \forall x \forall y\, p(x,y) \end{aligned}$$

4. Dimostrare che:

$$\begin{aligned} &arc(a,b),\; arc(b,c),\; arc(c,d), \\ &\forall x \forall y (arc(x,y) \rightarrow path(x,y,f(x,y))) \\ &\forall x \forall y \forall z\, (\exists w (arc(x,w) \wedge path(w,y,z)) \rightarrow path(x,y,f(x,z))) \\ &\qquad\qquad\qquad\qquad \vdash_{RES}\; \exists x\, path(a,d,x) \end{aligned}$$

Per ciascuno degli esercizi precedenti, considerare inoltre l'insieme di clausole ottenuto per risolvere l'esercizio.

- Quali raffinamenti della risoluzione sono completi per l'insieme di clausole che si ottiene?
- Le clausole ottenute sono tutte clausole Horn?
- L'insieme di clausole si può vedere come un programma logico e un goal? In caso affermativo, riscrivere le clausole nella notazione della programmazione logica ed costruire una dimostrazione SLD con la regola di calcolo "scelta del primo atomo". Calcolare inoltre la sostituzione di risposta che si ottiene da tale dimostrazione.