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

Dimostrazione per saturazione di livelli

- Livello 0: tutte le clausole dell'insieme S da refutare
- Livello $k > 0$: tutti i risolventi di una clausola del livello $k - 1$ contro una clausola di un livello $i < k$ $(0, 1, \ldots, k - 1)$.

Esempio: dimostrazione per saturazione di livelli

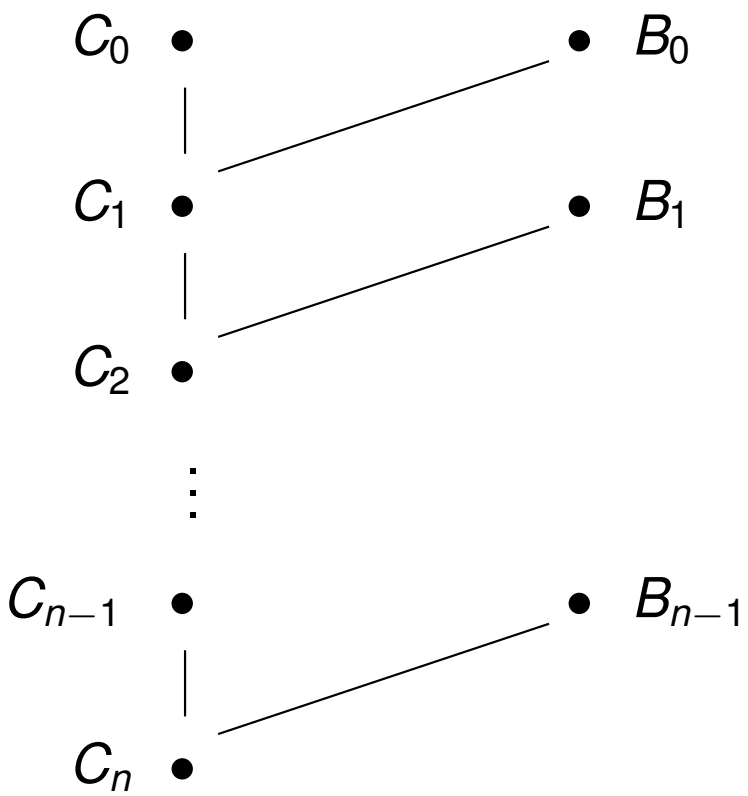
$$S = \{P \vee Q,\ \neg P \vee R,\ \neg Q \vee R,\ \neg R\}$$

Livello 0		
1.	$\boldsymbol{P \vee Q}$	in S
2.	$\boldsymbol{\neg P \vee R}$	in S
3.	$\boldsymbol{\neg Q \vee R}$	in S
4.	$\boldsymbol{\neg R}$	in S
Livello 1		
5.	$\boldsymbol{Q \vee R}$	$1, 2$
6.	$P \vee R$	$1, 3$
7.	$\neg P$	$2, 4$
8.	$\neg Q$	$3, 4$

Livello 2		
9.	$\boldsymbol{R}$	$3, 5$
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	$5, 8$
Livello 3		
17.	$\square$	$4, 9$

Le clausole 6-8 e 10-16 sono inutili.

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 $j < i$

Ad ogni passo di risoluzione una delle due clausole genitrici è quella ottenuta al passo precedente.

Punti di scelta: Clausola iniziale e clausole laterali

Risoluzione lineare: esempi

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

$$\dfrac{\dfrac{\dfrac{P \vee Q \quad \neg P \vee R}{Q \vee R} \qquad \neg Q \vee R}{R} \qquad \neg R}{\square}$$

Risoluzione lineare: esempi

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

$$\dfrac{\dfrac{\dfrac{P \vee Q \quad \neg P \vee R}{Q \vee R} \qquad \neg Q \vee R}{R} \qquad \neg R}{\square}$$

$$\dfrac{\dfrac{\dfrac{\neg R \quad \neg P \vee R}{\neg P} \qquad P \vee Q}{Q} \qquad \neg Q \vee R}{R} \qquad \neg R}{\square}$$

Risoluzione lineare: esempi

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

$$\dfrac{\dfrac{\dfrac{P \vee Q \quad \neg P \vee R}{Q \vee R} \qquad \neg Q \vee R}{R} \qquad \neg R}{\square}$$

$$\dfrac{\dfrac{\dfrac{\neg R \quad \neg P \vee R}{\neg P} \qquad P \vee Q}{Q} \qquad \neg Q \vee R}{R} \qquad \neg R}{\square}$$

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

$$\dfrac{\dfrac{\dfrac{\dfrac{P \vee Q \quad \neg P \vee Q}{Q} \qquad P \vee \neg Q}{P} \qquad \neg P \vee \neg Q}{\neg Q} \qquad Q}{\square}$$

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.

Proprietà della risoluzione lineare

La risoluzione lineare è completa (rispetto alla refutazione):

Se S è insoddisfacibile allora esiste una refutazione lineare di S, cioè esiste un modo di scegliere una clausola iniziale e, passo passo, le clausole laterali, in modo da ottenere una derivazione lineare di $\square$.

Questo non significa che qualsiasi sia la clausola iniziale e qualsiasi sia la scelta delle clausole laterali si può arrivare a $\square$.

Proprietà della risoluzione lineare

La risoluzione lineare è completa (rispetto alla refutazione):

Se S è insoddisfacibile allora esiste una refutazione lineare di S, cioè esiste un modo di scegliere una clausola iniziale e, passo passo, le clausole laterali, in modo da ottenere una derivazione lineare di $\square$.

Questo non significa che qualsiasi sia la clausola iniziale e qualsiasi sia la scelta delle clausole laterali si può arrivare a $\square$.

Teorema che può fornire una guida (semantica) alla **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 (con un solo letterale).

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

Livello 0			
1.	$\boldsymbol{P \vee Q}$	S	
2.	$\boldsymbol{\neg P \vee R}$	S	
3.	$\boldsymbol{\neg Q \vee R}$	S	
4.	$\boldsymbol{\neg R}$	S	
Livello 1			
5.	$\boldsymbol{\neg P}$	$2, 4$	
6.	$\boldsymbol{\neg Q}$	$3, 4$	
Livello 2			
7.	$\boldsymbol{Q}$	$1, 5$	
8.	P	$1, 6$	
Livello 3			
9.	R	$3, 7$	
10.	$\square$	$6, 7$	

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

Clausole Horn

Clausola Horn: ha al massimo un letterale positivo

$$\frac{\text{Horn}}{\begin{array}{c} P \\ \neg P \\ P \vee \neg Q \vee \neg R \\ \neg Q \vee \neg R \end{array}}$$

Clausole Horn

Clausola Horn: ha al massimo un letterale positivo

Horn	non Horn
P	$P \vee Q$
$\neg P$	$P \vee Q \vee \neg R \vee \ldots$
$P \vee \neg Q \vee \neg R$	
$\neg Q \vee \neg R$	

La risoluzione unitaria è **completa per clausole Horn**: se S è un insieme insoddisfacibile di clausole Horn, allora esiste una refutazione unitaria di S.

Programmazione Logica

Le clausole Horn si possono suddividere in due categorie:

clausole definite $\begin{cases} A \\ A \vee \neg B_1 \vee \ldots \vee \neg B_n \end{cases}$

clausole negative $\begin{cases} \neg B_1 \vee \ldots \vee \neg B_n \\ \square \end{cases}$

Programmazione Logica

Le clausole Horn si possono suddividere in due categorie:

clausole definite

$$\begin{cases} A & A \\ A \vee \neg B_1 \vee \ldots \vee \neg B_n & A \text{ :- } B_1, \ldots, B_n \end{cases}$$

clausole negative

$$\begin{cases} \neg B_1 \vee \ldots \vee \neg B_n & \text{?- } B_1, \ldots, B_n \\ \square \end{cases}$$

Notazione della programmazione logica:

$$A \vee \neg B_1 \vee \ldots \vee \neg B_n \quad \leftrightarrow \quad B_1 \wedge \cdots \wedge B_n \rightarrow A \quad \approx \quad \begin{array}{l} A \leftarrow B_1 \wedge \cdots \wedge B_n \\ A \text{ :- } B_1, \ldots, B_n \end{array}$$

$$\neg B_1 \vee \ldots \vee \neg B_n \quad \leftrightarrow \quad B_1 \wedge \cdots \wedge B_n \rightarrow \bot \quad \approx \quad \begin{array}{l} \bot \leftarrow B_1 \wedge \cdots \wedge B_n \\ \text{?- } B_1, \ldots, B_n \end{array}$$

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

Ogni programma logico è soddisfacibile
(nell'interpretazione $\mathcal{M}$ tale che, per ogni simbolo di predicato R^n, $\mathcal{M}(R) = D^n$).

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)

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

Regola di risoluzione SLD con regola di calcolo R:

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

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

L'ordine dei letterali è importante.

Derivazioni SLD

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

Una derivazione SLD 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 in P, 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.

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

$$\vdots$$

Refutazione SLD : SLD-derivazione finita con $\square$ come ultimo goal.

Esempio

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

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

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

$$\square$$

Esempio

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

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

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

$$\square$$

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

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

$$\square$$

Completezza della risoluzione SLD

Se P è un programma logico, G un goal e R una **qualsiasi** 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

Esempio (2)

Sia KB (*knowledge base*) l'insieme di formule:

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

La forma a clausole di KB 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 $\square$ da $P \cup \{?\text{-}\ nonno(X, ugo)\}$.

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

$$\dfrac{\dfrac{\dfrac{?\text{-}\ n(X, ugo) \quad n(X_1, Y) :\text{-}\ p(X_1, Z), p(Z, Y)}{?\text{-}\ p(X, Z), p(Z, ugo)} \quad p(aldo, piero)}{?\text{-}\ p(piero, ugo)} \quad p(piero, ugo)}{\square}$$

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

Programmazione logica

Ma: un programma serve per calcolare dei valori e non solo per rispondere SI o NO (chi è un nonno di ugo?).

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:

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

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

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

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

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 **sostituzione di risposta 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 $t/y \in \theta$, $y \in X$
  (le variabili sostituite in θ sono **solo** quelle che occorrono nel goal)

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 sostituzione di risposta 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 sostituzione di risposta calcolata da R per $P \cup \{G\}$ è corretta.

Esempio:

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

$$\dfrac{\dfrac{\dfrac{\text{?- } col(c, Y) \qquad col(X, Y1) :- vic(X, Z), col(Z, Y1)}{\text{?- } vic(c, Z), col(Z, Y)}\ \theta_0 \qquad vic(c, c1)}{\text{?- } col(c1, Y)}\ \theta_1 \qquad col(c1, blu)}{\square}\ \theta_2$$

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

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

Restrizione alle variabili del goal: $\{blu/Y\}$

Da una derivazione diversa si otterrebbe $\{rosso/Y\}$ come sostituzione di risposta. Sono entrambe corrette: da P si può derivare sia che c è rosso, sia che è blu.

Perché la restrizione alle clausole Horn?

Per clausole generali non sempre esiste una sostituzione di risposta 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 che

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

2. Dimostrare che

$$\forall x \forall y \forall z\,(p(x,y) \wedge p(y,z) \rightarrow p(x,z)),$$
$$\forall x_1 \forall y_1 \exists z_1\,(p(x_1,z_1) \wedge p(z_1,y_1)) \;\vdash_{Res}\; \forall x \forall y\, p(x,y)$$

3. Dimostrare che:

$$arc(a,b),\ arc(b,c),\ arc(c,d),$$
$$\forall x \forall y(arc(x,y) \rightarrow path(x,y,f(x,y)))$$
$$\forall x \forall y \forall z\,(\exists w(arc(x,w) \wedge path(w,y,z)) \rightarrow path(x,y,f(x,z)))$$
$$\vdash_{Res}\ \exists x\, path(a,d,x)$$

Esercizi (segue)

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 refutazione SLD con la regola di calcolo "scelta del primo atomo". Calcolare inoltre la sostituzione di risposta che si ottiene da tale dimostrazione.