

ALGORITMO DI UNIFICAZIONE E RISOLUZIONE PREDICATIVA

Versione del 06/11/09

Consideriamo la formula

$$F = \forall x \forall y r(fx, y) \wedge \forall u \exists v \neg r(u, v).$$

La formula F è chiaramente insoddisfacibile; la sua forma clausale è data da un insieme S con due clausole $C_1 = \{r(fx, y)\}$ e $C_2 = \{\neg r(u, gu)\}$. Nessun passo di risoluzione proposizionale, però, ci porta alla clausola vuota.

Consideriamo ora una *sostituzione* σ che trasforma variabili in termini. Possiamo applicare la sostituzione ad letterale o ad una clausola sostituendo le variabili x del letterale o della clausola con i rispettivi termini $\sigma(x)$. Il letterale e la clausola così ottenuti si denotano con $\sigma(\ell), \sigma(C)$, ma anche con $\ell\sigma$ $C\sigma$.

Ad esempio, se σ è la sostituzione tale che $\sigma(y) = gfx\sigma(u) = fx$ e σ lascia ferme le altre variabili, si ha:

$$C_1\sigma = \{r(fx, gfx)\}, \quad C_2\sigma = \{\neg r(fx, gfx)\}$$

e dall'insieme $\{C_1\sigma, C_2\sigma\}$ con un singolo passo di risoluzione proposizionale si ottiene la clausola vuota. Inoltre, come vedremo, $C \models C\sigma$, quindi il procedimento sopra descritto ci permette di stabilire che l'insieme $S = \{C_1, C_2\}$ è insoddisfacibile.

Questo esempio si generalizza: un passo di risoluzione predicativo consiste nella ricerca di una sostituzione σ che *renda possibile* un passo proposizionale, cioè tale che esistano $\ell_1, \dots, \ell_k$ in C_1 e $\ell'_1, \dots, \ell'_h$ in C_2 con $\ell_1\sigma = \ell_k\sigma = \overline{\ell'_1\sigma} = \overline{\ell'_h\sigma} = p(\bar{t})$. Un tale σ si dice *unificatore* dell'insieme $\{\ell_1, \dots, \ell_k, \overline{\ell'_1}, \dots, \overline{\ell'_h}\}$.

Ex:

$$F = \forall x \forall y (q(x) \vee q(fy) \vee \neg r(x, y)) \wedge \forall u \forall v (\neg q(u) \vee \neg q(fv) \vee r(a, a))$$

$$C_1 = \{q(x), q(fy), \neg r(x, y)\}, \quad C_2 = \{\neg q(u), \neg q(fv), r(a, a)\}.$$

Se $\sigma(x) = fy, \sigma(u) = fy, \sigma(v) = y$, otteniamo

$$C_1\sigma = \{q(fy), \neg r(fy, y)\}, \quad C_2\sigma = \{\neg q(fy), r(a, a)\},$$

da cui, per risoluzione proposizionale si può ottenere la clausola

$$C_3 = \{\neg r(fy, y)\}, r(a, a)\}.$$

Quindi,

$$C_1, C_2 \models C_3.$$

Dobbiamo prima però risolvere il problema: dati t_1, t_2 termini, possiamo trovare un unificatore?

1. ALGORITMO DI UNIFICAZIONE E REGOLA DI RISOLUZIONE PREDICATIVA

1.1. Sostituzioni e sistemi di Herbrand.

Definizione 1.1. Una sostituzione è una funzione σ dall'insieme delle variabili del linguaggio ai termini tale che l'insieme $\{x \in Var : x \neq \sigma(x)\}$ è finito.

L'insieme $\{x \in Var : x \neq \sigma(x)\}$ viene detto *dominio* della sostituzione σ . L'immagine $\sigma(x)$ di una variabile x tramite una sostituzione σ viene anche denotata con $x\sigma$. Una sostituzione σ con dominio $\{x_1, \dots, x_n\}$ e $\sigma(x_i) = t_i$ per $i = 1, \dots, n$ viene anche denotata da $\{x_1|t_1, \dots, x_n|t_n\}$. Ad esempio indichiamo con $\{x|fa, y|w\}$ la sostituzione che manda x in fa , y in w e lascia ferme tutte le variabili diverse da x e y .

Una sostituzione viene estesa in maniera naturale all'insieme dei termini del linguaggio ponendo

$$\sigma(c) = c, \quad \sigma(f(t_1, \dots, t_n)) = f(\sigma(t_1), \dots, \sigma(t_n)).$$

L'immagine $\sigma(t)$ di un termine t tramite una sostituzione σ viene denotata anche con $t\sigma$.

Date due sostituzioni σ e τ indichiamo con $\sigma\tau$ la funzione che ad ogni variabile x associa il termine $(x\sigma)\tau$ (che in notazione funzionale è uguale a $\tau(\sigma(x))$). Verificare per esercizio che tale funzione è, in effetti, una sostituzione, cioè che l'insieme $\{x : x \neq x\sigma\tau\}$ è finito.

Una sostituzione η si dice un'istanza di una sostituzione σ se esiste una sostituzione τ tale che $\sigma\tau = \eta$. Ad esempio, se

$$\sigma = \{x|fu, y|g(h(gv, u)), z|w\}, \quad \eta = \{x|fga, y|g(h(gfa, ga), z|w\},$$

la sostituzione η è un'istanza della sostituzione σ perché se $\tau = \{u|ga, v|fa\}$ si ha $\sigma\tau = \eta$.

Definizione 1.2. Un sistema di Herbrand è un insieme finito di coppie di termini

$$(t_1, s_1), \dots, (t_n, s_n).$$

Una soluzione (detta anche unificatore) di un sistema di Herbrand $(t_1, s_1), \dots, (t_n, s_n)$ è una sostituzione σ tale che $t_i\sigma = s_i\sigma$, per ogni $i = 1, \dots, n$. Un unificatore di un sistema di Herbrand si dice di massima generalità (m.g.u.) se ogni altro unificatore η del sistema è istanza di σ . Due sistemi di Herbrand si dicono equivalenti se hanno le stesse soluzioni.

Un sistema di Herbrand viene anche denotato con

$$\begin{cases} t_1 = s_1; \\ \cdot \\ \cdot \\ \cdot \\ t_n = s_n \end{cases}$$

Lemma 1.1. Se η unifica il sistema

$$\begin{cases} x_1 = r_1; \\ \cdot \\ \cdot \\ \cdot \\ x_n = r_n \end{cases}$$

dove le variabili x_i sono a due a due distinte, allora la sostituzione $\{x_1/r_1, \dots, x_n/r_n\}\eta$ coincide con η .

In particolare, sotto l'ipotesi che η unifichi il sistema di cui sopra abbiamo che per termini t ed s qualsiasi vale

$$\eta \text{ unifica } t \text{ con } s \Leftrightarrow \eta \text{ unifica } t\{x_1/r_1, \dots, x_n/r_n\} \text{ con } s\{x_1/r_1, \dots, x_n/r_n\}.$$

Dimostrazione. Basta verificare che le funzioni $\{x_1/r_1, \dots, x_n/r_n\}\eta$ e η hanno lo stesso valore su ogni variabile y . Se $y = x_i$ per un qualche i abbiamo

$$x_i\{x_1/r_1, \dots, x_n/r_n\}\eta = r_i\eta = x_i\eta.$$

Se $y \neq x_i$ per ogni i si ha

$$y\{x_1/r_1, \dots, x_n/r_n\}\eta = y\eta.$$

Dato un termine r qualsiasi si avrà allora:

$$r\eta = r(\{x_1/r_1, \dots, x_n/r_n\}\eta) = (r\{x_1/r_1, \dots, x_n/r_n\})\eta;$$

quindi, dati due termini t, s si avrà

$$t\eta = s\eta \Leftrightarrow t\{x_1/r_1, \dots, x_n/r_n\}\eta = s\{x_1/r_1, \dots, x_n/r_n\}\eta.$$

□

1.2. Algoritmo di unificazione. Dato un sistema di Herbrand $E = \{t_1, s_1), \dots, (t_n, s_n)\}$ vogliamo decidere in maniera algoritmica se il sistema E ammette almeno una soluzione, e, in tal caso, trovare tutte le possibili soluzioni di E .

Notiamo per prima cosa che, per un tipo particolare di sistemi di Herbrand, i cosiddetti sistemi *risolti*, il problema ha una semplice soluzione.

Definizione 1.3. *Un sistema di Herbrand del tipo*

$$\begin{cases} x_1 = r_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n = r_n \end{cases}$$

tale che se $x_i \neq x_j$ per $i \neq j$ e le variabili x_i non compaiono a destra delle equazioni del sistema, si dice risolto

Lemma 1.2. *Dato un sistema risolto*

$$\begin{cases} x_1 = r_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n = r_n \end{cases}$$

la sostituzione $\sigma = \{x_1/r_1, \dots, x_n/r_n\}$ è un m.g.u. del sistema.

Dimostrazione. La sostituzione $\sigma = \{x_1/r_1, \dots, x_n/r_n\}$ unifica il sistema perché $x_i\sigma = r_i$ e $r_i\sigma = r_i$ (infatti r_i non contiene variabili del dominio di σ). Quindi σ è un unificatore del sistema.

Mostriamo ora che σ è un m.g.u. del sistema: se η unifica il sistema, allora, applicando il lemma 1.1 otteniamo

$$\sigma\eta = \{x_1/r_1, \dots, x_n/r_n\}\eta = \eta.$$

Quindi η si ottiene come istanza di σ (tramite lo stesso η !).

□

L'algoritmo di unificazione prende in input un sistema di Herbrand e cerca di trasformarlo in un sistema equivalente e risolto. Un passo dell'algoritmo consiste nel scegliere una coppia (un'equazione) del sistema di Herbrand e operare a seconda della forma dell'equazione nel modo seguente:

- (1) se è del tipo $c = c$, $x = x$, si elimina l'equazione e si prosegue con il sistema così ottenuto;
- (2) se è del tipo $f(t_1, \dots, t_n) = g(r_1, \dots, r_k)$, con f diverso da g oppure $c = d$ con c diverso da d oppure $c = f(t_1, \dots, t_n)$ oppure $f(t_1, \dots, t_n) = c$ per c, d costanti, l'algoritmo si ferma con la risposta *non unificabile*;
- (3) se è del tipo $x = t$ con x che compare in t e t diversa da una variabile, l'algoritmo si ferma con la risposta *non unificabile*;
- (4) se è del tipo $f(t_1, \dots, t_n) = f(s_1, \dots, s_n)$ si sostituisce l'equazione con le n equazioni

$$\begin{cases} t_1 = s_1 \\ . \\ . \\ . \\ t_n = s_n, \end{cases}$$

si prosegue con il sistema così ottenuto;

- (5) se è del tipo $t = x$ con t diverso da una variabile, si sostituisce l'equazione con $x = t$;
- (6) se è del tipo $x = t$ con x che non compare in t , si sostituisce ogni altra equazione $r = s$ del sistema con l'equazione $r\{x|t\} = s\{x|t\}$;

l'algoritmo si ferma con risposta *unificabile* quando non è possibile fare più alcun passo, mentre si ferma con risposta *non unificabile* quando si trova un'equazione non unificabile (casi 2,3 della lista)

Notiamo che l'algoritmo si ferma con risposta *unificabile* se e solo se il sistema a cui si è arrivati è un sistema risolto.

Lemma 1.3. *Correttezza e Completezza dell'algoritmo di unificazione: l'algoritmo di unificazione si ferma dopo un numero finito di passi partendo da un qualsiasi sistema di Herbrand. Il sistema in input è unificabile se e solo se l'algoritmo raggiunge un sistema risolto, e, in questo caso il sistema di partenza ha lo stesso m.g.u. del sistema finale,*

Dimostrazione. Ci basta far vedere che l'algoritmo termina sempre dopo un numero finito di passi e che ogni passo dell'algoritmo se non si ferma con risposta *non unificabile* trasforma il sistema in un sistema equivalente. Notiamo inoltre che nei passi 2,3 il sistema raggiunto è effettivamente non unificabile.

Terminazione. Sia associata ad ogni sistema di Herbrand E la terna (a_0, a_1, a_2) , dove:

a_0 è il numero di variabili non risolte del sistema (una variabile si dice non risolta se appare a destra di un'equazione o in più di un'equazione o nella forma $x = x$).

a_1 è il numero di occorrenze di simboli funzionali o costanti a sinistra delle equazioni;

a_2 è il numero di equazioni del tipo $x = x$.

Quindi ad ogni sistema è associata una terna di numeri naturali. Queste terne possono essere ordinate tramite l'ordine lessicografico così definito:

$$(a, b, c) < (a', b', c') \Leftrightarrow (a < a') \vee (a = a' \wedge b < b') \vee (a = a' \wedge b = b' \wedge c < c').$$

In questo ordine non esistono catene infinite discendenti. Per dimostrare la terminazione dell'algoritmo di risoluzione basta allora notare che ad ogni passo dell'algoritmo la terna associata diminuisce nell'ordine lessicografico.

Correttezza Ogni passo trasforma il sistema in uno equivalente. Questo è ovvio per passi di tipo 1, 2, 3, 4, 5. Per quanto riguarda i passi di tipo 6, notiamo che, grazie al Lemma 1.1, un sistema del tipo

$$\begin{cases} x = t; \\ t_1 = s_1 \\ \cdot \\ \cdot \\ t_n = s_n \end{cases}$$

ha le stesse soluzioni del sistema

$$\begin{cases} x = t; \\ t_1\{x/t\} = s_1\{x/t\} \\ \cdot \\ \cdot \\ t_n\{x/t\} = s_n\{x/t\}. \end{cases}$$

□

1.2.1. *Esercizi.* Applicare l'algoritmo di unificazione ai seguenti sistemi. Quando possibile, trovare un m.g.u. del sistema.

$$\begin{cases} h(x, fy) = v \\ w = fh(a, ffx) \\ h(a, ffu) = v \\ fh(a, ffa) = w. \end{cases}$$

$$\begin{cases} f(x, f(hx, a)) = f(y, z) \\ z = f(y, a) \end{cases}$$

$$\begin{cases} g(y) = x \\ f(x, hx, y) = f(gz, w, z) \end{cases}$$

2. RISOLUZIONE

Come vedremo, l'algoritmo di unificazione viene usato all'interno della regola di risoluzione predicativa: dato un letterale $p(t_1, \dots, t_n)$ e una sostituzione σ si definisce

$$\sigma(p(t_1, \dots, t_n)) := p(\sigma(t_1), \dots, \sigma(t_n)), \quad \sigma(\neg p(t_1, \dots, t_n)) := \neg p(\sigma(t_1), \dots, \sigma(t_n)).$$

Un insieme di letterali $\{\ell_1, \dots, \ell_k\}$ si dice *unificabile* se esiste una sostituzione σ tale che $\sigma(\ell_1) = \dots = \sigma(\ell_k)$.

Data una clausola $C = \{\ell_1, \dots, \ell_k\}$ e una sostituzione σ denotiamo con $C\sigma$ la clausola

$$C\sigma = \{\sigma(\ell_1), \dots, \sigma(\ell_k)\}.$$

Si noti che la formula che corrisponde alla clausola C è $\forall \bar{x}(\ell_1 \vee \dots \vee \ell_k)$, dove $\bar{x}$ contiene tutte le variabili libere di $(\sigma(\ell_1), \dots, \sigma(\ell_k))$, mentre la formula che corrisponde alla clausola $C\sigma$ è $\forall \bar{z}(\sigma(\ell_1) \vee \dots \vee \sigma(\ell_k))$.

$\dots \vee \sigma(\ell_k))$, dove $\bar{z}$ contiene tutte le variabili libere di $(\sigma(\ell_1) \vee \dots \vee \sigma(\ell_k))$. Usando il Lemma di sostituzione, si dimostra (vedi gli esercizi alla fine del paragrafo) che vale

$$C \models C\sigma.$$

2.1. Regola di Risoluzione Predicativa. Siano

$$C_1 = \{\ell_1, \dots, \ell_k\} \cup G, \quad C_2 = \{\ell'_1, \dots, \ell'_h\} \cup H,$$

due clausole predicative **a variabili separate** tali che l'insieme

$$\{\ell_1, \dots, \ell_k, \bar{\ell}'_1, \dots, \bar{\ell}'_h\}$$

sia unificabile. Se σ è un m.g.u di tale insieme, allora il risolvente di C_1, C_2 rispetto a σ è la clausola

$$Ris_\sigma(C_1, C_2) = G\sigma \cup H\sigma.$$

Se le clausole C_1, C_2 non sono a variabili separate, prima di applicare la regola di risoluzione si avrà l'accortezza di cambiare il nome delle variabili di una delle due clausole, in modo da ottenere clausole a variabili separate.

Esempio: Consideriamo le clausole

$$C_1 = \{p(x, fx), p(x, z), p(y, y), \neg q(y, x)\}, \quad C_2 = \{\neg p(fh(z, a), fu), \neg p(u, v), r(u), \neg q(u, v)\}.$$

Prima di cercare di risolvere tali clausole, le rinominiamo per ottenere clausole a variabili separate (che chiamiamo ancora C_1, C_2)

$$C_1 = \{p(x, fx), p(x, s), p(y, y), \neg q(y, x)\}, \quad C_2 = \{\neg p(fh(z, a), fu), \neg p(u, v), r(u), \neg q(u, v)\}.$$

Consideriamo i letterali $p(x, fx), p(x, s)$ in C_1 e i letterali $\neg p(fh(z, a), fu), \neg p(u, v)$ in C_2 . L'insieme

$$\{p(x, fx), p(x, s), p(fh(z, a), fu), p(u, v)\}$$

è unificabile. Per accorgersene, impostiamo un sistema di Herbrand (in cui abbiamo saltato certe equazioni cancellabili quali $x = x$ o ripetute):

$$\begin{cases} fx = s \\ x = fh(z, a) \\ fx = fu \\ x = u \\ fx = v \end{cases}$$

Utilizzando l'algoritmo di unificazione si ottiene in un numero finito di passi il sistema risolto

$$\begin{cases} s = ffh(z, a) \\ u = fh(z, a) \\ x = fh(z, a) \\ v = ffh(z, a) \end{cases}$$

che fornisce il seguente m.g.u.

$$\sigma = \{s/fh(z, a), u/fh(z, a), x/fh(z, a), v = ffh(z, a)\}.$$

Otteniamo quindi per risoluzione predicativa la clausola

$$Ris_\sigma(C_1, C_2) = \{p(y, y), \neg q(x, y)\}\sigma \cup \{r(u), \neg q(u, v)\}\sigma =$$

$$= \{p(y, y), \neg q(fh(z, a), y), r(fh(z, a)), \neg q(fh(z, a), fh(z, a))\}.$$

Notiamo inoltre che in questo esercizio non è possibile cancellare con risoluzione più letterali di quelli scelti: l'insieme di letterali

$$\{p(x, fx), p(x, s), p(y, y), p(fh(z, a), fu), p(u, v)\}$$

non è infatti unificabile (si dovrebbe risolvere un sistema contenente il sottosistema

$$\begin{cases} x = y \\ fx = y, \end{cases}$$

che non è unificabile.

3. CORRETTEZZA E COMPLETEZZA DI RISOLUZIONE PREDICATIVA

Come nel caso proposizionale, un albero di risoluzione della clausola C a partire da clausole in S è un albero binario etichettato, in cui la radice è etichettata da C , le foglie sono etichettate da (rinomine) di clausole in S e un nodo interno è etichettato da un risolvente predicativo delle clausole che etichettano i due figli del nodo.

Come nel caso proposizionale (ma con una sostanziale differenza, vedi Osservazione), possiamo utilizzare questi alberi di risoluzione per stabilire l'insoddisfacibilità di un insieme di clausole S , cercando un albero di risoluzione che abbia radice etichettata con la clausola vuota e foglie etichettate da (rinomine di) clausole in S . Un tale albero si chiama *albero di refutazione* da S .

La possibilità di rinominare le clausole di S è essenziale per la completezza del metodo di risoluzione. In altri termini, esistono insiemi di clausole insoddisfacibili per cui non esiste alcun albero di risoluzione della clausola vuota a partire da clausole in S (senza rinomina). Un esempio è dato dall'insieme di clausole $S = \{C_1, C_2\}$ dove $C_1 = \{p(x)\}$, $C_2 = \{\neg p(fx)\}$. Se rinominiamo C_1 sostituendo la variabile x con y otteniamo la clausola vuota da tale rinomina e da C_2 , con un singolo passo di risoluzione. Invece non esiste alcun albero di risoluzione della clausola vuota dalle clausole in S non rinominate, perché non esiste nessuna sostituzione che unifichi i termini x e $f(x)$.

Come nel caso proposizionale, l'esistenza di alberi di refutazione da S di altezza n è legata alla presenza della clausola vuota all'interno degli insiemi risolvanti:

Definizione 3.1. *Dato un insieme di clausole S definiamo $Ris^n(S)$ induttivamente come segue:*

$$Ris^0(S) = S, \quad Ris^{n+1}(S) = Ris^n(S) \cup \{C : C \text{ è un risolvente di (rimnomine) di due clausole in } Ris^n(S)\}.$$

Diremo che C deriva per risoluzione da S se esiste un numero n tale che $C \in Ris^n(S)$.

Si dimostra facilmente per induzione su n che $C \in Ris^n(S)$ se e solo se esiste un albero di risoluzione di altezza minore o uguale ad n , con foglie etichettate da clausole in S e radice etichettata da C .

Teorema 3.1. *(Correttezza del metodo di risoluzione)*

$$\text{Se } C \text{ deriva per risoluzione da } S \Rightarrow S \models C.$$

Dimostrazione. Sia n il più piccolo numero per cui $C \in Ris^n(S)$. Dimostreremo che $S \models C$ per induzione su n .

Base $n = 0$: in questo caso abbiamo $C \in S$ e $S \models C$ è vero.

Passo induttivo: supponendo che $C \in \text{Ris}^{n+1}(S)$. Allora $C \in \text{Ris}^n(S)$ e $S \models C$ segue per ipotesi induttiva oppure esistono $C_1, C_2 \in \text{Ris}^n(S)$ tali che C è risolvente di C_1, C_2 . Per ipotesi induttiva sappiamo che $S \models C_i$ per $i = 1, 2$ quindi per ottenere la tesi ci basta verificare che

$$C_1, C_2 \models C$$

Sia σ il m.g.u. utilizzato nella risoluzione di $C = \{\ell_1, \dots, \ell_k\}$ da C_1, C_2 . Sappiamo che esiste un letterale ℓ tale che $C_1\sigma = \{\ell\} \cup C'_1$, $C_2\sigma = \{\bar{\ell}\} \cup C'_2$ e $C = C'_1 \cup C'_2$. Come nel caso proposizionale, segue facilmente che $C_1\sigma, C_2\sigma \models C$, e, poiché $C_i \models C_i\sigma$ per $i = 1, 2$ otteniamo $C_1, C_2 \models C$. $\square$

Come già accadeva nel caso proposizionale, la completezza del metodo si ottiene solo per $C = \square$. Ricordiamo che dato un insieme Γ di enunciati universali, vale:

Lemma 3.2.

$$\Gamma \text{ è soddisfacibile} \Leftrightarrow \Gamma^c \text{ è (proposizionalmente) soddisfacibile}.$$

Siamo finalmente in grado di dimostrare la completezza di Risoluzione Predicativa.

Teorema 3.3. *Se S è insoddisfacibile esiste un n tale che $\square \in \text{Res}^n(S)$.*

Dimostrazione. Sia S un insieme di clausole insoddisfacibili, per il Lemma 3.2 l'insieme delle istanze chiuse S^c di S è proposizionalmente insoddisfacibile. Dal Teorema di completezza proposizionale, esiste allora una refutazione proposizionale di S^c , cioè un albero T' di risoluzione proposizionale con radice etichettata dalla clausola vuota e le foglie etichettate da clausole in S^c . Mostriamo ora come sia possibile cambiare le etichette dei nodi di T' in modo da ottenere un albero di refutazione predicativa T con foglie etichettate da (rinomine) di clausole in S . A questo scopo mostriamo che, più in generale, se T' è un albero di risoluzione proposizionale della clausola C' da S^c allora è possibile rietichettare i nodi di T' con clausole predicative in modo che:

- (1) le foglie siano etichettate da rinomine di clausole in S ;
- (2) esista una sostituzione θ ground (cioè: che manda variabili in termini chiusi) tale che per ogni nodo dell'albero, l'etichetta proposizionale E' del nodo sia uguale a $E\theta$, dove E è la corrispondente etichetta predicativa.

La dimostrazione procede per induzione sull'altezza h dell'albero T' .

Se $h = 0$, abbiamo $C' \in S^c$ quindi C' è istanza chiusa di una clausola $C \in S$ e basta rietichettare l'unico nodo di T' con C .

Se $h > 0$, consideriamo l'ultimo passo di risoluzione in T' , che risolve due clausole $C'_1 = \{p(\bar{t})\} \cup G'$, $C'_2 = \{\neg p(\bar{t})\} \cup H'$ per ottenere la clausola $C' = G' \cup H'$. L'albero T' è allora composto da due sottoalberi, T'_1 con radice etichettata da C'_1 , T'_2 con radice etichettata da C'_2 , più un ultimo passo di risoluzione proposizionale che porta a C' . Per induzione, possiamo rietichettare gli alberi T'_i per $i = 1, 2$ in modo da ottenere due alberi di risoluzione predicativa T_1, T_2 e due sostituzioni ground θ_1, θ_2 che mappano le etichette predicative dei due alberi nelle corrispondenti etichette proposizionali. In particolare, se C_i è l'etichetta della radice di T_i allora $C_i\theta_i = C'_i$. A patto di rinominare le clausole in uno dei due alberi, possiamo supporre che le due sostituzioni siano a variabili separate. Questo implica che la sostituzione θ_1 non ha alcun effetto sulle etichette di nodi dell'albero T_2 e quindi, se E è l'etichetta di un nodo in T_2 allora $E\theta_1\theta_2 = E\theta_2 = E'$, dove E' è la corrispondente etichetta in T'_2 . Inoltre, poiché θ_1 è ground, se D è un'etichetta di un nodo in T_1 la clausola $D\theta_1$ non ha variabili libere e quindi $D\theta_1\theta_2 = D\theta_1 = D'$, dove D' è la corrispondente etichetta in T'_1 . In particolare, $C_i\theta_1\theta_2 = C'_i$ per $i = 1, 2$.

Possiamo allora usare la sostituzione $\theta_1\theta_2$ sui due sottoalberi, e per concludere la dimostrazione ci basterà trovare un risolvente predicativo C di C_1, C_2 tale che $C\theta_1\theta_2 = C'$.

Poiché $C'_1 = \{p(\bar{t})\} \cup G'$ è istanza di C_1 tramite la sostituzione $\theta_1\theta_2$, esisteranno dei letterali $\ell_1, \dots, \ell_k$ in C_1 tali che $\ell_j\theta_1\theta_2 = p(\bar{t})$, per $j = 1, \dots, k$ e $(C_1 \setminus \{\ell_1, \dots, \ell_k\})\theta_1\theta_2 = G'$. Analogamente, esisteranno dei letterali $\ell'_1, \dots, \ell'_h$ in C_2 tali che $\ell'_j\theta_1\theta_2 = \neg p(\bar{t})$, per $j = 1, \dots, h$ e $(C_2 \setminus \{\ell'_1, \dots, \ell'_h\})\theta_1\theta_2 = H'$.

In altre parole, l'insieme

$$\{\ell_1, \dots, \ell_k, \overline{\ell'_1}, \dots, \overline{\ell'_h}\}$$

è unificabile.

Sia σ l'unificatore di massima generalità corrispondente e C il risolvente di C_1, C_2 rispetto a σ :

$$C = (C_1 \setminus \{\ell_1, \dots, \ell_k\})\sigma \cup (C_2 \setminus \{\ell'_1, \dots, \ell'_h\})\sigma.$$

Per concludere la dimostrazione ci basta mostrare che $C\theta_1\theta_2 = C'$. Dalle proprietà dell'unificatore di massima generalità e dal fatto che $\theta_1\theta_2$ unifica l'insieme di letterali

$$\{\ell_1, \dots, \ell_k, \overline{\ell'_1}, \dots, \overline{\ell'_h}\};$$

abbiamo $\sigma\theta_1\theta_2 = \theta_1\theta_2$. Ne segue:

$$\begin{aligned} C\theta_1\theta_2 &= [(C_1 \setminus \{\ell_1, \dots, \ell_k\})\sigma \cup (C_2 \setminus \{\ell'_1, \dots, \ell'_h\})\sigma]\theta_1\theta_2 = \\ &= [(C_1 \setminus \{\ell_1, \dots, \ell_k\}) \cup (C_2 \setminus \{\ell'_1, \dots, \ell'_h\})]\sigma\theta_1\theta_2 = [(C_1 \setminus \{\ell_1, \dots, \ell_k\}) \cup (C_2 \setminus \{\ell'_1, \dots, \ell'_h\})]\theta_1\theta_2 = \\ &= (C_1 \setminus \{\ell_1, \dots, \ell_k\})\theta_1\theta_2 \cup (C_2 \setminus \{\ell'_1, \dots, \ell'_h\})\theta_1\theta_2 = G' \cup H' = C'. \end{aligned}$$

□

Abbiamo quindi dimostrato che, se esiste un albero di risoluzione della clausola proposizionale C' da S^c di altezza n , allora esiste una clausola predicativa C tale che:

- (1) C' è istanza di C ;
- (2) esiste un albero di risoluzione predicativa di C da S di altezza n .

Applicando quanto detto all'albero T' di refutazione proposizionale di S^c , con $C' = \square$, otteniamo che la clausola predicativa C di cui sopra non può essere altro che $\square$, e quindi l'albero T è un albero di refutazione di S di altezza n ; possiamo allora dedurre che $\square \in Res^n(S)$.

ESERCIZI SU RISOLUZIONE PREDICATIVA E UNIFICAZIONE

1. Data una clausola C e una sostituzione σ . dimostrare che vale $C \models C\sigma$.
2. Nei tre casi seguenti, stabilire se sia possibile derivare la clausola vuota dalle clausole C_1, C_2 con un singolo passo di risoluzione. Indicare in ogni caso il sistema di Herbrand e i passi dell'algoritmo di unificazione utilizzati per risolvere il problema.

$$(a) \quad C_1 = \{s(h(x, fy), w), s(v, fh(a, ffx))\}, \quad C_2 = \{\neg s(h(a, ffu), fh(a, ffa)), \neg s(z, fh(a, ffa))\};$$

$$(b) \quad C_1 = \{q(x, fy), q(fy, w)\}, \quad C_2 = \{\neg q(z, ffz), \neg q(v, ffv)\};$$

$$(c) \quad C_1 = \{q(x, fy), p(y, w)\}, \quad C_2 = \{\neg q(z, fa), \neg p(fv, ffv)\};$$

3. Siano ψ, θ gli enunciati seguenti:

$$\psi := \forall x \exists y [(r(x, fy) \rightarrow r(fx, fx)) \wedge (\neg r(x, fy) \rightarrow \forall z r(fz, fz))],$$

$$\theta := \exists y r(y, y).$$

La formula ψ ha come conseguenza logica la formula θ ? Se la risposta è affermativa, dimostrarlo utilizzando il metodo di risoluzione; qual è il numero minimo di passi di risoluzione necessario per arrivare alla clausola vuota?

4. Determinare se

$$\forall x \forall y \exists z (q(fz) \rightarrow s(x, y)) \wedge \forall y (\neg q(y) \vee \exists x \neg s(x, y)) \models \exists y \neg q(y);$$

se la risposta è positiva, darne una dimostrazione utilizzando il metodo di risoluzione.

5. Utilizzando il metodo di risoluzione, dimostrare che il seguente insieme di enunciati Γ è insoddisfacibile,:

$$\Gamma = \{\forall x \forall y \exists z (r(x, y) \rightarrow q(fz)), \quad \forall y (q(y) \vee \exists x \neg r(x, y)), \quad \neg \exists y q(y)\}.$$

Esiste una refutazione che utilizza solo due risoluzioni?

6. Trovare un unificatore di massima generalità per il seguente sistema di Herbrand, o dimostrare che tale unificatore non esiste:

$$\begin{cases} f(g(x, z)) = f(g(g(fa, y), ft)) \\ g(y, u) = g(x, f(g(a, y))) \end{cases}$$

7. Considerate le clausole

$$\begin{aligned} C_1 &= \{r(h(x, a, y), h(y, f(g(x, y)), b)), r(x, b)\} \\ C_2 &= \{\neg r(h(b, z, f(u)), h(f(z), f(g(f(u), f(z))), b))\}, \\ C_3 &= \{\neg r(h(b, z, f(u)), h(f(a), f(g(b, w)), v))\}. \end{aligned}$$

Per ognuna delle coppie di clausole $\{C_1, C_2\}$, $\{C_1, C_3\}$ e $\{C_2, C_3\}$ stabilite se si può applicare un passo corretto dell'algoritmo di risoluzione. Quando la risposta è negativa giustificarla, quando la risposta è positiva effettuate tutti i possibili passi di risoluzione, indicando la sostituzione utilizzata e la clausola prodotta.

8. Una *risoluzione lineare da input* di una clausola C a partire da un insieme di clausole S è una successione $C_0, B_1, C_1, B_2, C_2, \dots, C_{n-1}, B_n, C_n$ tale che $C_n = C$, $C_0, B_0, B_1, \dots, B_n$ sono clausole in S e C_{i+1} è ottenuto per risoluzione da C_i, B_{i+1} . Se $C = \square$, diciamo che S ammette una *refutazione lineare da input*.

- (a) La refutazione lineare da input non è, in generale, completa: dimostrare che l'insieme di clausole proposizionali

$$S = \{\{p, q\}, \quad \{p, \neg q\}, \quad \{\neg p, q\}, \quad \{\neg p, \neg q\}\},$$

è insoddisfacibile, ma non esiste una refutazione lineare da input di S

- (b) Dimostrare invece che, se $\mathcal{P}$ è un programma Prolog e G è un goal e $\mathcal{P} \cup \{G\}$ è insoddisfacibile, esiste sempre una refutazione lineare da input che parte dal goal (cioè: $C_0 = G$), e tale che le clausole $B_0, B_1, \dots, B_n$ sono in $\mathcal{P}$ (una refutazione con queste proprietà si dice refutazione lineare da input di G da $\mathcal{P}$).

(suggerimento: dimostrare il lemma per induzione sul numero n di variabili prop. Q tali che Q o $\neg Q$ appartengono ad almeno una clausola di $\mathcal{P}$; nel passo induttivo, provare che, siccome $S = \mathcal{P} \cup \{G\}$ è insoddisfacibile, il programma $\mathcal{P}$ deve contenere almeno un fatto $\{Q\}$ per una qualche variabile proposizionale Q . Considerare poi l'insieme di clausole S^Q e usare l'induzione).