

Risoluzione con unificazione

Marco Piastra

Risoluzione proposizionale

Procedura per stabilire se $\Gamma \models \varphi$

a) Refutazione $\Gamma \cup \{\neg\varphi\}$ e traduzione in forma normale congiuntiva (FNC)

$\beta_1 \wedge \beta_2 \wedge \dots \wedge \beta_n$ dove ogni β_i è una disgiunzione di letterali del tipo A o $\neg A$

b) Traduzione di $\Gamma \cup \{\neg\varphi\}$ in forma a clausole (FC)

$\{\beta_1, \beta_2, \dots, \beta_n\}$ dove ogni β_i è una fbf separata, in cui si omette il simbolo \wedge

c) Applicazione esaustiva della regola di inferenza per risoluzione

1) Selezione di due clausole da risolvere $\{\beta_1, \beta_2, \dots, \beta_n, \alpha\}, \{\neg\alpha, \gamma_1, \gamma_2, \dots, \gamma_m\}$

1) Generazione del risolvente

$\{\beta_1, \beta_2, \dots, \beta_n, \alpha\}, \{\neg\alpha, \gamma_1, \gamma_2, \dots, \gamma_m\} \vdash \{\beta_1, \beta_2, \dots, \beta_n, \gamma_1, \gamma_2, \dots, \gamma_m\}$

Condizioni di terminazione:

1) Derivazione della clausola vuota (*successo*)

2) Non sono possibili nuove risoluzioni - *punto fisso* (*fallimento*)

Refutazione e forma a clausole in L_{PO}

a) Refutazione: $\Gamma \cup \{\neg\varphi\}$

b) Traduzione in forma normale prenessa e *skolemizzazione* $sko(\Gamma \cup \{\neg\varphi\})$:

Tutte le formule sono nella forma:

$$\forall x_1 \forall x_2 \dots \forall x_n \psi \quad (\text{la matrice } \psi \text{ non contiene quantificatori})$$

Essendo tutti universali, i quantificatori si possono omettere

c) Traduzione delle matrici ψ in FNC (con le stesse regole del caso proposizionale)

Si eliminano \rightarrow e \leftrightarrow in base alle regole di riscrittura

Si muove \neg all'interno:

$$\neg(\alpha \wedge \beta) \Leftrightarrow (\neg\alpha \vee \neg\beta)$$

$$\neg(\alpha \vee \beta) \Leftrightarrow (\neg\alpha \wedge \neg\beta)$$

Si distribuisce \vee :

$$((\alpha \wedge \beta) \vee \gamma) \Leftrightarrow ((\alpha \vee \gamma) \wedge (\beta \vee \gamma))$$

e quindi si traduce in forma a clausole (FC) (con le stesse regole del caso proposizionale)

Esempio: traduzione in FC di $\forall x (P(x) \rightarrow \exists y (Q(x,y) \wedge R(y)))$

1: $\forall x \exists y (P(x) \rightarrow (Q(x,y) \wedge R(y)))$

(forma normale prenessa)

2: $\forall x (P(x) \rightarrow (Q(x,k(x)) \wedge R(k(x))))$

(skolemizzazione, nuova funzione $k/1$)

3: $P(x) \rightarrow (Q(x,k(x)) \wedge R(k(x)))$

(eliminazione dei quantificatori)

4: $\neg P(x) \vee (Q(x,k(x)) \wedge R(k(x)))$

(equivalenza di \rightarrow)

5: $(\neg P(x) \vee Q(x,k(x))) \wedge (\neg P(x) \vee R(k(x)))$

(FNC, distributività di \vee)

6: $\{\neg P(x), Q(x,k(x))\}, \{\neg P(x), R(k(x))\}$

(FC)

Unificazione

▪ Unificatore

Una sostituzione $\sigma = [x_1/t_1, x_2/t_2 \dots x_n/t_n]$ che rende risolvibili due letterali α e $\neg\beta$

in simboli: $\sigma(\alpha) \equiv \sigma(\beta)$

- Non sono ammesse sostituzioni *ricorsive*: in x_i/t_i la variabile x_i **non può** comparire in t_i
- Non sempre esiste un unificatore:
ad esempio $\{P(g(x, f(a)), a)\}$ e $\{\neg P(g(b, f(w)), k(w))\}$ non sono unificabili

▪ Unificatore più generale (MGU - *most general unifier*)

Se esiste un unificatore di α e $\neg\beta$ esiste anche un unificatore più generale MGU μ

$$\text{MGU } \mu \Leftrightarrow \forall \sigma \exists \sigma' : \sigma = \mu \cdot \sigma'$$

cioè qualsiasi altro unificatore può essere ottenuto per composizione da μ

(intuitivamente, μ è la minima sostituzione indispensabile)

Esiste un algoritmo che trova μ (se la coppia α e $\neg\beta$ è unificabile, ovviamente)

Unificare è necessario

■ Problema: $\Gamma \models \varphi$?

$\Gamma \equiv \{\forall x (Filosofo(x) \rightarrow Umano(x)), \forall x (Umano(x) \rightarrow Mortale(x)), Filosofo(Socrate)\}$
 $\varphi \equiv Mortale(Socrate)$

Procedura (*risoluzione per refutazione*):

1: $\{\forall x (Filosofo(x) \rightarrow Umano(x)), \forall x (Umano(x) \rightarrow Mortale(x)), Filosofo(Socrate), \neg Mortale(socrate)\}$

($\Gamma \cup \{\neg\varphi\}$ è già in forma prenessa, non serve la skolemizzazione)

2: $\{\{Umano(x), \neg Filosofo(x)\}, \{Mortale(x), \neg Umano(x)\}, \{Filosofo(socrate)\}, \{\neg Mortale(socrate)\}\}$

(forma a clausole)

Applicazione della regola di risoluzione

3: $\{Umano(x), \neg Filosofo(x)\}, \{Mortale(x), \neg Umano(x)\} \vdash \{\neg Filosofo(x), Mortale(x)\}$

4: $\{Umano(socrate)\}, \{Mortale(x), \neg Umano(x)\} \vdash ???$

(la regola di risoluzione non si può applicare direttamente, però ...)

Costruzione del MGU

▪ Algoritmo di Martelli e Montanari

Input: $\{s_1 = t_1, s_2 = t_2 \dots s_n = t_n\}$ (equazioni tra termini: gli argomenti dei due letterali)

Procedura:

Applicare le seguenti regole, in ordine qualsiasi
(ciascuna applicazione riscrive l'insieme di equazioni)

- | | |
|-------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| (1) $f(s_1, \dots, s_n) = f(t_1, \dots, t_n)$ | <i>replace by the equations
$s_1 = t_1, \dots, s_n = t_n,$</i> |
| (2) $f(s_1, \dots, s_n) = g(t_1, \dots, t_m)$ where $f \neq g$ | <i>halt with failure,</i> |
| (3) $x = x$ | <i>delete the equation,</i> |
| (4) $t = x$ where t is not a variable | <i>replace by the equation $x = t,$</i> |
| (5) $x = t$ where x does not occur in t
and x occurs elsewhere | <i>apply the substitution $\{x/t\}$
to all other equations</i> |
| (6) $x = t$ where x occurs in t and x differs from t | <i>halt with failure.</i> |

La procedura termina quando non vi sono più regole applicabili
o quando si ha un fallimento (regole (2) e (6))

Si ha **successo** sse tutte le equazioni sono del tipo $x_i = t_i$

Esempi di costruzione del MGU

Esempio: $\{f(x, a) = f(g(z), y), h(u) = h(d)\}$

$\{x = g(z), y = a, h(u) = h(d)\}$

$\{x = g(z), y = a, u = d\}$

Regola (1) su $f(x, a) = f(g(z), y)$

Regola (1) su $h(u) = h(d)$, MGU

Esempio: $\{f(x, a) = f(g(z), y), h(x, z) = h(u, d)\}$

$\{x = g(z), y = a, h(x, z) = h(u, d)\}$

$\{x = g(z), y = a, h(g(z), z) = h(u, d)\}$

$\{x = g(z), y = a, u = g(z), z = d\}$

$\{x = g(d), y = a, u = g(d), z = d\}$

Regola (1) su $f(x, a) = f(g(z), y)$

Regola (5) su $x = g(z)$

Regola (1) su $h(g(z), z) = h(u, d)$

Regola (5) su $z = d$, MGU

Esempio: $\{f(x, a) = f(g(z), y), h(x, z) = h(d, u)\}$

$\{x = g(z), y = a, h(x, z) = h(d, u)\}$

$\{x = g(z), y = a, h(g(z), z) = h(d, u)\}$

Regola (1) su $f(x, a) = f(g(z), y)$

Non vi sono regole applicabili
e $h(g(z), z) = h(d, u)$ non è del tipo atteso
(fallimento)

Risoluzione con unificazione in L_{PO}

Procedura per stabilire se $\Gamma \models \varphi$ in L_{PO}

- a) Refutazione $\Gamma \cup \{\neg\varphi\}$,
- b) Forma normale prenessa e skolemizzazione $sko(\Gamma \cup \{\neg\varphi\})$
- c) Traduzione di $sko(\Gamma \cup \{\neg\varphi\})$ in FNC e quindi in forma a clausole (FC)
- d) Applicazione esaustiva della regola di risoluzione:
 - 1) Selezione di due clausole da risolvere $\{\beta_1, \beta_2, \dots, \beta_n, \alpha\}, \{\neg\alpha', \gamma_1, \gamma_2, \dots, \gamma_m\}$
 - 2) *Standardizzazione* delle variabili
(creazione di due nuove clausole con *ridenominazione* delle variabili)
 - 3) Costruzione del MGU μ (se esiste) dei due letterali α e α'
 - 4) Applicazione di μ alle due clausole e generazione del risolvente
 $\{\beta_1, \beta_2, \dots, \beta_n, \alpha\}[\mu], \{\neg\alpha', \gamma_1, \gamma_2, \dots, \gamma_m\}[\mu] \vdash \{\beta_1, \beta_2, \dots, \beta_n, \gamma_1, \gamma_2, \dots, \gamma_m\}[\mu]$

Condizioni di terminazione:

- 1) Derivazione della clausola vuota (*successo*)
- 2) Non sono possibili nuove risoluzioni - *punto fisso* (*fallimento*)

Il metodo può anche *divergere* (i.e. continuare all'infinito)

Esempio di risoluzione con unificazione

■ Problema: $\Gamma \models \varphi$?

$\Gamma \equiv \{\forall x (Filosofo(x) \rightarrow Umano(x)), \forall x (Umano(x) \rightarrow Mortale(x)), Filosofo(Socrate)\}$
 $\varphi \equiv Mortale(Socrate)$

Procedura (*risoluzione per refutazione*):

1: $\{\forall x (Filosofo(x) \rightarrow Umano(x)), \forall x (Umano(x) \rightarrow Mortale(x)), Filosofo(Socrate), \neg Mortale(socrate)\}$

($\Gamma \cup \{\neg\varphi\}$ è già in forma prenessa, non serve la skolemizzazione)

2: $\{\{Umano(x), \neg Filosofo(x)\}, \{Mortale(x), \neg Umano(x)\}, \{Filosofo(socrate)\}, \{\neg Mortale(socrate)\}\}$

Applicazione iterativa della regola di risoluzione

3: $\{Filosofo(socrate)\}, \{Umano(x_1), \neg Filosofo(x_1)\}, [x_1/socrate] \vdash \{Umano(socrate)\}$

4: $\{Umano(socrate)\}, \{Mortale(x_2), \neg Umano(x_2)\}, [x_2/socrate] \vdash \{Mortale(socrate)\}$

5: $\{Mortale(socrate)\}, \{\neg Mortale(socrate)\}, [] \vdash \{\}$

(Successo)

Esempio di risoluzione con unificazione

$\Gamma \equiv \{\forall x (Filosofo(x) \rightarrow Umano(x)), \forall x (Umano(x) \rightarrow Mortale(x)), Filosofo(Socrate)\}$
 $\varphi \equiv Mortale(Socrate)$

Procedura (risoluzione per refutazione):

1: ...

2: $\{\{Umano(x), \neg Filosofo(x)\}, \{Mortale(x), \neg Umano(x)\}, \{Filosofo(socrate)\}, \{\neg Mortale(socrate)\}\}$

Applicazione iterativa della regola di risoluzione

3: $\{Mortale(x_1), \neg Umano(x_1)\}, \{Umano(x_2), \neg Filosofo(x_2)\}, [x_1/x_2] \vdash \{Mortale(x_2), \neg Filosofo(x_2)\}$

4: $\{Mortale(x_3), \neg Filosofo(x_3)\}, \{Filosofo(socrate)\}, [x_3/socrate] \vdash \{Mortale(socrate)\}$

5: $\{Mortale(socrate)\}, \{\neg Mortale(socrate)\}, [] \vdash \{\}$

(Successo)

Diversa scelta delle clausole da risolvere (rispetto al caso precedente)

Notare che nella risoluzione 4 si unificano due letterali con variabili:
il risolvente contiene variabili (si dice anche *lifting*)

Ulteriore esempio

■ Il metodo di risoluzione può divergere

Problema: $\forall x (Q(f(x)) \rightarrow P(x)) \models \exists x (P(f(x)) \wedge \neg Q(f(x)))$

Refutazione:

$\{ \forall x (Q(f(x)) \rightarrow P(x)) \} \cup \{ \neg \exists x (P(f(x)) \wedge \neg Q(f(x))) \}$

Forma normale prenessa:

$\{ \forall x (Q(f(x)) \rightarrow P(x)) \} \cup \{ \forall x \neg (P(f(x)) \wedge \neg Q(f(x))) \}$

(Sono due enunciati in forma universale, non serve *skolemizzazione*)

Forma a clausole:

$\{ Q(f(x)) \rightarrow P(x) \} \cup \{ \neg (P(f(x)) \wedge \neg Q(f(x))) \}$

$\{ \neg Q(f(x)) \vee P(x) \} \cup \{ \neg P(f(x)) \vee Q(f(x)) \}$

$\{ \neg Q(f(x)) \vee P(x) \}, \{ \neg P(f(x)) \vee Q(f(x)) \}$

Applicazione iterativa della regola di risoluzione (notare la standardizzazione):

1: $\{ \neg Q(f(x_1)), P(x_1) \}, \{ \neg P(f(x_2)), Q(f(x_2)) \}, [x_1/f(x_2)] \vdash \{ \neg Q(f(f(x_2))), Q(f(x_2)) \}$

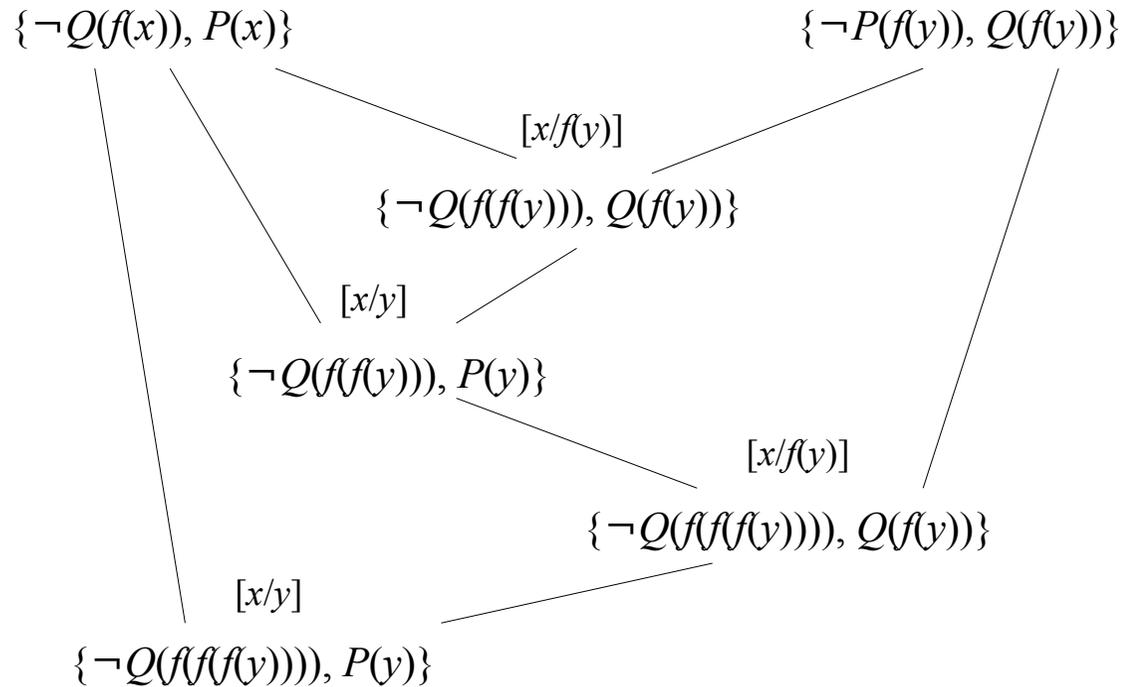
2: $\{ \neg Q(f(x_3)), P(x_3) \}, \{ \neg Q(f(f(x_4))), Q(f(x_4)) \}, [x_3/x_4] \vdash \{ \neg Q(f(f(x_4))), P(x_4) \}$

3: $\{ \neg Q(f(f(x_5))), P(x_5) \}, \{ \neg P(f(x_6)), Q(f(x_6)) \}, [x_5/f(x_6)] \vdash \{ \neg Q(f(f(f(x_6)))), Q(f(x_6)) \}$

4: $\{ \neg Q(f(x_7)), P(x_7) \}, \{ \neg Q(f(f(f(x_8)))) \}, [x_7/x_8] \vdash \{ \neg Q(f(f(f(x_8)))) \}, P(x_8) \}$

...

Vista alternativa dell'esempio precedente



.
. .
. .

La standardizzazione delle variabili
spesso non viene mostrata, per semplicità

Completezza del metodo di risoluzione

- Il metodo di risoluzione con unificazione è **corretto** in L_{PO}

Se il metodo trova una contraddizione in $sko(\Gamma \cup \{\neg\varphi\})$ allora $\Gamma \models \varphi$

- Il metodo di risoluzione con unificazione è **completo** per L_{PO} ?

Sì, nei limiti della semi-decidibilità di L_{PO} (Robinson, 1963)

Se $\Gamma \models \varphi$, il metodo trova una contraddizione in $sko(\Gamma \cup \{\neg\varphi\})$

In generale (ma non nel caso peggiore) il metodo a risoluzione è più efficiente dell'enumerazione ricorsiva e verifica di $H(sko(\Gamma \cup \{\neg\varphi\}))$

Il vantaggio principale è il *lifting*

Se $\Gamma \not\models \varphi$, il metodo può divergere

Tuttavia, il metodo (inteso come algoritmo) potrebbe divergere anche quando $\Gamma \models \varphi$

Elementi critici:

- Criterio di selezione delle clausole da risolvere
- Strategia di esplorazione delle alternative

Esempio: il mondo delle liste

▪ Liste di oggetti $[a, b, c, \dots]$

$cons(s, x)$

funzione, associa ad un oggetto (es. a) ed una lista (es. $[b, c]$)

la lista ottenuta inserendo l'oggetto all'inizio (es. $[a, b, c]$)

$Append(x, y, z)$

predicato, associa alle liste x e y la concatenazione z

nil

costante, indica la lista vuota.

Notazione abbreviata (Prolog): $[] \Leftrightarrow nil$

$[a] \Leftrightarrow cons(a, nil)$

$[a, b] \Leftrightarrow cons(a, cons(b, nil))$

$[a|[b, c]] \Leftrightarrow cons(a, [b, c])$

Assiomi (AL)

$\forall x Append(nil, x, x)$

$\forall x \forall y \forall z (Append(x, y, z) \rightarrow \forall s Append([s, x], y, [s, z]))$

Esempi (conseguenze logiche)

$AL + \exists z Append([a], [b, c], z) \models Append([a], [b, c], [a, b, c]) = [z/[a, b, c]]$

$AL + \exists x \exists y Append(x, y, [a, b]) \models Append([a], [b], [a, b]) = [x/[a], x/[b]]$

$\models Append(nil, [a, b], [a, b]) = [x/nil, y/[a, b]]$

$\models Append([a, b], nil, [a, b]) = [x/[a, b], y/nil]$

Esempio: il mondo delle liste

Problema: $\forall x \text{ Append}(\text{nil}, x, x) \models \exists y \forall x \text{ Append}(\text{nil}, \text{cons}(y, x), \text{cons}(a, x))$

1: $\forall x \text{ Append}(\text{nil}, x, x), \neg \exists y \forall z \text{ Append}(\text{nil}, \text{cons}(y, z), \text{cons}(a, z))$

(refutazione e ridenominazione delle variabile x)

2: $\forall x \text{ Append}(\text{nil}, x, x), \forall y \exists z \neg \text{Append}(\text{nil}, \text{cons}(y, z), \text{cons}(a, z))$ (forma normale prenessa)

3: $\{\text{Append}(\text{nil}, x, x)\}, \{\neg \text{Append}(\text{nil}, \text{cons}(y, k(y)), \text{cons}(a, k(y)))\}$

($k/1$ funzione di Skolem, forma a clausole)

(N.B. il Prolog non fa la *skolemizzazione*: deve farla il programmatore)

La coppia di **letterali**

$\text{Append}(\text{nil}, x, x), \neg \text{Append}(\text{nil}, \text{cons}(y, k(y)), \text{cons}(a, k(y)))$

... è compatibile (stesso predicato *Append/3*) ma i letterali hanno argomenti **diversi**

Se tuttavia si applica una sostituzione $\sigma = [x/\text{cons}(a, k(a)), y/a]$ si ottiene

$\{\text{Append}(\text{nil}, \text{cons}(a, k(a)), \text{cons}(a, k(a)))\}, \{\neg \text{Append}(\text{nil}, \text{cons}(a, k(a)), \text{cons}(a, k(a)))\}$

Da cui, per risoluzione, si ottiene la clausola vuota

La sostituzione σ si dice **unificatore** delle due clausole

va applicata integralmente a tutte e due le clausole da risolvere