Azzolini Riccardo 2020-06-15

Risoluzione al primo ordine

1 Risoluzione nel calcolo dei predicati

Sì è già visto che, per ogni formula del primo ordine H, esiste una formula $Ex^S(H)$ chiusa e in forma normale di Skolem tale che

$$H \text{ è soddisfacibile} \iff Ex^S(H) \text{ è soddisfacibile}$$

Poi, considerando una formula in forma di Skolem,

$$\varphi = \forall x_1 \dots \forall x_n \psi$$

siccome la matrice ψ non contiene quantificatori, la si può trasformare in una CNF.

Dunque, complessivamente, data una qualunque formula del primo ordine, si può costruire una formula chiusa in forma di Skolem con matrice in CNF, che è equisoddisfacibile alla formula di partenza e può essere rappresentata come insieme di clausole (supponendo implicitamente che tutte le variabili presenti nelle clausole di tale insieme siano quantificate universalmente).

Ciò consente di applicare il calcolo di risoluzione per studiare la soddisfacibilità di una qualunque formula del primo ordine. Per estendere la risoluzione al caso predicativo, bisogna però introdurre alcune modifiche, che riguardano essenzialmente la gestione dei termini.

Ad esempio, se si considera la formula (in questo caso già chiusa)

$$\forall x(A(x) \vee B(x)) \wedge \forall z(A(f(c)) \to C(z))$$

e la si trasforma in forma di Skolem con matrice in CNF,

$$\varphi = \forall x\big((A(x) \vee B(X)) \wedge (\neg A(f(c)) \vee C(x))\big)$$

si ha che φ è rappresentata dall'insieme di clausole:

$$\mathcal{S}_\varphi = \big\{\underbrace{\{A(x), B(x)\}}_{\mathcal{C}_1}, \underbrace{\{\neg A(f(c)), C(x)\}}_{\mathcal{C}_2}\big\}$$

Alle clausole $\mathcal{C}_1$ e $\mathcal{C}_2$ è possibile applicare la regola di risoluzione (analogamente al caso proposizionale) se si istanzia la variabile x con il termine $f(c)$:

$$\mathcal{S}_\varphi[f(c)/x] = \big\{\{A(f(c)), B(f(c))\}, \{\neg A(f(c)), C(f(c))\}\big\}$$

Adesso, le due clausole contengono i letterali complementari $A(f(c))$ e $\neg A(f(c))$, sui quali si applica la risoluzione:

$$\frac{\{A(f(c)), B(f(c))\} \qquad \{\neg A(f(c)), C(f(c))\}}{B(f(c)), C(f(c))}\,\text{ris}$$

Per generalizzare il ragionamento, bisogna introdurre un metodo per individuare le sostituzioni da effettuare al fine di poter applicare la regola di risoluzione.

2 Sostituzioni

Una sostituzione può essere rappresentata come

$$\sigma = [t_1/x_1, \ldots, t_n/x_n]$$

che indica la sostituzione della variabile x_i con il termine t_i per ogni $i = 1, \ldots, n$.

Generalizzando l'applicazione a una singola formula, una sostituzione può essere applicata a un insieme: se $\mathcal{E}$ è un insieme di formule (oppure un insieme di termini[1]), si scrive $\mathcal{E}\sigma$ per indicare l'insieme ottenuto applicando la sostituzione σ a tutte le formule (o termini) di $\mathcal{E}$.

In seguito, sarà necessario applicare l'operazione di **composizione** alle sostituzioni: la composizione di due sostituzioni σ e τ è la sostituzione (indicata con $\sigma \circ \tau$ oppure con $\sigma\tau$) che si ottiene applicando prima σ e poi τ. Bisogna anche introdurre l'elemento neutro rispetto alla composizione: la **sostituzione vuota** ϵ, tale che $\mathcal{E}\epsilon = \mathcal{E}$.

2.1 Esempi

- Siano $\mathcal{E} = \{P(f(x), y), A(x) \to B(f(y))\}$ e $\sigma = [g(x)/y]$. Allora:

$$\mathcal{E}\sigma = \{P(f(x), g(x)), A(x) \to B(f(g(x)))\}$$

- Siano $\mathcal{E} = \{P(f(x), y), A(x) \to B(f(y))\}$, $\sigma = [g(x)/y]$ e $\tau = [c/x]$. Applicando la composizione di sostituzioni $\sigma\tau$ si ottiene:

$$\begin{aligned}
\mathcal{E}\sigma\tau &= \{P(f(x), y), A(x) \to B(f(y))\}\underbrace{[g(x)/y]}_{\sigma}\underbrace{[c/x]}_{\tau} \\
&= \{P(f(x), g(x)), A(x) \to B(f(g(x)))\}\underbrace{[c/x]}_{\tau} \\
&= \{P(f(c), g(c)), A(c) \to B(f(g(c)))\}
\end{aligned}$$

[1] La risoluzione al primo ordine può essere presentata ragionando equivalentemente sulle formule o sui termini. Perciò, molte delle definizioni che verranno date in seguito sono valide sia sulle formule che sui termini.

Allora, riscritta come unica sostituzione, la composizione $\sigma\tau$ diventa:

$$\sigma\tau = [g(c)/y, c/x]$$

2.2 Composizione di sostituzioni

In generale, se $\sigma = [t_1/x_1, \ldots, t_n/x_n]$ e $\tau = [s_1/y_1, \ldots, s_m/y_m]$, la composizione $\sigma\tau$ si ottiene "concatenando" le due sostituzioni, applicando la sostituzione τ a tutti i termini di σ,

$$[t_1\tau/x_1, \ldots, t_n\tau/x_n, s_1/y_1, \ldots, s_m/y_m]$$

e togliendo tutte le coppie s_i/y_i (provenienti da τ) che agiscono su variabili già sostituite da σ, cioè tali che $y_i \in \{x_1, \ldots, x_n\}$.

Ad esempio, date $\sigma = [c/x, d/y]$ e $\tau = [e/y, f/z]$:

$$\sigma\tau = [c/x, d/y][e/y, f/z] = [c/x, d/y, f/z]$$

Osservazione: La composizione di sostituzioni *non è commutativa*. Ad esempio:

$$A(x,y)[c/x, d/y][e/x] = A(x,y)[c/x, d/y] = A(c,d)$$
$$A(x,y)[e/x][c/x, d/y] = A(x,y)[e/x, d/y] = A(e,d)$$

3 Unificatori

Definizione: Una sostituzione σ è un **unificatore** per un insieme di espressioni (formule o termini) $\mathcal{E} = \{E_1, \ldots, E_N\}$ se $\mathcal{E}\sigma$ ha un solo elemento, cioè se σ rende uguali tutti gli elementi di $\mathcal{E}$:

$$|\mathcal{E}\sigma| = 1 \iff E_1\sigma = E_2\sigma = \cdots = E_n\sigma$$

3.1 Esempio

Si consideri l'insieme di formule:

$$\mathcal{E} = \{A(x), A(f(y)), A(f(g(z)))\}$$

- La sostituzione $\sigma = [f(g(z))/x, g(z)/y]$ è un unificatore per $\mathcal{E}$:

$$\mathcal{E}\sigma = \{A(x), A(f(y)), A(f(g(z)))\}[f(g(z))/x, g(z)/y] = \{A(f(g(z)))\}$$

- Anche $\tau = [f(g(a))/x, g(a)/y, a/z]$ è un unificatore per $\mathcal{E}$:

$$\mathcal{E}\tau = \{A(x), A(f(y)), A(f(g(z)))\}[f(g(a))/x, g(a)/y, a/z] = \{A(f(g(a)))\}$$

Confrontando gli insiemi ottenuti applicando questi due unificatori, si osserva che $\mathcal{E}\tau$ può essere ottenuto applicando una sostituzione a $\mathcal{E}\sigma$,

$$\mathcal{E}\sigma[a/z] = \{A(f(g(z)))\}[a/z] = \{A(f(g(a)))\} = \mathcal{E}\tau$$

mentre non vale il contrario (perché a, essendo una costante, non può essere sostituita: la sostituzione si applica solo alle variabili). Si dice allora che l'unificatore σ è *più generale di* τ.

4 Unificatore più generale

Definizione: Un unificatore σ per un insieme di espressioni $\mathcal{E}$ è un **unificatore più generale** (o **MGU**, *Most General Unifier*) se, per ogni altro unificatore τ di $\mathcal{E}$, esiste una sostituzione θ tale che $\tau = \sigma\theta$.

Quando esiste, l'unificatore più generale di un insieme è *unico* (a meno di rinominare le variabili che occorrono nell'unificatore), e può essere generato dall'**algoritmo di Robinson**, che verrà presentato in seguito.

Osservazione: Nell'esempio precedente, si è mostrato solo che σ è più generale *di* τ, ovvero che, per *uno specifico* altro unificatore τ, esiste $\theta = [a/z]$ tale che $\tau = \sigma\theta$. Invece, per verificare che (in questo caso) σ è effettivamente anche un unificatore più generale "di tutti", sarebbe necessario mostrare che ciò avviene per *ogni* altro unificatore.

5 Insieme di disaccordo

Definizione: L'**insieme di disaccordo** di un insieme di formule $\mathcal{E}$ è un insieme $D(\mathcal{E})$ di espressioni (formule o termini), ottenute leggendo contemporaneamente tutte le formule di $\mathcal{E}$ e raccogliendo in $D(\mathcal{E})$ i primi sottotermini per cui le formule differiscono.

Ad esempio:

- Se
$$\mathcal{E} = \{A(x, f(y)), A(x, f(g(y))), A(x, h(z))\}$$
allora:
$$D(\mathcal{E}) = \{f(y), f(g(y)), h(z)\}$$
- Se invece si considerassero solo le prime due formule di $\mathcal{E}$,
$$\mathcal{E}' = \{A(x, f(y)), A(x, f(g(y)))\}$$
si avrebbe:
$$D(\mathcal{E}') = \{y, g(y)\}$$

- Dato $\mathcal{E} = \{A(a), A(b)\}$, si ha $D(\mathcal{E}) = \{a, b\}$, e nessuna sostituzione può rendere uguali a e b (in quanto costanti), quindi $\mathcal{E}$ non ha unificatori (e dunque, in particolare, non ha un unificatore più generale).

6 Algoritmo di Robinson

Input: l'insieme di espressioni $\mathcal{E}$.
Output: l'MGU di $\mathcal{E}$, se esiste, altrimenti `null`.

$k = 0$
$\sigma_0 = \epsilon$
`while` $(|\mathcal{E}\sigma_k| \neq 1)$ {
  calcola l'insieme di disaccordo $D(\mathcal{E}\sigma_k)$
  `if` (esistono $x, t \in D(\mathcal{E}\sigma_k)$ tali che $x \in VAR$ non occorre in t) `then`
    $\sigma_{k+1} = \sigma_k \circ [t/x]$
    $k = k + 1$
  `else`
    `return null`
}
`return` σ_k

Quest'algoritmo è un processo iterativo (la variabile k conta le iterazioni), che calcola l'MGU "per approssimazioni successive", componendo sostituzioni σ_k a partire da quella vuota $\sigma_0 = \epsilon$. A ogni iterazione:

1. Si calcola l'insieme di disaccordo, che non può essere vuoto perché $\mathcal{E}\sigma_k$ contiene più di un elemento (e quindi ci sono sicuramente delle differenze tra i vari elementi).
2. Si cercano nell'insieme di disaccordo una variabile x e un termine t tali che x non occorra in t (questa condizione verrà motivata più avanti). Se esistono, si costruisce una nuova sostituzione σ_{k+1}, componendo quella ottenuta all'iterazione precedente, σ_k, con $[t/x]$. Altrimenti, se tali x e t non esistono, non si riesce a trovare un MGU, e perciò l'algoritmo termina restituendo `null`.

Se la condizione dell'`if` è sempre soddisfatta (cioè non viene restituito `null`), il ciclo termina quando $\mathcal{E}\sigma_k$ ha un singolo elemento, ovvero quando σ_k è un unificatore per $\mathcal{E}$.

Osservazione: Il ciclo termina sicuramente, perché ogni nuova sostituzione elimina da $\mathcal{E}$ una variabile, quindi, prima o poi, o $\mathcal{E}\sigma_k$ "collassa" in una sola formula, oppure non si riesce più a soddisfare la condizione dell'`if`.

6.1 Esempio di applicazione

Input: $\mathcal{E} = \{P(a, y), P(x, f(b))\}$

1. $k = 0,\ \sigma_0 = \epsilon$
2. Siccome $|\mathcal{E}\sigma_0| = |\{P(a, y), P(x, f(b))\}| \neq 1$, viene eseguita la prima iterazione del ciclo `while`:
   a) $D(\mathcal{E}\sigma_0) = \{a, x\}$
   b) La condizione dell'`if` è soddisfatta, considerando la variabile x e il termine a (in cui, appunto, x non occorre).
   c) $\sigma_1 = \sigma_0 \circ [a/x] = \epsilon \circ [a/x] = [a/x]$
3. $|\mathcal{E}\sigma_1| = |P(a, y), P(a, f(b))| \neq 1$, quindi si esegue la seconda iterazione:
   a) $D(\mathcal{E}\sigma_1) = \{y, f(b)\}$
   b) La condizione dell'`if` è soddisfatta dalla variabile y e dal termine $f(b)$.
   c) $\sigma_2 = \sigma_1 \circ [f(b)/y] = [a/x, f(b)/y]$
4. $|\mathcal{E}\sigma_2| = |\{P(a, f(b))\}| = 1$, dunque il ciclo termina, e viene restituito l'unificatore σ_2.

6.2 Teorema di correttezza dell'algoritmo

Teorema: L'algoritmo di Robinson restituisce l'MGU di un insieme $\mathcal{E}$ se questo insieme è unificabile, altrimenti restituisce `null` (che indica che l'insieme non è unificabile).

6.3 Occur check

Nella condizione dell'`if`,

$$\text{esistono } x, t \in D(\mathcal{E}\sigma_k) \text{ tali che } x \in VAR \text{ non occorre in } t$$

la verifica che t non contenga la variabile x prende il nome di **occur check**.

L'occur check è necessario per garantire la terminazione, ma può peggiorare di molto le prestazioni dell'algoritmo, facendo sì che esso richieda, nel caso peggiore, tempo esponenziale nella dimensione dell'input. Allora, nella pratica, l'occur check viene solitamente omesso, accettando il rischio che l'algoritmo possa non terminare, e si cerca invece di effettuare dei test euristici che garantiscano la terminazione su un insieme significativo di casi.

La situazione tipica in cui, senza l'occur check, non si ha la terminazione è esemplificata dall'insieme $\mathcal{E} = \{A(x), A(f(x))\}$:

- iterazione 1:

$$D(\mathcal{E}\sigma_0) = \{x, f(x)\} \qquad \sigma_1 = [f(x)/x] \qquad \mathcal{E}\sigma_1 = \{A(f(x)), A(f(f(x)))\}$$

- iterazione 2:

$$D(\mathcal{E}\sigma_1) = \{x, f(x)\} \qquad \sigma_2 = [f(f(x))/x] \qquad \mathcal{E}\sigma_2 = \{A(f(f(x))), A(f(f(f(x))))\}$$

- e così via, all'infinito.

In questo caso molto semplice, un'implementazione "furba" dell'algoritmo di Robinson si potrebbe accorgere che l'insieme $\mathcal{E}$ non è unificabile prima ancora di applicare l'algoritmo vero e proprio; in altri casi, quando le interazioni tra le variabili sono più complesse, non è detto che si riesca a capire a priori se ci saranno problemi legati all'occur check.

7 Risolvente e regola di risoluzione

Rispetto al caso proposizionale, nel caso predicativo la regola di risoluzione diventa un po' più complicata: in particolare, invece di agire su un solo letterale per ciascuna delle due clausole, si trattano insieme più letterali alla volta. Prima di dare la definizione formale, si illustra intuitivamente il funzionamento della regola mediante un esempio.

7.1 Esempio

Si considerino le clausole

$$\mathcal{C}_1 = \{A(x, f(y)), A(x, z), C(z)\} \qquad \mathcal{C}_2 = \{\neg A(f(x_1), z_1), B(x_1)\}$$

Esse non contengono direttamente una coppia complementare, ma si nota che ci sono dei "candidati promettenti" per cercare di ottenerne una: le formule che in una delle due clausole hanno un certo simbolo di predicato (in questo caso A), e nell'altra hanno lo stesso simbolo di predicato ma negato. In particolare, le coppie candidate sono:

- $A(x, f(y)) \in \mathcal{C}_1,\ \neg A(f(x_1), z_1) \in \mathcal{C}_2$;
- $A(x, z) \in \mathcal{C}_1,\ \neg A(f(x_1), z_1) \in \mathcal{C}_2$.

Quando si vuole applicare la risoluzione, si cerca per prima cosa di trovare un unificatore per tutte queste formule candidate, cioè, in questo caso, per:

$$L_1 = A(x, f(y)) \qquad L_2 = A(x, z) \qquad L_1' = \neg A(f(x_1), z_1)$$

È però necessario gestire un piccolo dettaglio tecnico: l'algoritmo di unificazione cerca il primo punto in cui le formule considerate vanno in disaccordo, ma qui alcune iniziano con A, mentre altre con $\neg A$. Per rendere unificabile l'insieme, e "portare" l'algoritmo

a ragionare sui termini, bisogna far sì che le formule inizino tutte allo stesso modo, complementando i letterali provenienti da una delle due clausole (per convenzione, la prima). L'insieme da unificare diventa allora

$$\{\overline{L_1}, \overline{L_2}, L_1'\} = \{\overbrace{\neg \underbrace{A(x, f(y))}_{L_1}}^{\overline{L_1}}, \overbrace{\neg \underbrace{A(x, z)}_{L_2}}^{\overline{L_2}}, \underbrace{\neg A(f(x_1), z_1)}_{L_1'}\}$$

e, applicando a esso l'algoritmo di Robinson, si trova l'unificatore più generale

$$\sigma = [f(x_1)/x, f(y)/z, f(y)/z_1]$$

tale che

$$|\{\overline{L_1}, \overline{L_2}, L_1'\}\sigma| = |\{\neg A(f(x_1), f(y))\}| = 1$$

ovvero

$$\overline{L_1}\sigma = \overline{L_2}\sigma = L_1'\sigma = \neg A(f(x_1), f(y))$$

Così, applicando σ a $\mathcal{C}_1$ e $\mathcal{C}_2$, i due letterali candidati $L_1, L_2 \in \mathcal{C}_1$ diventano uguali ($L_1\sigma = L_2\sigma$), e $L_1' \in \mathcal{C}_2$ diventa il letterale complementare ai due in $\mathcal{C}_1$.

Infine, il risolvente $\mathcal{R}$ si ottiene applicando a $\mathcal{C}_1\sigma$ e $\mathcal{C}_2\sigma$ la stessa regola di risoluzione usata nel caso proposizionale, o, equivalentemente, rimuovendo tutti i letterali candidati da $\mathcal{C}_1$ e $\mathcal{C}_2$, calcolando l'unione delle due clausole, e applicando a quest'ultima la sostituzione:

$$\begin{aligned}\mathcal{R} &= \big((\mathcal{C}_1 \setminus \{L_1, L_2\}) \cup (\mathcal{C}_2 \setminus \{L_1'\})\big)\sigma \\ &= \{C(z)\}\sigma \cup \{B(x_1)\}\sigma \\ &= \{C(f(y)), B(x_1)\}\end{aligned}$$

7.2 Definizione

Definizione: Siano $\mathcal{C}_1, \mathcal{C}_2$ due clausole della logica dei predicati, che si suppone non contengano variabili in comune (ma, tramite un'opportuna sostituzione, è sempre possibile riportarsi a questo caso). Se esistono dei letterali $L_1, \ldots, L_n \in \mathcal{C}_1$, $L_1', \ldots, L_m' \in \mathcal{C}_2$ e una sostituzione σ che è un MGU per $\{\overline{L_1}, \ldots, \overline{L_n}, L_1', \ldots, L_m'\}$, allora il **risolvente** di $\mathcal{C}_1$ e $\mathcal{C}_2$ (rispetto a $L_1, \ldots, L_n, L_1', \ldots, L_m'$ e all'MGU σ) è la clausola:

$$\mathcal{R} = \big((\mathcal{C}_1 \setminus \{L_1, \ldots, L_n\}) \cup (\mathcal{C}_2 \setminus \{L_1', \ldots, L_m'\})\big)\sigma$$

Come regola, la risoluzione si scrive dunque nel modo seguente:

$$\frac{\mathcal{C}_1', L_1, \ldots, L_n \qquad \mathcal{C}_2', L_1', \ldots, L_m'}{\mathcal{C}_1'\sigma, \mathcal{C}_2'\sigma}\,\text{ris} \quad \text{con } \sigma \text{ MGU per } \{\overline{L_1}, \ldots, \overline{L_n}, L_1', \ldots, L_m'\}$$

(dove $\mathcal{C}_1 = \mathcal{C}_1' \cup \{L_1, \ldots, L_n\}$ e $\mathcal{C}_2 = \mathcal{C}_2' \cup \{L_1', \ldots, L_m'\}$). Nel caso dell'esempio precedente:

$$\frac{\{A(x, f(y)), A(x, z), C(z)\} \qquad \{\neg A(f(x_1), z_1), B(x_1)\}}{\{C(f(y)), B(x_1)\}}\,\text{ris}$$

8 Correttezza della regola di risoluzione

Lemma (correttezza): Se $\mathcal{R}$ è un risolvente di $\mathcal{C}_1$ e $\mathcal{C}_2$, allora $\mathcal{R}$ è conseguenza logica di $\mathcal{C}_1$ e $\mathcal{C}_2$.

Dimostrazione: Per definizione,

$$\mathcal{R} = \big((\mathcal{C}_1 \setminus \{L_1, \ldots, L_n\}) \cup (\mathcal{C}_2 \setminus \{L'_1, \ldots, L'_m\})\big)\sigma$$

dove σ è un MGU per $\{\overline{L_1}, \ldots, \overline{L_n}, L'_1, \ldots, L'_m\}$. Allora, σ è un MGU anche per i sottoinsiemi di tale insieme, in particolare per $\{\overline{L_1}, \ldots, \overline{L_n}\}$ e $\{L'_1, \ldots, L'_m\}$, e inoltre, togliendo i complementi, rimane ancora un MGU per $\{L_1, \ldots, L_n\}$.

Applicando l'unificatore a $\{L_1, \ldots, L_n\}$ e $\{L'_1, \ldots, L'_m\}$, si ottengono due insiemi di un letterale ciascuno, e, per costruzione, questi due letterali sono uno il complemento dell'altro:

$$\{L_1, \ldots, L_n\}\sigma = \{\overline{L}\} \qquad \{L'_1, \ldots, L'_m\}\sigma = \{L\}$$

In questo modo, si è identificata una coppia complementare tra le clausole $\mathcal{C}_1\sigma$ e $\mathcal{C}_2\sigma$, e il risolvente può essere riscritto come:

$$\mathcal{R} = (\mathcal{C}_1\sigma \setminus \{\overline{L}\}) \cup (\mathcal{C}_2\sigma \setminus \{L\})$$

Adesso, bisogna mostrare che questo risolvente è conseguenza logica delle clausole di partenza, ovvero che, dati arbitrariamente un modello $\mathcal{M}$ e un assegnamento e, se $(\mathcal{M}, e) \models \mathcal{C}_1, \mathcal{C}_2$ allora $(\mathcal{M}, e) \models \mathcal{R}$.

Supponendo appunto che $(\mathcal{M}, e) \models \mathcal{C}_1, \mathcal{C}_2$, si ha innanzitutto che $(\mathcal{M}, e) \models \mathcal{C}_1\sigma$ e $(\mathcal{M}, e) \models \mathcal{C}_2\sigma$. Infatti, le eventuali variabili nelle clausole sono da considerare implicitamente quantificate universalmente (perché le clausole rappresentano una formula chiusa in forma di Skolem). Di conseguenza, le clausole sono verificate in $(\mathcal{M}, e)$ se e solo se valgono per ogni assegnamento di elementi del dominio alle variabili, e allora continuano a valere quando si sostituiscono una o più variabili con dei termini qualunque.[2]

Poi, siccome i letterali $\overline{L}$ e L sono complementari, $(\mathcal{M}, e)$ verifica sicuramente uno e uno solo dei due.

- Se è verificato $\overline{L}$, allora:

$$(\mathcal{M}, e) \models \overline{L} \implies (\mathcal{M}, e) \not\models L$$

[2] In generale, sostituendo una variabile con un termine, l'insieme di elementi da considerare può essere al massimo ridotto, e mai invece ampliato (perché a una variabile sono assegnabili tutti i possibili elementi del dominio, mentre un termine potrebbe rappresentarne solo alcuni – come caso estremo, una costante ne rappresenta uno solo), quindi una formula quantificata universalmente continua a valere.

Sapendo che $(\mathcal{M}, e) \models \mathcal{C}_2\sigma$, deve esserci almeno un letterale nella clausola (disgiunzione) $\mathcal{C}_2$ che è verificato:

$$\begin{aligned} &\implies \widetilde{\exists} L' \neq L \in \mathcal{C}_2\sigma \text{ tale che } (\mathcal{M}, e) \models L' \\ &\implies (\mathcal{M}, e) \models (\mathcal{C}_2\sigma \setminus \{L\}) \\ &\implies (\mathcal{M}, e) \models \underbrace{(\mathcal{C}_2\sigma \setminus \{L\}) \cup (\mathcal{C}_1\sigma \setminus \{\overline{L}\})}_{\mathcal{R}} \\ &\implies (\mathcal{M}, e) \models \mathcal{R} \end{aligned}$$

- Se è verificato L, il ragionamento è analogo:

$$\begin{aligned} (\mathcal{M}, e) \models L \implies &(\mathcal{M}, e) \not\models \overline{L} \\ \implies &\widetilde{\exists} L' \neq \overline{L} \in \mathcal{C}_1\sigma \text{ tale che } (\mathcal{M}, e) \models L' \\ \implies &(\mathcal{M}, e) \models (\mathcal{C}_1\sigma \setminus \{\overline{L}\}) \\ \implies &(\mathcal{M}, e) \models \mathcal{R} \end{aligned}$$

9 Derivazione per risoluzione e refutazione

Le definizioni di derivazione per risoluzione e di refutazione sono identiche al caso proposizionale (ciò che cambia è solo come è fatta la regola di risoluzione):

- *Definizione*: Una clausola $\mathcal{C}$ è **derivabile per risoluzione** da un insieme di clausole $\mathcal{S}$ se esiste una sequenza $\mathcal{C}_1, \ldots, \mathcal{C}_n$ di clausole tale che:
  - $\mathcal{C}_n = \mathcal{C}$
  - $\widetilde{\forall} i = 1, \ldots, n$ si ha che:
    - $\mathcal{C}_i \in \mathcal{S}$,
    - oppure $\mathcal{C}_i$ è la conclusione della regola di risoluzione applicata a due clausole $\mathcal{C}_j, \mathcal{C}_k$ della sequenza, con $j < i$ e $k < i$.

  Per indicare che $\mathcal{C}$ è derivabile per risoluzione da $\mathcal{S}$, si scrive $\mathcal{S} \vdash_{\mathrm{R}} \mathcal{C}$.
- *Definizione*: Una **refutazione** di $\mathcal{S}$ è una derivazione della clausola vuota $\square$ da $\mathcal{S}$. $\mathcal{S}$ è **refutabile** se $\mathcal{S} \vdash_{\mathrm{R}} \square$.

10 Teorema di validità e completezza

Teorema: Un insieme di clausole $\mathcal{S}$ è insoddisfacibile se e solo se $\mathcal{S} \vdash_{\mathrm{R}} \square$.

- La validità,
$$\mathcal{S} \vdash_{\mathrm{R}} \square \implies \mathcal{S} \text{ insoddisfacibile}$$
segue immediatamente dal lemma di correttezza della regola di risoluzione, ragionando per induzione sulla lunghezza della derivazione.

- La completezza,
$$\mathcal{S} \text{ insoddisfacibile} \implies S \vdash_{\mathrm{R}} \square$$
ha una dimostrazione abbastanza complessa (come tipicamente accade per il teorema di completezza di qualunque calcolo), che utilizza il teorema di Herbrand e un lemma fondamentale chiamato *Lifting Lemma*.