3.2 Calcolo della risoluzione

Il calcolo della risoluzione serve a stabilire l'insoddisfacibilità di proposizioni in forma normale congiuntiva. Ai fini della soddisfacibilità è equivalente considerare una congiunzione o l'insieme dei suoi congiunti, per cui le proposizioni in forma normale congiuntiva si identificano con insiemi di clausole. Un insieme S di clausole è soddisfacibile se e solo se esiste una i tale che $i \models C$ per ogni $C \in S$.

Poiché in una clausola c'è solo il connettivo $\vee$ applicato ai letterali, e si possono eliminare ripetizioni, e l'ordine non è rilevante per la commutatività della disgiunzione, una clausola è completamente determinata dall'insieme dei suoi letterali, e spesso è rappresentata come insieme di letterali $\{L_1, \ldots, L_n\}$. Noi manterremo la scrittura di $\vee$ e la rappresentazione come liste, ricordando tuttavia che, in questo paragrafo, una clausola indicherà in generale una qualunque delle clausole equivalenti che si ottengono permutando i letterali. Useremo peraltro in qualche caso, nella trattazione teorica, la notazione insiemistica, comoda per semplicità di espressione; ad esempio la notazione $C \cup \{L\}$ servirà a indicare una clausola che contiene il letterale L, non importa in quale posizione, e $C_1 \cup C_2$ la disgiunzione delle due clausole C_1 e C_2 da cui siano state eliminate le ripetizioni, dovute alla eventuale presenza di uno stesso letterale sia in C_1 sia in C_2. La lettera C in generale indicherà una clausola.

Una clausola unitaria è una clausola con un solo letterale; la terminologia insiemistica suggerisce di considerare per estensione anche la clausola con nessun letterale, o l'insieme vuoto di letterali: questa clausola, che sarebbe $\emptyset$, viene tradizionalmente indicata con $\Box$ ed è l'unica clausola che è insoddisfacibile [1]. Infatti $i \models C$ se e solo se esiste un letterale $L \in C$ tale che $i \models L$, e se C è vuota nessuna interpretazione la soddisfa, perché nessuna interpretazione dà il valore 1 a qualche letterale di C. (Se nel linguaggio c'è $\bot$ la clausola unitaria $\bot$ è insoddisfacibile, e non c'è bisogno di utilizzare $\Box$.) Si noti invece che l'insieme vuoto di clausole è soddisfacibile, perché tutte le sue clausole sono soddisfatte da (una qualunque) i.

Se S_1 è soddisfacibile e $S_2 \subseteq S_1$, allora S_2 è soddisfacibile dalle stesse interpretazioni che soddisfano S_1. Per le clausole invece se $C_1 \subseteq C_2$ allora se $i \models C_1$ allora $i \models C_2$.

[1] Da non confondere con lo stesso segno $\Box$ usato per indicare la fine di una dimostrazione.

Conviene indicare con L^c il letterale *complementare* di L, cioè L^c è $\neg P$ se L è P, e L^c è P se L è $\neg P$. Si dice anche che L e L^c formano una coppia complementare.

Definizione 3.2.1 (Regola di risoluzione)**.** *Dalle clausole* $C_1 \cup \{L\}$ *e* $C_2 \cup \{L^c\}$ *si deriva* $C_1 \cup C_2$*, schematicamente*

$$\frac{C_1 \cup \{L\} \qquad C_2 \cup \{L^c\}}{C_1 \cup C_2}$$

Le clausole $C_1 \cup \{L\}$ e $C_2 \cup \{L^c\}$ si chiamano clausole *genitrici* della regola, $C_1 \cup C_2$ la clausola *risolvente*, e L il letterale *risolto*.

Anche L^c s potrebbe chiamare il letterale risolto, ma quando è in gioco una coppia complementare è sufficiente menzionare uno solo dei due. Nelle premesse della regola, C_1 o C_2 o entrambe possono essere vuote.

Con l'iterazione della regola di risoluzione si costruiscono derivazioni per risoluzione, che si possono presentare sia come alberi che come successioni finite di clausole. Gli alberi sono alberi etichettati binari in cui come al solito identifichiamo i nodi con le clausole che sono le etichette; gli alberi sono rappresentati ora in modo naturale, con la radice in basso, le foglie in alto; le foglie non hanno predecessori, ogni altro nodo ha due predecessori.

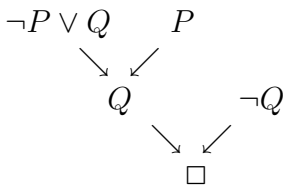
Una *derivazione per risoluzione* della clausola C dall'insieme di clausole S è un albero etichettato in cui la radice è C, le foglie sono clausole di S e ogni nodo che non sia una foglia è la risolvente dei due predecessori.

Come successione di clausole, una derivazione per risoluzione di una clausola C da S è una succesione $C_1, \ldots, C_n$ dove C_n è C e ogni C_i o $\in S$ o è la risolvente di due clausole precedenti della successione.

Se esiste una derivazione di C da S si dice che C è derivabile da S e si scrive $S \vdash C$. Una derivazione di $\Box$ da S si chiama una *refutazione* di S. La clausola vuota $\Box$ non può che essere la risolvente di due clausole unitarie L e L^c.

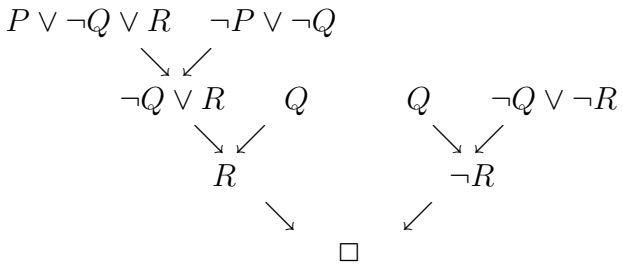
La definizione di derivazione da S ha senso anche per S infiniti; ma in ogni derivazione da S occorrono solo un numero finito di elementi di S.

Esempio
Sia $S = \{\neg P \vee Q, P, \neg Q\}$.



è una refutazione di S: $S \vdash \square$.

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



è una derivazione di $\square$ da S: $S \vdash \square$.

Il motivo per cui una derivazione di $\square$ da S si chiama una refutazione di S è che

Teorema 3.2.2 (Correttezza)**.** *Se $S \vdash \square$, allora S è insoddisfacibile.*

Dimostrazione. Se i è un'interpretazione che soddisfa S, allora i soddisfa tutte le proposizioni in una derivazione da S, che quindi non può terminare con $\square$.

Infatti se i soddisfa le premesse $C_1 \cup \{L\}$ e $C_2 \cup \{L^c\}$ di una applicazione della regola di risoluzione, allora per la prima genitrice o i soddisfa un letterale di C_1, e quindi $C_1 \cup C_2$, oppure soddisfa L; in questo caso, per la seconda genitrice i deve soddisfare un letterale di C_2, e quindi $C_1 \cup C_2$. $\square$

Prima di dimostrare la completezza, introduciamo la seguente operazione su insiemi di clausole, con la notazione

$$S^L = \{C - \{L^c\} : C \in S \, e \, L \notin C\}.$$

Analogamente per S^{L^c}, osservando che $(L^c)^c = L$,

$$S^{L^c} = \{C - \{L\} : C \in S \, e \, L^c \notin C\}.$$

L'insieme S^L si ottiene da S tenendo le clausole che non contengono né L né L^c, eliminando quelle che contengono L e modificando quelle che contengono L^c con l'eliminazione di L^c.

Lemma 3.2.3. *Se S è insoddisfacibile, anche S^L e S^{L^c} lo sono.*

Dimostrazione. Supponiamo che S^L sia soddisfatto da i (per S^{L^c} il discorso è analogo). Poiché né L né L^c occorrono in S^L, i non è definita per L e la si può estendere a i' ponendo $i'(L) = 1$.

Le clausole di S che sono anche in S^L perché non contengono né L né L^c e che sono soddisfatte da i continuano a essere soddisfatte da i'. Le clausole di S che si ottengono da clausole di S^L reinserendo il letterale L^c che era stato tolto, continuano a essere soddisfatte da i'. Le clausole di S che non sono in S^L perché contengono L sono soddisfatte da $i'(L) = 1$. □

Vale anche il viceversa (esercizio).

Se S è finito, S^L e S^{L^c} hanno meno letterali di S. Iterando il passaggio da S a S^L e S^{L^c}, determinando $(S^L)^M$, $(S^{L^c})^M$ e via riducendo, si perviene a insiemi la cui insoddisfacibilità è facilmente decidibile. L'organizzazione di questo procedimento sui vari insiemi di clausole non è semplice, e useremo S^L e S^{L^c} solo per dimostrare, per S finito, il

Completezza. Se S è insoddisfacibile, allora $S \vdash \square$. □

Dimostrazione. La dimostrazione è per induzione sul numero di letterali degli insiemi di clausole. S^L e S^{L^c} hanno meno letterali di S e sono insoddisfacibili. Per ipotesi induttiva, $S^L \vdash \square$ e $S^{L^c} \vdash \square$. Se nella refutazione di S^L si reintroduce L^c nelle clausole di S^L che sono foglie e che si ottengono da clausole di S per cancellazione di L^c, questo letterale si trasmette nelle clausole risolventi, e la derivazione diventa una derivazione di L^c da S. Analogamente, se nella refutazione di S^{L^c} si reintroduce L nelle clausole da cui era stato cancellato, si ottiene una derivazione di L da S.

Accostando queste due derivazioni e aggiungendo una risoluzione tra L e L^c, si ottiene una refutazione di S.

Per la base dell'induzione, si può considerare il caso di zero letterali, e allora S deve contenere solo la clausola $\square$, che è quindi derivabile da S. □

Prima di provare a costruire una refutazione di S, nell'intento di dimostrarne l'insoddisfacibilità, conviene eseguire su S alcune semplificazioni, se possibile.

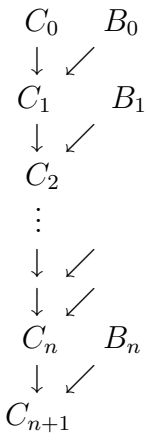
Ad esempio se in qualche clausola di S occorre un letterale L tale che il suo complementare non occorre in nessuna clausola, allora le clausole contenenti L possono essere eliminate; infatti S è soddisfacibile se e solo se il sottoinsieme S' ottenuto in questo modo è soddisfacibile: se S è soddisfacibile, anche S' lo è; viceversa, se S' che non contiene né L né L^c è soddisfatto da i, estendendo i ponendo $i'(L) = 1$ si ha un'interpretazione che soddisfa S.

Se qualche clausola di S è una tautologia, si può eliminare perché l'insieme ridotto è soddisfacibile se e solo se S lo è. Ma si può fare di più, chiedendo che in una derivazione da S ogni volta che l'applicazione della regola di risoluzione produrrebbe una risolvente che è una tautologia, perché in C_1 c'è un letterale L_1 diverso dal letterale risolto, il cui complementare L_1^c occorre in C_2, la risoluzione non venga eseguita. Con questa *restrizione della tautologia*, per cui in una derivazione da S non deve occorrere in nessun nodo una tautologia, si può dimostrare, rifacendo la dimostrazione del teorema di completezza, che il calcolo della risoluzione resta completo.

Restrizioni del genere sull'applicazione della regola di risoluzione danno origine ai cosiddetti *raffinamenti* della risoluzione.

3.3 Risoluzione lineare ordinata

La restrizione della risoluzione lineare ordinata consiste nel chiedere che ogni risolvente sia immediatamente usata come genitrice di una nuova risoluzione, in cui l'altra genitrice o appartiene all'insieme S o è una delle clausole ottenute precedentemente nella catena di risoluzioni che ha portato alla presente clausola, in modo che una derivazione da S assume il seguente aspetto di *derivazione lineare*:



Le clausole C_i si chiamano *centrali* e tutte salvo la prima, detta *top* e appartenente a S, sono risolventi; le clausole B_i sono dette *laterali* e o appartengono a S oppure sono una delle C_j con $j < i$. Per queste laterali non è rispettata la condizione che le foglie siano elementi di S, ma si può ripristinare (distruggendo la linearità) aggiungendo sopra ad esse la derivazione da cui sono dedotte.

Alla struttura lineare della derivazione si aggiunge la condizione che le clausole non sono più considerate come insiemi di letterali, cioè a meno di equivalenze logiche; ogni clausola intesa come disgiunzione è considerata diversa dalle clausole equivalenti che si ottengono per permutazioni dei letterali; le clausole sono successioni ordinate di letterali, inclusa eventualmente la successione vuota, sempre indicata con $\Box$.

Si evitano peraltro le ripetizioni di letterali convenendo che quando un letterale occorre più di una volta in una clausola, si cancellando automaticamente tutte le occorrenze salvo la prima (operazione detta anche di *merging left*).

Anche la regola di risoluzione deve essere modificata per adattarsi alla natura ordinata delle clausole, e deve indicare quale letterale scegliere come letterale risolvente e in quale delle due clausole genitrici (che anch'esse sono date in modo ordinato come prima e seconda genitrice). Per comodità di scrittura conveniamo che sia l'ultimo letterale della prima clausola genitrice (ma non cambierebbe nulla con un'altra scelta, ad esempio il primo) nel formulare la

Definizione 3.3.1 (Regola di risoluzione ordinata)**.** *Se L è l'ultimo letterale di una prima clausola, ed L^c occorre in una seconda clausola $D_1 \vee L^c \vee D_2$, dove D_1 o D_2 possono essere la clausola vuota, a significare che L^c può essere in una posizione qualunque, allora*

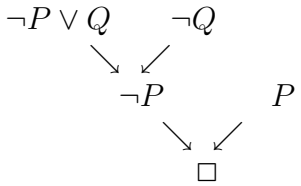
$$\frac{C \vee L \qquad D_1 \vee L^c \vee D_2}{C \vee D_1 \vee D_2}$$

Si noti che se un letterale di una clausola ha il complementare in un'altra, ma non è all'ultimo posto della sua, non può essere il letterale risolto di una risoluzione delle due clausole: ad esempio $P \vee Q$ e $\neg P \vee Q$ non possono essere risolte rispetto a P con una risoluzione ordinata.

Una *derivazione per risoluzione lineare ordinata* di C da S è una derivazione lineare di C da S in cui le clausole sono considerate ordinate e le risoluzioni sono risoluzioni ordinate tra la centrale e la laterale, nell'ordine.

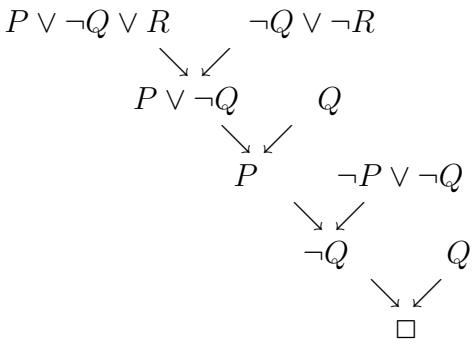
Esempio

La derivazione di p. 48 è una derivazione lineare ma non lineare ordinata. Essa può essere sostituita, per refutare $S = \{\neg P \vee Q, P, \neg Q\}$, dalla derivazione lineare ordinata:



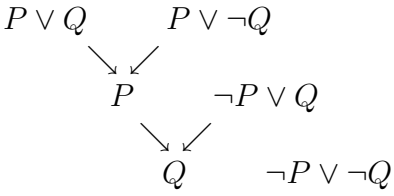
Esempio

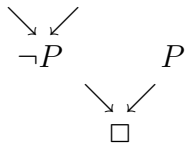
La refutazione di p. 48 di $S = \{P \vee \neg Q \vee R, \neg P \vee \neg Q, Q, \neg Q \vee \neg R\}$ non è lineare. Essa può tuttavia essere sostituita dalla refutazione lineare ordinata:



Esempio

Nelle due refutazioni degli esempi precedenti, le clausole laterali sono clausole di S. Nella seguente refutazione lineare ordinata di $\{P \vee Q, \neg P \vee Q, P \vee \neg Q, \neg P \vee \neg Q\}$





l'ultima clausola laterale P è una clausola centrale precedente.

Non sorprenda il secondo esempio, perché vale la completezza per la risoluzione lineare ordinata. La dimostrazione si svolge come quella per la risoluzione ordinaria, manipolando ed attaccando derivazioni che esistono per ipotesi induttiva per insiemi ridotti; questa volta tuttavia le due derivazioni vanno attaccate in serie per formare un albero lineare, e quindi bisogna poter controllare il *top* della seconda. Occorre poter affermare cioè che se un insieme è insoddisfacibile esiste una sua refutazione che ha come *top* una clausola particolare scelta da noi.

Questo è vero, come vedremo, nella forma: se S è insoddisfacibile e $S - \{C\}$ è soddisfacibile, allora esiste una refutazione di S con *top* C. Il risultato è anche operativamente significativo e importante, perché gli insiemi di clausole su cui normalmente si lavora derivano da una situazione del genere: si ha una teoria $\{A_1, \ldots, A_n\}$ che si sa o si presume consistente, e si vuole sapere se $A_1, \ldots, A_n \models B$, trasformando il problema in quello della insoddisfacibilità di $\{A_1, \ldots, A_n, \neg B\}$; allora $\neg B$ è il candidato *top* della refutazione (almeno se $A_1, \ldots, A_n, \neg B$ sono clausole).

Per la dimostrazione serve una versione più forte, anche se non operativa, in cui si possa assumere una clausola qualunque come *top*; a questa si arriva con il concetto di *insieme insoddisfacibile minimale*. Un insieme S di clausole si dice insoddisfacibile minimale se S è insoddisfacibile e ogni $S' \subset S$ è soddisfacibile.

Ogni insieme insoddisfacibile di clausole contiene un sottoinsieme insoddisfacibile minimale (esercizio). Anzi, per ogni insieme S insoddisfacibile e tale che $S - \{C\}$ è soddisfacibile, esiste un sottoinsieme insoddisfacibile di S che contiene C ed è tale che ogni suo sottoinsieme che contiene C è soddisfacibile (esercizio).

Possiamo ora enunciare

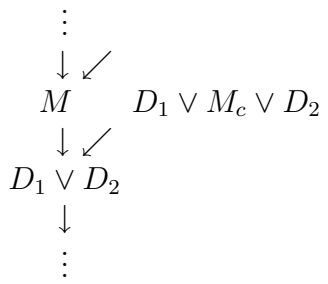
Completezza. Se S è insoddisfacibile e $S - \{C\}$ è soddisfacibile, allora esista una refutazione lineare ordinata di S con top C. □

Dimostrazione. Possiamo supporre che S sia insoddisfacibile minimale, nel senso detto sopra, vale a dire ogni suo sottoinsieme proprio che contiene C è soddisfacibile.

Sia C della forma $M \vee D \vee L$ (se C è unitaria uguale a L si procede come sotto, identificando M ed L). Si consideri S^{M^c} che è insoddisfacibile, e si osservi che $S^{M^c} - \{D \vee L\}$ è soddisfacibile. Infatti se i è un'interpretazione che soddisfa $S - \{C\}$, ma non C altrimenti soddisferebbe S, allora $i(M) = 0$; allora cancellando M da clausole di $S - \{C\}$ soddisfatte da i per ottenere $S^{M^c} - \{D \vee L\}$ si ottengono clausole ancora soddisfatte da i.

Per ipotesi induttiva, esiste una refutazione lineare ordinata di S^{M^c} con *top* $D \vee L$. Reintroduciamo M nelle clausole da cui era stato tolto, nella stessa posizione in cui era quando è stato tolto, per riavere le stesse clausole di S; ma M è reintrodotto quindi all'inizio del *top*, ed essendo all'inizio M si trasmette lungo l'albero di derivazione restando sempre al primo posto e non interferendo con le risoluzioni compiute. Si ottiene così una derivazione per risoluzione lineare ordinata da S di M con *top* C.

In S esiste una clausola che contiene M^c, perché i letterali si presentano sempre in coppie complementari, altrimenti li elimineremmo preliminarmente, e sia $D_1 \vee M^c \vee D_2$. $S - \{D_1 \vee M^c \vee D_2\}$ contiene C ed è soddisfacibile, ma una j che lo soddisfi deve dare il valore zero a M^c. Se consideriamo ora S^M abbiamo come sopra che questo è insoddisfacible e $S^M - \{D_1 \vee D_2\}$ è soddisfacibile; quindi per ipotesi induttiva esiste una refutazione lineare ordinata di S^M con *top* $D_1 \vee D_2$. Appendiamo questa alla precedente derivazione di M da S con il legame



Nella parte sotto $D_1 \vee D_2$ ci sono in generale clausole laterali di S^M; se reintroduciamo M^c otteniamo clausole di S, ma il letterale M^c che passa nelle centrali può a un certo punto essere l'ultimo di una centrale, e bloccare l'applicazione della risoluzione eseguita nella orginaria derivazione con *top* $D_1 \vee D_2$ da S^M; ad esempio potrebbe succedere che da

$$\begin{array}{ll} \downarrow & \\ E & F \\ \downarrow \swarrow & \\ G & \end{array}$$

si ottiene

$$\begin{array}{ll} \downarrow & \\ E & F \vee M^c \\ \downarrow \swarrow & \\ G \vee M^c & \end{array}$$

e non si può proseguire con la risoluzione sull'ultimo letterale di G; ma a questo punto possiamo inserire una risoluzione con la clausola centrale precedente M, e ripristinare l'originaria centrale

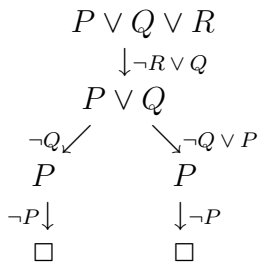
$$\begin{array}{ll} \downarrow & \\ E & F \vee M^c \\ \downarrow \swarrow & \\ G \vee M^c & M \\ \downarrow \swarrow & \\ G & \end{array}$$

Alla fine si ottiene una derivazione per risoluzione lineare ordinata di $\Box$ da S con *top* C. $\Box$

Si noti che se C è unitaria S^{M^c} è $\Box$; in pratica serve solo la seconda derivazione da S^M, con *top* C.

La risoluzione lineare ordinata è compatibile anche con la restrizione della tautologia, perché la dimostrazione si estende a questa ulteriore restrizione: se le refutazioni assunte per ipotesi induttiva soddisfano la restrizione della tautologia, le operazioni eseguite su di esse per combinarle in una refutazione di S non introducono alcuna tautologia.

Una derivazione lineare può essere ridotta da un albero a un ramo se le frecce che partono dalle clausole laterali sono cancellate e le clausole stesse sono ad esempio scritte a fianco delle frecce che uniscono le due centrali consecutive. Ma allora con un albero si possono rappresentare tutte le derivazioni con *top* fissato, ogni ramo rappresentando una derivazione lineare. Ad esempio, per l'insieme $\{P \vee Q \vee R, \neg R \vee Q, \neg Q, \neg Q \vee P, \neg P\}$ tutte le refutazioni lineari ordinate da S con *top* $P \vee Q \vee R$ sono rappresentate dall'albero



Senza il vincolo dell'ordine in generale l'albero delle derivazioni sarebbe molto più folto.

3.4 Esercizi

1. Descrivere l'insieme S di clausole associate alla forma normale congiuntiva delle seguenti proposizioni:

   $(P \rightarrow Q \vee R) \wedge \neg R \wedge (Q \rightarrow \neg P)$

   $\neg(P \rightarrow Q) \vee (P \wedge \neg Q)$

   $(P \rightarrow Q) \rightarrow ((\neg P \rightarrow Q) \rightarrow \neg Q)$

2. Determinare S^P e $S^{\neg P}$ per i seguenti insiemi di clausole:

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

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

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

3. Dimostrare che se S^L e S^{L^c} sono insoddisfacibili, anche S lo è.

4. Spiegare come devono essere fatti gli insiemi di clausole a cui si perviene iterando le riduzioni S^L e S^{L^c} perché S sia insoddisfacibile.

5. Verificare l'insoddisfacibilità o meno dei seguenti insiemi di clausole calcolando iterativamente le riduzioni S^L e S^{L^c}:

   $\{P \vee R, Q \vee \neg R, \neg Q, \neg P \vee T, \neg U, U \vee \neg T\}$

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

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

6. Cercare una refutazione per risoluzione lineare dei seguenti insiemi di clausole:

   $\{P \vee R, Q \vee \neg R, \neg Q, \neg P \vee T, \neg U, U \vee \neg T\}$

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

7. Trovare una refutazione lineare ordinata dei seguenti insiemi di clausole:

   $\{P \vee R, Q \vee \neg R, \neg Q, \neg P \vee T, \neg U, U \vee \neg T\}$

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

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

8. Cercare una refutazione per risoluzione non lineare dei seguenti insiemi di clausole:

   $\{P \vee R, Q \vee \neg R, \neg Q, \neg P \vee T, \neg U, U \vee \neg T\}$

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

9. Verificare che per l'insieme insoddisfacibile $\{P \vee Q \vee R, \neg P \vee Q, \neg Q, \neg R \vee Q, \neg P \vee \neg R\}$ non c'è una refutazione lineare ordinata con *top* $\neg P \vee \neg R$ che rispetti anche la restrizione della tautologia, e dare una spiegazione.

10. Trovare tutte le refutazioni lineari ordinate di $S = \{P \vee \neg Q \vee R, \neg P \vee \neg Q, Q, \neg Q \vee \neg R\}$.

11. Rappresentare per ogni clausola di S dell'esercizio precedente l'albero delle derivazioni lineari, anche non ordinate, da S che hanno quella clausola come *top*.

12. Dimostrare che ogni insieme insoddisfacibile di clausole contiene un sottoinsieme insoddisfacibile minimale.

3.5 Clausole di Horn e Programmazione Logica

Una clausola si chiama *clausola di Horn* se contiene al più un letterale positivo. Le clausole si possono classificare in *negative* se tutti i loro letterali sono negativi, *positive* se tutti i loro letterali sono positivi, e altrimenti *miste*.

Una clausola di Horn può dunque essere negativa, e allora si chiama anche *goal*, per il motivo che vedremo, oppure positiva, ma allora è unitaria, e si chiama anche *fatto*, oppure mista, ma con un solo letterale positivo, e si chiama anche *legge*.

Un insieme di clausole di Horn, se è insoddisfacibile, non può contenere soltanto leggi e fatti, perché altrimenti l'interpretazione *positiva*, quella che dà il valore 1 a tutte le lettere, lo soddisferebbe; deve contenere quindi almeno un *goal*. Analogamente non può contenere solo clausole negative o miste, altrimenti l'interpretazione *negativa*, quella che dà il valore 0 a tutte le lettere, lo soddisferebbe.

Per gli insiemi di clausole di Horn è completa una ulteriore restrizione della risoluzione lineare ordinata, quella della *risoluzione a input*, o input–risoluzione (che si potrebbe definire anche solo sulla base della risoluzione lineare), data dalla

Definizione 3.5.1. *Una derivazione per risoluzione lineare ordinata si dice una input–derivazione se nessuna clausola laterale è uguale a una centrale precedente.*

Una input–derivazione di $\square$ da S si chiama una input–refutazione di S.

Il vantaggio di una input–derivazione su una derivazione lineare ordinata è che non c'è bisogno di registrare e conservare in memoria, in un'esecuzione meccanica, le varie clausole centrali generate, ma solo l'ultima, insieme alle clausole originali di S.

Completezza. Se S è un insieme insoddisfacibile di clausole di Horn, esiste una input–refutazione di S. $\square$

Dimostrazione. Si consideri una refutazione lineare ordinata di S che abbia come *top* un *goal*. Una tale derivazione esiste, per la completezza della risoluzione lineare ordinata, e supponendo S insoddisfacibile minimale, o di essersi ristretti a un suo sottonsieme insoddisfacibile minimale, esiste con *top* qualunque; ma ogni insieme insoddisfacibile di clausole di Horn contiene almeno un *goal*. Scegliamo un *top* negativo.

La clausola *top* negativa non può che essere risolta con una clausola mista o positiva, ma la risolvente è di nuovo negativa. Iterando, tutte le centrali sono negative, e nessuna può essere risolta con una centrale precedente, anch'essa negativa.

Quindi una refutazione lineare ordinata di S con *top* una clausola negativa soddisfa automaticamente la restrizione dell'input. $\square$

Si potrebbe definire una nozione duale rispetto alle clausole di Horn, clausole in cui occorre al più un letterale negativo. Anche per queste vale la

completezza della input–risoluzione, con derivazioni che hanno come *top* una clausola positiva.

Nella programmazione logica si usano clausole di Horn, di linguaggi predicativi, ma con un calcolo soggiacente che può essere illustrato con la logica proposizionale, e non è altro che la input–risoluzione, in altra notazione.

Si torna ad usare il condizionale, anche se rovesciato, e una clausola mista $\neg Q_n \vee \ldots \vee \neg Q_1 \vee P$, che è logicamente equivalente a $Q_n \wedge \ldots \wedge Q_1 \rightarrow P$ sarà rappresentata da

$$P \leftarrow Q_1, \ldots, Q_n$$

anche se P fosse in un posto qualunque. P si chiama *testa* e $Q_1, \ldots, Q_n$ *corpo* della legge.

Quindi per uniformità di notazione una clausola unitaria P sarà rappresentata da

$$P \leftarrow$$

e un *goal* $\neg Q_n \vee \ldots \vee \neg Q_1$ da

$$\leftarrow Q_1, \ldots, Q_n$$

Tralasciamo altri dettagli zuccorosi dei veri e propri linguaggi di programmazione come PROLOG, come un punto alla fine di ogni clausola o un punto interrogativo per il *goal*. Un insieme $\mathcal{P}$ di clausole di Horn positive o miste scritte nel modo indicato si chiama anche *programma*.

Esempio

L'insieme di clausole di Horn $S = \{P \vee \neg R \vee \neg Q, Q, R\}$ si rappresenta con il programma

$$\begin{array}{l} Q \leftarrow \\ R \leftarrow \\ P \leftarrow Q, R \end{array}$$

Un'*interrogazione* di un programma si realizza chiedendo se una proposizione atomica, o una congiunzione di proposizioni atomiche, sono conseguenza logica del programma. Questo equivale a chiedere se aggiungendo la loro negazione all'insieme di clausole del programma si ottiene un insieme insoddisfacibile. Al programma si aggiunge pertanto l'interrogazione nella forma di un *goal* semplice, la negazione di una lettera, ovvero una clausola negativa.

Esempio

Il precedente programma può essere interrogato aggiungendo il *goal* $\neg P$, ottenendo

$$
\begin{array}{l}
Q \leftarrow \\
R \leftarrow \\
P \leftarrow Q, R \\
\leftarrow P
\end{array}
$$

Il teorema sulla completezza della input–risoluzione suggerisce di partire per una input–refutazione dal *goal*. Nella notazione della programmazione logica, che seguiamo sull'esempio di sopra, la risoluzione di $\neg P$ con la clausola $P \vee \neg R \vee \neg Q$, che dà origine alla nuova clausola negativa, nuovo *goal*, $\neg R \vee \neg Q$, viene sostituita dalla identificazione del P del *goal* $\leftarrow P$ con la testa di una legge o con un fatto, in questo caso la legge $P \leftarrow Q, R$, e nello spostamento del problema del *soddisfacimento* del *goal* P con quello del soddisfacimento del nuovo *goal* multiplo $\leftarrow Q, R$. Formalmente, P viene rimpiazzato in $\leftarrow P$ dal corpo Q, R della clausola con la cui testa P si identifica. Si dice anche, in vista dell'estensione ai linguaggi predicativi, che P *unifica* con la testa di una clausola.

Un *goal* multiplo si soddisfa cercando di soddisfare il primo *sottogoal* Q con lo stesso procedimento di prima, che corrisponde a cercare di risolvere la clausola $\neg R \vee \neg Q$ rispetto al suo ultimo letterale. Poiché Q unifica con il fatto $Q \leftarrow$, Q è soddisfatto, e il *goal* $\leftarrow Q, R$ si riduce, sostituendo a Q il corpo del fatto $Q \leftarrow$, che è vuoto, a $\leftarrow R$. Anche questo *sottogoal* R unifica con il fatto $R \leftarrow$, ed è perciò soddisfatto. Il *goal* diventa $\leftarrow$, o la clausola vuota, e il *goal* originario è soddisfatto.

Se il procedimento non arrivasse a questa conclusione, ma si interrompesse con qualche *sottogoal* non soddisfatto, o entrasse in ciclo, non si direbbe ancora che il *goal* $\leftarrow P$ non è soddisfatto, o *fallisce*; si esplorerebbero prima altre possibilità di unificazione dei vari *sottogoal*, e solo se tutte fallissero si direbbe che il *goal* è fallito, il che corrisponde alla non esistenza di una input–refutazione con il *goal* come *top*, o alla soddisfacibilità delle clausole del programma più $\neg P$.

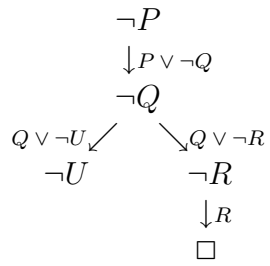
Si faccia attenzione che dire nel contesto della programmazione che un *goal* $\leftarrow P$ è soddisfatto dal programma $\mathcal{P}$ significa dire nel contesto logico che l'insieme $\mathcal{P} \cup \{\neg P\}$ è insoddisfacibile

Ad esempio, per il programma

$$P \leftarrow Q$$
$$Q \leftarrow U$$
$$Q \leftarrow R$$
$$R \leftarrow$$

il *goal* $\leftarrow P$ unifica con la prima clausola e il nuovo *goal* è $\leftarrow Q$, che unifica con la seconda, ma diventa $\leftarrow U$ e la ricerca si interrompe. Se si torna indietro e si unifica $\leftarrow Q$ con la testa della terza clausola, si ottiene $\leftarrow R$ che è soddisfatto unificando con $R \leftarrow$.

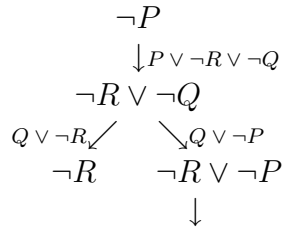
Le due strade seguite corrispondono ai due rami del seguente albero delle derivazioni lineari ordinate da $\{P \vee \neg Q, Q \vee \neg U, Q \vee \neg R, R, \neg P\}$ con *top* $\neg P$:



Invece il programma

$$P \leftarrow Q, R$$
$$Q \leftarrow R$$
$$Q \leftarrow P$$

interrogato con il *goal* $\leftarrow P$ esplora due strade, una che fallisce con il *sottogoal* $\leftarrow R$, l'altra che entra in ciclo, in corrispondenza al seguente albero delle derivazioni da $\{P \vee \neg R \vee \neg Q, Q \vee \neg R, Q \vee \neg P\}$ con *top* $\neg P$,



dove il ramo di sinistra è terminato e quello di destra con $P \vee \neg R \vee \neg Q$ dà origine a un ciclo infinito.

Esempio

L'insieme di clausole di Horn $S = \{P \vee \neg R \vee \neg Q, Q \vee \neg T \vee \neg U, R \vee \neg U, U, T, Q\}$ è soddisfacibile, con l'intepretazione positiva, in quanto non ci sono clausole negative. $S \cup \{\neg P\}$ è insoddisfacibile. Lo verifichiamo nella forma della programmazione logica considerando il programma associato a S:

$$
\begin{array}{l}
P \leftarrow Q, R \\
Q \leftarrow U, T \\
R \leftarrow U \\
U \leftarrow \\
T \leftarrow \\
Q \leftarrow
\end{array}
$$

a cui aggiungiamo il *goal*

$$\leftarrow P$$

e calcoliamo la successione dei nuovi *goal*

$$
\begin{array}{lll}
1 & P \leftarrow Q, R & \\
2 & Q \leftarrow U, T & \\
3 & R \leftarrow U & \\
4 & U \leftarrow & \\
5 & T \leftarrow & \\
6 & Q \leftarrow & \\
7 & \leftarrow P & P \text{ unifica con } 1 \\
8 & \leftarrow Q, R & Q \text{ unifica con } 2 \\
9 & \leftarrow U, T, R & U \text{ unifica con } 4 \\
10 & \leftarrow T, R & T \text{ unifica con } 5 \\
11 & \leftarrow R & R \text{ unifica con } 3 \\
12 & \leftarrow U & U \text{ unifica con } 4 \\
13 & \leftarrow &
\end{array}
$$

Si noti che al passo 8 Q unifica anche con 6, per cui si sarebbe potuto continuare con

$$
\begin{array}{ll}
9 & \leftarrow R \\
10 & \leftarrow U \\
11 & \leftarrow
\end{array}
$$

Queste sono le due input–refutazioni dell'insieme $S = \{P \vee \neg R \vee \neg Q, Q \vee \neg T \vee \neg U, R \vee \neg U, U, T, Q, \neg P\}$ con *top* $\neg P$. Il motore inferenziale dei linguaggi di programmazione logica si basa tuttavia oltre che sulla regola di risoluzione anche su alcune restrizioni pratiche, per ragioni di controllo; ad esempio le clausole del programma sono date in modo ordinato, e per soddisfare un *sottogoal* le clausole di programma sono passate in rassegna nell'ordine fissato, a cercare una clausola con testa che unifichi con il *sottogoal*, e viene scelta la prima che si incontra. Questo vincolo distrugge la completezza, perchè ad esempio il programma

$$\begin{array}{l} P \leftarrow Q \\ Q \leftarrow P \\ P \leftarrow \end{array}$$

entra in ciclo con il *goal* $\leftarrow P$ e il *goal* non è soddisfatto, nonostante P sia conseguenza logica di $\{P \rightarrow Q, Q \rightarrow P, P\}$. Per via di queste e altre restrizioni, si impone un'attenta scrittura ordinata dei programmi, o l'introduzione di comandi permessi dal linguaggio che impediscano di tornare a clausole già considerate.

Il procedimento inferenziale sopra descritto, nella notazione della programmazione logica, corrispondente alla input–risoluzione, si chiama anche nella letteratura specifica SLD–*risoluzione*, SLD per *Selection Left for Definite clauses*: "clausole definite è un altro nome per le clausole di Horn; la "scelta a sinistra si riferisce alla scelta di prendere in considerazione sempre il primo, a sinistra, *sottogoal* di un *goal* multiplo; la scelta corrisponde a quella della risoluzione ordinata di considerare l'ultimo, a destra, letterale come letterale da risolvere, secondo la rappresentazione del *goal* $\neg Q_n \vee \ldots \vee \neg Q_1$ come $\leftarrow Q_1, \ldots, Q_n$.

3.6 Esercizi

1. Verificare e spiegare perché non può esistere alcuna input–refutazione dell'insieme $S = \{P \vee Q, \neg P \vee Q, \neg Q \vee P, \neg P \vee \neg Q\}$ dell'esempio di p. 52-3.

2. Verificare che l' insieme insoddisfacibile di clausole $S = \{P \vee \neg R, Q \vee \neg R, R \vee \neg Q \vee P, \neg P \vee \neg Q, Q \vee R\}$ non ammette una input-refutazione.

3. Spiegare perché l'insieme $\{P \vee Q, \neg P \vee \neg Q, P, Q\}$ pur non essendo un insieme di clausole di Horn ammette una input–refutazione.

4. Trovare una input–refutazione dell'insieme $S = \{\neg P \vee \neg Q \vee \neg R, \neg P \vee \neg Q \vee R, Q \vee \neg U, U \vee \neg P, P\}$.

5. Trovare una input–refutazione dell'insieme $S = \{P \vee Q \vee R, \neg R \vee P \vee Q, \neg Q \vee P, \neg P\}$ con *top* $P \vee Q \vee R$ e spiegare perché è prevedibile che esista.

6. Trovare tutte le input–refutazioni dell'insieme $S = \{\neg P \vee \neg Q \vee \neg R, \neg P \vee R \vee \neg Q, Q \vee \neg R, R \vee \neg P, \neg Q \vee \neg R, P\}$ e rappresentarle con gli alberi di tutte le input–derivazioni con *top* fissato.

7. Riscrivere gli insiemi di clausole di Horn dei precedenti esercizi nella notazione della programmazione logica e svolgerli nella forma di interrogazioni di programmi.

8. Dato il programma

$$\begin{aligned} P &\leftarrow Q, T \\ U &\leftarrow R \\ R &\leftarrow P, Q \\ T &\leftarrow Q \\ Q &\leftarrow \end{aligned}$$

   dimostrare che il *goal* $\leftarrow P$ è soddisfatto, e che anche $\leftarrow Q, R$ lo è.

9. Dimostrare che anche i *goal* $\leftarrow U$ e $\leftarrow T$ sono soddisfatti, e che questo significa che non solo l'interpretazione positiva soddisfa il programma, ma che è l'unica che lo soddisfa.

10. Dimostrare che se in un programma ci sono solo fatti, solo i *goal* che unificano con quei fatti sono soddisfatti, e che se in un programma ci sono solo leggi nessun *goal* è soddisfatto; questo significa che per ogni *goal* $\leftarrow P$ l'insieme di $\neg P$ e delle clausole del programma è soddisfacibile; quale è l'interpretazione che lo soddisfa?

11. Verificare che per il programma

$$\begin{aligned} U &\leftarrow R \\ R &\leftarrow P, U \\ P &\leftarrow U \\ U &\leftarrow \end{aligned}$$

il *goal* $\leftarrow U$ non è soddisfatto se si impone di prendere in considerazione le clausole di programma solo nell'ordine in cui sono scritte.