Validità e soddisfacibilità

Una sentenza è valida se è vera in *tutti* i modelli,
p.e., $Vero$, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

La validità è collegata all'inferenza tramite il Teorema di Deduzione:
$KB \models \alpha$ se e solo se $(KB \Rightarrow \alpha)$ è valida

Una sentenza è soddisfacibile se è vera in qualche modello
p.e., $A \vee B$, C

Una sentenza è insoddisfacibile se in nessun modello è vera
e.g., $A \wedge \neg A$

Soddisfacibilità è connessa all'inferenza dal seguente risultato:
$KB \models \alpha$ se e solo se $(KB \wedge \neg\alpha)$ è insoddisfacibile
cioè, prova α tramite *reductio ad absurdum*

Metodi di prova

I metodi di prova si suddividono in (approssimativamente) due tipologie:

Applicazione di regole di inferenza

– Generazione (corretta) di nuove sentenze a partire da vecchie sentenze

– Prova = una sequenza di applicazione di regole di inferenza

Può utilizzare regole di inferenza come operatori in un algoritmo standard di ricerca

– Tipicamente richiede la rappresentazione delle sentenze in una forma normale

Model checking

enumerazione della tabella di verità (sempre esponenziale in n)

backtracking migliorato, p.e., Davis–Putnam–Logemann–Loveland

ricerca euristica nello spazio dei modelli (corretta ma incompleta)

p.e., algoritmi hill-climbing basati sulla minimizzazione dei conflitti

Forward e backward chaining

Forma di Horn (ristretta)

KB = *congiunzione* di *Clausole di Horn*

Clausola di Horn =

- ♢ simbolo proposizionale; o
- ♢ (congiunzione di simboli) $\Rightarrow$ simbolo

P.e., $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$

Modus Ponens (per la Forma di Horn): completeto per basi di conoscenza in forma di Horn

$$\frac{\alpha_1, \ldots, \alpha_n, \qquad \alpha_1 \wedge \cdots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

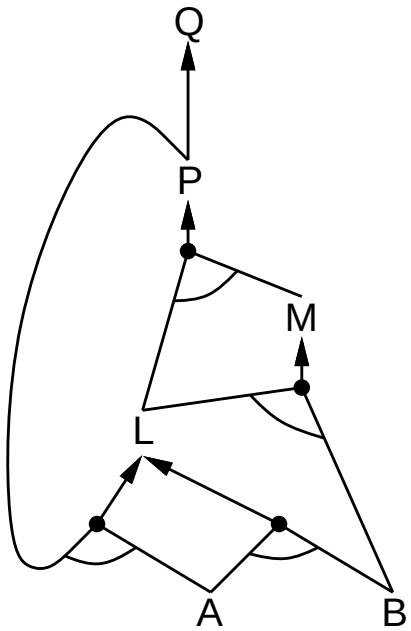
Può essere usato con forward chaining o backward chaining.

Questi algoritmi sono molto immediati e hanno complessità *lineare* in tempo

Forward chaining

Idea: applicare ogni regola le cui premesse sono soddisfatte in KB,
aggiungere le conclusioni a KB, fino a quando non si trova la query

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



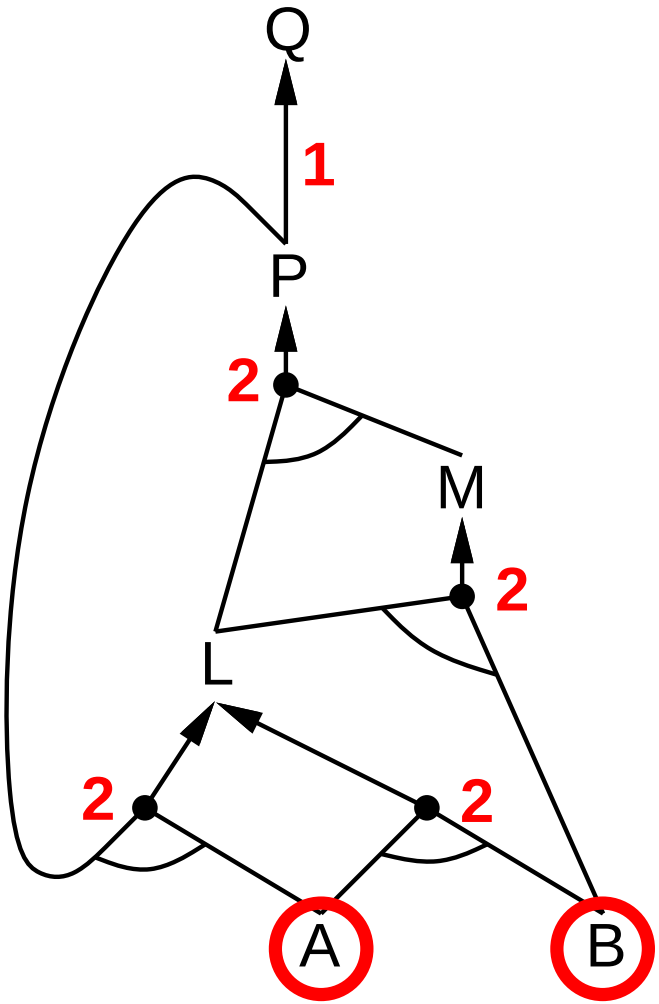
Algoritmo di Forward chaining

function CP-CA-IMPLICA?(KB, q) **returns** *true* or *false*

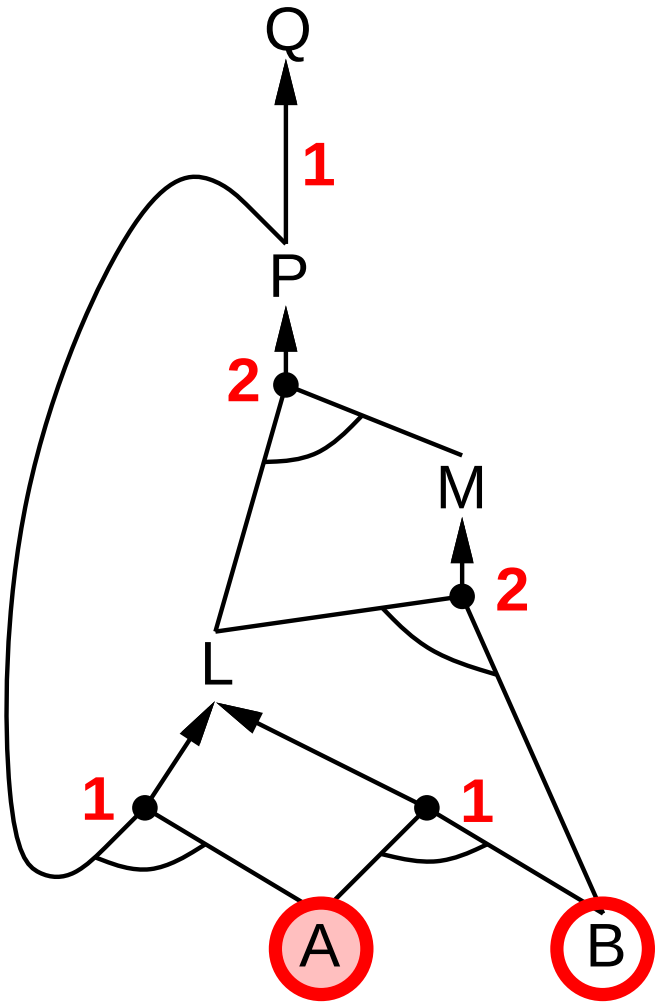
local variables: *conto*, una tabella, indicizzata per clausola, che contiene inizialmente il numero di premesse
inferiti, una tabella, indicizzata per simbolo, in cui ogni elemento è inizialmente *false*
agenda, una lista di simboli, che contiene inizialmente quelli noti come veri nella KB

while *agenda* non è vuota **do**
  $p \leftarrow$ POP(*agenda*)
  unless *inferiti*[p] **do**
    inferiti[p] $\leftarrow$ *true*
    for each clausola di Horn c in cui appare la premessa p **do**
      decrementa *conto*[c]
      if *conto*[c] $= 0$ **then do**
        if TESTA[c] $= q$ **then return** *true*
        PUSH(TESTA[c], *agenda*)
return *false*

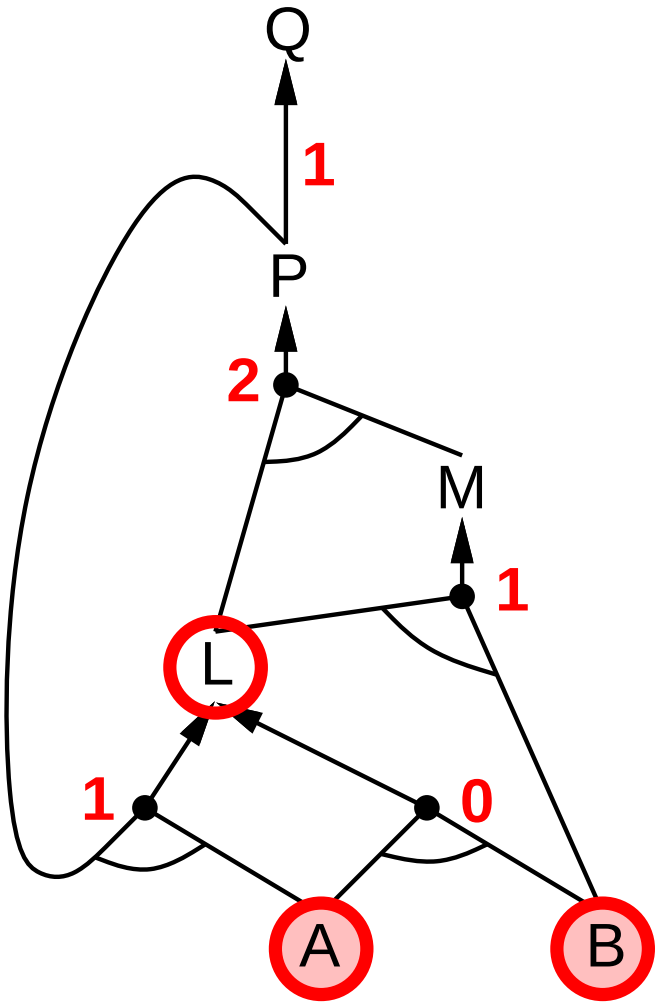
Esempio di Forward chaining



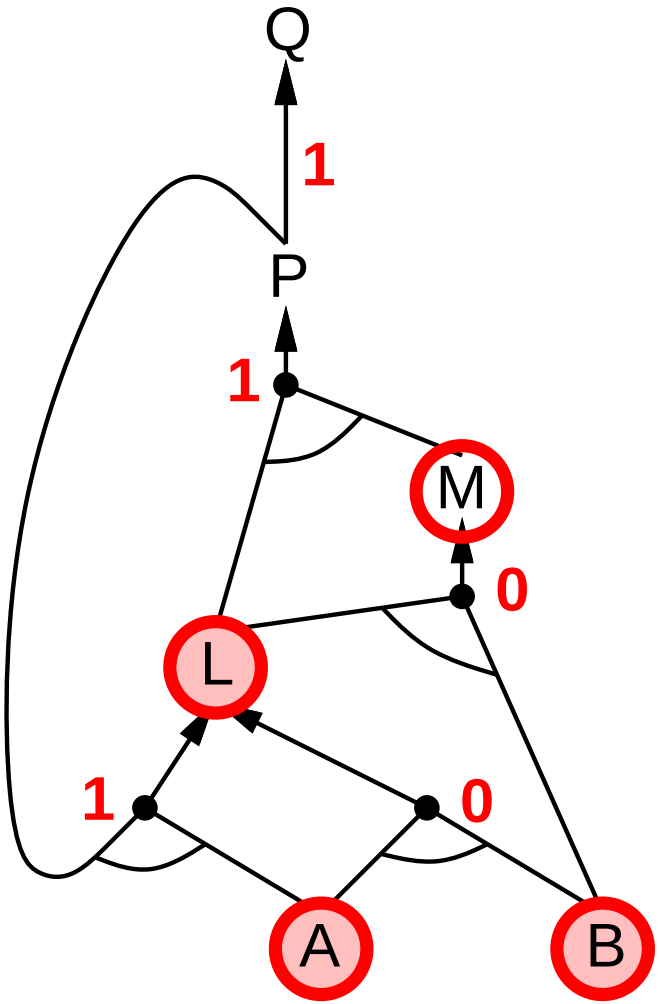
Esempio di Forward chaining



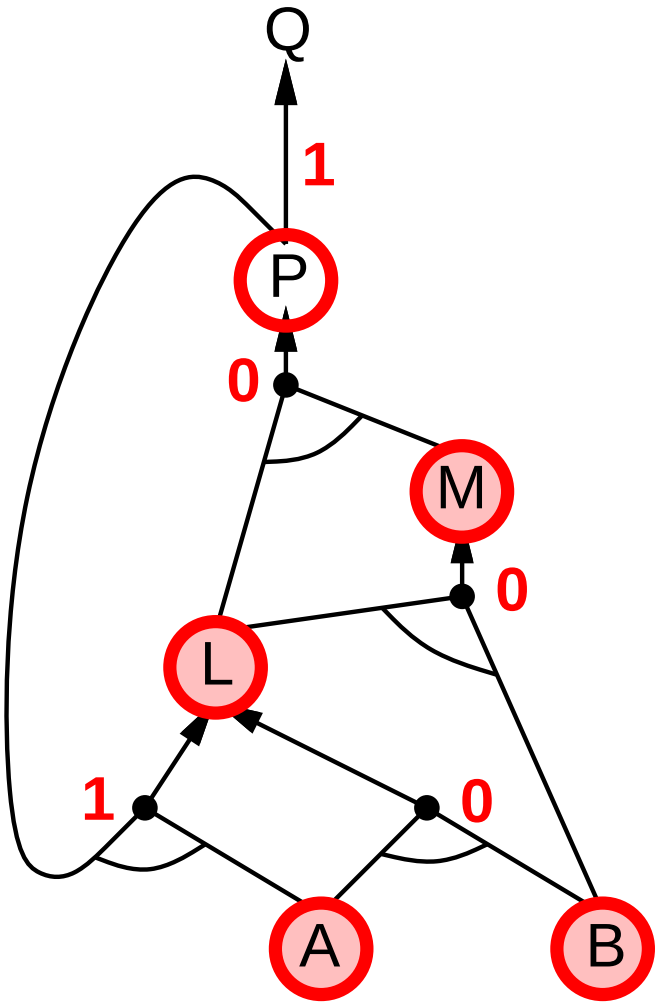
Esempio di Forward chaining



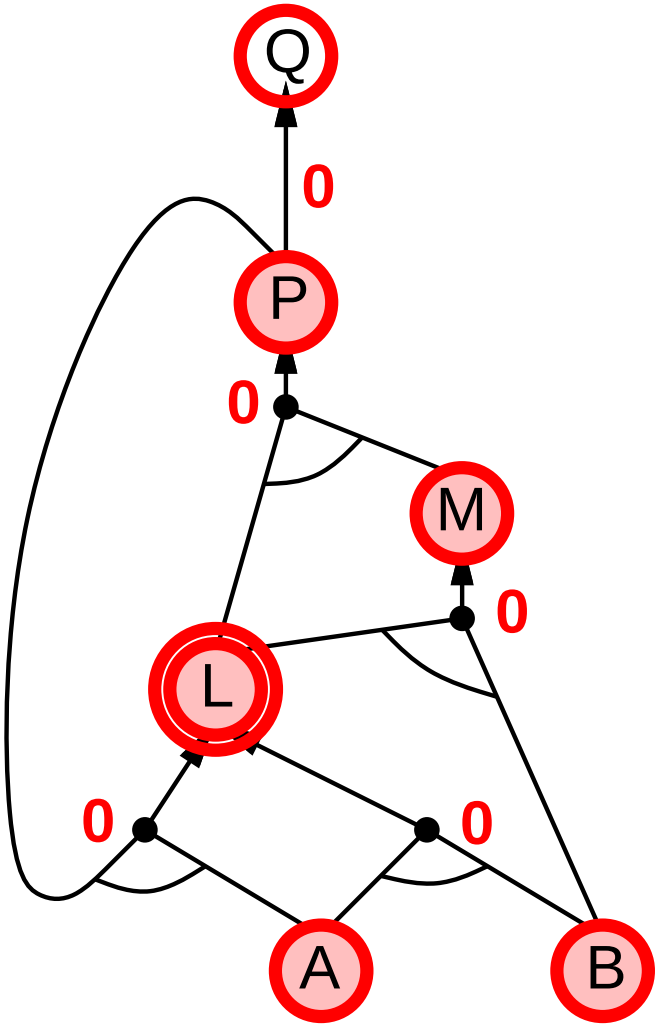
Esempio di Forward chaining



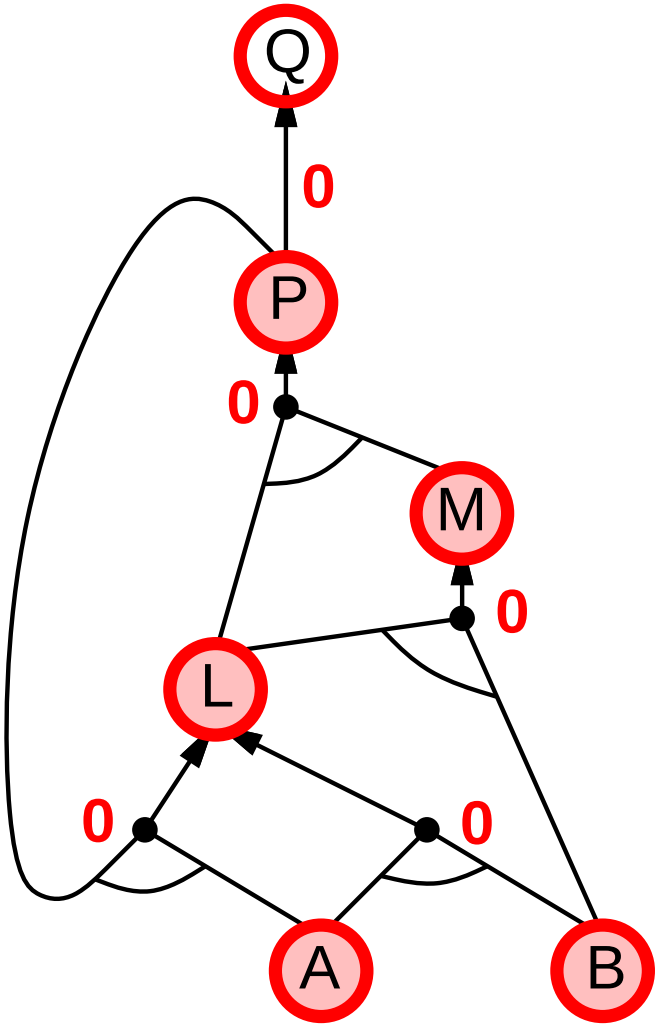
Esempio di Forward chaining



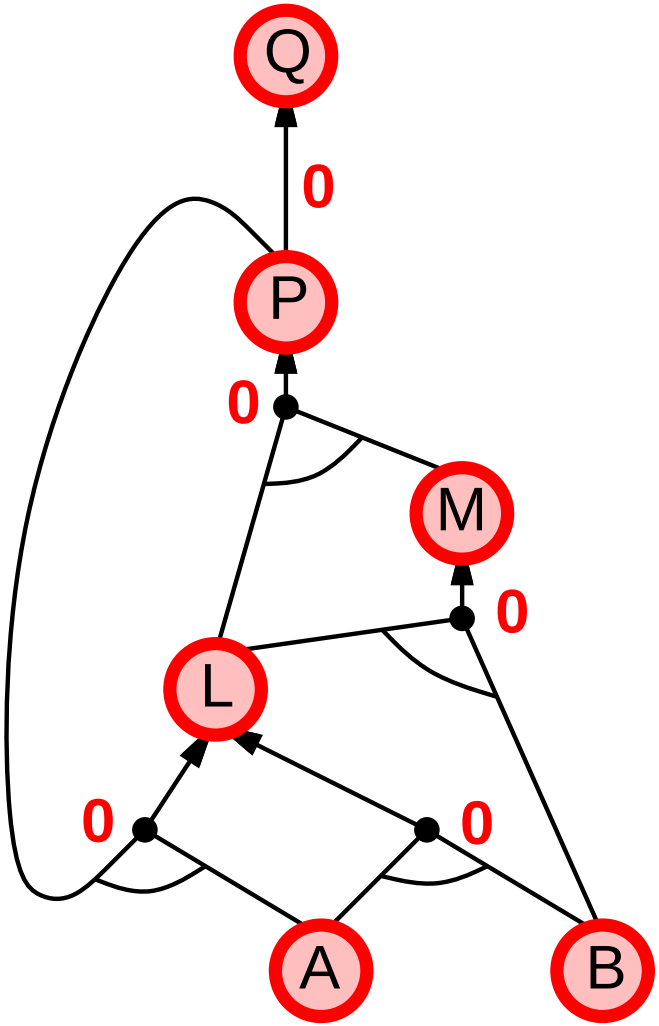
Esempio di Forward chaining



Esempio di Forward chaining



Esempio di Forward chaining



Prova di completezza

FC deriva ogni sentenza atomica che è conseguenza di KB

1. FC raggiunge un punto fisso dove nessuna nuova sentenza atomica è derivata

2. Si consideri lo stato finale come un modello m, assegnando vero/falso ai simboli

3. Ogni clausola nella KB originale è vera in m
   Prova: Si supponga che una clausola $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ sia falsa in m
   Allora $a_1 \wedge \ldots \wedge a_k$ è vera in m e b è falsa in m
   Perciò l'algoritmo non ha raggiunto un punto fisso!

4. Quindi m è un modello per KB

5. Se $KB \models q$, q è vera in *ogni* modello di KB, incluso m

Backward chaining

Idea: si lavora all'indietro (backwards) a partire dalla query q:

per provare q per mezzo di BC,

controlla se q è già conosciuta, o

prova tramite BC tutte le premesse di una regola che deriva q

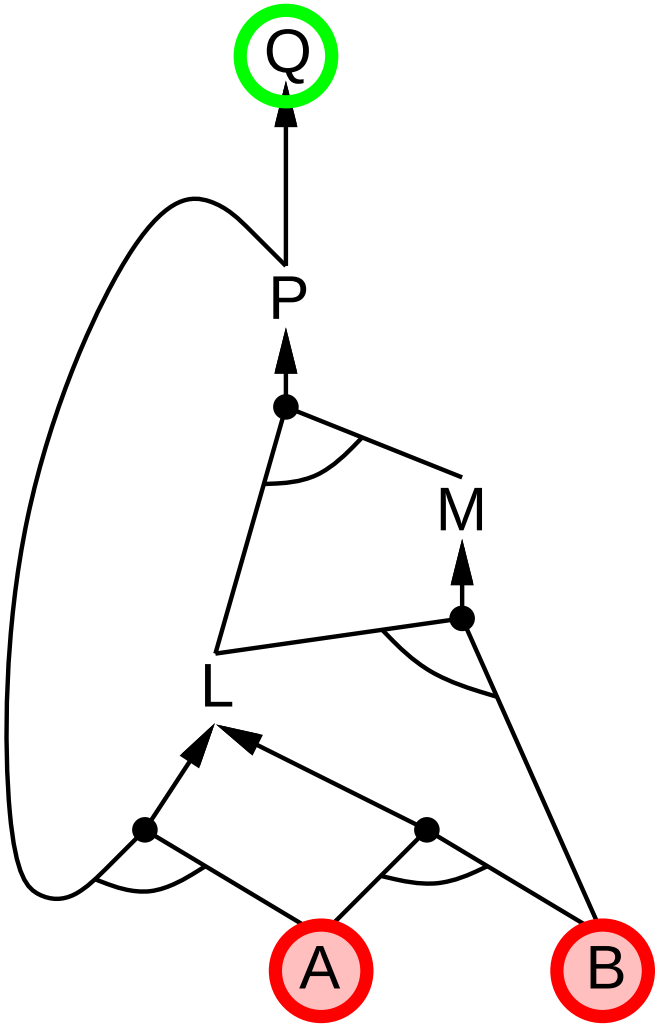
Evitare cicli: controlla se un nuovo sottogoal è già presente nella pila dei goal

Evitare di ripetere del lavoro: controlla se un nuovo sottogoal

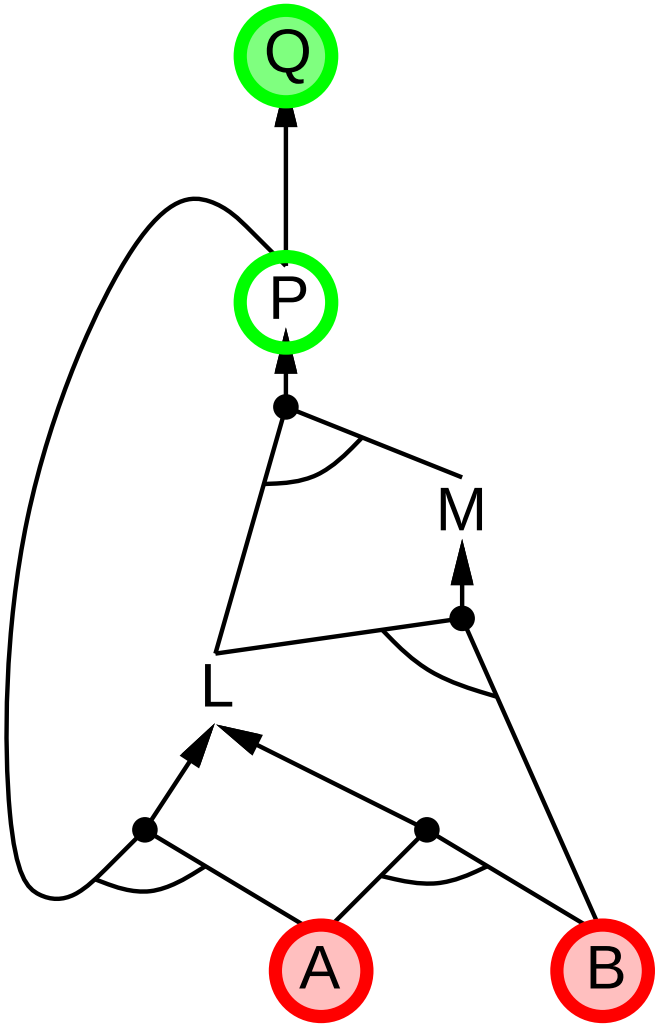
1) è stato già provato vero, o

2) è già fallito

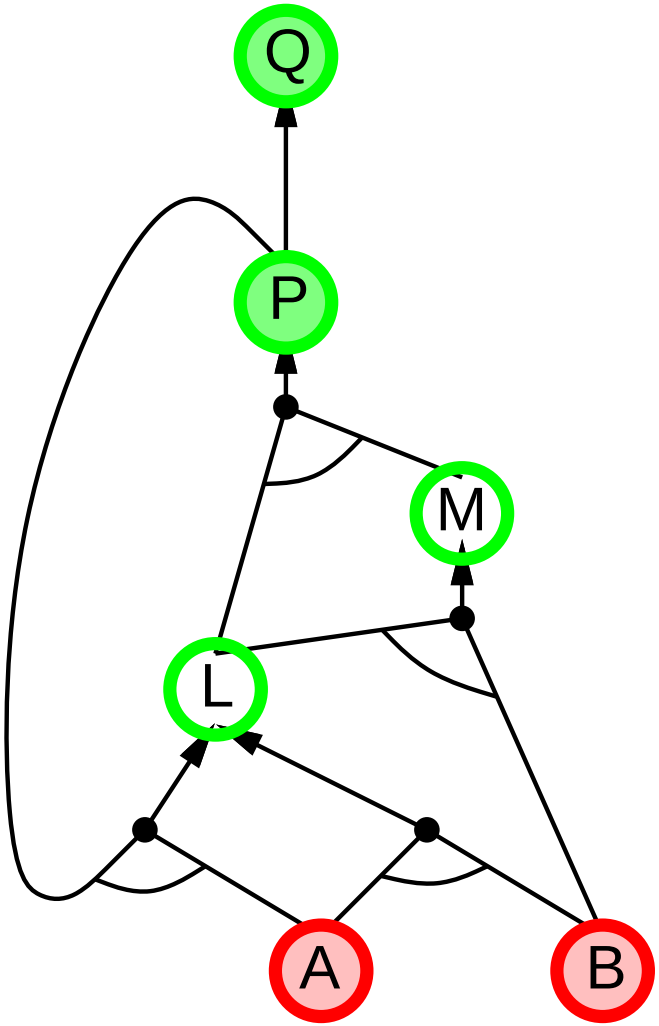
Esempio di Backward chaining



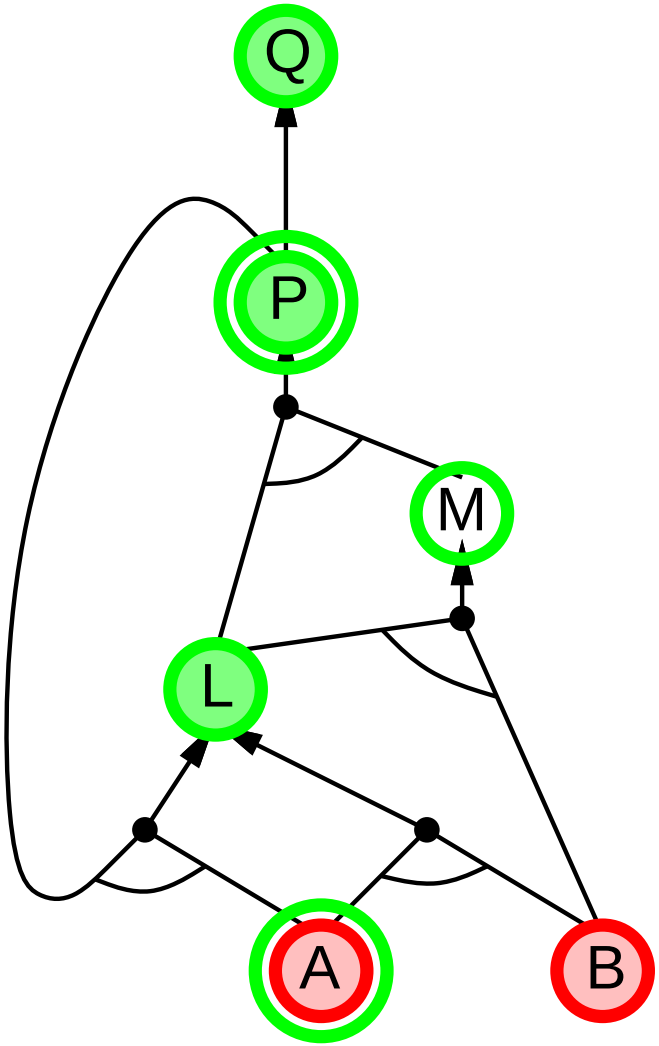
Esempio di Backward chaining



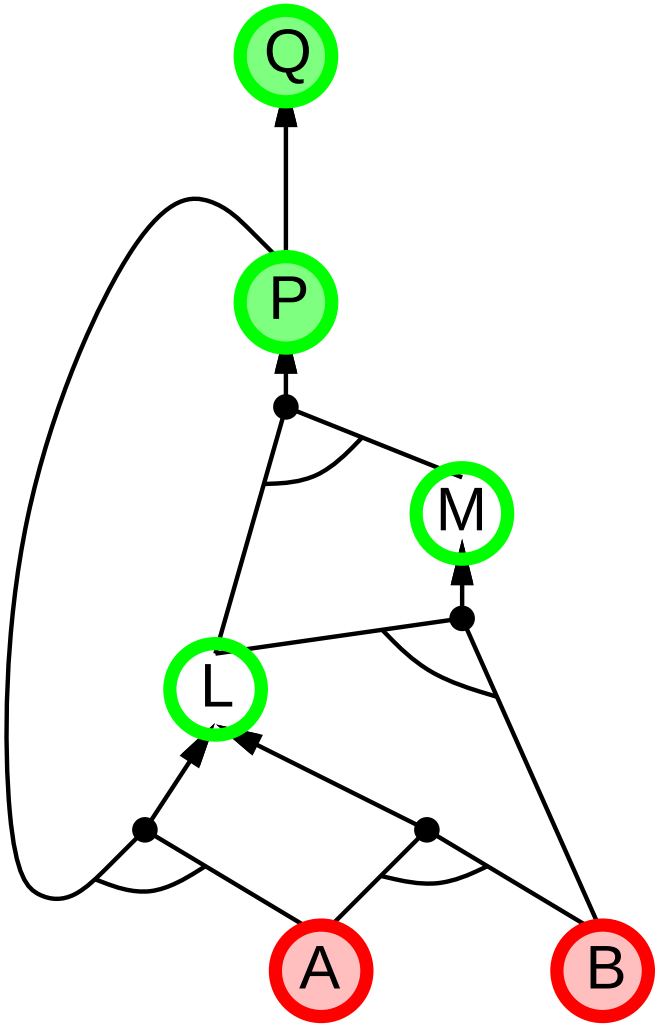
Esempio di Backward chaining



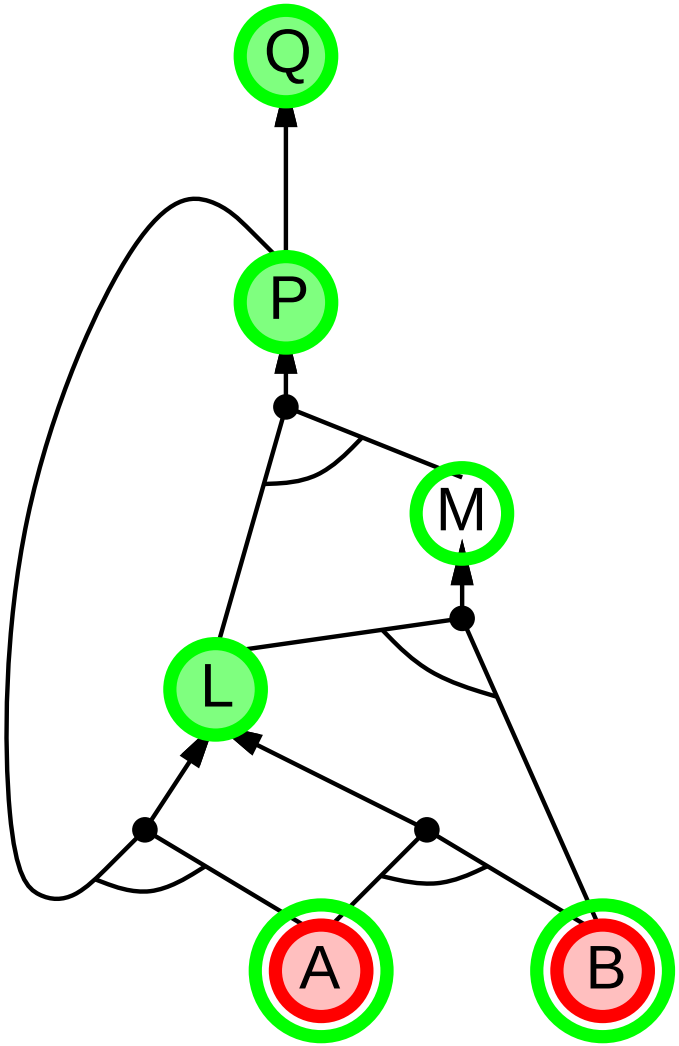
Esempio di Backward chaining



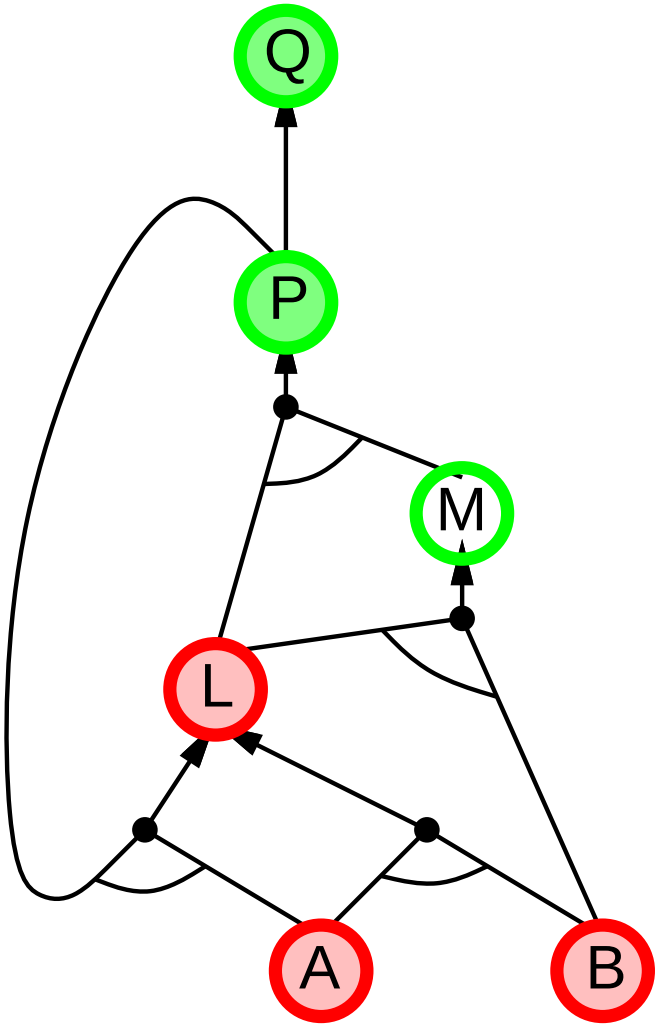
Esempio di Backward chaining



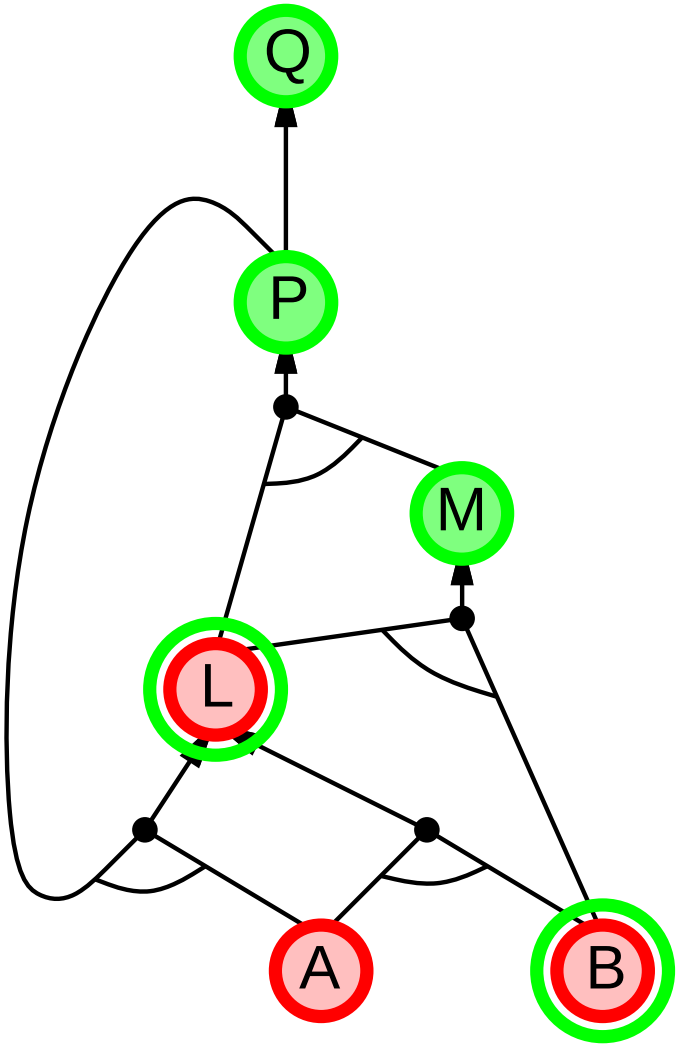
Esempio di Backward chaining



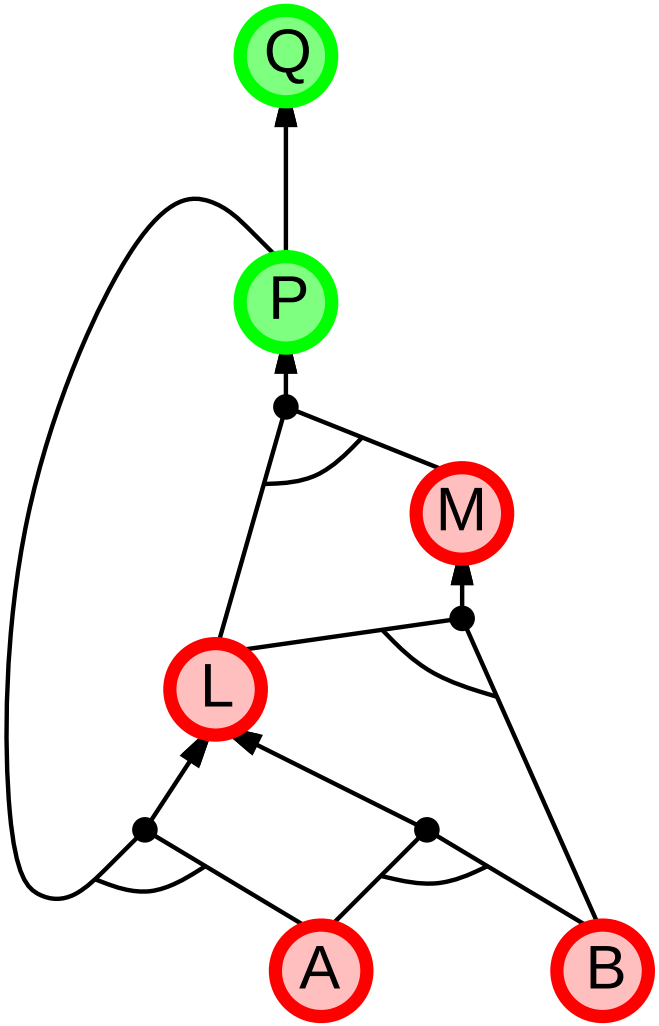
Esempio di Backward chaining



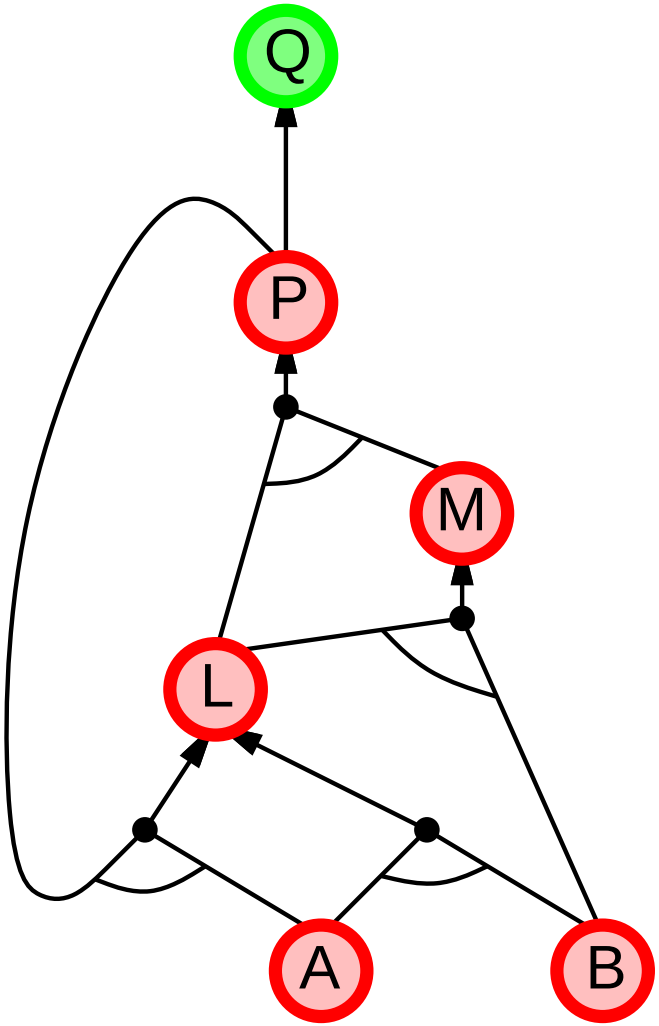
Esempio di Backward chaining



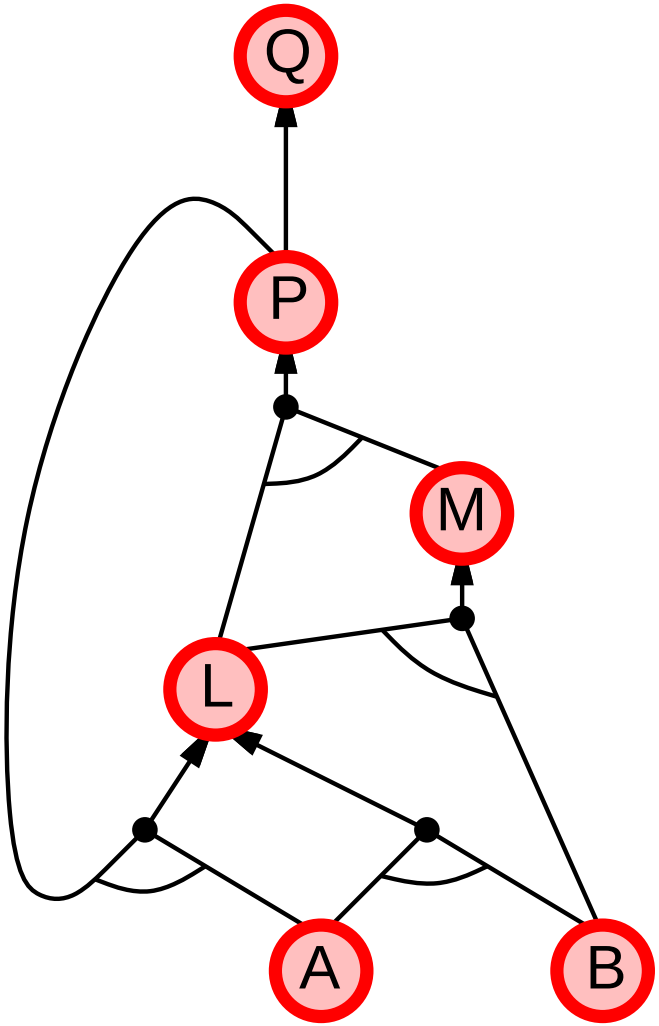
Esempio di Backward chaining



Esempio di Backward chaining



Esempio di Backward chaining



Forward contro backward chaining

FC è data-driven (guidato dai dati), conf. elaborazione inconscia, automatica,

p.e., riconoscimento di oggetti, decisioni di routine

Rischia di eseguire molto lavoro irrilevante per il goal

BC è goal-driven (guidato dal goal), appropriato per il problem-solving,

p.e., Dove sono le mie chiavi? Come faccio ad entrare nel dottorato di ricerca ?

La complessità di BC può essere *molto minore* che lineare nella dimensione di KB

Risoluzione

Forma Normale Congiuntiva (CNF—universale)

congiunzione di $\underbrace{\textit{disgiunzioni}}_{\textit{clausole}}$ di *letterali*

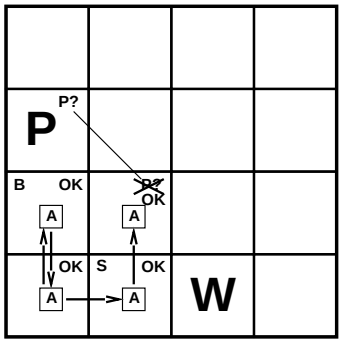
P.e., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Risoluzione regola di inferenza (per CNF): completa per la logica proposizionale

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

dove ℓ_i e m_j sono letterali complementari. P.e.,

$$\frac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$$



La Risoluzione è corretta e completa per la logica proposizionale

Conversione in CNF

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminare $\Leftrightarrow$, rimpiazzando $\alpha \Leftrightarrow \beta$ con $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminare $\Rightarrow$, rimpiazzando $\alpha \Rightarrow \beta$ con $\neg\alpha \vee \beta$.

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Spostare $\neg$ all'interno usando le regole di de Morgan e la doppia negazione:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Applicare la legge distributiva ($\vee$ su $\wedge$) e portare tutto su un livello:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Algoritmo di Risoluzione

Prova per contraddizione, cioè, mostra che $KB \wedge \neg\alpha$ è insoddisfacibile

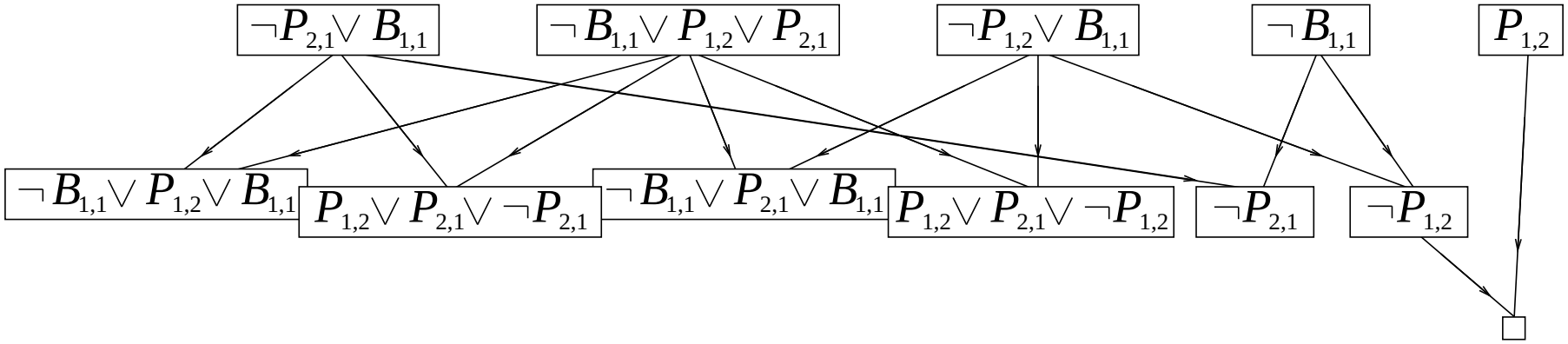
function CP-RISOLUZIONE(KB, α) **returns** *true* oppure *false*

$clausole \leftarrow$ l'insieme di clausole nella rappresentazione CNF di $KB \wedge \neg\alpha$
$new \leftarrow \{\}$
loop do
  for each C_i, C_j **in** $clausole$ **do**
    $resolvents \leftarrow$ CP-RISOLVI(C_i, C_j)
    if $resolvents$ contiene la clausola vuota **then return** *true*
    $new \leftarrow new \cup resolvents$
  if $new \subseteq clausole$ **then return** *false*
  $clausole \leftarrow clausole \cup new$

CP-RISOLVI(C_i, C_j) restituisce l'insieme dei risolventi ottenuti applicando la regola di risoluzione in tutti i modi possibili per le clausole C_i e C_j

Esempio di Risoluzione

$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$ $\alpha = \neg P_{1,2}$



Completezza della Risoluzione

Il teorema di completezza per la risoluzione nella logica proposizionale è chiamato **ground resolution theorem**:

> *Se un insieme di clausole S è insoddisfacibile, allora la chiusura della risoluzione di tali clausole $RC(S)$ contiene la clausola vuota.*

La chiusura della risoluzione di un insieme di clausole corrisponde all'insieme ottenuto come punto fisso dalla applicazione ripetuta della risoluzione. In caso di terminazione con fallimento dell'algoritmo visto, corrisponde all'insieme $clauses$ alla fine della esecuzione.

La dimostrazione del teorema si ottiene dimostrando quanto segue:

> *Se la chiusura $RC(S)$ non contiene la clausola vuota, allora S è soddisfacibile.*

vediamo la dimostrazione di questa affermazione ...

Completezza della Risoluzione

Se $RC(S)$ non contiene la clausola vuota, allora si può costruire un modello per S. Infatti, si dia un ordine arbitrario ai simboli di predicato che compaiono in S, ottenendo $P_1, P_2, \ldots, P_k$. Poi si segua la seguente procedura per costruire il modello:

`for` $i = 1$ `to` k

- *se esiste una clausola in* $RC(S)$ *contenente* $\neg P_i$ *tale che tutti gli altri letterali della clausola sono falsi a causa del valore di verità già assegnato ai* $P_1, P_2, \ldots, P_{i-1}$*, allora assegna valore di verità* $falso$ *a* P_i
- *altrimenti assegna valore di verità* $vero$ *a* P_i

Mostriamo ora che tale procedura termina sempre con un modello per S.

Facciamo tale dimostrazione per induzione su i: supponiamo che sia possibile costruire il modello parziale per i simboli fino a P_{i-1} e mostriamo che tale modello può essere esteso per i simboli fino a P_i

Completezza della Risoluzione

Caso base $i = 1$.

In questo caso, in S non possono necessariamente essere presenti simultaneamente sia la clausola P_1 e $\neg P_1$, perchè altrimenti $RC(S)$ conterrebbe la clausola vuota. Quindi per $i = 1$ la procedura descritta si può applicare senza problemi: $P_1 \leftarrow falso$ se è presente $\neg P_1$, altrimenti $P_1 \leftarrow vero$.

Ipotesi induttiva vera per $i - 1$.

Consideriamo una clausola C in $RC(S)$ che contiene P_i. Si hanno problemi ad assegnare un valore di verità a P_i solo se

- $C \equiv B \vee \neg P_i$, con B clausola che contiene solo simboli P_j con $j < i$
- esiste in $RC(S)$ una clausola $C' \equiv B' \vee P_i$ con B' clausola che contiene solo simboli P_j con $j < i$

Completezza della Risoluzione

Ma se questo succede, in $RC(S)$ deve essere presente anche la clausola $B \vee B'$ (che contiene solo simboli P_j con $j < i$) altrimenti $RC(S)$ non è la chiusura, ed a causa della ipotesi induttiva, l'assegnamento parziale fino a P_{i-1} non può rendere falsa sia B che B'. Quindi, se è falsa B, $P_i \leftarrow falso$, se invece è falsa B', $P_i \leftarrow vero$, ottenendo un modello parziale per i simboli fino all'indice i.

Si conclude che quando i raggiunge k, otteniamo un modello completo per S e quindi abbiamo dimostrato che S è soddisfacibile.

Riassunto

Gli agenti logici applicano l'inferenza ad una base di conoscenza
per derivare nuova informazione e prendere decisioni

Concetti base della logica:
- sintassi: struttura formale di sentenze
- semantica: verità di sentenze rispetto a modelli
- entailment: verità necessaria di una sentenza data un'altra
- inferenza: derivazione di sentenze a partire da altre sentenze
- correttezza: le derivazioni producono solo sentenze conseguenti
- completezza: le derivazioni producono tutte le sentenze conseguenti

Il mondo dei Wumpus world richiede la capacità di rappresentare informazione parziale e negata, ragionamento per casi, etc.

Forward e Backward chaining sono lineari, e completi per clausole di Horn
La Risoluzione è completa per la logica proposizionale

La logica proposizionale manca di potere espressivo