Resolution

- Regola di resolution:

NB: i due letterali P_i e Q_j sono complementari (uno è la negazione dell'altro)

$$\frac{P_1 \vee P_2 \vee \ldots \vee P_{i-1} \vee P_i \vee P_{i+1} \vee \ldots \vee P_n \qquad Q_1 \vee Q_2 \vee \ldots \vee Q_{j-1} \vee Q_j \vee Q_{j+1} \vee \ldots \vee Q_m}{P_1 \vee P_2 \vee \ldots \vee P_{i-1} \vee P_{i+1} \vee \ldots \vee P_n \vee Q_1 \vee Q_2 \vee \ldots \vee Q_{j-1} \vee Q_{j+1} \vee \ldots \vee Q_m}$$

La formula derivata è detta **resolvent**, in essa *ogni letterale compare una volta sola (**FATTORIZZAZIONE**), esempio:*

A ∨ A diventa A

Esempio

Letterali complementari

$$\frac{A \lor B \lor \neg C \qquad C \lor A}{A \lor B}$$

La rimozione dei letterali complementari produce il risolvente: $A \lor B \lor A$ che si semplifica nelle'equivalente $A \lor B$ applicando la fattorizzazione

Relazione con il modus ponens

$$\frac{B \vee C \qquad \neg C \vee A}{A \vee B}$$

Il modus ponens è un caso speciale di risoluzione

Lo si evidenzia tramite l'eliminazione dell'implicazione

$$\frac{C \qquad C \Rightarrow A}{A}$$

Modus Ponens

$$\frac{C \qquad \neg C \vee A}{A}$$

Modus ponens dopo l'eliminazione dell'implicazione

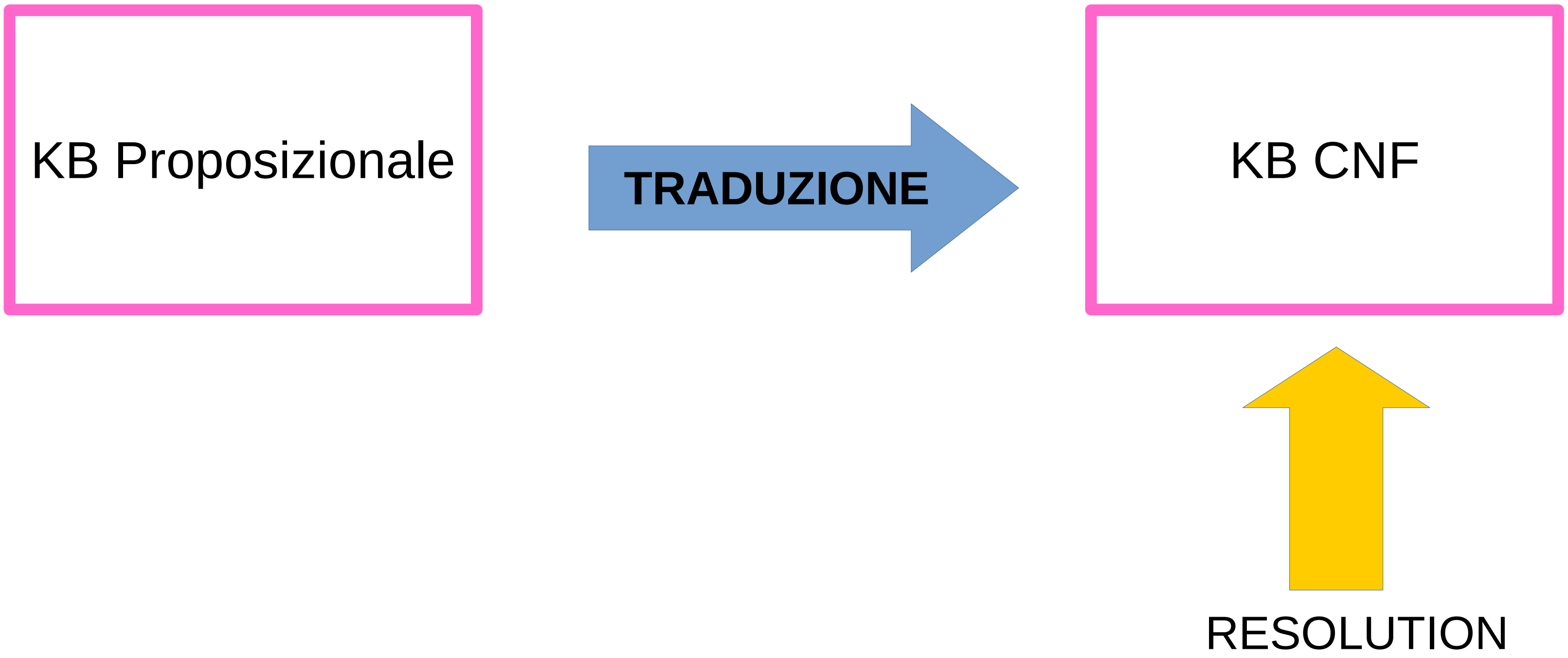
Agente guidato dalla conoscenza e inferenza

```
Agente ha: KB
Tempo = 0

Function KB-Agent(percezione) returns azione
{
1.    tell(KB, costruisci-formulaP(percezione, tempo))
2.    risposta ← ask(KB, costruisci-interrogazioneA(tempo))
3.    tell(KB, costruisci-formulaA(azione, tempo))
4.    tempo ← tempo + 1
5.    return risposta
}
```

1. L'agente sia dotato di una KB iniziale
2. La KB viene aggiornata con l'aggiunta di fatti che dipendono dalla "percezione" e dalle "azioni" eseguite
3. Ask interroga la KB per ottenere l 'azione da eseguire: questa richiesta attiva un processo di inferenza in cui la query, negata, viene aggiunta alla KB e, applicando iterativamente la resolution, viene ottenuta la risposta

Prerequisito: KB in Conjunctive Normal Form



Formule proposizionali e clausole

- CNF: conjunctive normal form
  data una qualsiasi formula proposizionale esiste una congiunzione di clausole ad essa equivalente

GRAMMATICA DELLE CLAUSOLE

- CNFsentence → Clause ∧ … ∧ Clause
- Clause → Literal ∨ … ∨ Literal
- Literal → Symbol | ¬Symbol
- Symbol → P | Q | …

Esempi e controesempi

- $\neg A \wedge (B \vee C)$
- $(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$
- $A \vee B$
- $A \wedge B$

Clausole

- $\neg(B \vee C)$
- $(A \wedge B) \vee C$
- $A \wedge (B \vee (D \wedge E))$.

Formule proposizionali che sembrano ma non sono clausole

https://en.wikipedia.org/wiki/Conjunctive_normal_form

Formule proposizionali e clausole

Algoritmo di traduzione in clausole

1) Eliminare la biimplicazione: $((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$
2) Eliminare l'implicazione: $(\neg\alpha \vee \beta)$
3) Portare il not all'interno (De Morgan ed eliminazione della doppia negazione):
   - $(\neg\alpha \vee \neg\beta)$ oppure $(\neg\alpha \wedge \neg\beta)$
   - α equivale a $\neg\neg\alpha$
4) Distribuire l'or sull'and dove possibile: $((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$

Formule proposizionali e clausole

Algoritmo di traduzione in clausole

1) Eliminare la biimplicazione
2) Eliminare l'implicazione
3) Portare il not all'interno (De Morgan ed eliminazione della doppia negazione)
4) Distribuire l'or sull'and dove possibile

- $\neg(B \vee C)$
- $(A \wedge B) \vee C$
- $A \wedge (B \vee (D \wedge E))$.

$\neg B \wedge \neg C$

$(A \vee C) \wedge (B \vee C)$

$A \wedge (B \vee D) \wedge (B \vee E)$

Esempio: da formule a clausole

La KB proposizionale vista tradotta in clausole:

C1) ¬ Piove ∨ Atmosfera_umida
C2) ¬ Notte ∨ Vento or Atmosfera_umida
C3a) ¬ Atmosfera_umida ∨ Prato_bagnato
C3b)¬ Atmosfera_umida ∨ Strada_Bagnata
C4) ¬ Innaffiatore_on ∨ Prato_bagnato
C5) ¬ Piove ∨ Ombrello_aperto
C6) ¬ Sole ∨ ¬ Vento or Innaffiatore_on
C7) ¬ Sole ∨ ¬ Vento ∨ Atmosfera_asciutta
C8) ¬ Sole ∨ ¬ Notte
C10) ¬ Atmosfera_asciutta ∨ ¬ Atmosfera_umida

La traduzione in clausola della R8) e della R9) producono esattamente la stessa clausola (in particolare la C8)

Algoritmo di risoluzione

Function CP-RISOLUZIONE(KB, A) **returns** true | false
Inputs: KB è la base di conoscenza,
A è una query espressac come formula proposizionale

clausole ← clausole della rappresentazione CNF di KB and not A
New ← { }
Loop do {
For each Ci, Cj in clausole **do {**
resolvents ← IR-RISOLUZIONE(Ci, Cj)
If resolvents contiene la clausoal vuota **return** true
new ← new U resolvents
}
If new incluso in clausole **return** false
clausole ← clausole U resolvents
}

NOTA: nella II edizione versione italiana qui compare erroneamente una chiamata ricorsiva

Nota: CP-RISOLUZIONE sta per algoritmo di risoluzione per logica proposizionale
IR-RISOLUZIONE sta per regola di inferenza di risoluzione per logica proposizionale

Completezza della risoluzione

- TEOREMA:

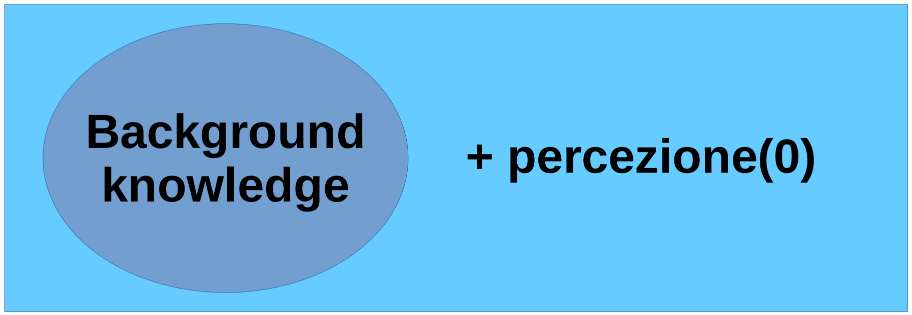
se un insieme di clausole è insoddisfacibile la chiusura della risoluzione contiene la clausola vuota

Non lo dimostriamo

Esempio: pioggia, atmosfera, strada

- KB: quella già vista, tradotta in clausole
- Percezione: da aggiungere alla KB
- Interrogazione: inferenza di formule

Esempio: percezione (proposizionale)

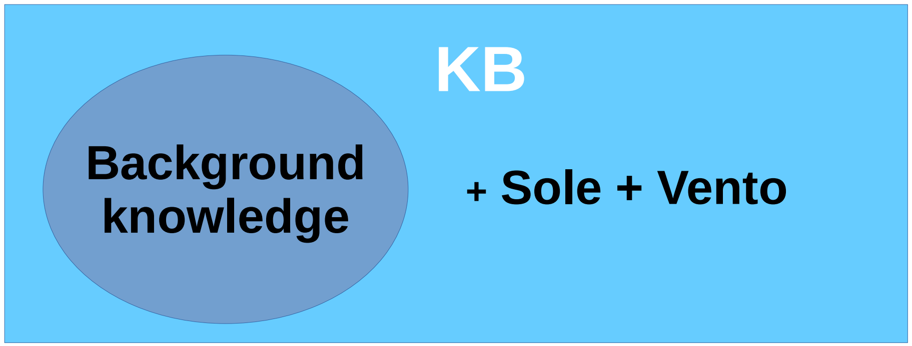


La percezione produce dei fatti che Vengono aggiunti alla KB. Supponiamo che I fatti siano:

F1) Sole

F2) Vento

Esempio: ragionamento



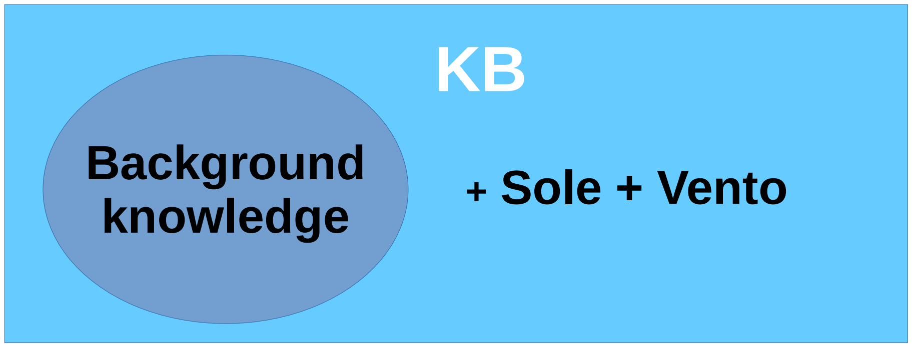
⊨ ¬ Piove ?

Ci domandiamo se:

KB ⊨ ¬ Piove

cioè se **¬ Piove** sia conseguenza logica di **KB**, ovvero se **¬ Piove** sia vero in tutti i modelli in cui **KB** è vera

Esempio: ragionamento



$\models$ **¬ Piove ?**

Useremo una dimostrazione per refutazione, cioè cercheremo di dimostrare **(KB ∧ Piove) ≡ False** utilizzando la **resolution**

Inferire il goal negato

Per verificare se “¬ Piove” è una conseguenza logica della KB precedentemente riportata si parte negando il goal ed ottenendo quindi la clausola:

GN) Piove

A questo punto si può applicare la *dimostrazione per refutazione :*

1) si aggiunge il goal negato GN) alla KB trasformata in forma clausale,
2) e si dimostra che questa è insoddisfacibile generando la clausola vuota mediante risoluzione

Inferire il goal negato

Risolvendo:

GN) con C1) si ha:

Piove ¬ Piove ∨ Atmosfera_umida

C21) Atmosfera_umida

C21) con C10) si ottiene:

Atmosfera_umida ¬ Atmosfera_asciutta ∨ ¬ Atmosfera_umida

C22) ¬ Atmosfera_asciutta

Inferire il goal negato

Risolvendo:

C22) con C7) si ha

$$\frac{\neg\ Atmosfera\_asciutta \qquad \neg\ Sole \lor \neg\ Vento \lor Atmosfera\_asciutta}{C23)\quad \neg\ Sole \lor \neg\ Vento}$$

C23) con F1) si ha

$$\frac{\neg\ Sole \lor \neg\ Vento \qquad Sole}{C24)\quad \neg\ Vento}$$

che a sua volta mediante la risoluzione con F2) (Vento) genera la **clausola vuota**. La risposta sarà true.

Clausole di Horn

Horn clauses, forward and backward chaining

- In molti contesti pratici sono utilizzate clausole dalla forma molto specifica, per le quali sono stati studiati meccanismi di inferenza ad hoc:
  - Clausole di Horn
  - Forward Chaining e Backward Chaining

Horn Clauses (clausole di Horn)

- **Definizione**: una **clausola di Horn** è una disgiunzione di letterali di cui al più uno è positivo
  - Se la clausola contiene esattamente un letterale positivo è detta **clausola definita**
  - **Esempi**: $\neg B \vee C$ oppure $\neg A \vee \neg B \vee C$ oppure $\neg A \vee \neg B$ sono clausole di Horn, le prime due sono anche clausole definite
- Catturano delle implicazioni in cui la formula implicante è una congiunzione di letterali positivi e la formula implicata è un singolo letterale positivo, esempi: $B \Rightarrow C$ oppure $A \wedge B \Rightarrow C$
- Costituiscono la base della programmazione logica

Clausole di Horn: vantaggi

- Su clausole di Horn è possibile applicare meccanismi di inferenza molto **naturali per gli esseri umani**
- Consentono di verificare la consequenzialità logica in un **tempo che cresce linearmente con la dimensione della KB** (quindi l'inferenza nel caso proposizionale è computazionalmente economica)

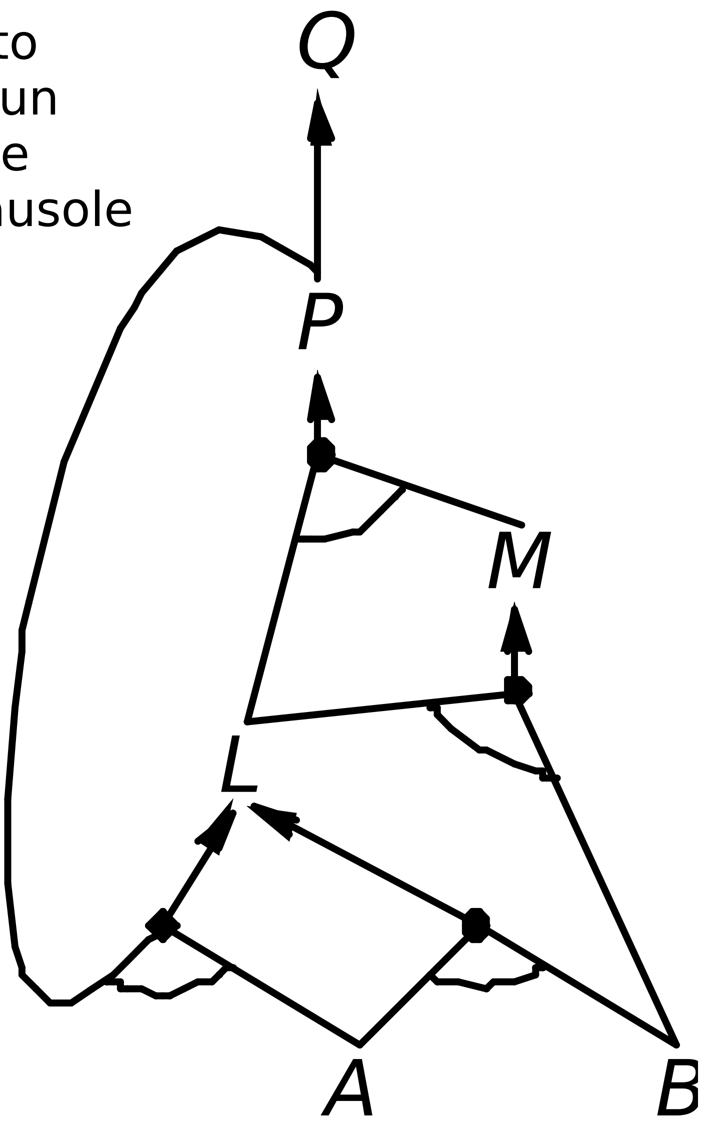
Forward Chaining (concatenazione in avanti)

- Permette di derivare una <u>query data da un singolo simbolo proposizionale</u> da una <u>KB costituita da clausole di Horn</u>
- Procedimento iterativo, <u>guidato dai dati</u>:
  1) Si parte dai fatti conosciuti
  2) Si applica il modus ponens (da F $\Rightarrow$ Q e F derivo Q) , ragionamento deduttivo
  3) Se tutte le premesse di un'implicazione sono vere, si aggiunge il letterale implicato all'insieme dei fatti conosciuti
  4) Terminazione: o si ottiene la query (return true) o a un certo punto non si potranno fare altre inferenze (return false)
- Ha <u>complessità lineare</u> nella dimensione della KB

Esempio

La figura rappresenta la seguente KB sotto forma di grafo AND-OR. Gli archi uniti da un archetto rappresentano letterali in AND, le frecce rappresentano degli OR dati da clausole aventi come testa lo stesso letterale

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

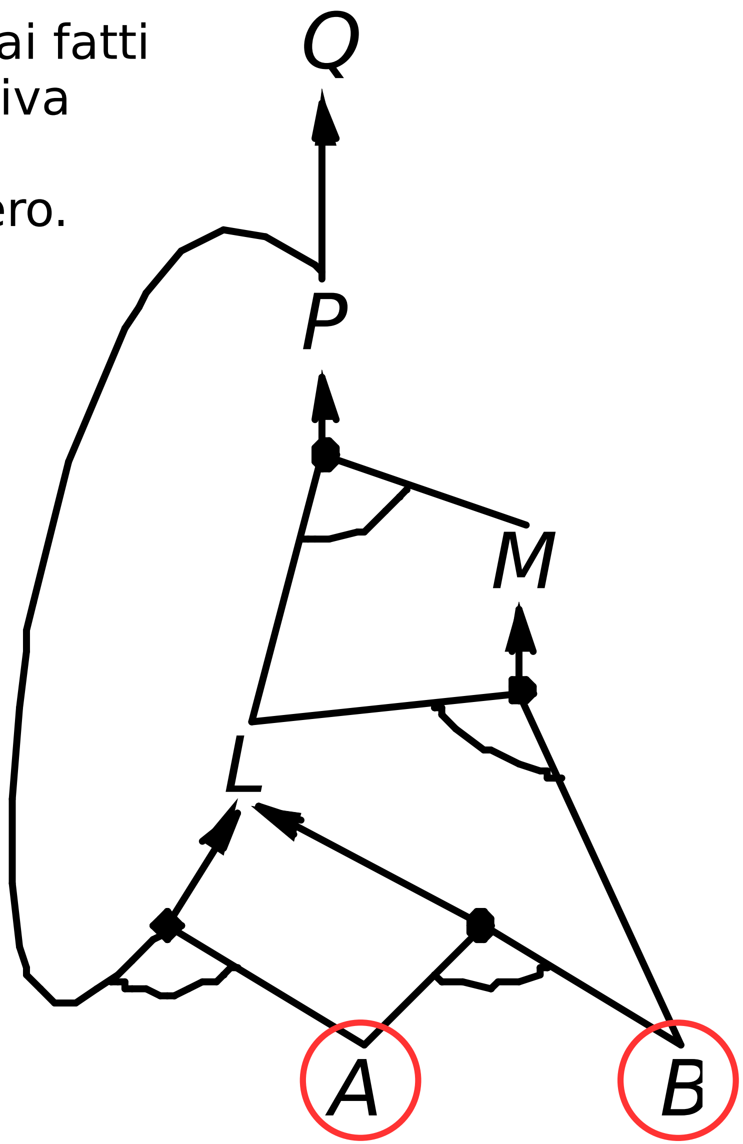


Esempio

Partendo dai nodi attivati direttamente dai fatti l’inferenza si propaga: un arco AND si attiva quando tutti i lsuoi congiunti sono veri;
un arco OR quando uno dei disgiunti è vero.

P ⇒ Q
L ∧ M ⇒ P
B ∧ L ⇒ M
A ∧ P ⇒ L
A ∧ B ⇒ L

Fatti: A, B
Si vuole dimostrare Q



Esempio

I cerchi rossi evidenziano alcuni letterali che risulteranno veri con questo procedimento:
A e B perché attivati dai fatti, L perché A e B in and costituiscono l'antecedente di una regola che ha L come conseguente, ecc.

$P \Rightarrow Q$

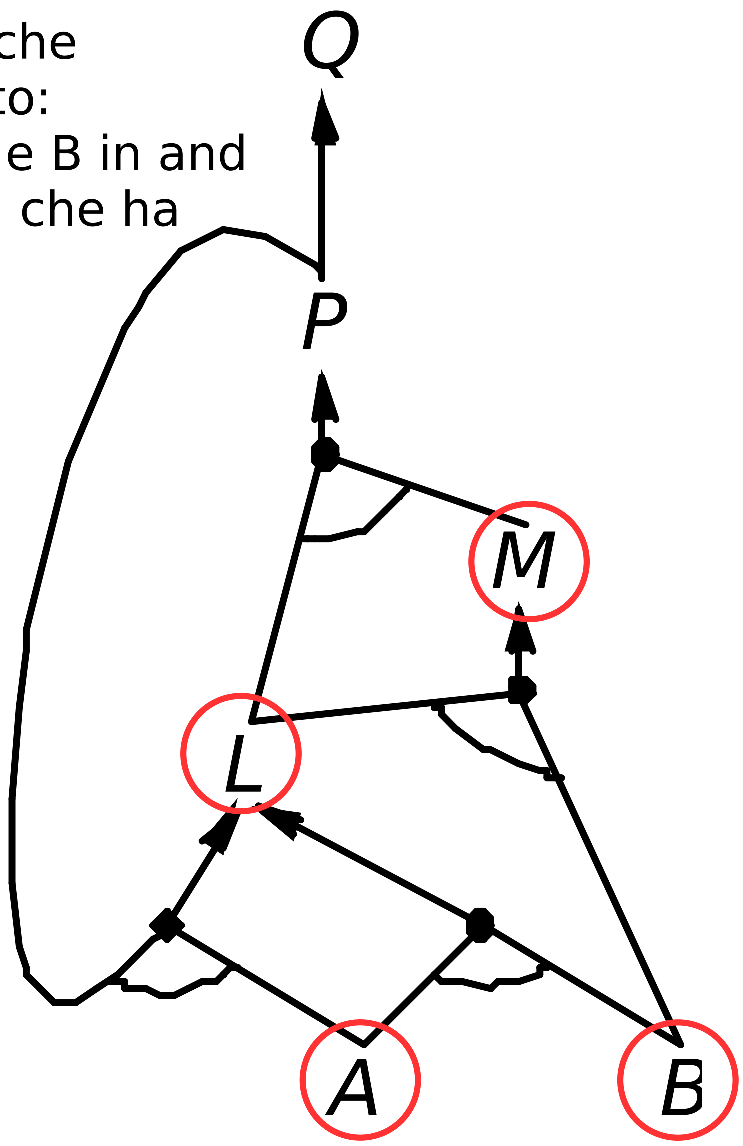
$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

Fatti: A, B



Commenti sul forward chaining

- **Complessità lineare**
- **Completo**: permette di derivare tutte le formule atomiche dimostrabili a partire dalla KB
- **Inconscio**: è guidato dai dati e non usa l'informazione relativa al goal (la formula che stiamo cercando di dimostrare)
- È adeguato a risolvere problemi come per esempio il riconoscimento di oggetti
- Può attivare molte **inferenze inutili** ai fini della dimostrazione della formula in oggetto

Backward chaining (concatenazione all'indietro)

- Parte dalla formula da dimostrare (**goal**):
  - Se risulta già vera termina restituendo true
  - Altrimenti cerca clausole di Horn di cui la formula è conclusione e cerca di dimostrarne le premesse usando come informazione aggiuntiva i fatti noti

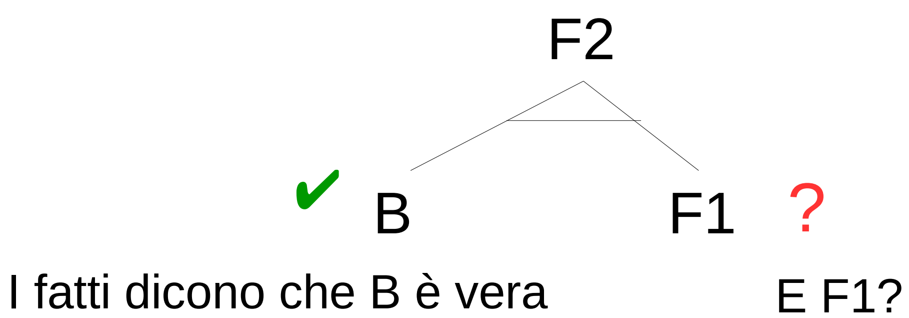
Backward chaining

Supponiamo di avere:

R1) $A \wedge C \Rightarrow F1$

R2) $B \wedge F1 \Rightarrow F2$

E di voler dimostrare che dati A, B e C, F2 è vera. F2 non appartiene ai fatti noti ma abbiamo R2



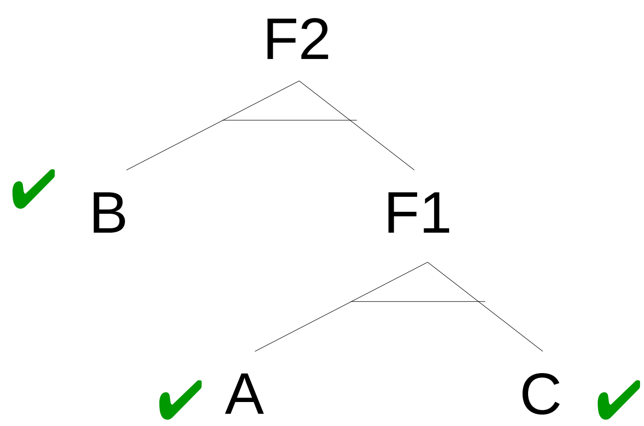
Backward chaining

Supponiamo di avere:

R1) $A \wedge C \Rightarrow F1$

R2) $B \wedge F1 \Rightarrow F2$

… F1 non appartiene ai fatti noti ma abbiamo R1, che ci dice che F1 è vera quando A e C sono veri. I fatti dicono che A e C sono veri, F2 è vera



I fatti dicono che B è vera

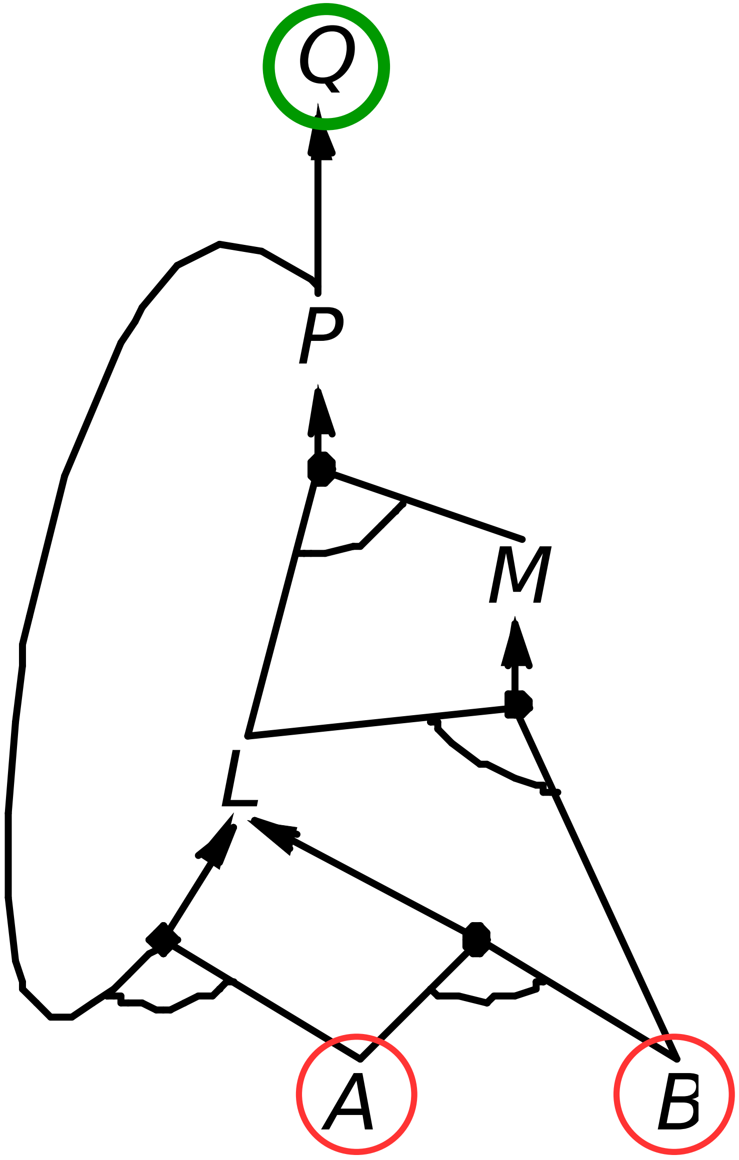
Esempio

Stessa KB, fatti e obiettivo di prima

Si sfruttano due informazioni:
- Una è costituita dall'obiettivo
- L'altra è costituita dai fatti

Nella ricerca:
- Evitare i loop
- Se un sottogoal è già stato dimostrato, non dimostrarlo di nuovo

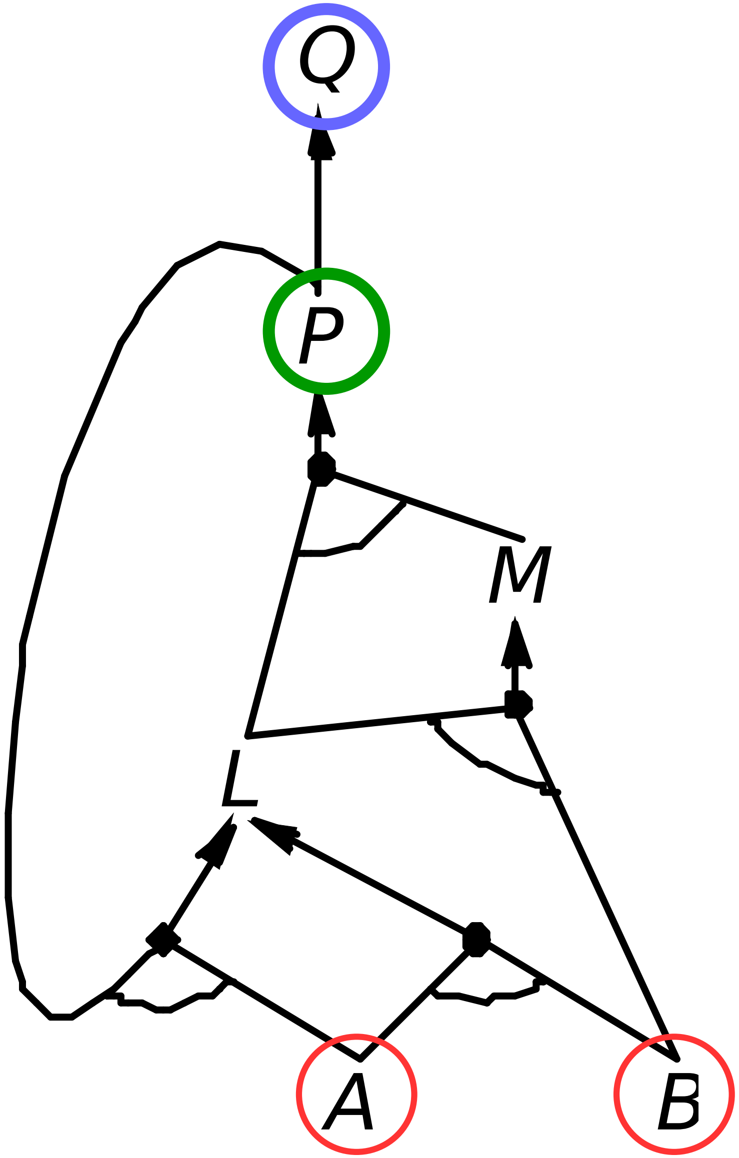
BC:
- Q è vera se P è vera ...



Esempio

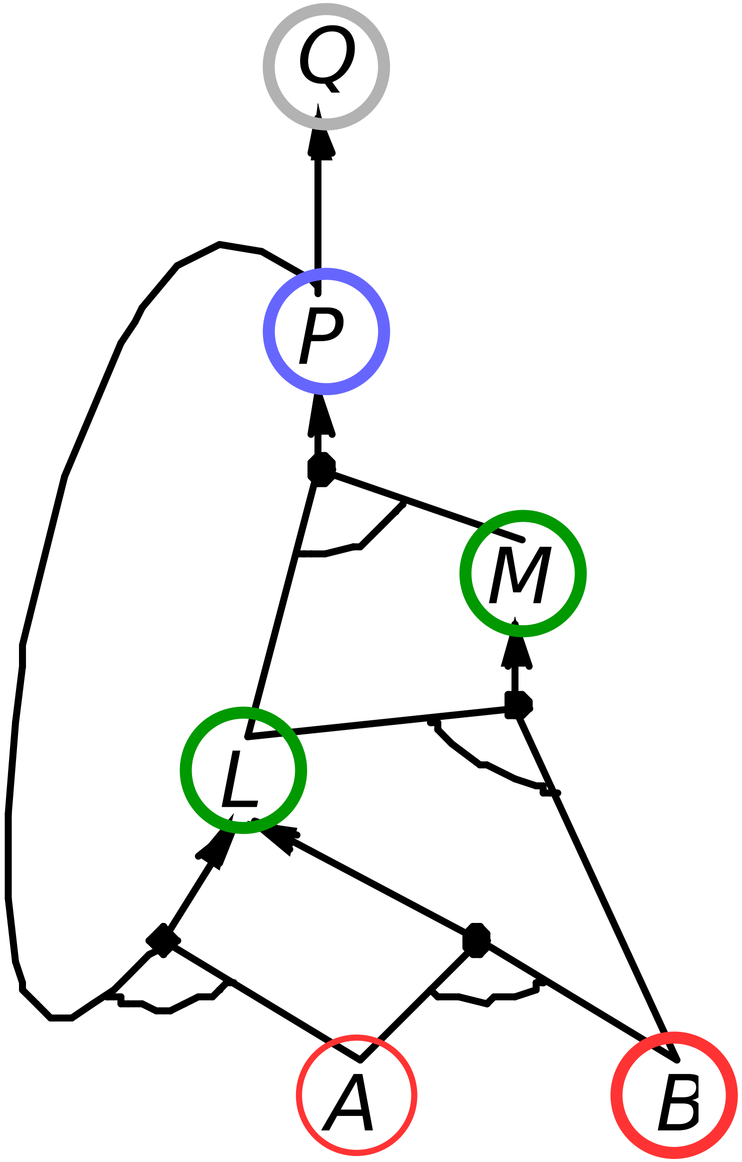
BC:

- P è vera se L e M sono vere



Esempio

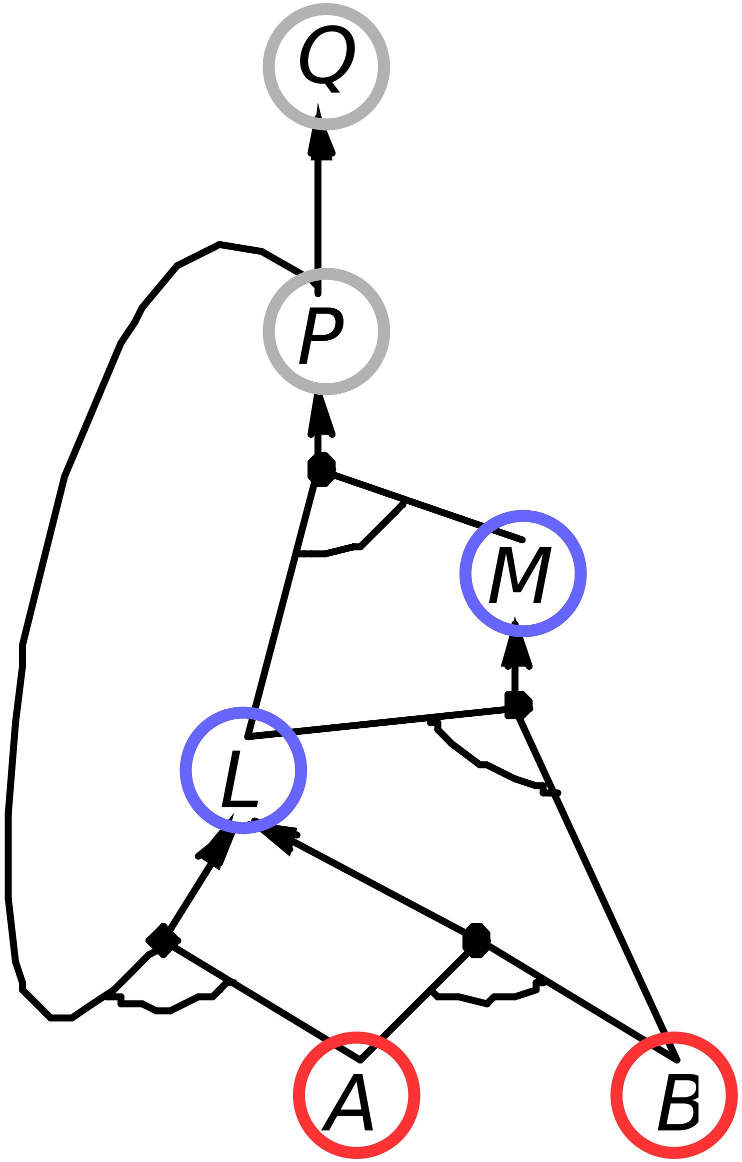
BC:
- P è vera se L e M sono vere
- M è vera se L e B sono vere:
  B è un fatto, L è da dimostrare



Esempio

BC:

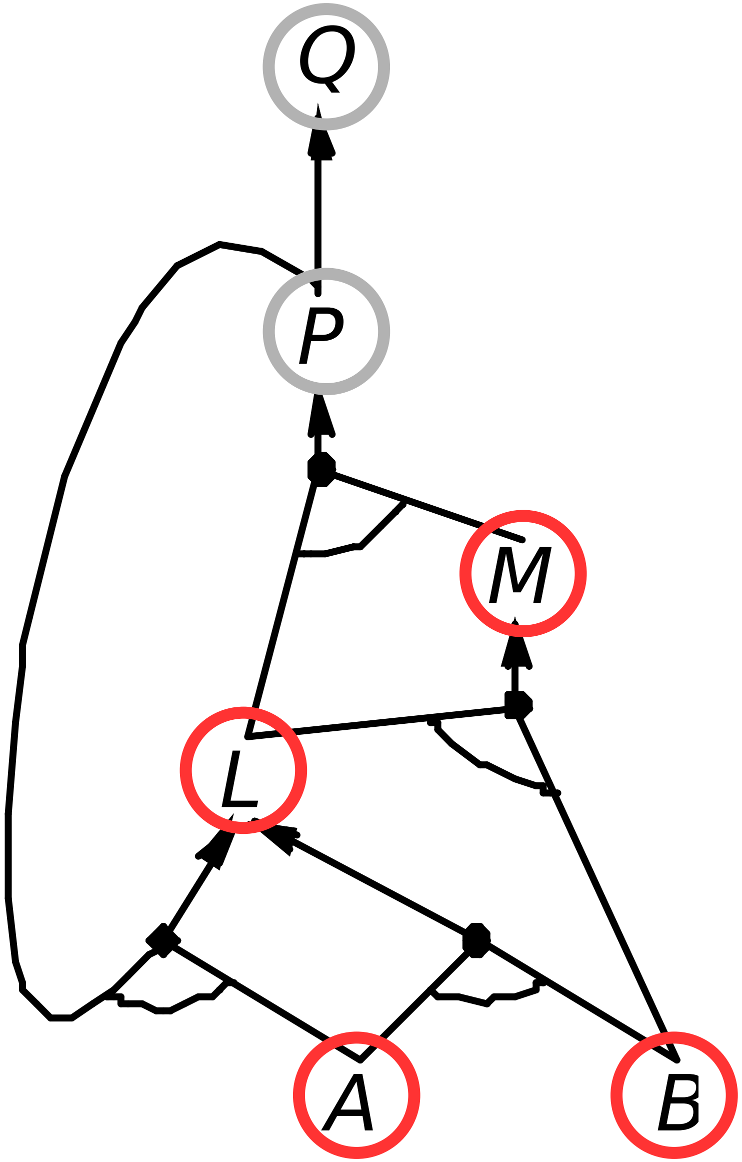
- P è vera se L e M sono vere
- M è vera se L e B sono vere:
  B è un fatto, L è da dimostrare
- L è vera se A e B sono vere:
  A e B sono fatti!
  L è vera anche quando A e P sono vere ma di P non conosciamo il valore di verità e comunque ci basta che una delle due regole sia vera



Esempio

BC:

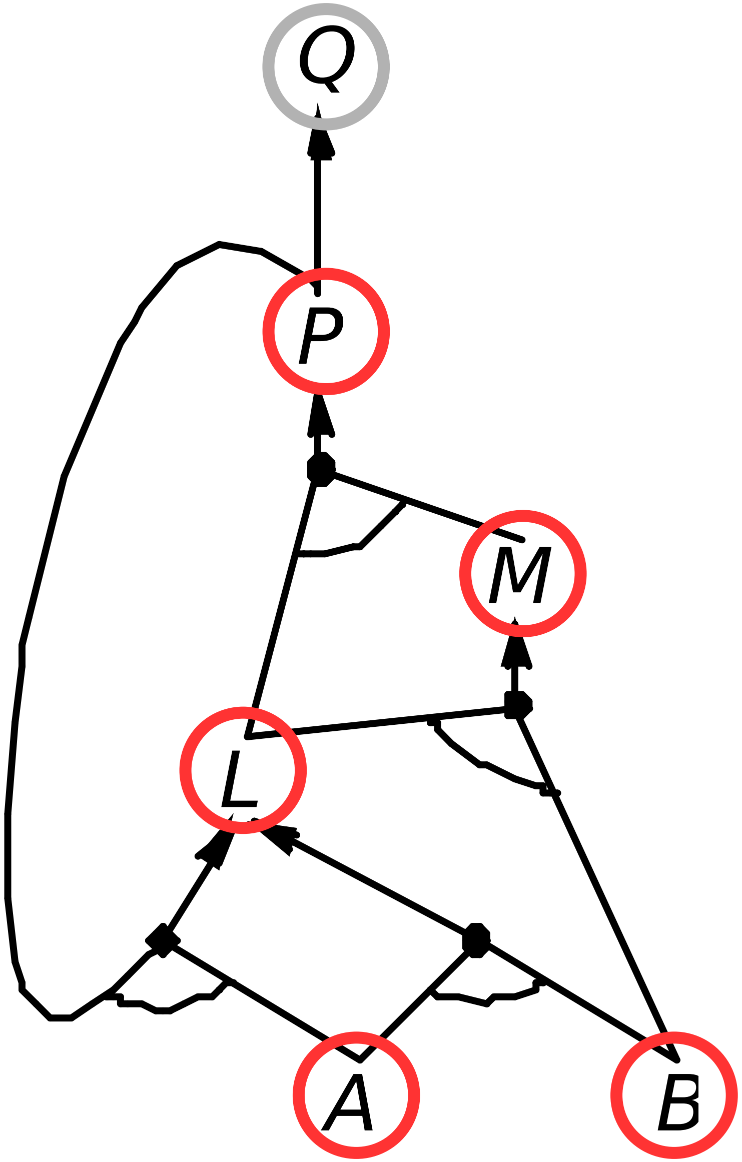
- I valori di verità si propagano dal basso verso l'alto (come valori di ritorno)



Esempio

BC:

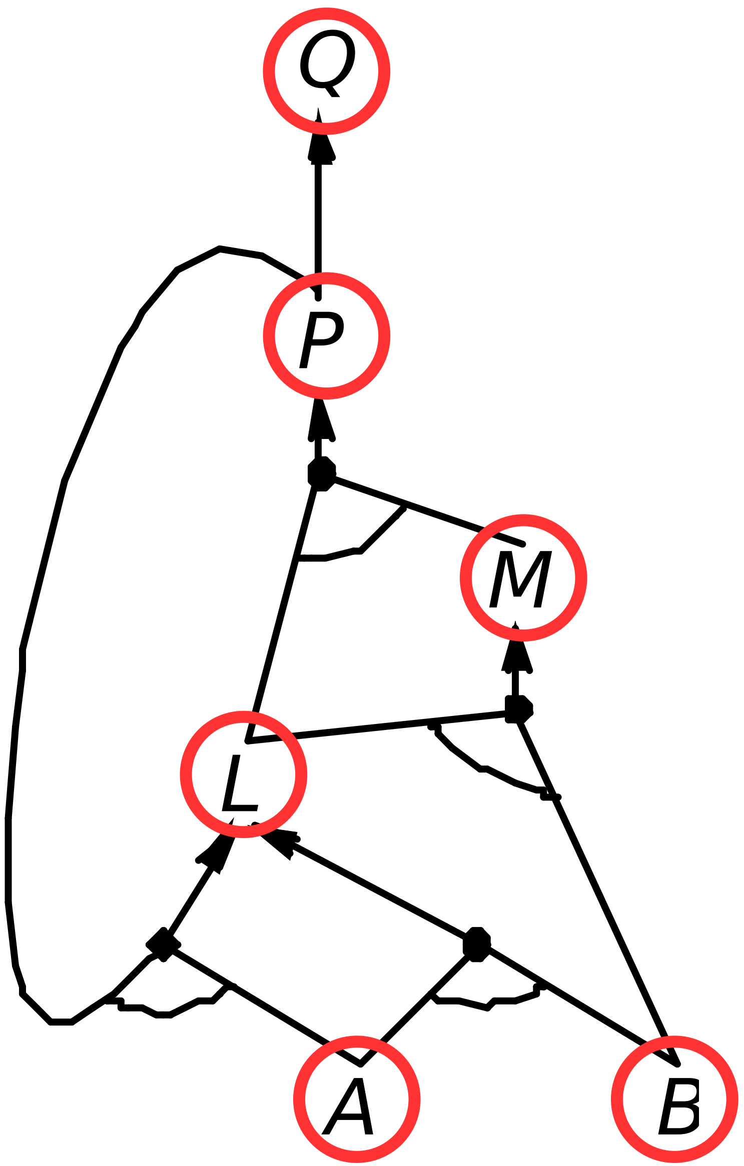
- P risulterà vera



Esempio

BC:

- E infine anche Q risulterà vera
- Il loop da P a L sarà ignorato



Commento su backward chaining

- Realizza una forma di ragionamento **guidato dagli obiettivi**
- È usato nel **theorem proving** e nella **programmazione logica** come meccanismo di inferenza
- Spesso è **più efficiente** del forward chaining in quanto l'uso del goal focalizza la ricerca
- La complessità temporale è **meno che lineare**