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:

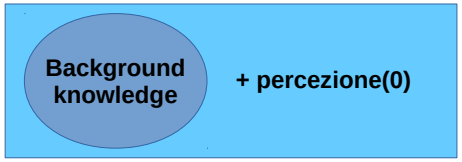
se un insieme di clausole è insoddisfacibile la chiusura della risoluzione contiene la clausola vuota

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)

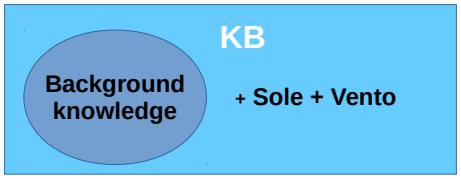


La percezione produce dei fatti che Vengono aggiunti alla KB. Supponiamo che I fatti siano:

F1) Sole

F2) Vento

Esempio: ragionamento



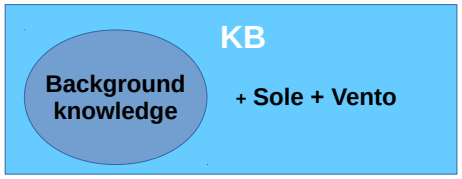
$\models \neg$ **Piove ?**

Ci domandiamo se:

KB $\models \neg$ **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



$\models \neg$ **Piove ?**

Useremo una dimostrazione per refutazione, cioè cercheremo di dimostrare **(KB** $\wedge$ **Piove)** $\equiv$ **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 $\vee$ Atmosfera_umida

C21) Atmosfera_umida

C21) con C10) si ottiene:

Atmosfera_umida ¬ Atmosfera_asciutta $\vee$ ¬ Atmosfera_umida

---

C22) ¬ Atmosfera_asciutta

Inferire il goal negato

Risolvendo:

C22) con C7) si ha

¬ Atmosfera_asciutta ¬ Sole ∨ ¬ Vento ∨ Atmosfera_asciutta

C23) ¬ Sole ∨ ¬ Vento

C23) con F1) si ha

¬ Sole ∨ ¬ Vento Sole

C24) ¬ 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: ¬ B ∨ C oppure ¬ A ∨ ¬ B ∨ C oppure ¬ A ∨ ¬ 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 ⇒ C oppure A ∧ B ⇒ 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)

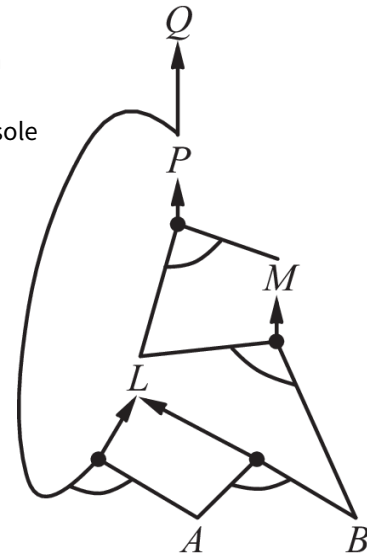
Forward Chaining (concatenazione in avanti)

- Permette di derivare una query data da un singolo simbolo proposizionale da una KB costituita da clausole di Horn
- Procedimento iterativo, guidato dai dati:
  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 complessità lineare 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

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

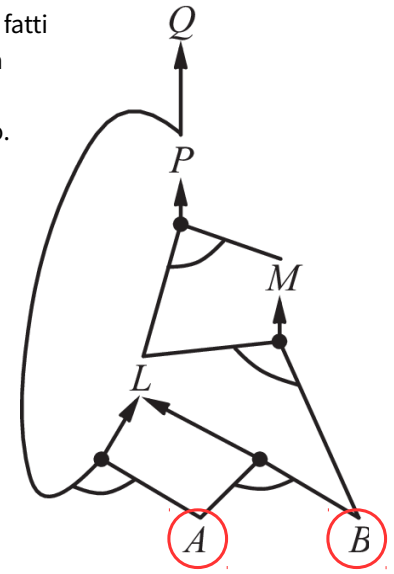


Esempio

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.

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

Fatti: A, B
Si vuole dimostrare Q

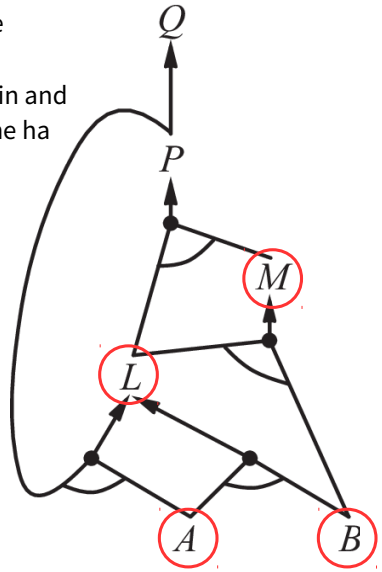


Esempio

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

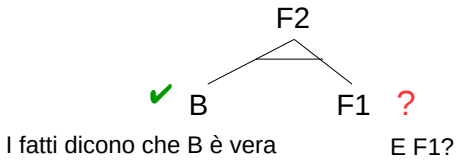
Backward chaining

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



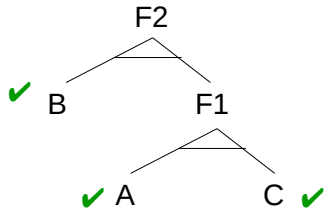
Backward chaining

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

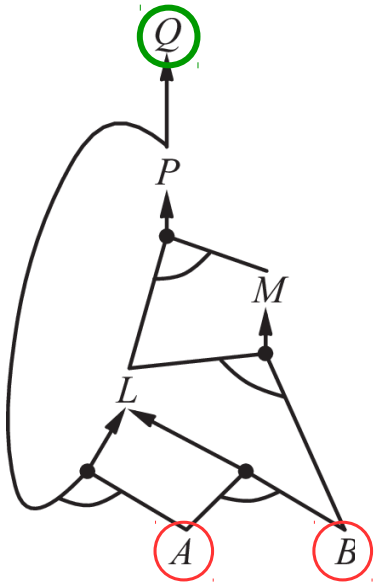
Esempio

Stessa KB, fatti e obiettivo di prima

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

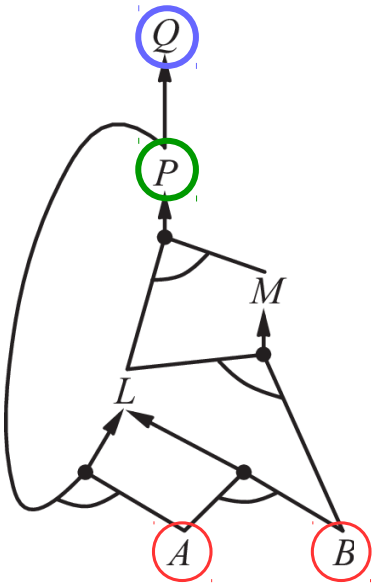
Nella ricerca:
- Evitare i loop
- Se un sottogoal è già stato dimostrato, non dimostrarlo di nuovo

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