

Abbiamo visto...

- Gli agenti logici applicano **inferenze** a una **base di conoscenza** per derivare nuove informazioni.
- Concetti base della logica:
 - **sintassi**: struttura formale delle sentenze
 - **semantica**: **verità** di sentenze rispetto ad **interpretazioni/modelli**
 - **conseguenza logica (entailment)**: sentenza necessariamente vera data un'altra sentenza
 - **inferenza**: derivare (sintatticamente) sentenze da altre sentenze
 - **correttezza (soundness)**: la derivazione produce solo sentenze che sono conseguenza logica.
 - **Completezza (completeness)**: la derivazione puo' produrre tutte le conseguenze logiche.

1

Logica Proposizionale:

- E' la logica piu' semplice, ma non molto espressiva. Non possiamo esprimere variabili (solo enumerazione di tutti gli elementi)
- Se S, S_1, S_2 sono sentenze allora sono anche sentenze:
 - $\neg S$ (**negazione**)
 - $S_1 \wedge S_2$ (**coniunzione**)
 - $S_1 \vee S_2$ (**disgiunzione**)
 - $S_1 \Rightarrow S_2$ (**implicazione**)
 - $S_1 \Leftrightarrow S_2$ (**bicondizionale**)

2

Dimostrazioni in logica proposizionale

- Vedremo la dimostrazione basata su
 - Risoluzione (corretta e completa per clausole generali)
 - Forward chaining (corretta e completa per clausole Horn)
 - Backward chaining (corretta e completa per clausole Horn)
- Una qualunque fbf della logica proposizionale si puo' trasformare in un equivalente insieme di clausole generali (formule SP o PS vedi reti logiche ed algebra di Boole).

3

IL PRINCIPIO DI RISOLUZIONE

- Sistema di deduzione per la logica a clausole per il quale valgono interessanti proprietà.
- Regola di inferenza: **Principio di Risoluzione** [Robinson 65], che si applica a teorie del primo ordine in **forma a clausole**.
- È la regola di inferenza base utilizzata nella programmazione logica.

4

CLAUSOLE

- Una **clausola** è una disgiunzione di letterali (cioè formule atomiche negate e non negate), in cui tutte le variabili sono quantificate universalmente in modo implicito.
- Una clausola generica può essere rappresentata come la disgiunzione:

$$A_1 \vee A_2 \vee \dots \vee A_n \vee \sim B_1 \vee \dots \vee \sim B_m$$

dove A_i ($i=1, \dots, n$) e B_j ($j=1, \dots, m$) sono atomi.

- Una clausola nella quale non compare alcun letterale, sia positivo sia negativo, è detta clausola vuota e verrà indicata con \square . interpretato come contraddizione: disgiunzione falso $\vee \sim$ vero
- Un sottoinsieme delle clausole è costituito dalle **clausole definite**, nelle quali si ha sempre un solo letterale positivo:

$$A_1 \vee \sim B_1 \vee \dots \vee \sim B_m$$

5

IL PRINCIPIO DI RISOLUZIONE

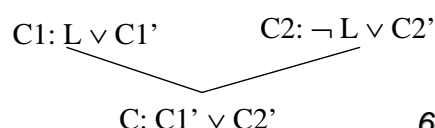
- Il principio di risoluzione, che si applica a formule in forma a clausole, è utilizzato dalla maggior parte dei risolutori automatici di teoremi.
- **Logica Proporzionale:** clausole prive di variabili.
- Siano C_1 e C_2 due clausole prive di variabili:

$$C_1 = A_1 \vee \dots \vee A_n \qquad C_2 = B_1 \vee \dots \vee B_m$$

- Se esistono in C_1 e C_2 due letterali **opposti**, A_i e B_j , ossia tali che $A_i = \sim B_j$, allora da C_1 e C_2 , (clausole **parent**) si può derivare una nuova clausola C_3 , denominata **risolvente**, della forma:

$$C_3 = A_1 \vee \dots \vee A_{i-1} \vee A_{i+1} \vee \dots \vee A_n \vee B_1 \vee \dots \vee B_{j-1} \vee B_{j+1} \vee \dots \vee B_m$$

- C_3 è conseguenza logica di $C_1 \cup C_2$.



6

ESEMPI

$$\begin{array}{ccc} C_1 = p(0,0) & & C_2 = \sim p(0,0) \vee p(0,s(0)) \\ & \searrow & \swarrow \\ & C_3 = p(0,s(0)) & \end{array}$$

$$\begin{array}{ccc} C_1 = p \vee q \vee \sim a \vee \sim b & & C_2 = f \vee a \\ & \searrow & \swarrow \\ & C_3 = p \vee q \vee \sim b \vee f & \end{array}$$

7

DIMOSTRAZIONE PER CONTRADDIZIONE ATTRAVERSO LA RISOLUZIONE (1)

- Dati gli assiomi propri H di una teoria e una formula F , derivando da $H \cup \{\sim F\}$ la contraddizione logica si dimostra che F è un teorema della teoria.
- 1) Ridurre H e il teorema negato $\sim F$ in forma a clausole.
 H trasformato nell'insieme di clausole H^C : $H \rightarrow H^C$
 F negata e trasformata nell'insieme di clausole F^C : $\sim F \rightarrow F^C$
- 2) **All'insieme $H^C \cup F^C$ si applica la risoluzione**
Se F è un teorema della teoria, allora la risoluzione deriva la contraddizione logica (clausola vuota) in un numero finito di passi.
- **Contraddizione:** Nella derivazione compariranno due clausole del tipo A e $\sim B$ con A e B formule atomiche unificabili.

8

DIMOSTRAZIONE PER CONTRADDIZIONE ATTRAVERSO LA RISOLUZIONE (2)

- Per dimostrare F , il metodo originario (Robinson) procede generando i risolventi per **tutte le coppie** di clausole dell'insieme di partenza $C_0 = H^C \cup F^C$ che sono aggiunti a C_0 . Procedimento iterato, fino a derivare, se è possibile, la clausola vuota.
- 1. $C_{i+1} = C_i \cup \{\text{risolventi delle clausole di } C_i\}$
- 2. Se C_{i+1} contiene la clausola vuota, termina.
- Altrimenti ripeti il passo 1.

9

ESEMPIO

$$H = \{(a \rightarrow c \vee d) \wedge (a \vee d \vee e) \wedge (a \rightarrow \sim c)\}$$

$$F = \{d \vee e\}$$

- La trasformazione in clausole di H e $\sim F$ produce:

$$H^C = \{\sim a \vee c \vee d, a \vee d \vee e, \sim a \vee \sim c\}$$

$$F^C = \{\sim d, \sim e\} \text{ (cioe' } \sim(d \vee e)\text{)}$$

- Si vuole dimostrare che $H^C \cup F^C$:

$$\{\sim a \vee c \vee d, \quad (1)$$

$$a \vee d \vee e, \quad (2)$$

$$\sim a \vee \sim c, \quad (3)$$

$$\sim d, \quad (4)$$

$$\sim e\} \quad (5)$$

- è contraddittorio.

10

ESEMPIO

- **Tutti i possibili risolventi al passo 1 sono:**

$\{c \vee d \vee e,$ (6) da (1) e (2)

$d \vee e \vee \sim c,$ (7) da (2) e (3)

$\sim a \vee c,$ (8) da (1) e (4)

$a \vee e,$ (9) da (2) e (4)

$a \vee d,$ (10) da (2) e (5)

$\sim a \vee d\}$ (11) da (1) e (3)

- **Al passo 2, da (10) e (11) viene derivato il risolvente:**

d (12)

e al passo 3, da (4) e (12) viene derivata anche la clausola vuota.

11

Algoritmo di Risoluzione per PL (Logica Proporzionale)

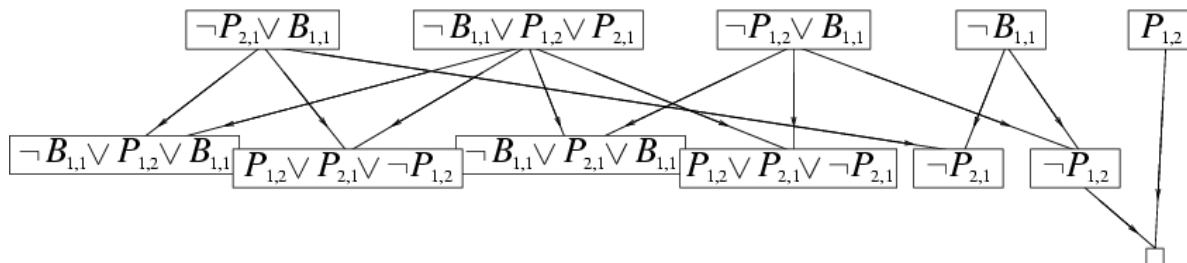
- Per contraddizione, i.e., $KB \wedge \neg \alpha$ e' insoddisfacibile

```
function PL-RESOLUTION( $KB, \alpha$ ) returns true or false
  clauses  $\leftarrow$  the set of clauses in the CNF representation of  $KB \wedge \neg \alpha$ 
  new  $\leftarrow \{ \}$ 
  loop do
    for each  $C_i, C_j$  in clauses do
      resolvents  $\leftarrow$  PL-RESOLVE( $C_i, C_j$ )
      if resolvents contains the empty clause then return true
      new  $\leftarrow$  new  $\cup$  resolvents
    if new  $\subseteq$  clauses then return false
  clauses  $\leftarrow$  clauses  $\cup$  new
```

12

Esempio di Risoluzione

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



13

Forward e backward chaining

- **Horn Form** (sottoinsieme della logica proposizionale)
 - KB = **congiunzione** di **clausole di Horn**
 - Clausole di Horn =
 - Proposizioni atomiche; o
 - (congiunzione di proposizioni atomiche) \Rightarrow proposizione atomica
 - E.g. $KB = C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$

- **Modus Ponens** (per Horn): completo per Horn KB

$$\frac{\alpha_1, \dots, \alpha_n, \quad \alpha_1 \wedge \dots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

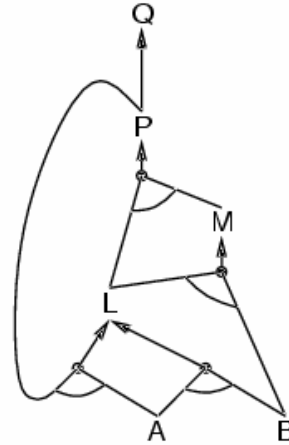
- Può essere usato sia per **forward chaining** o **backward chaining**.
Algoritmi molto naturali e con complessità lineare in tempo.

14

Forward chaining

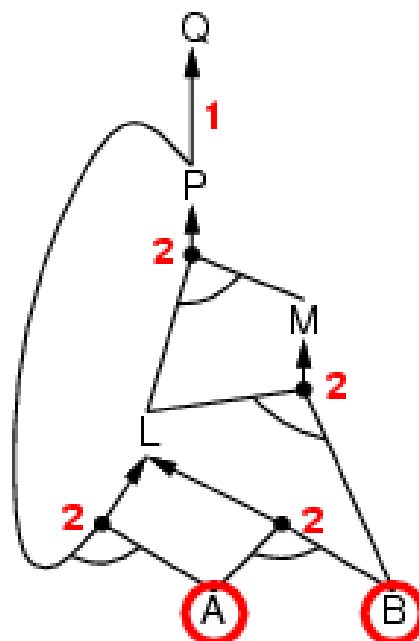
- Idea: applica tutte le regole le cui premesse sono soddisfatte nella *KB*,
 - Aggiungi le conclusioni alla *KB*, fino a trovare la query

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



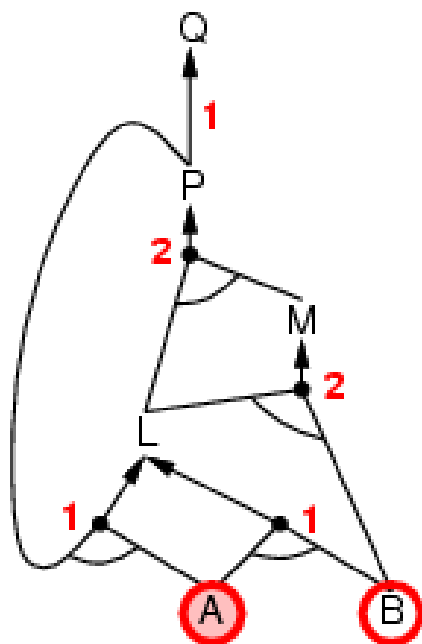
15

Forward chaining: esempio



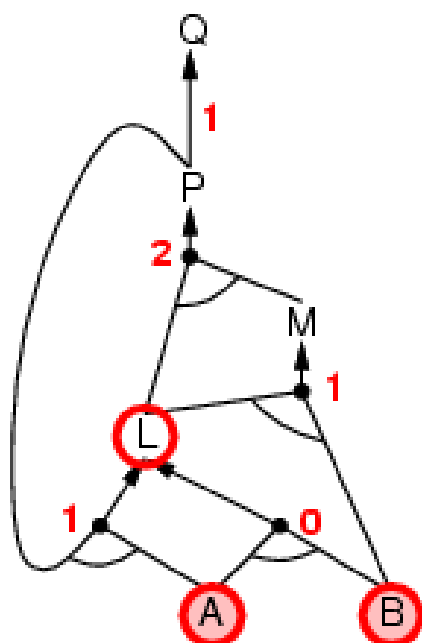
16

Forward chaining : esempio



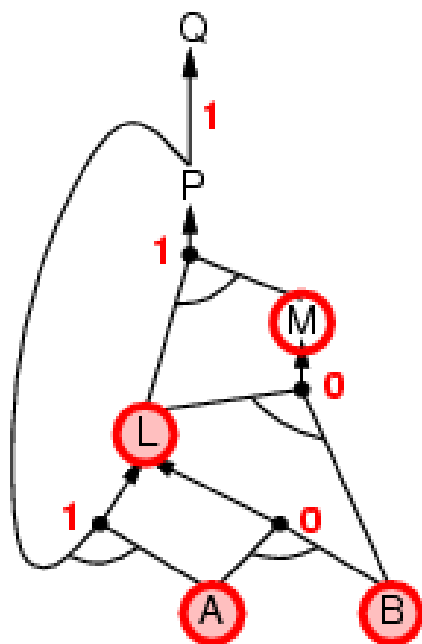
17

Forward chaining : esempio



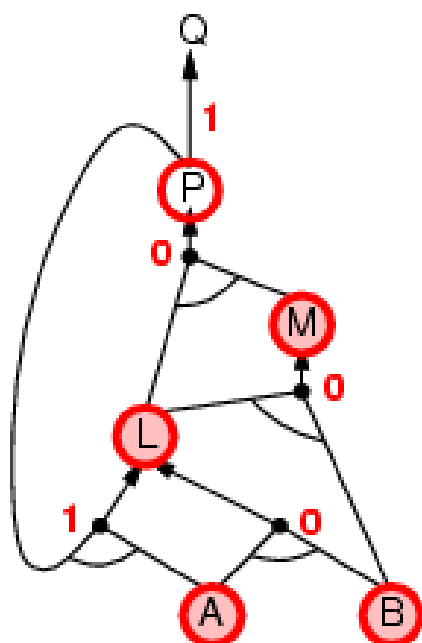
18

Forward chaining : esempio



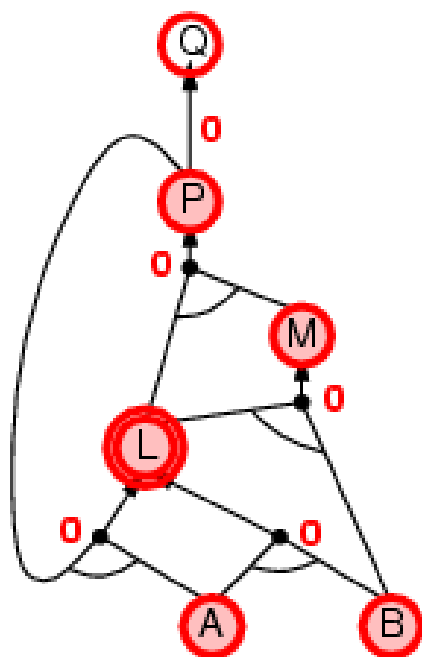
19

Forward chaining : esempio



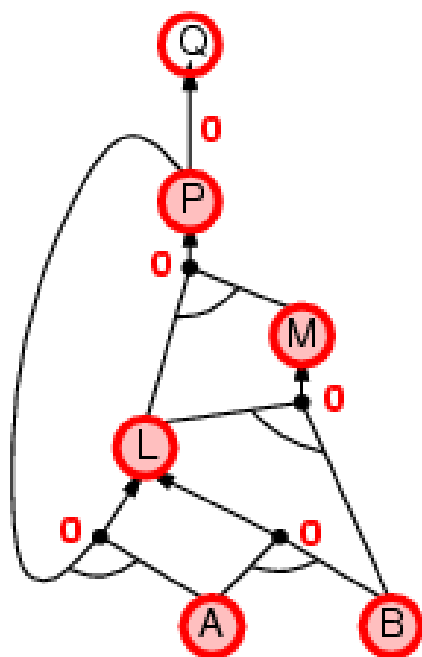
20

Forward chaining : esempio



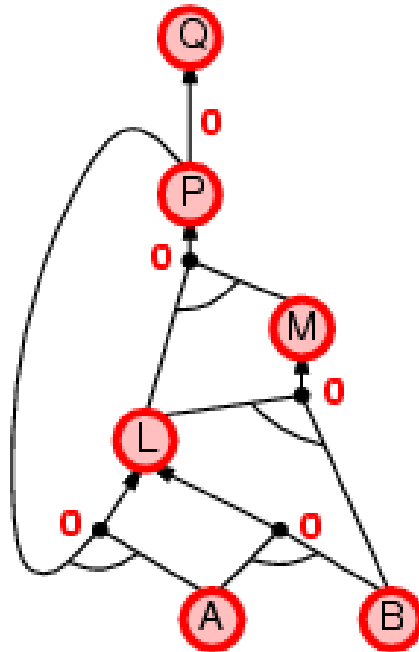
21

Forward chaining : esempio



22

Forward chaining : esempio



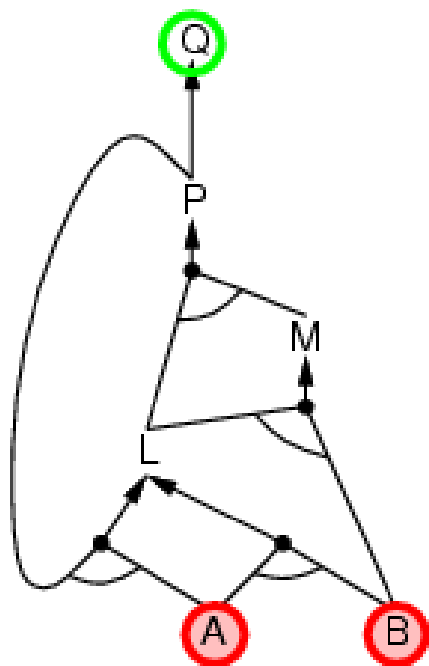
23

Backward chaining (BC)

- Idea: lavora backwards dalla query (goal) q :
 - Per dimostrare q
 - Controlla se q e' gia noto, oppure,
 - Dimostra attraverso BC tutte le premesse di una qualche regola che conclude q
- Per evitare loops: controlla se i nuovi sottogoals sono gia' nello stack dei goals
- Evita di ripetere lavoro controllando se i nuovi sottogoal sono gia' stati dimostrati veri o falsi.

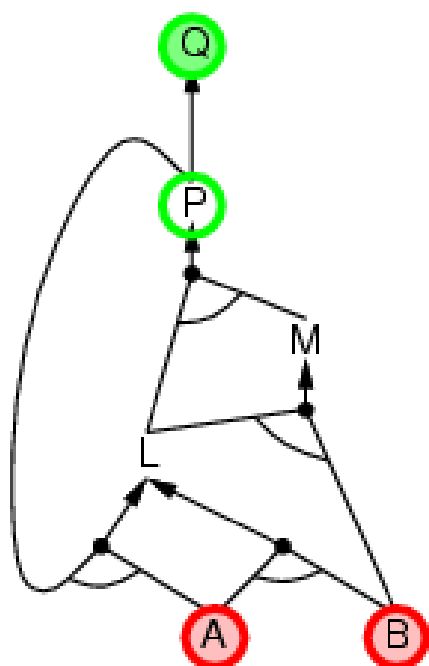
24

Backward chaining : esempio



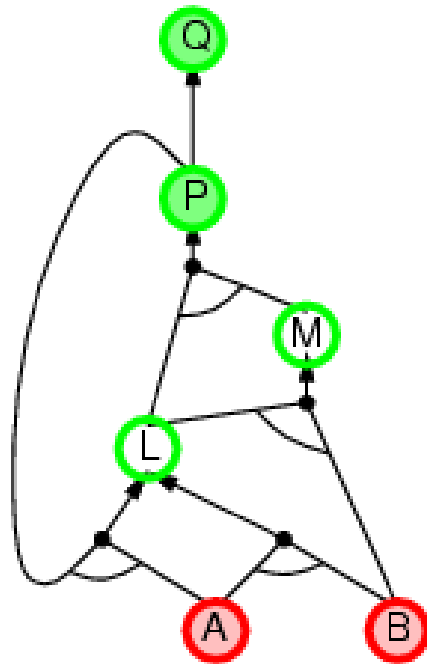
25

Backward chaining : esempio



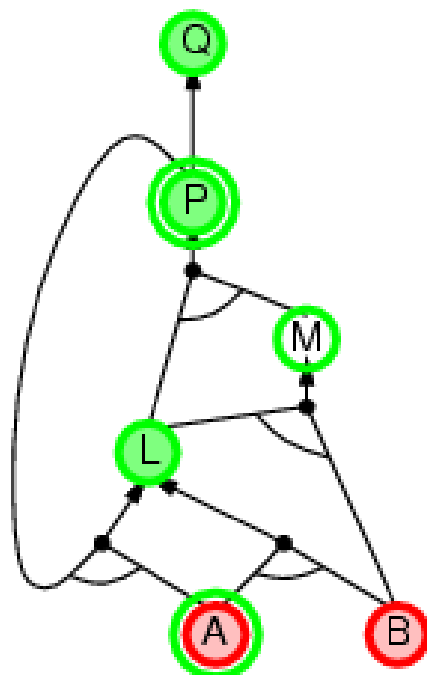
26

Backward chaining : esempio



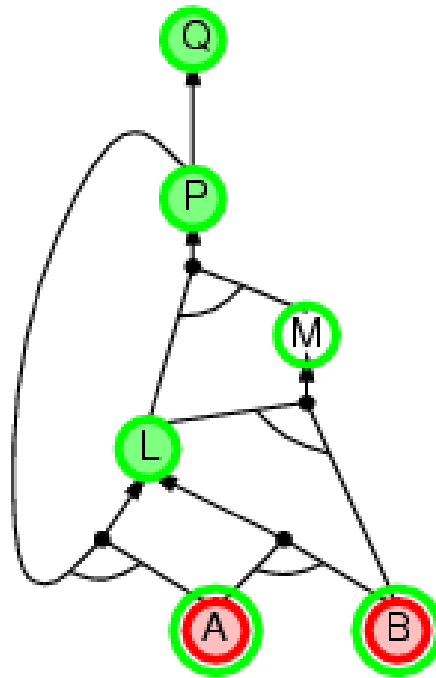
27

Backward chaining : esempio



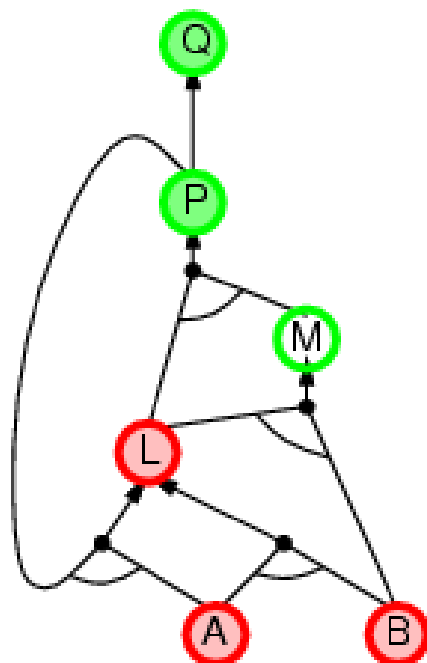
28

Backward chaining : esempio



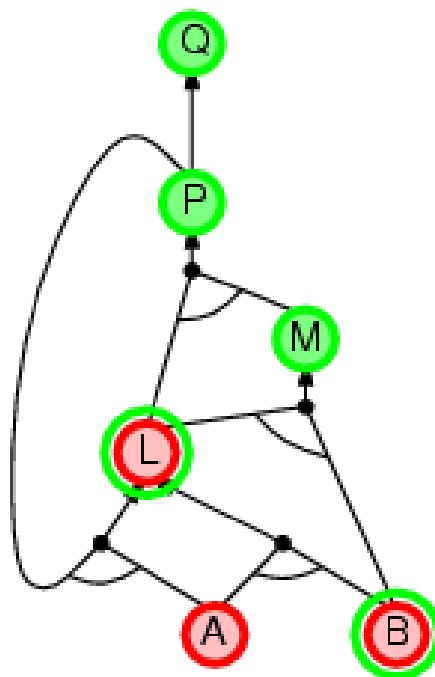
29

Backward chaining : esempio



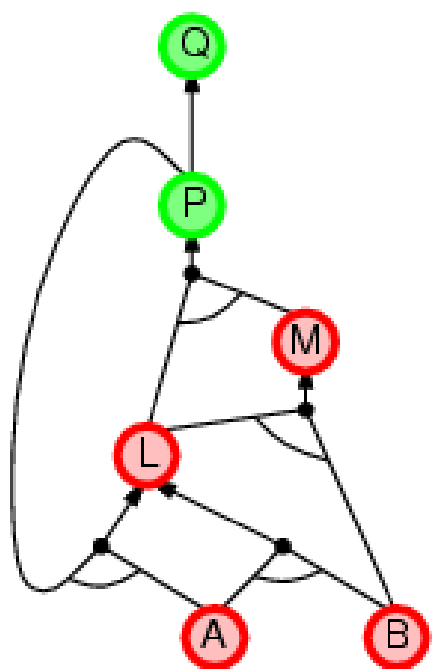
30

Backward chaining : esempio



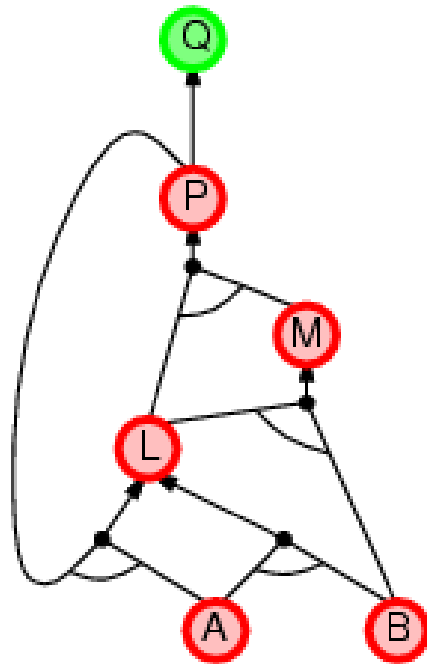
31

Backward chaining : esempio



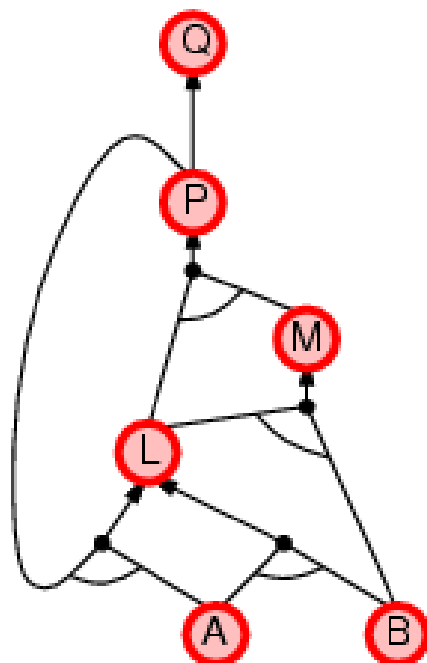
32

Backward chaining : esempio



33

Backward chaining : esempio



34

Forward vs. backward chaining

- FC e' **data-driven**,
 - e.g., monitoring, configurazione, interpretazione
 - Non si concentra sull'obiettivo
- BC e' **goal-driven**, quindi in generale piu' appropriato per il problem-solving

35

Sommario

- La risoluzione e' completa per la logica proposizionale
- Forward, backward chaining sono complete per le clausole di Horn
- La logica proposizionale e' troppo povera dal punto di vista espressivo.

36