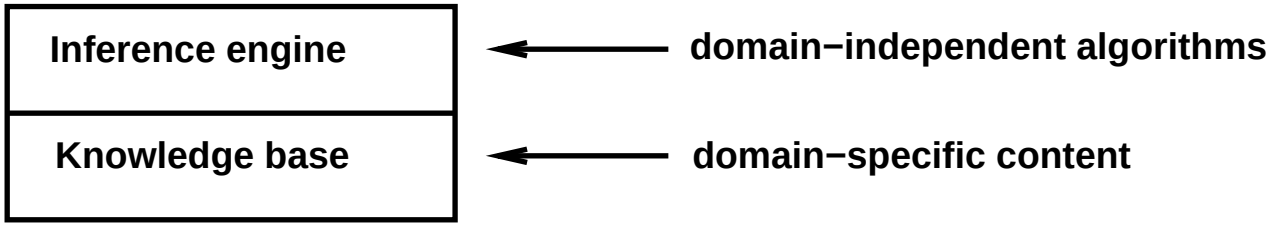
Agenti logici

Capitolo 7

Outline

- ♢ Agenti basati sulla conoscenza
- ♢ Il mondo dei Wumpus
- ♢ Logica in generale: modelli ed entailment
- ♢ Logica (Booleana) proposizionale
- ♢ Equivalenza, validità, soddisfacibilità
- ♢ Regole di inferenza e dimostrazione di teoremi
  - forward chaining
  - backward chaining
  - risoluzione

Base di conoscenza (Knowledge base)



Base di conoscenza = insieme di sentenze in un linguaggio formale

Approccio dichiarativo per la costruzione di un agente (o altro sistema):
DIRE (TELL) ad esso quello che ha bisogno di sapere

Quindi esso può CHIEDERE (ASK) a se stesso cosa fare—le risposte dovrebbero seguire dalla base di conoscenza (KB)

Gli agenti possono essere descritti al livello della conoscenza
cioè per quello che essi sanno, indipendentemente dall'implementazione

o a livello implementativo
cioè considerando le strutture dati nella KB e gli algoritmi che la manipolano

Un semplice agente basato sulla conoscenza

function KB-AGENT(*percept*) **returns** an *action*
  static: KB, a knowledge base
    t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE(*percept*, t))
  $action \leftarrow$ ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE($action$, t))
  $t \leftarrow t + 1$
  return *action*

L'agente deve essere capace di:

- Rappresentare stati, azioni, etc.
- Incorporare nuove percezioni
- Aggiornare le rappresentazioni interne del mondo (ambiente)
- Dedurre proprietà nascoste del mondo
- Dedurre le azioni appropriate da intraprendere

Il mondo dei Wumpus: descrizione PEAS

Misura di prestazione

oro +1000, morte -1000,

-1 per ogni spostamento, -10 per l'uso della freccia

Ambiente

Quadrati adiacenti ad un wumpus puzzano (stench)

Quadrati adiacenti ad una trappola (pit) sono ventilate

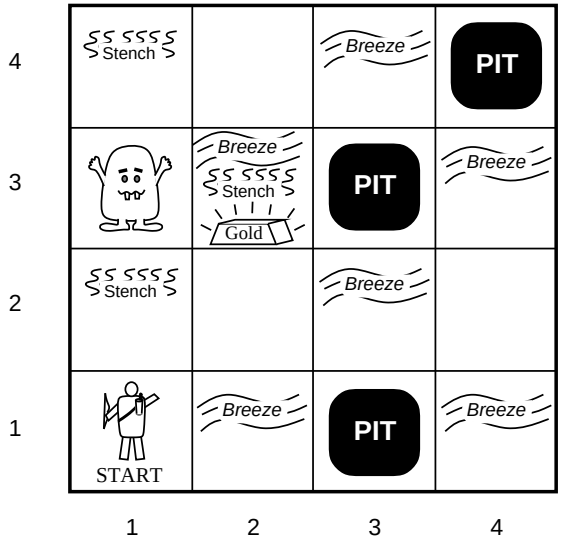
Luccichio (glitter) se e solo se l'oro è nello stesso quadrato

La freccia uccide il wumpus solo se l'agente è posto di fronte

La freccia può essere usata una sola volta

L'agente può prendere l'oro solo se si trova nello stesso quadrato

L'agente può lasciar cadere l'oro nel quadrato dove si trova



Sensori

brezza, luccichio, puzza

Attuatori

spostamento a sinistra, spostamento a destra,

avanti, prendi, lascia, lancia freccia

Caratterizzazione del mondo dei Wumpus

Osservabile??

Caratterizzazione del mondo dei Wumpus

Osservabile?? No—solo percezioni locali

Deterministico??

Caratterizzazione del mondo dei Wumpus

Osservabile?? No—solo percezioni locali

Deterministico?? Si—risultati delle azioni esattamente specificati

Episodico??

Caratterizzazione del mondo dei Wumpus

Osservabile?? No—solo percezioni locali

Deterministico?? Si—risultati delle azioni esattamente specificati

Episodico?? No—sequenziale al livello delle azioni

Statico??

Caratterizzazione del mondo dei Wumpus

Osservabile?? No—solo percezioni locali

Deterministico?? Si—risultati delle azioni esattamente specificati

Episodico?? No—sequenziale al livello delle azioni

Statico?? Si—Wumpus e trappole non si muovono

Discreto??

Caratterizzazione del mondo dei Wumpus

Osservabile?? No—solo percezioni locali

Deterministico?? Si—risultati delle azioni esattamente specificati

Episodico?? No—sequenziale al livello delle azioni

Statico?? Si—Wumpus e trappole non si muovono

Discreto?? Si

Agente Singolo??

Caratterizzazione del mondo dei Wumpus

Osservabile?? No—solo percezioni locali

Deterministico?? Si—risultati delle azioni esattamente specificati

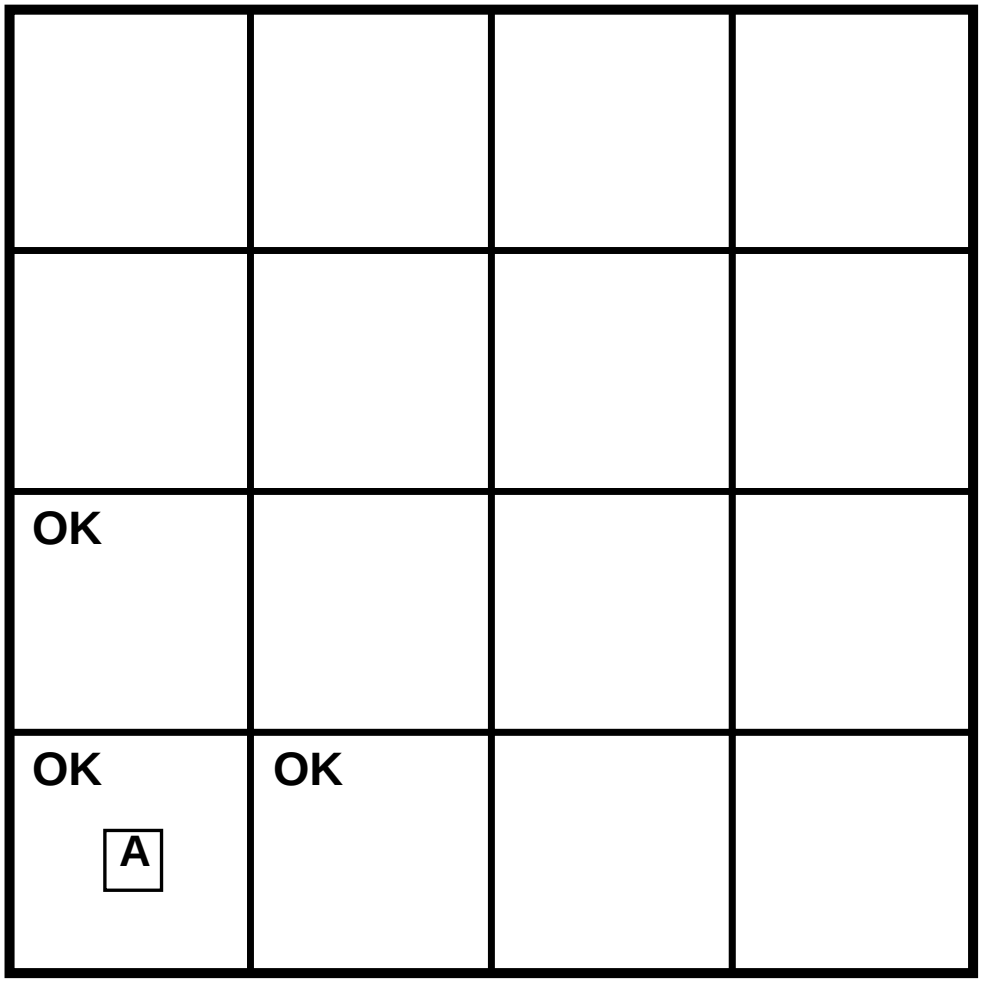
Episodico?? No—sequenziale al livello delle azioni

Statico?? Si—Wumpus e trappole non si muovono

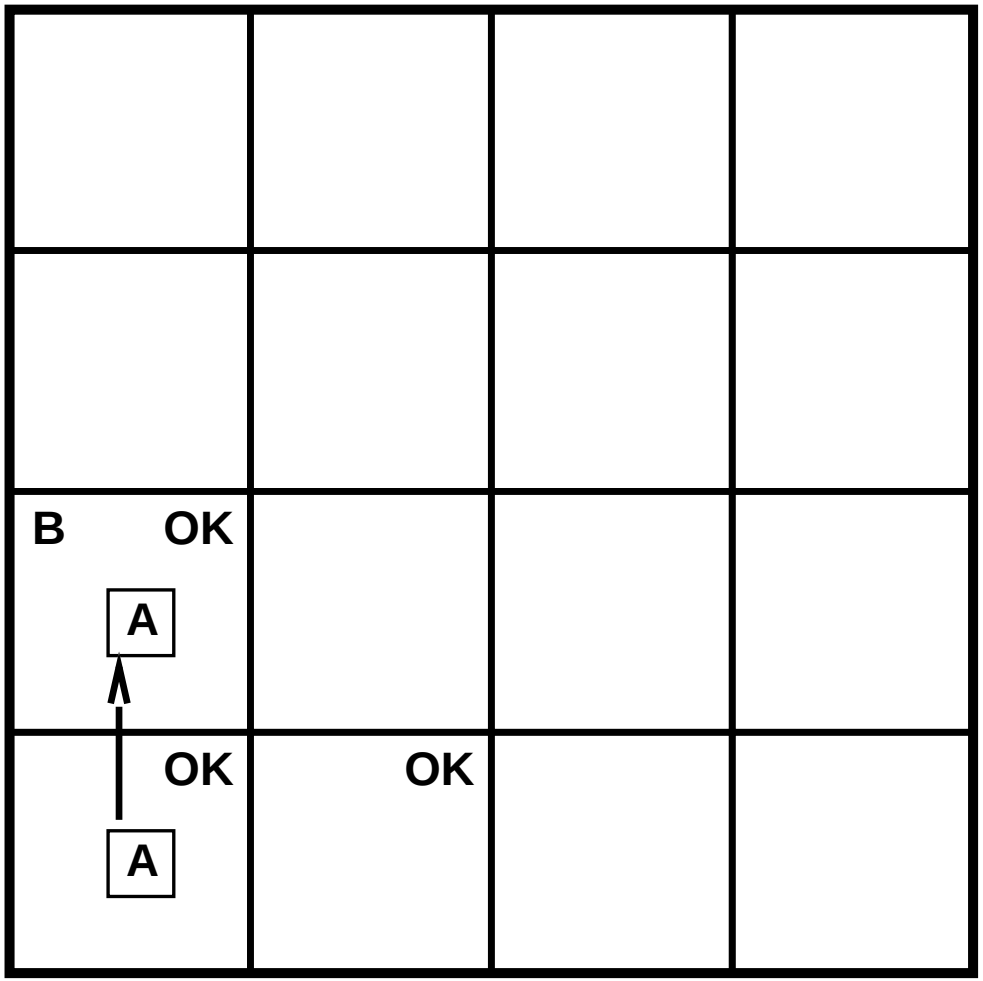
Discreto?? Si

Agente Singolo?? Si—Wumpus fa parte dell'ambiente

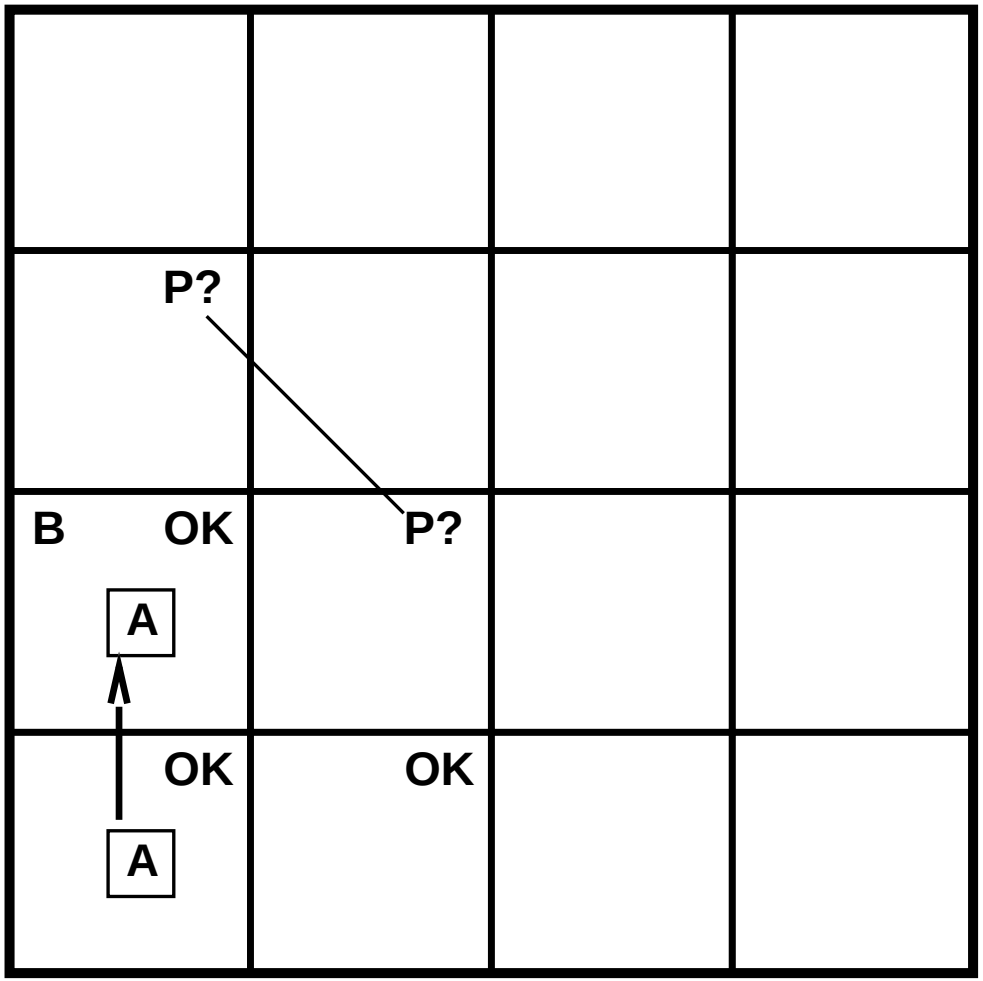
Esplorando il mondo dei wumpus



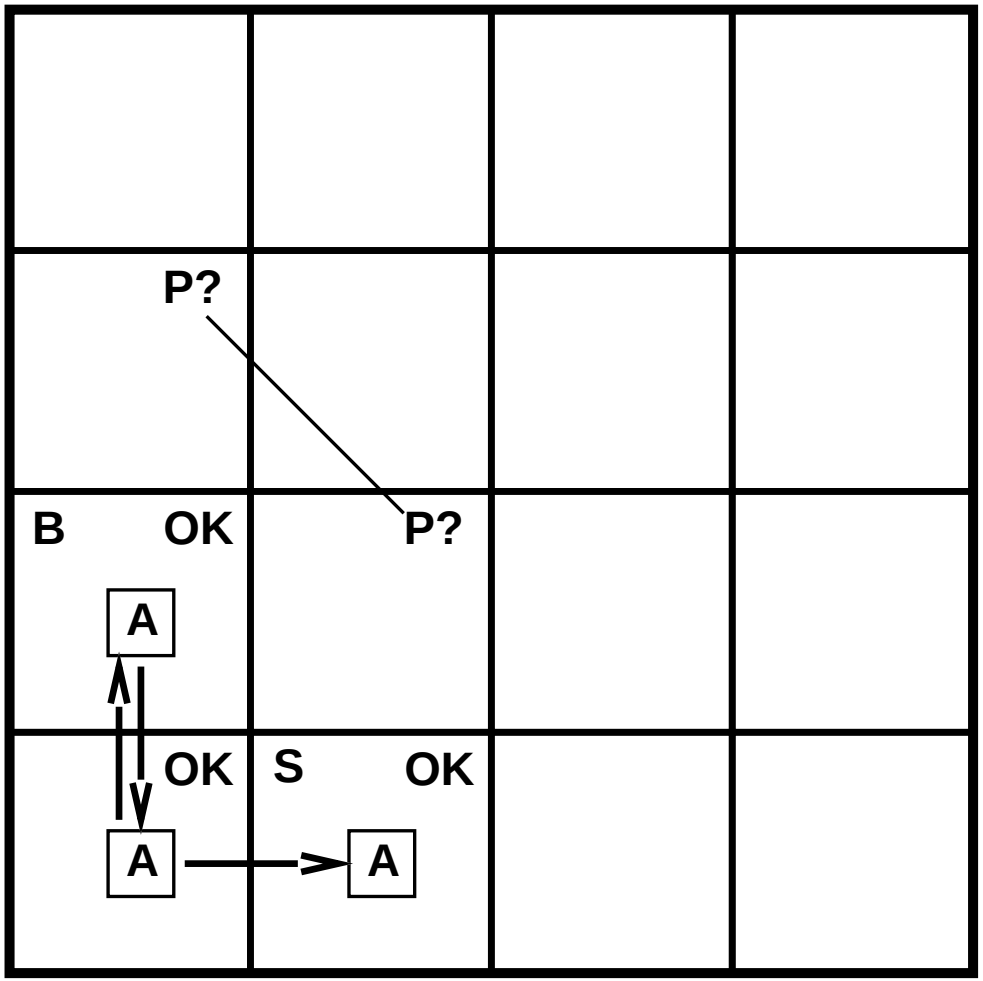
Esplorando il mondo dei wumpus



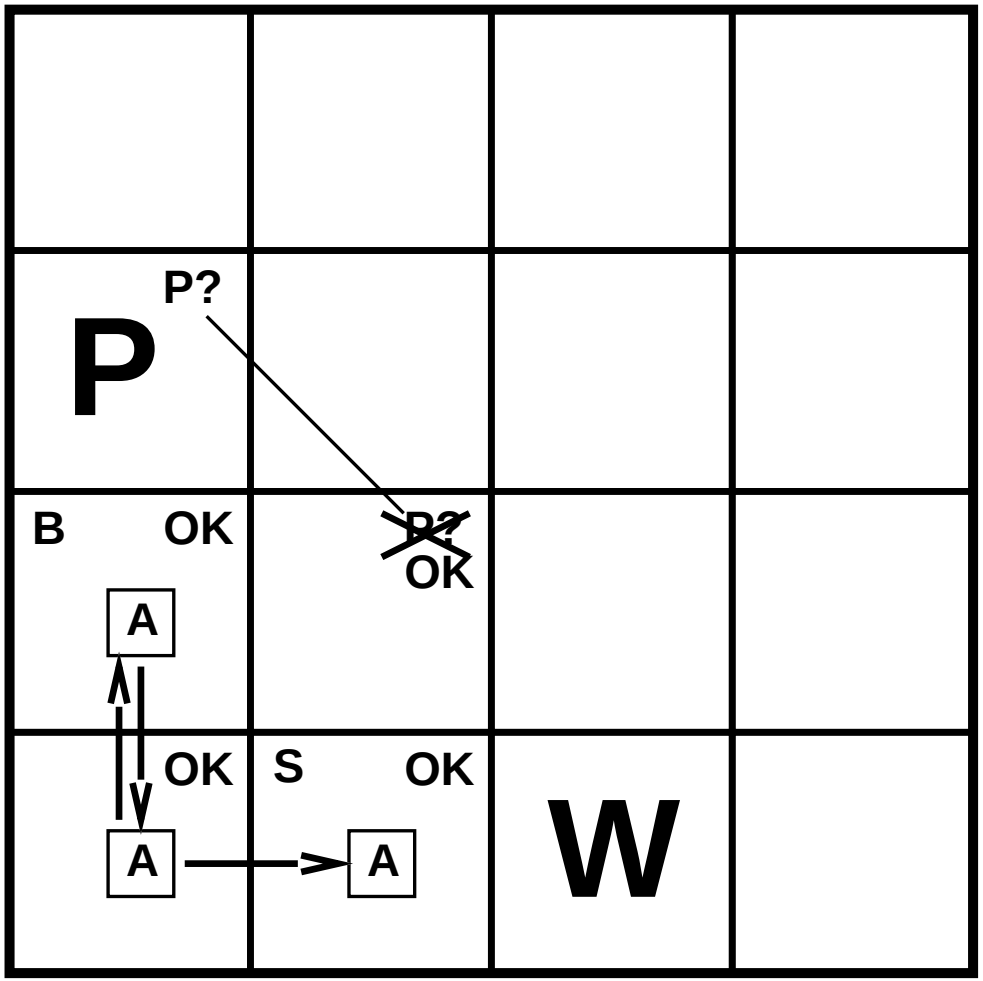
Esplorando il mondo dei wumpus



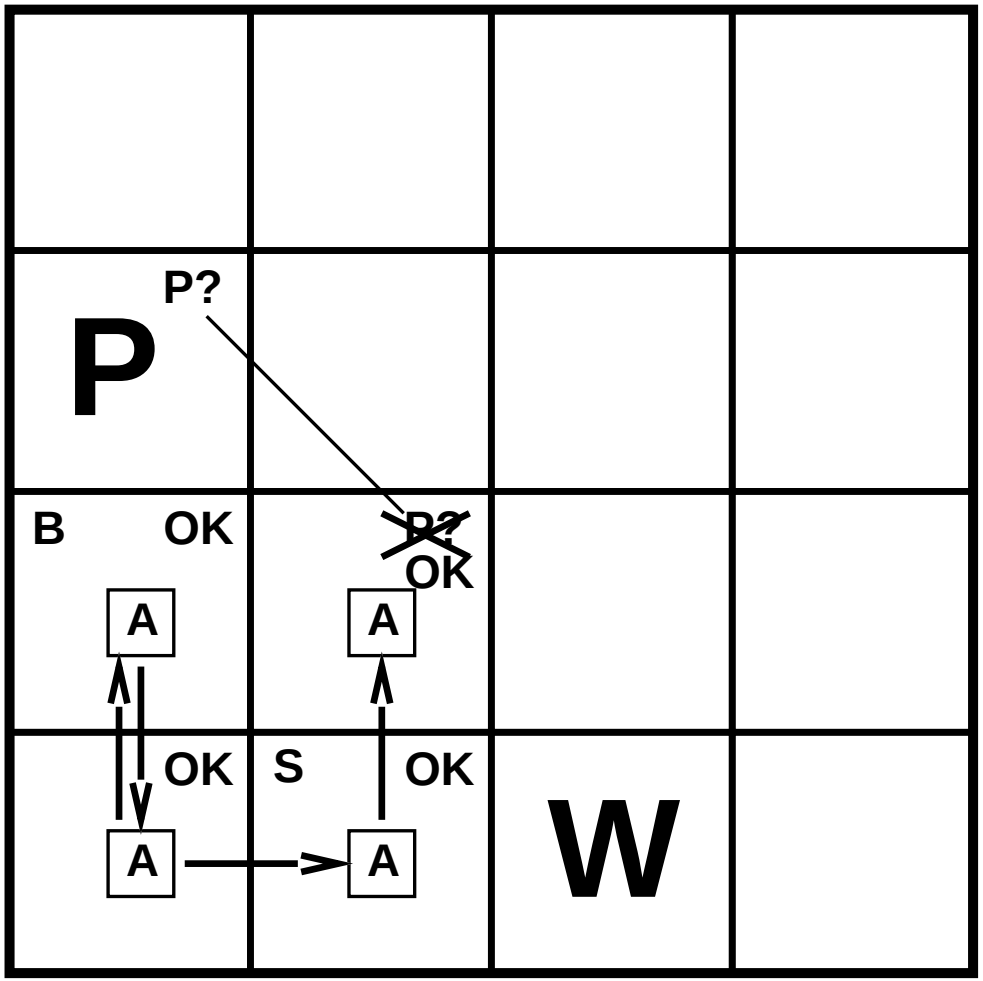
Esplorando il mondo dei wumpus



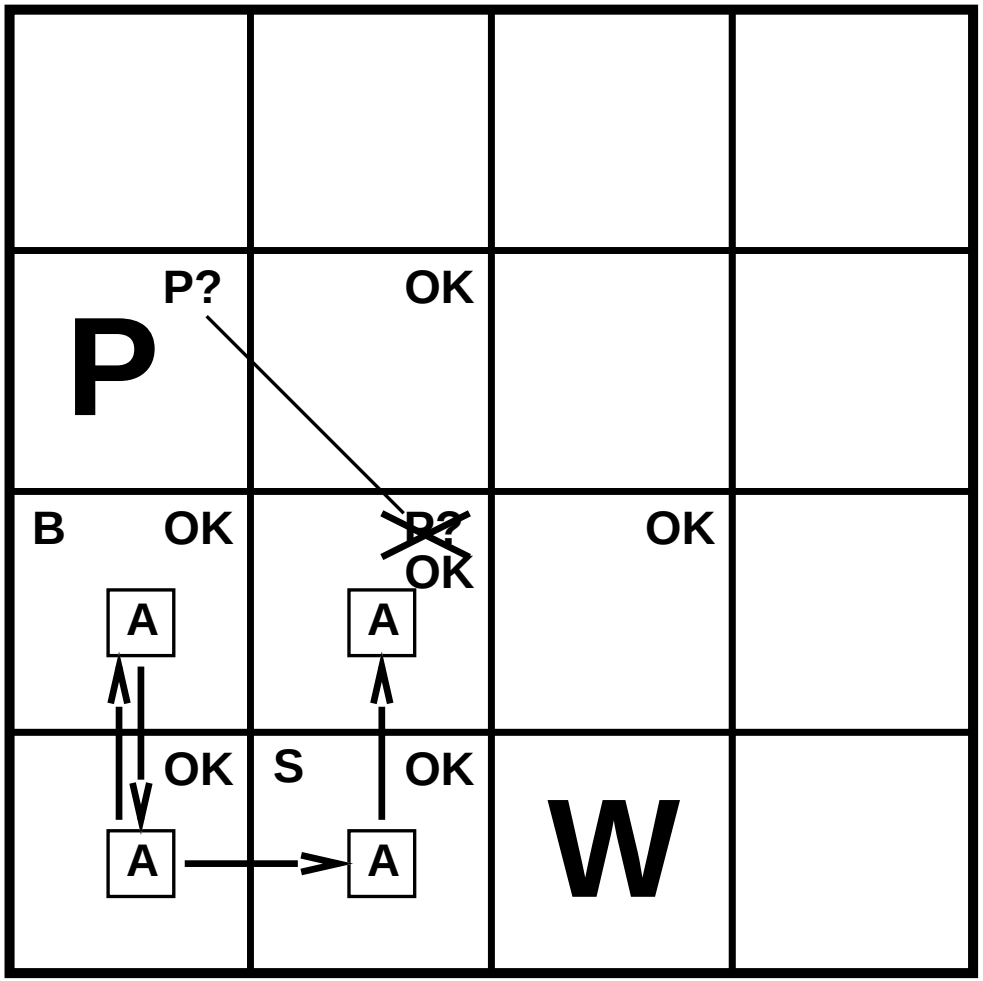
Esplorando il mondo dei wumpus



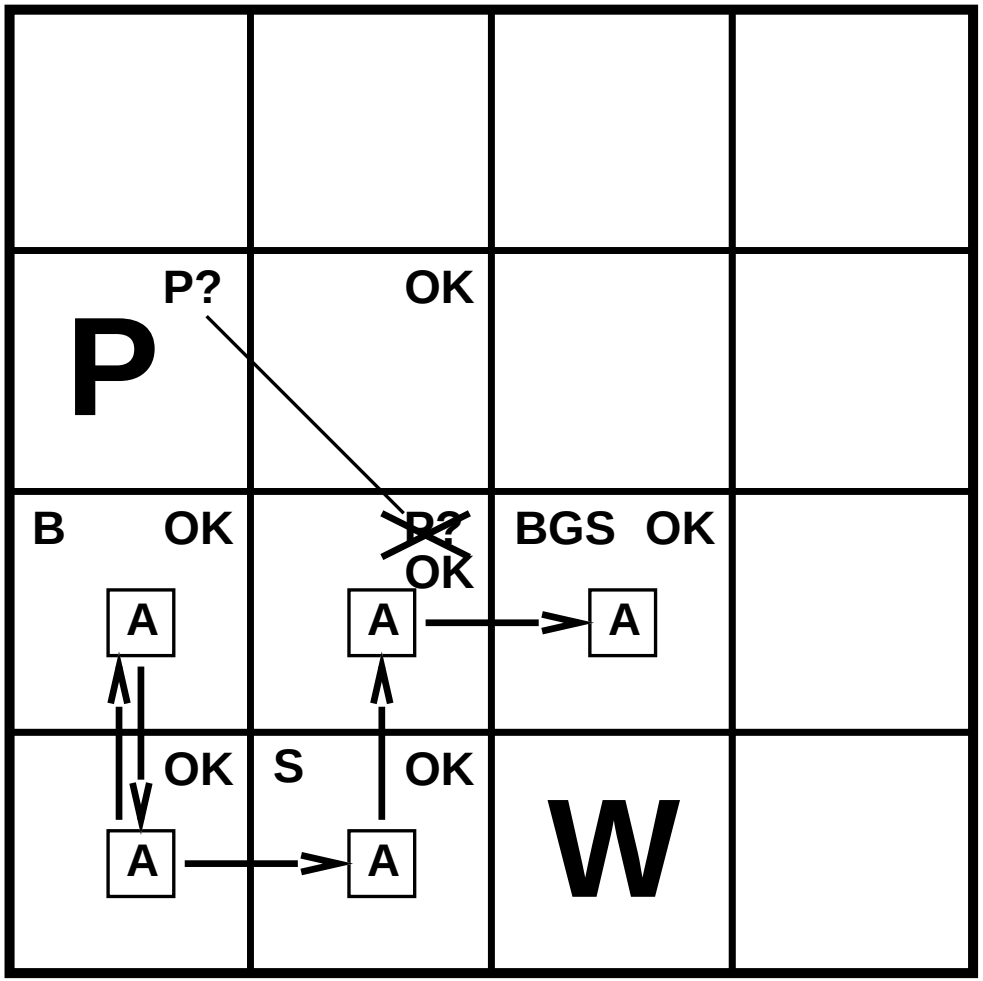
Esplorando il mondo dei wumpus



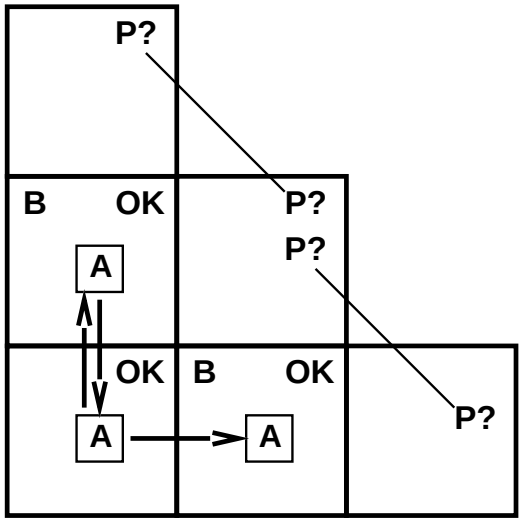
Esplorando il mondo dei wumpus



Esplorando il mondo dei wumpus

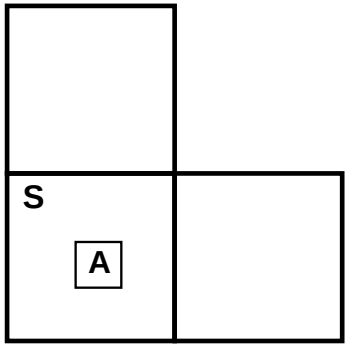


Altre situazioni critiche



Brezza in (1,2) e (2,1)
$\Rightarrow$ azioni non sicure

Assumendo le trappole uniformemente distribuite,
(2,2) contiene una trappola con prob 0.86, contro 0.31 per (1,2) e (2,1)



Puzza in (1,1)
$\Rightarrow$ non si può muovere
Si può usare una strategia di coercizione:
lancia la freccia di fronte
wumpus presente $\Rightarrow$ morto $\Rightarrow$ sicuro
wumpus assente $\Rightarrow$ sicuro

Logica in generale

Le logiche sono linguaggi formali per rappresentare l'informazione
in modo tale che si possano trarre delle conclusioni

La sintassi definisce le sentenze che appartengono al linguaggio

La semantica definisce il "significato" delle sentenze;
cioè, definisce il valore di verità di una sentenza in un dato mondo

Per esempio, nel linguaggio dell'aritmetica

$x + 2 \geq y$ è una sentenza; $x2 + y >$ non è una sentenza

$x + 2 \geq y$ è vera se e solo se il numero $x + 2$ non è minore del numero y

$x + 2 \geq y$ è vera in un mondo dove $x = 7,\ y = 1$
$x + 2 \geq y$ è falsa in un mondo dove $x = 0,\ y = 6$

Entailment

Entailment significa che una sentenza *segue da* un' altra (consegue):

$$KB \models \alpha$$

La base di conoscenza KB implica la sentenza α
se e solo se
α è vera in tutti i mondi dove KB è vera

Per esempio, la KB contenente "l'INTER ha vinto" e "il MILAN ha vinto"
ne consegue che "ha vinto l'INTER o ha vinto il MILAN"

Per esempio, $x + y = 4$ implica $4 = x + y$

L'implicazione è una relazione fra sentenze (cioè, *sintassi*)
che è basata sulla *semantica*

Modelli

I logici tipicamente pensano in termini di modelli, che formalmente sono mondi strutturati rispetto ai quali si può valutare la verità

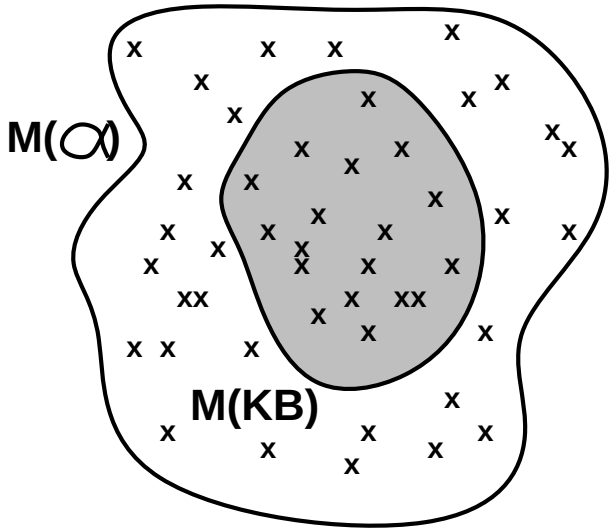
Diciamo che m è un modello di una sentenza α se α è vera in m

$M(\alpha)$ è l'insieme di tutti i modelli di α

Allora $KB \models \alpha$ se e solo se $M(KB) \subseteq M(\alpha)$

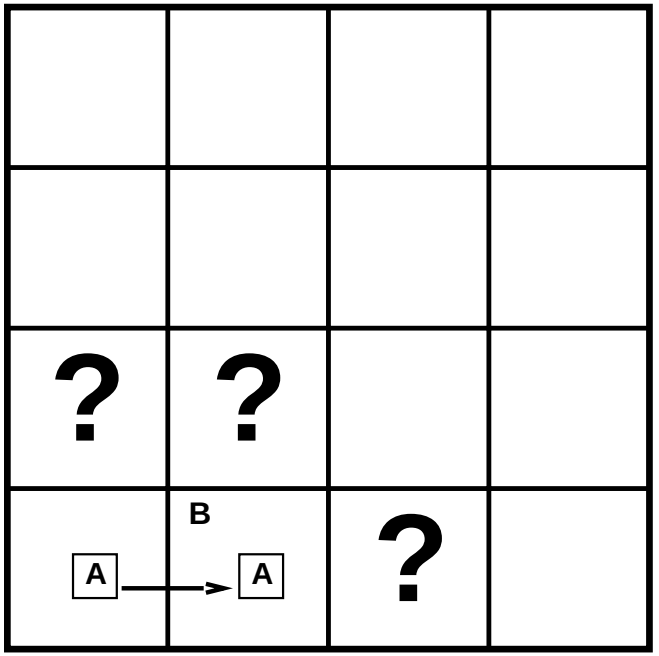
Per esempio, KB = l'INTER ha vinto e il MILAN ha vinto

α = l'INTER ha vinto



Entailment nel mondo dei wumpus

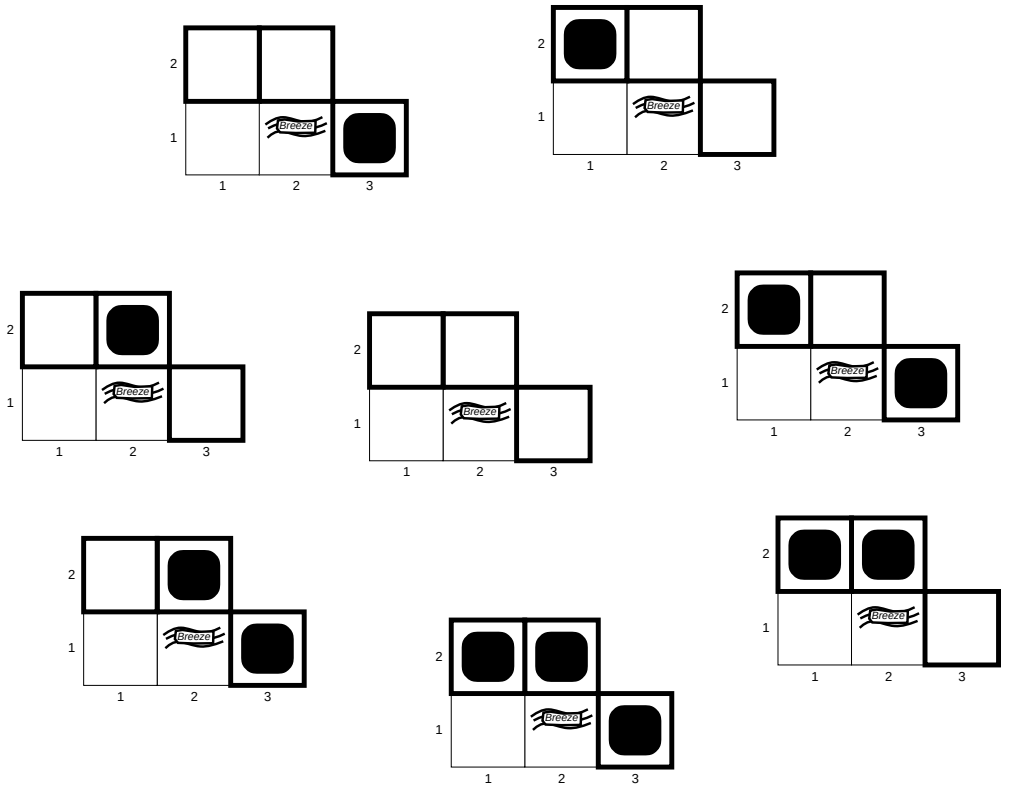
Situazione dopo aver riconosciuto che non c'è nulla in [1,1],
spostamento a destra, brezza in [2,1]



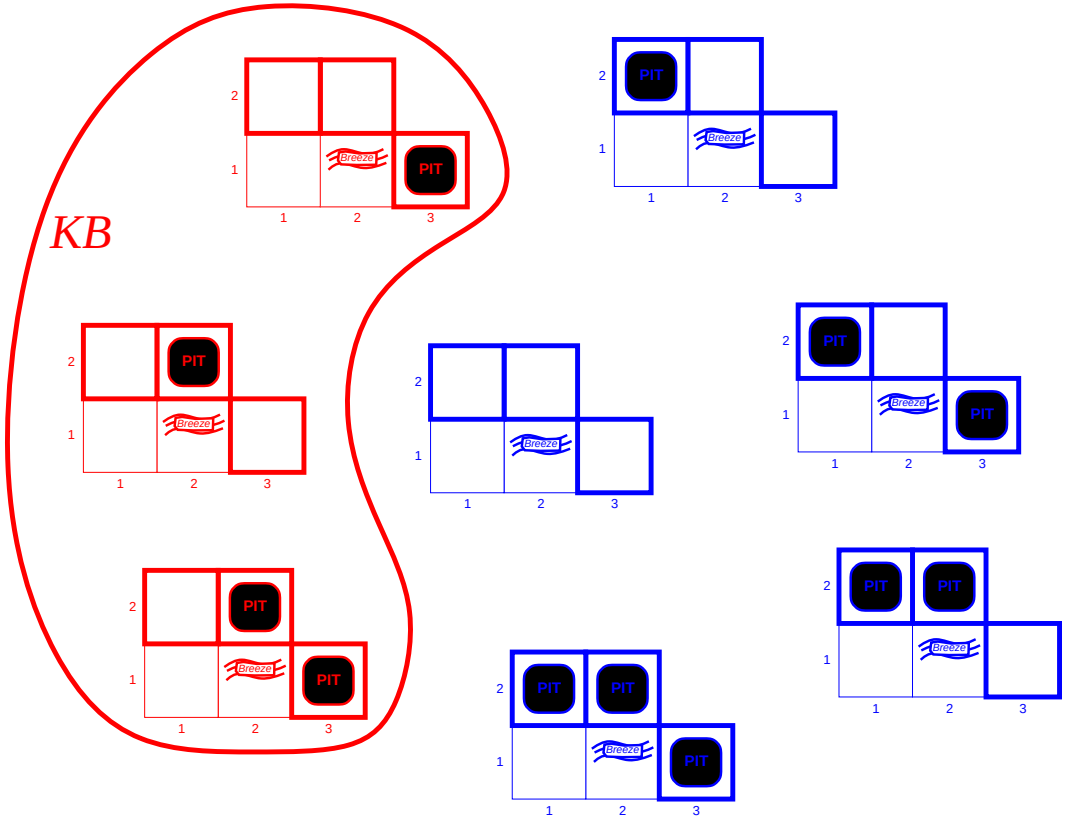
Consideriamo modelli possibili per i "?" assumendo solo trappole

3 scelte Booleane $\Rightarrow$ 8 possibili modelli

Modelli per il mondo dei wumpus

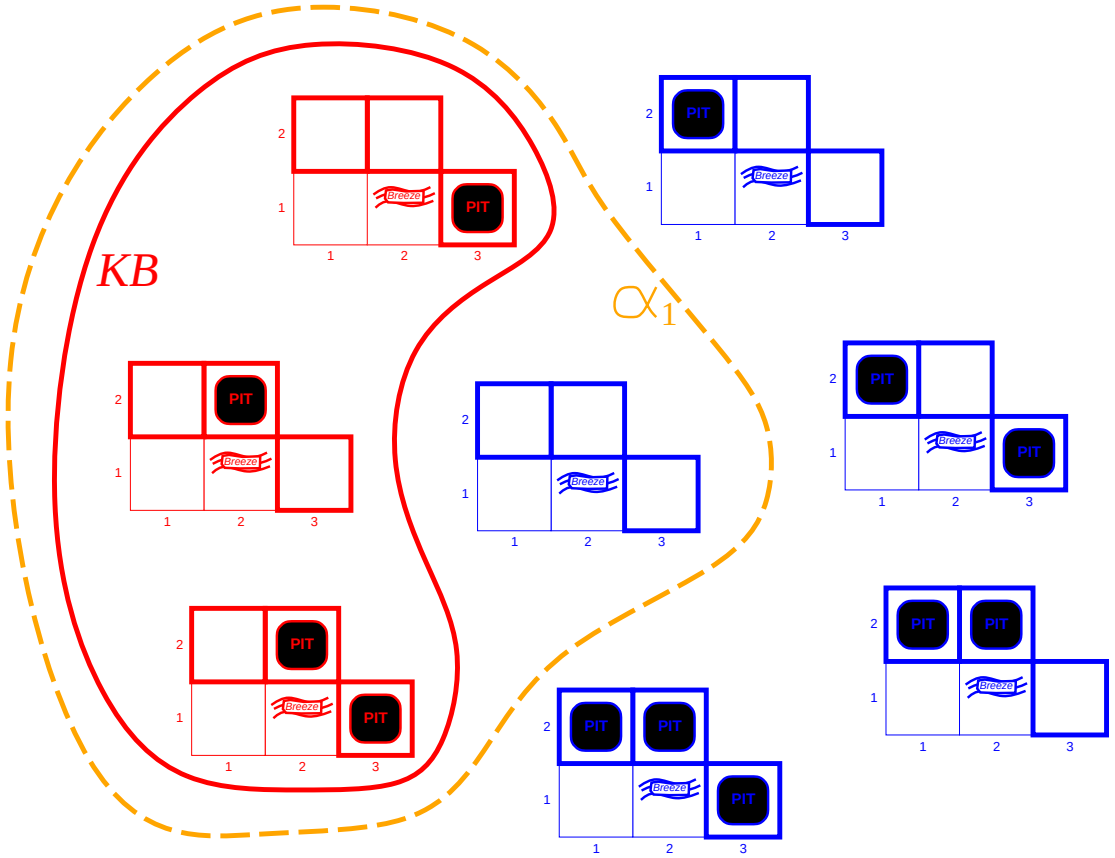


Modelli per il mondo dei wumpus



KB = regole del mondo dei wumpus + osservazioni

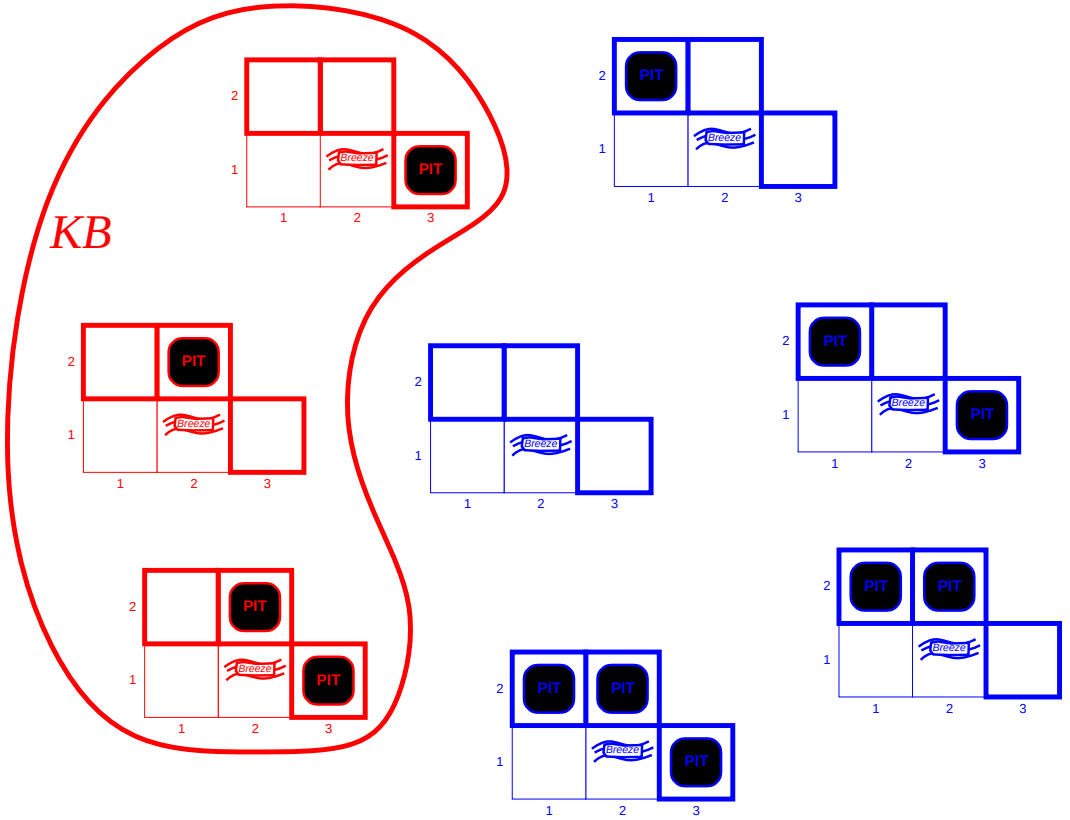
Modelli per il mondo dei wumpus



$KB =$ regole del mondo dei wumpus + osservazioni

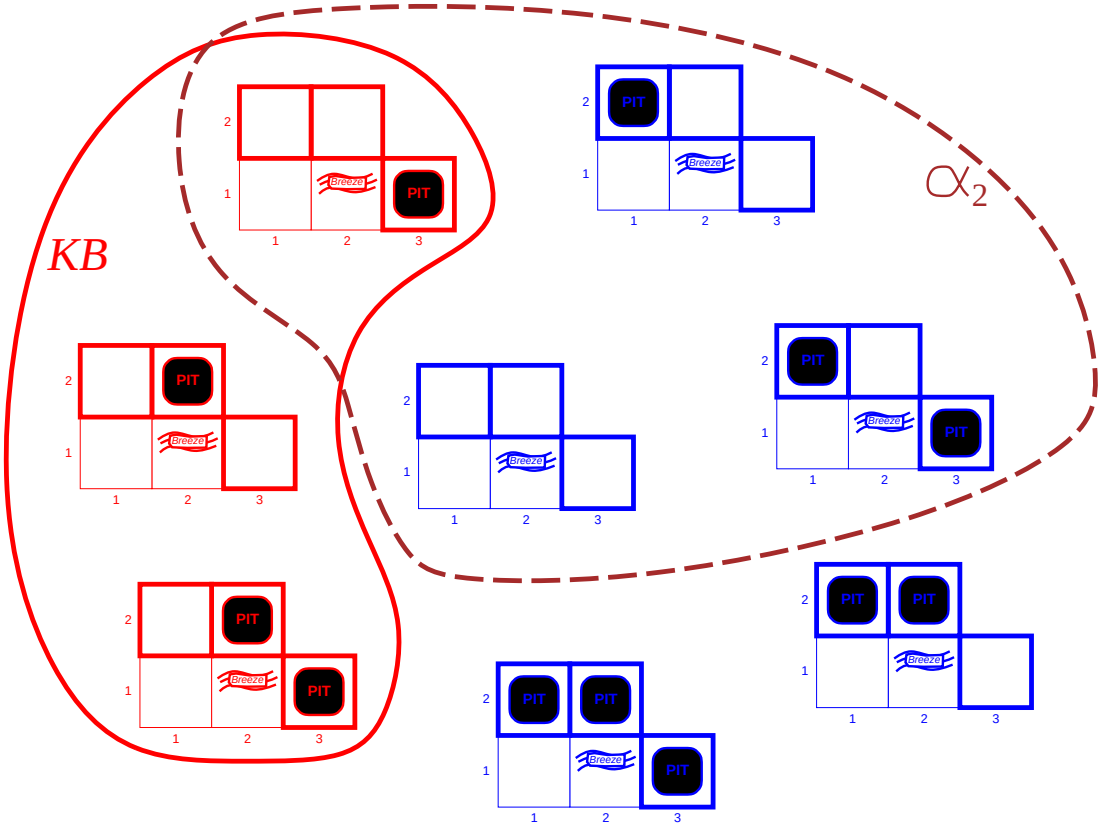
$\alpha_1 =$ "[1,2] è sicuro", $KB \models \alpha_1$, provato via model checking (test sui modelli)

Modelli per il mondo dei wumpus



KB = regole del mondo dei wumpus + osservazioni

Modelli per il mondo dei wumpus



$KB =$ regole del mondo dei wumpus $+$ osservazioni

$\alpha_2 =$ "[2,2] è sicuro", $KB \not\models \alpha_2$

Inferenza

$KB \vdash_i \alpha$ = la sentenza α può essere derivata da KB per mezzo della procedura i

Le conseguenze di KB costituiscono un "pagliaio"; α è un "ago".
Entailment = "ago nel pagliaio"; inferenza = trovarlo

Correttezza (Soundness): i è corretta se
ogni volta che $KB \vdash_i \alpha$, è anche vero che $KB \models \alpha$

Completezza (Completeness): i è completa se
ogni volta che $KB \models \alpha$, è anche vero che $KB \vdash_i \alpha$

Anticipazione: definiremo una logica (logica del primo ordine) che è abbastanza espressiva da essere in grado di esprimere quasi tutto ciò che interessa, e per cui esiste una procedura di inferenza corretta e completa.

In sintesi, la procedura risponde a tutte le domande la cui risposta segue da ciò che è conosciuto dalla KB.

Logica Proposizionale: Sintassi

La logica proposizionale è la più semplice—illustra idee di base

I simboli proposizionali P_1, P_2 etc costituiscono sentenze

Se S è una sentenza, $\neg S$ è una sentenza (negazione)

Se S_1 e S_2 sono sentenze, $S_1 \wedge S_2$ è una sentenza (congiunzione)

Se S_1 e S_2 sono sentenze, $S_1 \vee S_2$ è una sentenza (disgiunzione)

Se S_1 e S_2 sono sentenze, $S_1 \Rightarrow S_2$ è una sentenza (implicazione)

Se S_1 e S_2 sono sentenze, $S_1 \Leftrightarrow S_2$ è una sentenza (bicondizionale)

Logica Proposizionale: Semantica

Ogni modello specifica il valore di verità (vero/falso) per ogni simbolo proposizionale

P.e. $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
$vero$ $vero$ $falso$

(Con questi simboli, 8 possibili modelli possono essere enumerti automaticamente.)

Regole per valutare la verità rispetto ad un modello m:

$\neg S$	è vero sse	S	è falso				
$S_1 \wedge S_2$	è vero sse	S_1	è vero and	S_2	è vero		
$S_1 \vee S_2$	è vero sse	S_1	è vero or	S_2	è vero		
$S_1 \Rightarrow S_2$	è vero sse	S_1	è falso or	S_2	è vero		
cioè,	è falso sse	S_1	è vero and	S_2	è falso		
$S_1 \Leftrightarrow S_2$	è vero sse	$S_1 \Rightarrow S_2$	è vero and	$S_2 \Rightarrow S_1$	è vero		

Un semplice processo ricorsivo valuta una sentenza arbitraria, p.e.,
$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = vero \wedge (falso \vee vero) = vero \wedge vero = vero$

Tabelle di verità per i connettivi

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$falso$	$falso$	$vero$	$falso$	$falso$	$vero$	$vero$
$falso$	$vero$	$vero$	$falso$	$vero$	$vero$	$falso$
$vero$	$falso$	$falso$	$falso$	$vero$	$falso$	$falso$
$vero$	$vero$	$falso$	$vero$	$vero$	$vero$	$vero$

Sentenze nel mondo dei wumpus

Poniamo $P_{i,j}$ essere vero se c'è una trappola in $[i, j]$.
Poniamo $B_{i,j}$ essere vero se c'è brezza in $[i, j]$.

$$\neg P_{1,1}$$
$$\neg B_{1,1}$$
$$B_{2,1}$$

"Le trappole causano brezze in quadrati adiacenti"

Sentenze nel mondo dei wumpus

Poniamo $P_{i,j}$ essere vero se c'è una trappola in $[i, j]$.
Poniamo $B_{i,j}$ essere vero se c'è brezza in $[i, j]$.

$$\neg P_{1,1}$$
$$\neg B_{1,1}$$
$$B_{2,1}$$

"Le trappole causano brezze in quadrati adiacenti"

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$
$$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

"Un quadrato ha brezza *se e solo se* c'è una trappola adiacente"

Tabelle di verità per l'inferenza

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
$falso$	$falso$	$falso$	$falso$	$falso$	$falso$	$falso$	$falso$	$vero$
$falso$	$falso$	$falso$	$falso$	$falso$	$falso$	$vero$	$falso$	$vero$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$falso$	$vero$	$falso$	$falso$	$falso$	$falso$	$falso$	$falso$	$vero$
$falso$	$vero$	$falso$	$falso$	$falso$	$falso$	$vero$	$\underline{vero}$	$\underline{vero}$
$falso$	$vero$	$falso$	$falso$	$falso$	$vero$	$falso$	$\underline{vero}$	$\underline{vero}$
$falso$	$vero$	$falso$	$falso$	$falso$	$vero$	$vero$	$\underline{vero}$	$\underline{vero}$
$falso$	$vero$	$falso$	$falso$	$vero$	$falso$	$falso$	$falso$	$vero$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$vero$	$vero$	$vero$	$vero$	$vero$	$vero$	$vero$	$falso$	$falso$

Inferenza per mezzo di enumerazione

Una enumerazione a scandaglio (Depth-first) di tutti i modelli è corretta e completa

function TT-ENTAILS?(KB, α) **returns** *true* or *false*

  $symbols \leftarrow$ a list of the proposition symbols in KB and α
  return TT-CHECK-ALL($KB, \alpha, symbols, [\,]$)

---

function TT-CHECK-ALL($KB, \alpha, symbols, model$) **returns** *true* or *false*
  if EMPTY?($symbols$) **then**
    if PL-TRUE?($KB, model$) **then return** PL-TRUE?($\alpha, model$)
    else return *true*
  else do
    $P \leftarrow$ FIRST($symbols$); $rest \leftarrow$ REST($symbols$)
    return TT-CHECK-ALL($KB, \alpha, rest$, EXTEND($P, true, model$)) **and**
        TT-CHECK-ALL($KB, \alpha, rest$, EXTEND($P, false, model$))

$O(2^n)$ per n simboli; il problema è co-NP-complete

Equivalenza Logica

Due sentenze sono logicamente equivalenti sse sono vere negli stessi modelli:

$\alpha \equiv \beta$ se e solo se $\alpha \models \beta$ and $\beta \models \alpha$

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

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

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

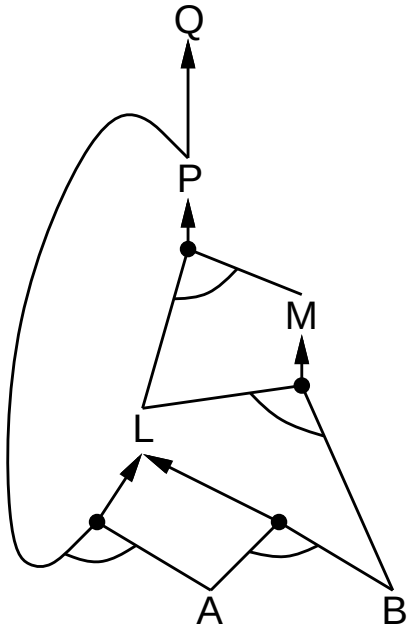
Può essere usato con forward chaining o backward chaining.

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

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

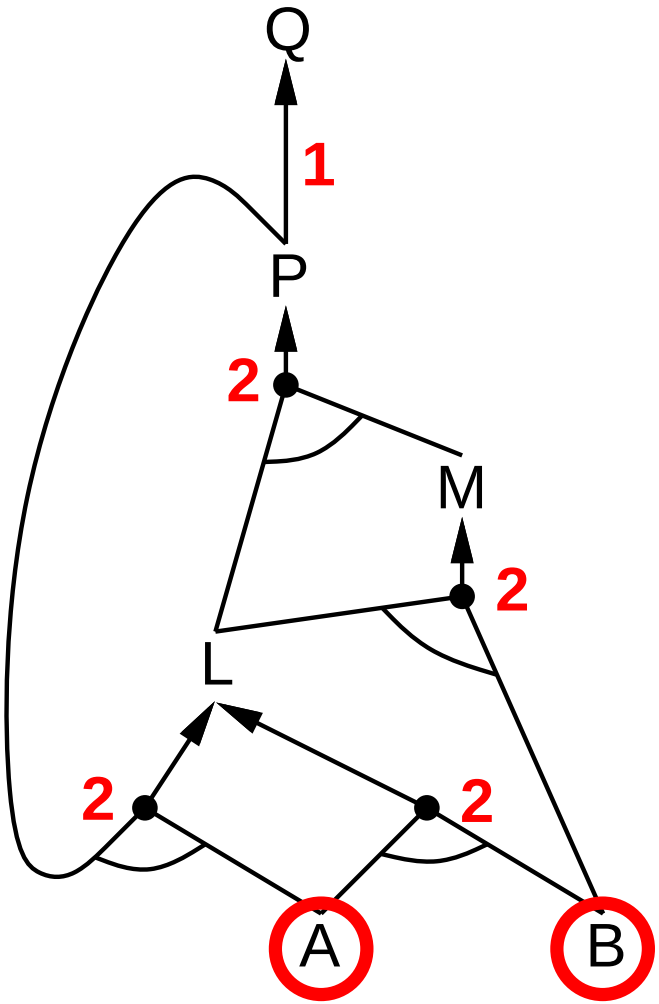


Algoritmo di Forward chaining

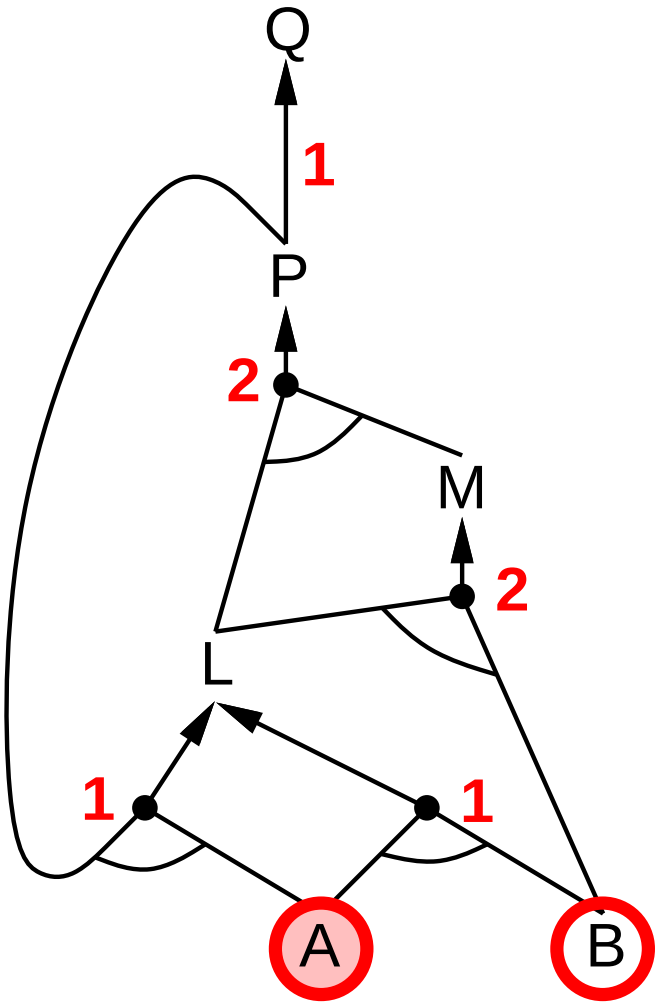
function PL-FC-ENTAILS?(KB, q) **returns** *true* or *false*
  local variables: *count*, a table, indexed by clause, initially the number of premises
    inferred, a table, indexed by symbol, each entry initially *false*
    agenda, a list of symbols, initially the symbols known to be true

  while *agenda* is not empty **do**
    $p \leftarrow$ POP(*agenda*)
    unless *inferred*[p] **do**
      inferred[p] $\leftarrow$ *true*
      for each Horn clause c in whose premise p appears **do**
        decrement *count*[c]
        if *count*[c] = 0 **then do**
          if HEAD[c] = q **then return** *true*
          PUSH(HEAD[c], *agenda*)
  return *false*

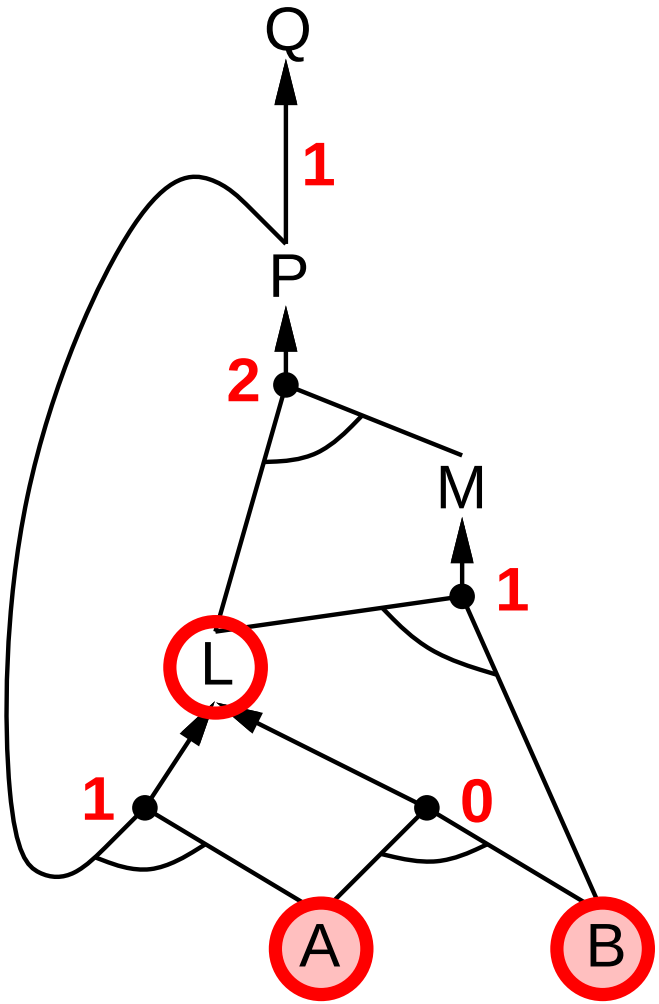
Esempio di Forward chaining



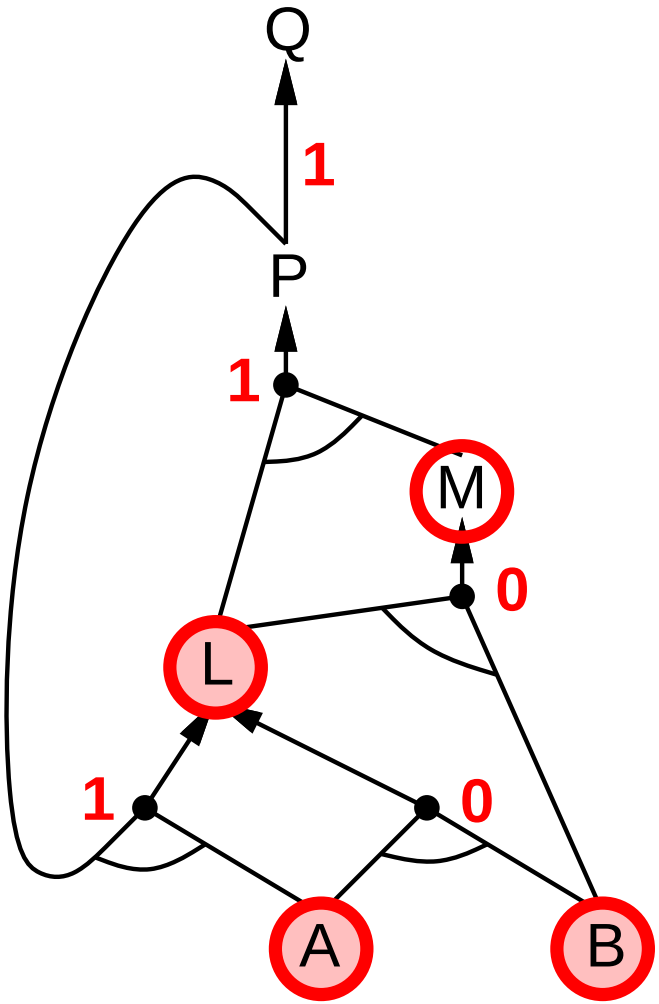
Esempio di Forward chaining



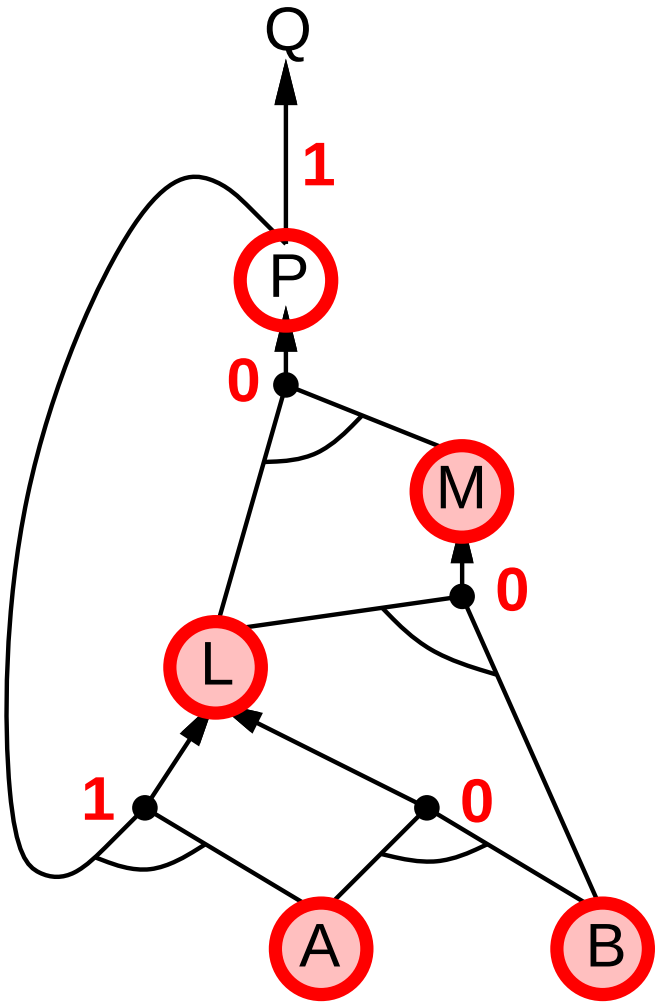
Esempio di Forward chaining



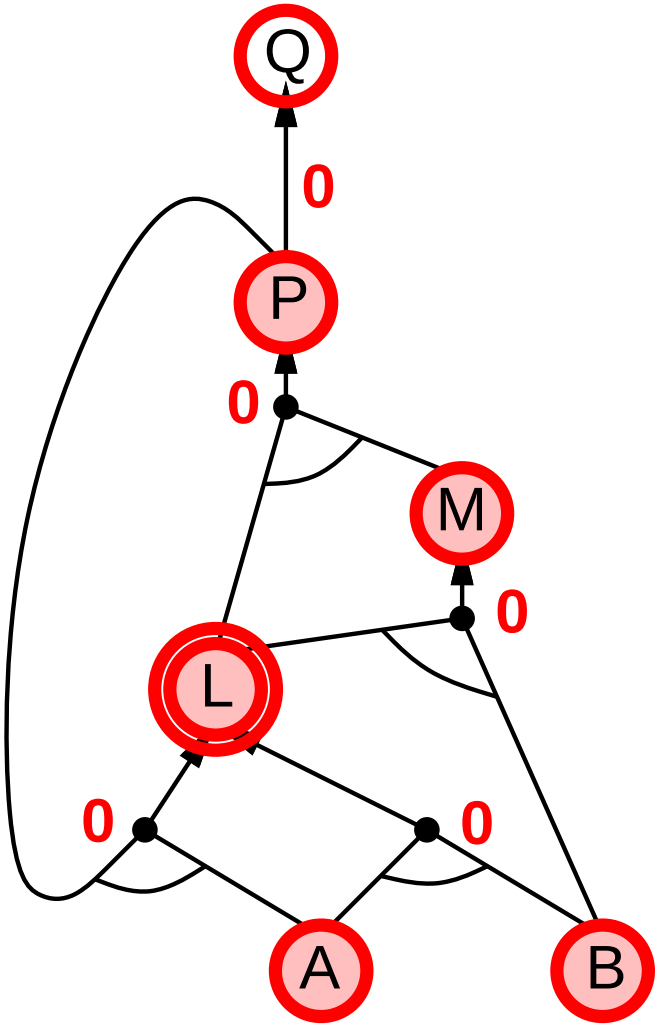
Esempio di Forward chaining



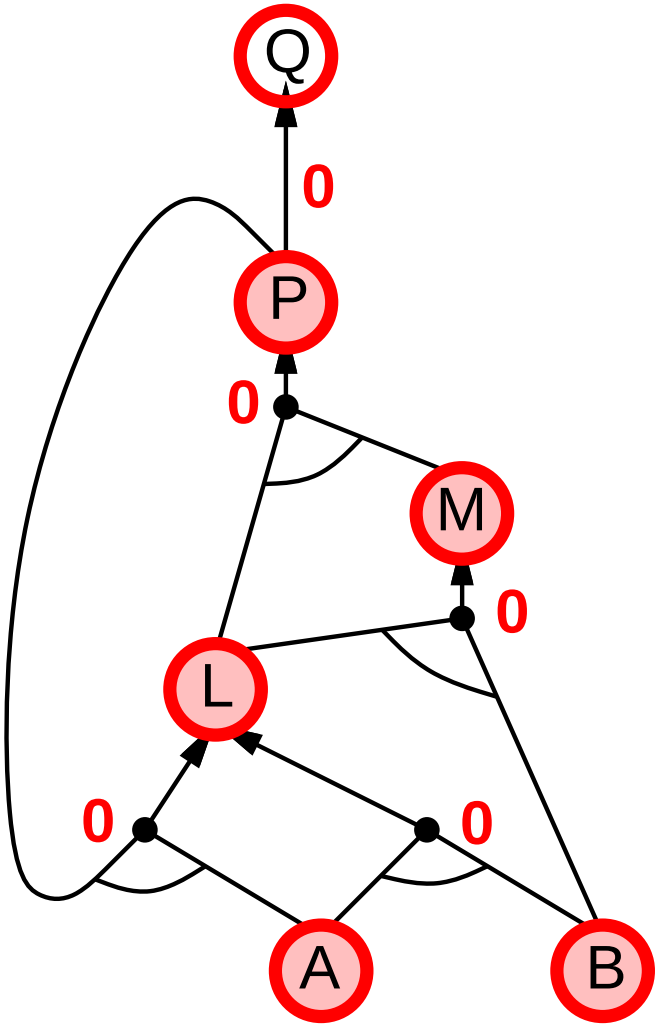
Esempio di Forward chaining



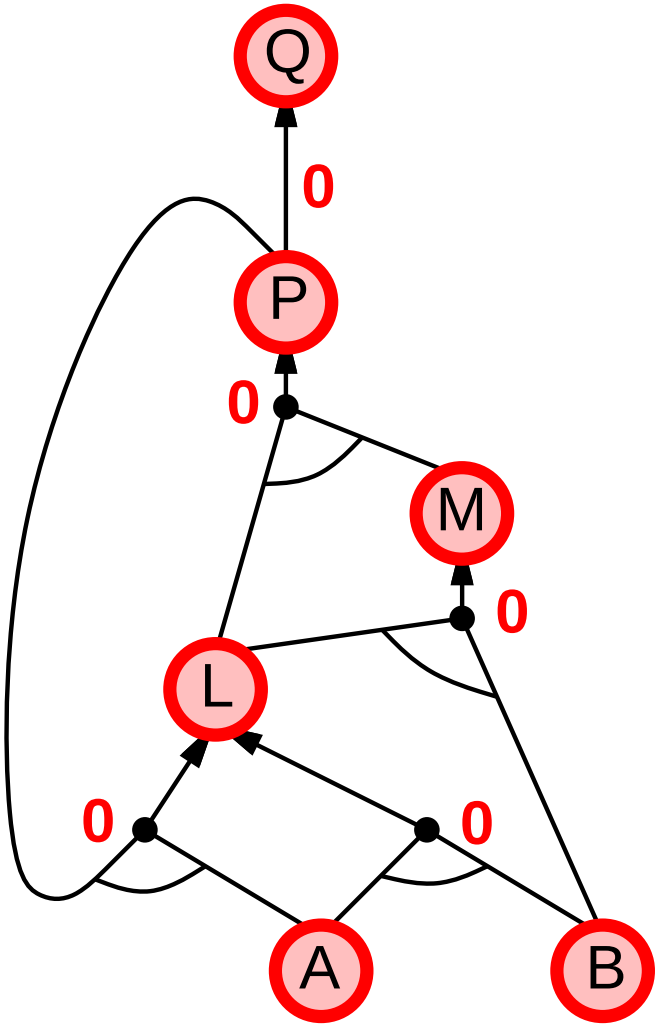
Esempio di Forward chaining



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

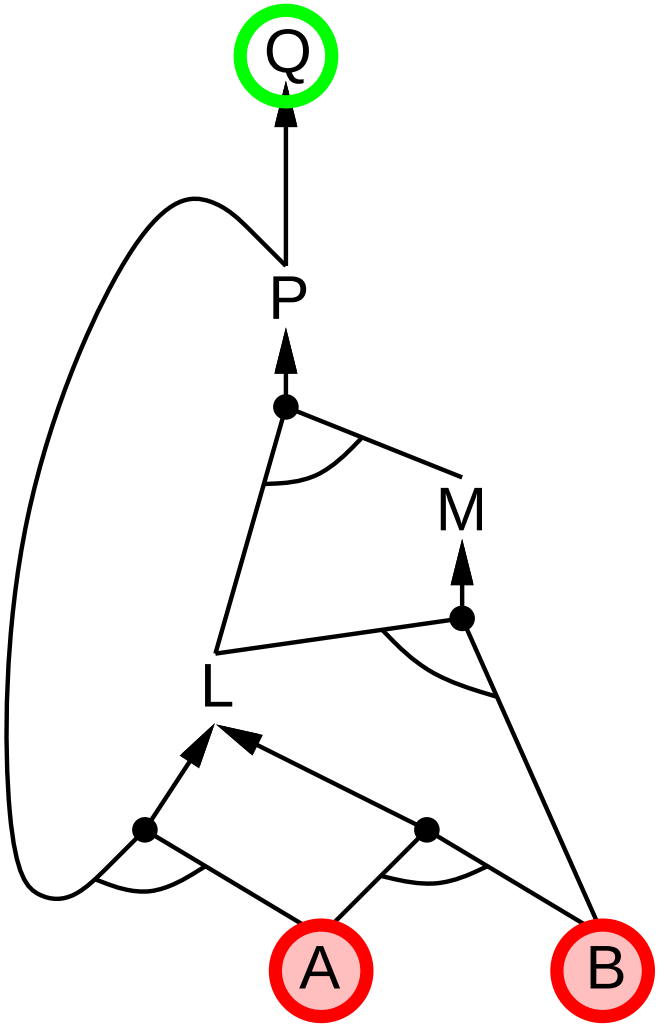
Evitare cicli: controlla se un nuovo sottogoal è già presente nella pila dei goal

Evitare di ripetere del lavoro: controlla se un nuovo sottogoal

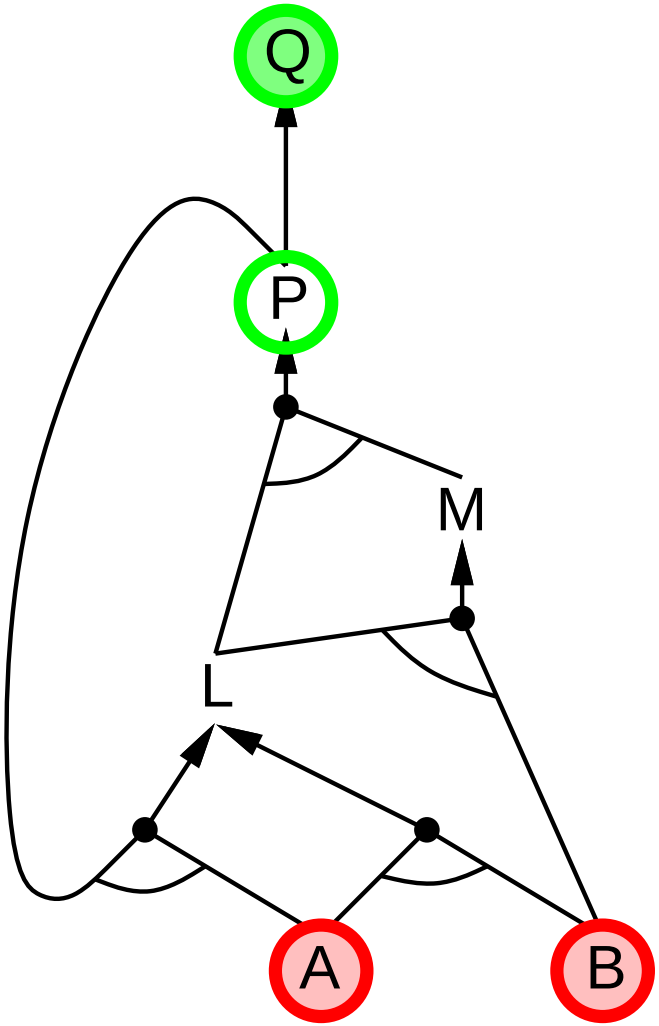
1) è stato già provato vero, o

2) è già fallito

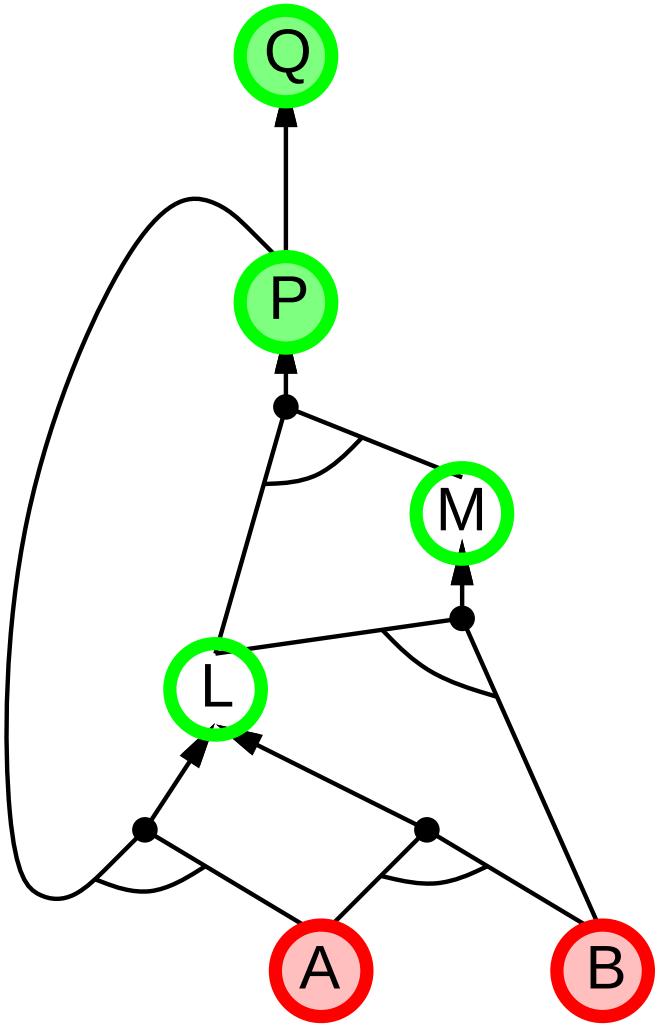
Esempio di Backward chaining



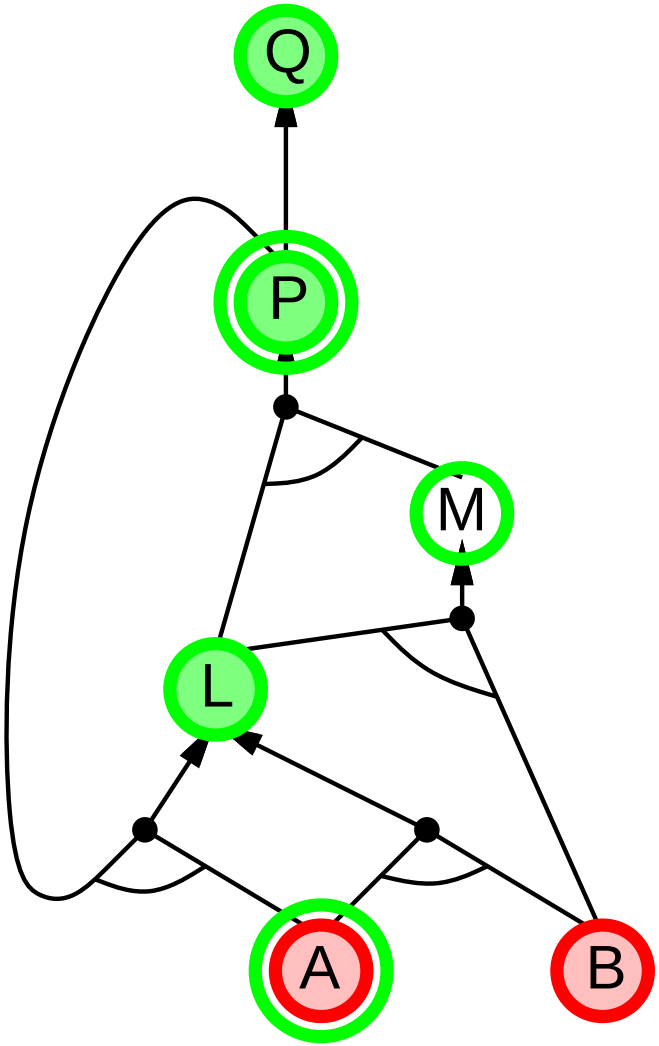
Esempio di Backward chaining



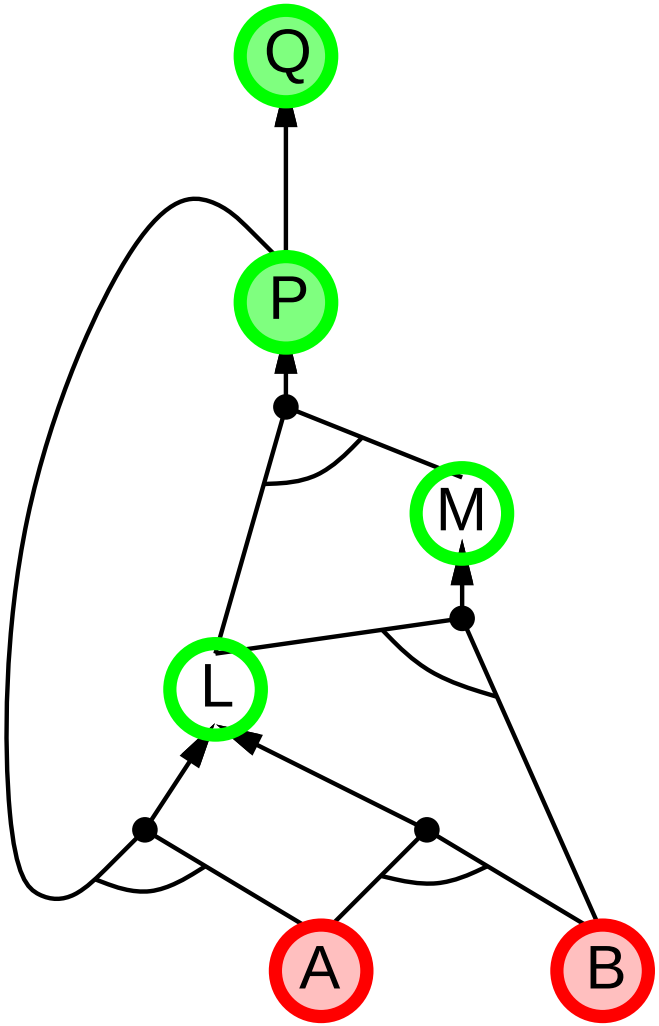
Esempio di Backward chaining



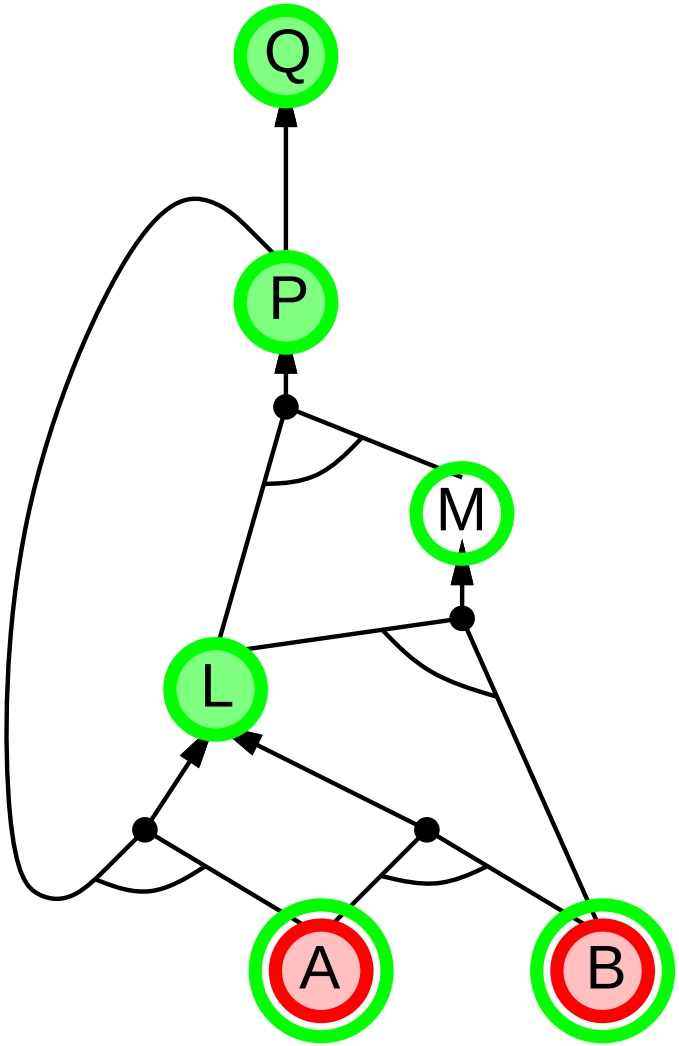
Esempio di Backward chaining



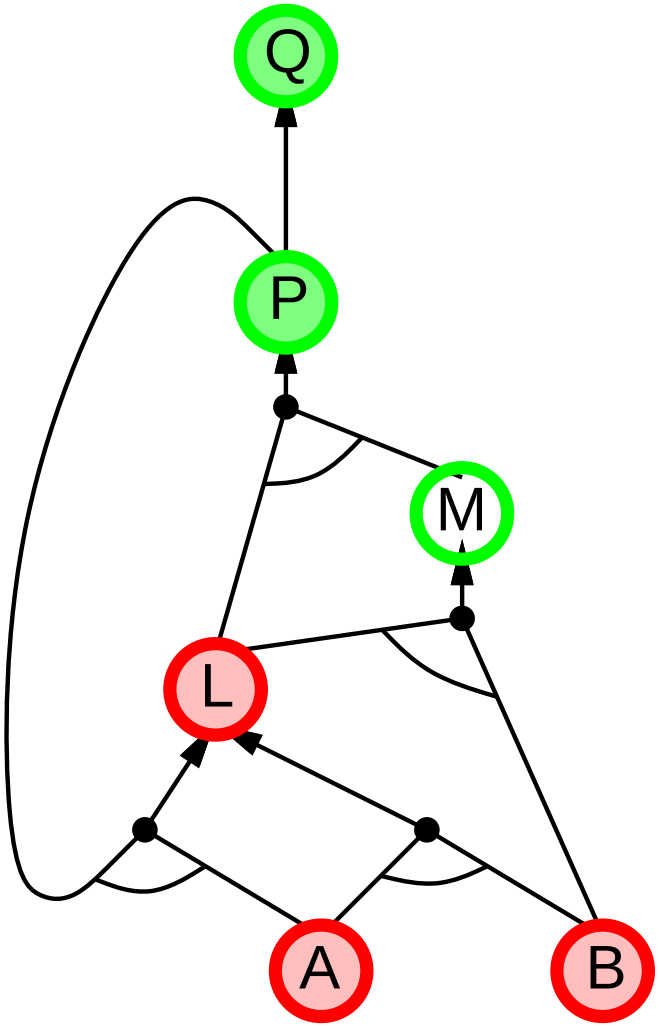
Esempio di Backward chaining



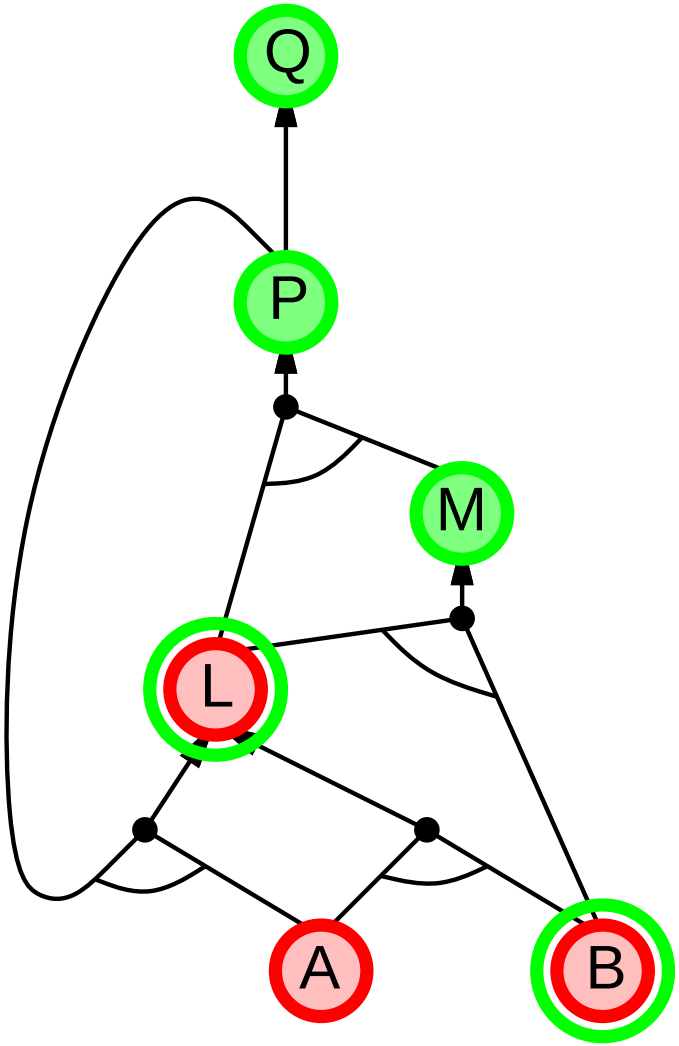
Esempio di Backward chaining



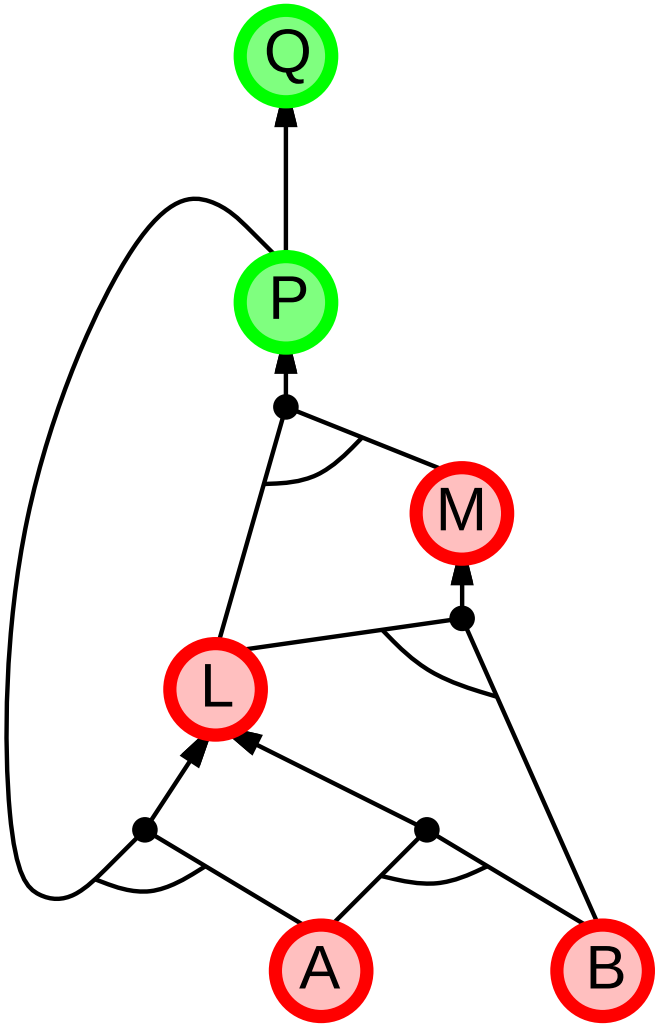
Esempio di Backward chaining



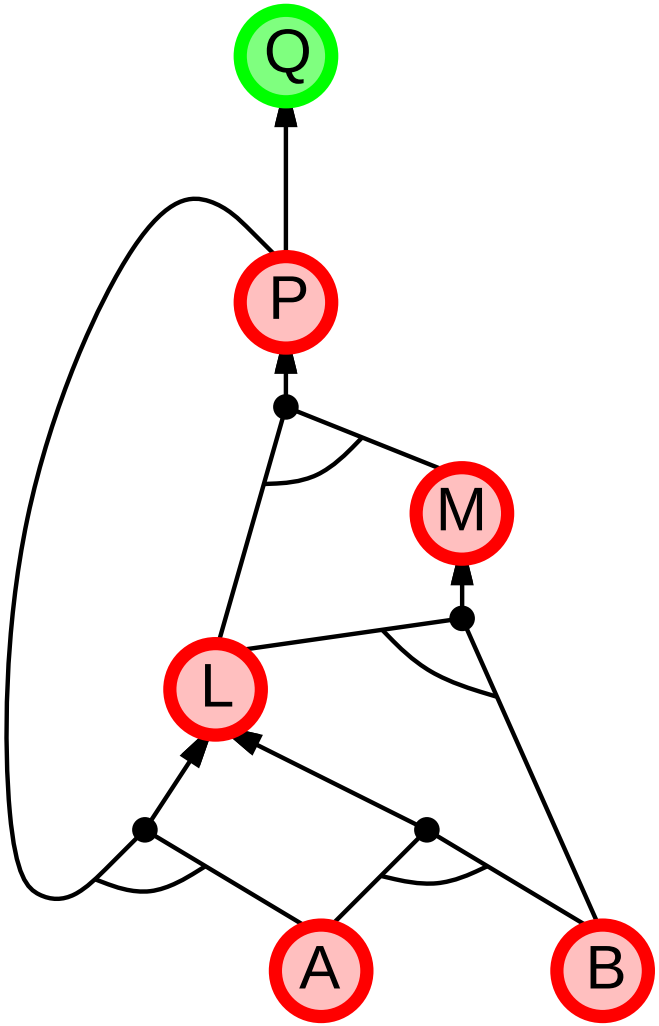
Esempio di Backward chaining



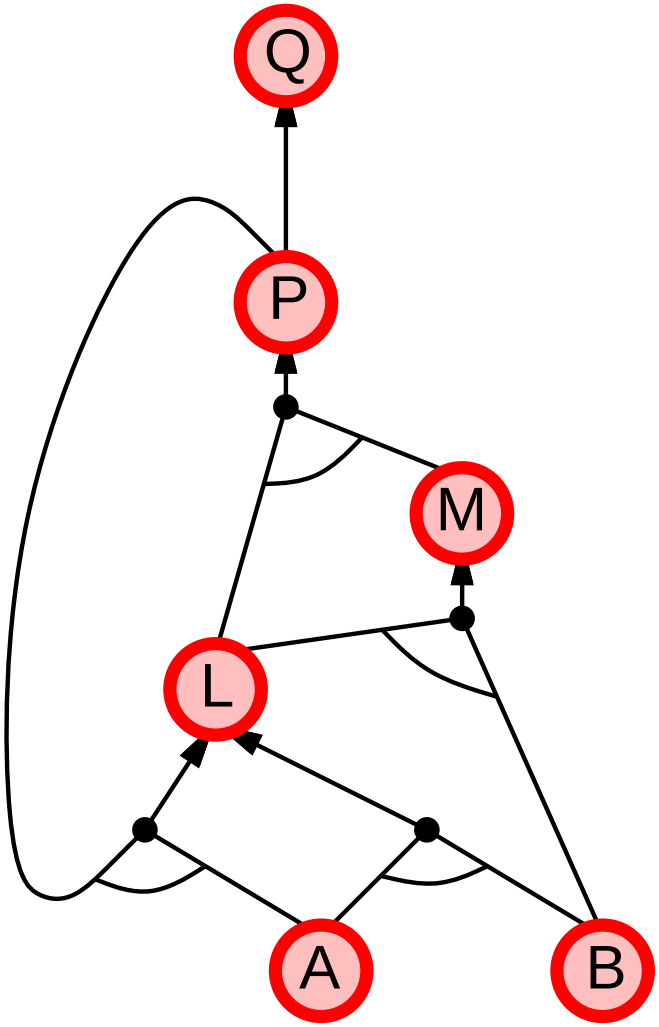
Esempio di Backward chaining



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*

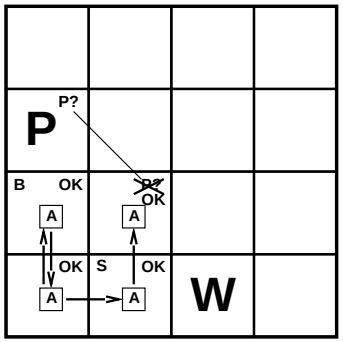
P.e., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

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

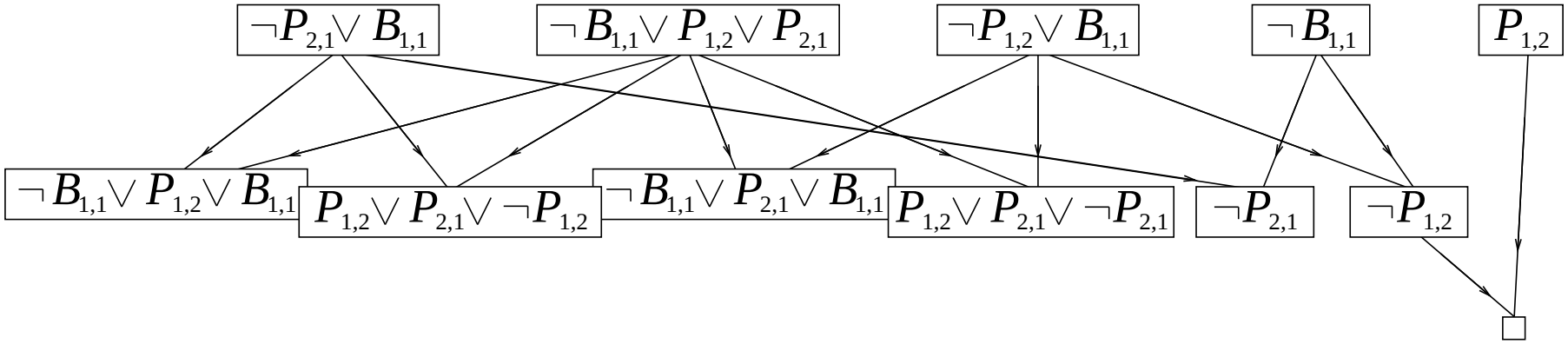
Prova per contraddizione, cioè, mostra che $KB \wedge \neg\alpha$ è insoddisfacibile

function PL-RESOLUTION(KB, α) **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*
  $clauses \leftarrow clauses \cup new$

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 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