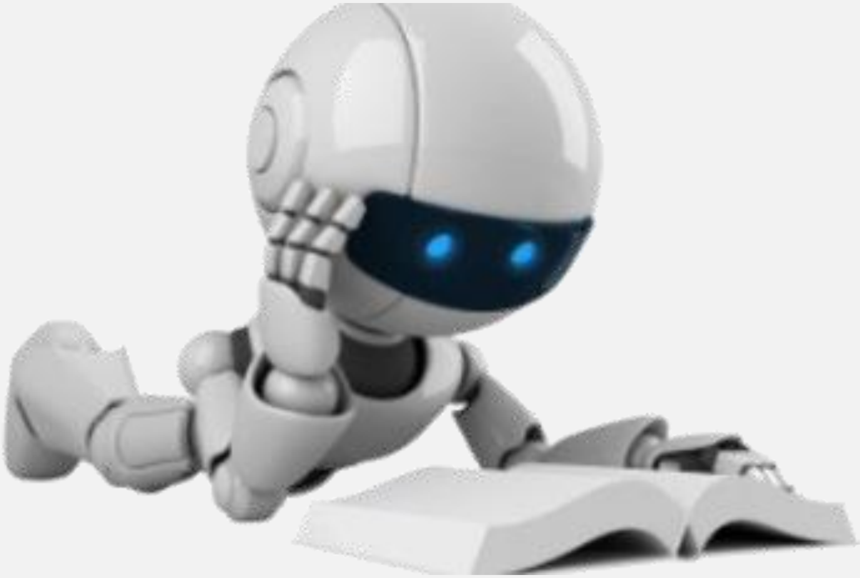
Agentes Lógicos



TÉCNICO LISBOA

Sumário

- Agentes baseados em conhecimento
- O mundo do Wumpus
- Lógica em geral
- Lógica proposicional (Booleana)
  - Equivalência, validade, satisfação
- Lógica de 1ª ordem
  - Representação em lógica de 1ª ordem
  - Inferência em lógica de 1ª ordem

Relação com o livro

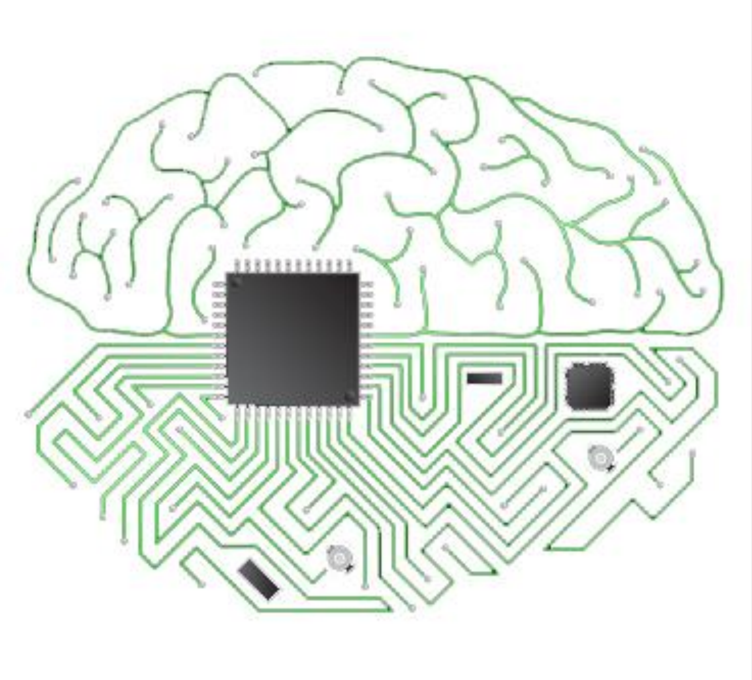
- Capítulo 7 (7.1, 7.2, 7.3, 7.5)
- Capítulo 8 (8.1, 8.2. 8.3)
- Capítulo 9 (9.2, 9.3, 9.4)

Outras secções assume-se que os alunos já deram na cadeira de Lógica para a Programação.

As áreas de IA

Robótica

Representação do Conhecimento e Raciocínio



Visão

Procura

Agentes

Planeamento de acções

Língua Natural

Aprendizagem

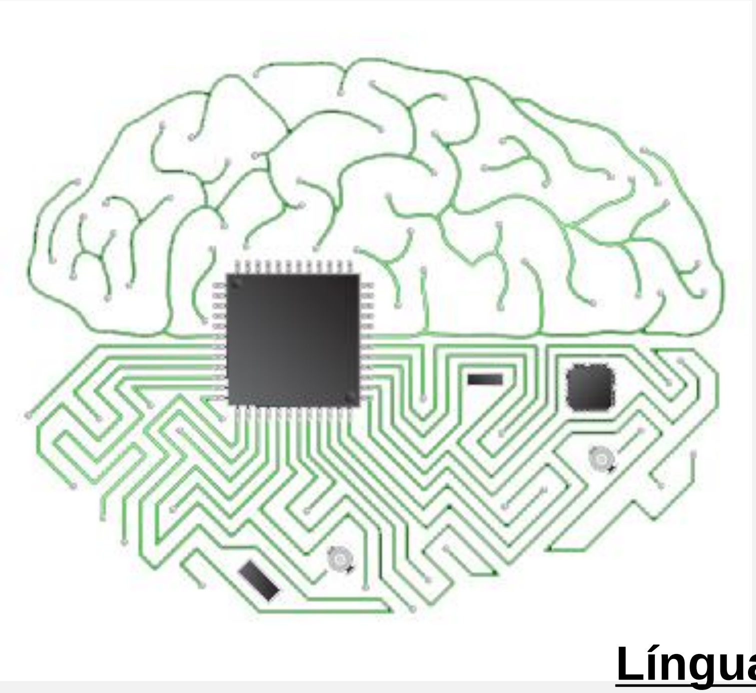
Jogos

As áreas de IA

Robótica

Representação do Conhecimento e Raciocínio

Visão



Procura

Agentes

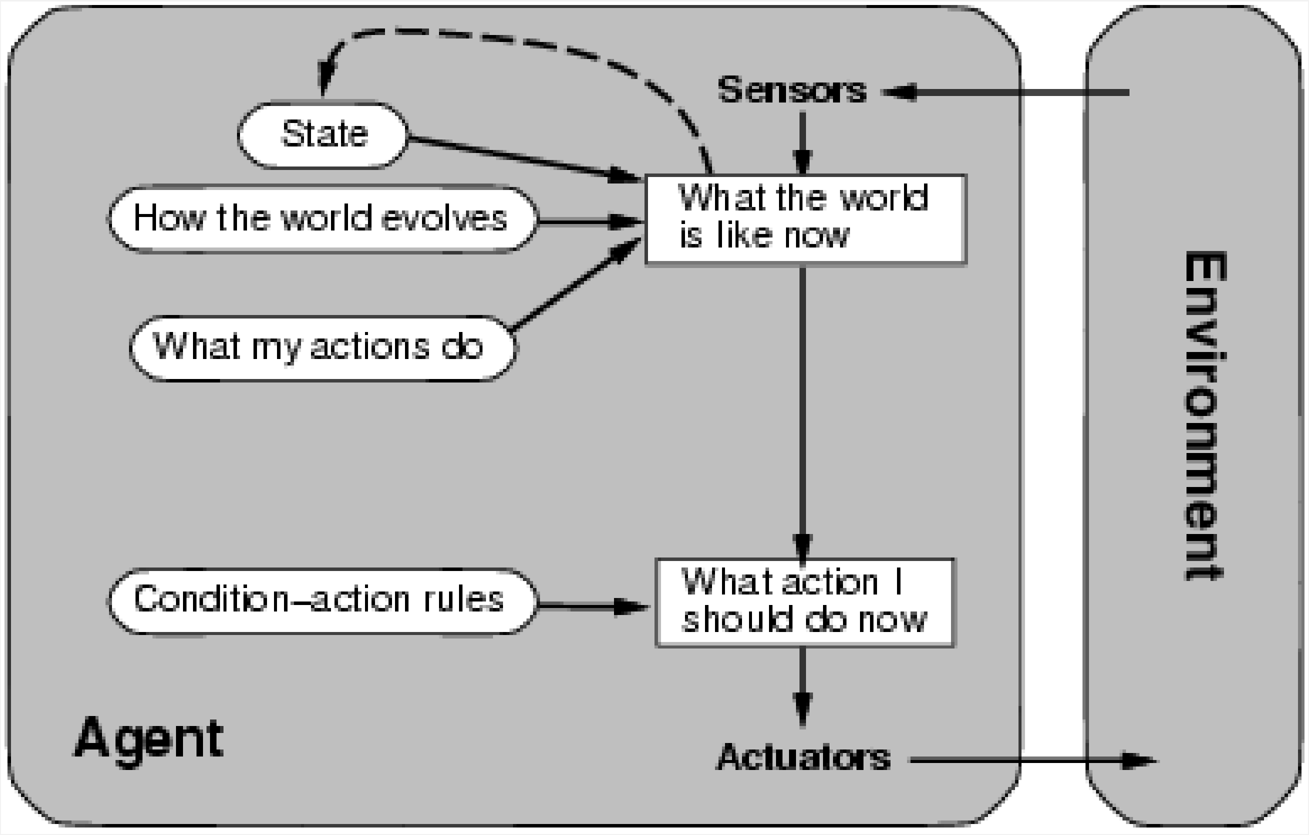
Planeamento de acções

Língua Natural

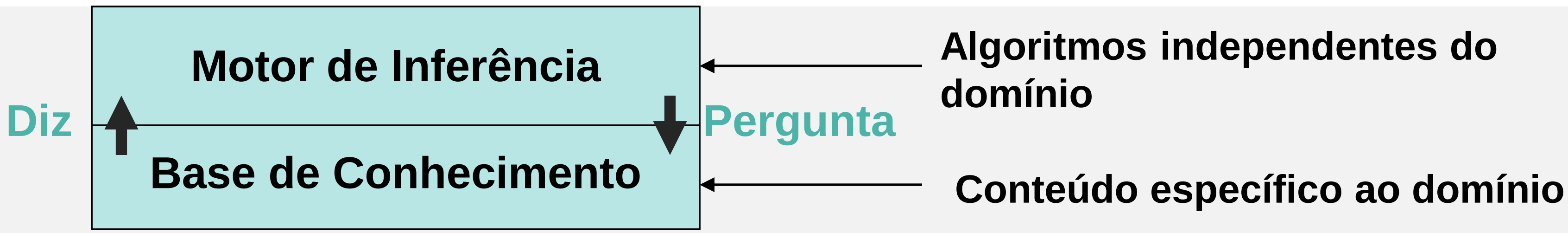
Aprendizagem

Jogos

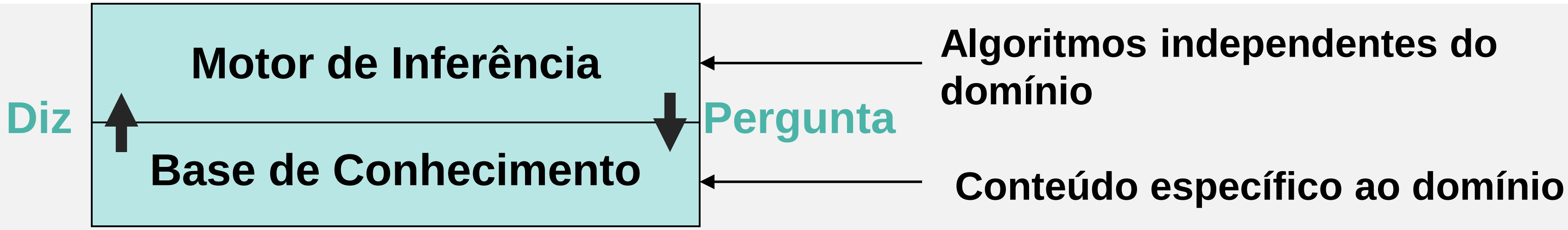
Agentes baseados no conhecimento



Bases de conhecimento



- Base de Conhecimento (BC ou KB, do Inglês *Knowledge Base*) = conjunto de frases numa linguagem formal



Abordagem declarativa para construir um agente (ou outro sistema):

- BC `Diz` o que é necessário o agente saber
- O agente `Pergunta` a si próprio o que fazer – respostas são obtidas a partir da BC
- Agentes podem ser considerados ao nível de conhecimento
  i.e., importa o que eles sabem, independentemente de como são implementados
- Ou ao nível da implementação
  – i.e., estruturas de dados na BC e algoritmos que a manipulam

Agente baseado em conhecimento

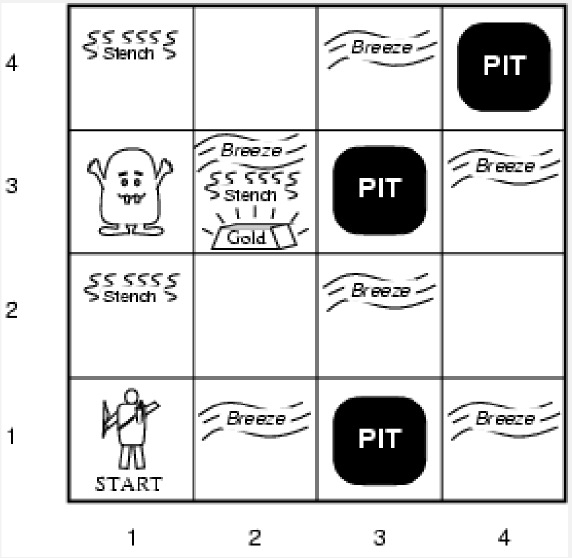
Função AgenteBC (*percepcao)* **devolve** *accao*
estático: *BC*, uma base de conhecimento
t, um contador, inicialmente a 0, que indica o tempo

`Diz`(*BC*,cria-precepcao-frase*(percepcao,t)*)
accao ← `Pergunta`(*BC*,cria-accao-pergunta*(t)*)
`Diz`(BC,cria-accao-frase*(accao,t)*)
t ← *t* +1
devolve *accao*

- O agente deve ser capaz de:
  - Representar estados, acções, etc.
  - Incorporar novas percepções
  - Actualizar representação interna do mundo
  - Deduzir propriedades implícitas no mundo
  - Deduzir acções mais apropriadas

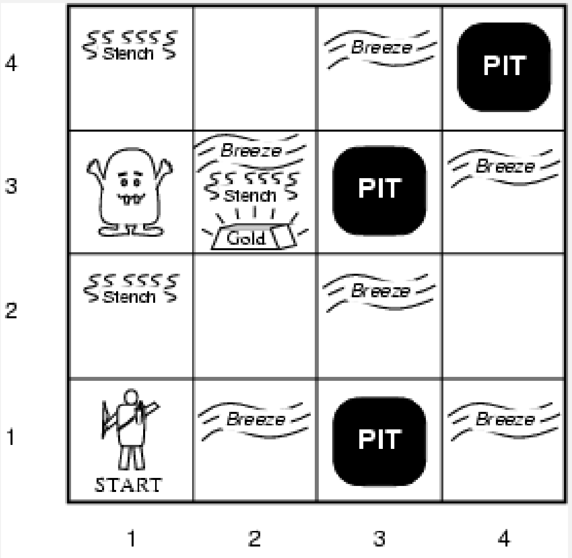
Mundo do Wumpus:

- Ambiente
  - Posições adjacentes a *pit* cheiram bem
  - Posições adjacentes ao *wumpus* cheiram mal
  - Brilho *sse* ouro está na mesma posição
  - Disparar gasta a única seta
  - Disparar mata o *wumpus* se estamos de frente para ele
  - Agarrar apanha o ouro que está na mesma posição
  - Largar liberta o ouro na posição
  - Agente morre na posição com *wumpus* (vivo) ou com *pit*



Mundo do Wumpus: descrição

- Sensores: CheirarMal, CheirarBem, Brilhar, Chocar, Gritar
- Actuadores: virar esquerda, virar direita, frente, agarrar, largar, disparar
- Medida de desempenho
  - ouro +1000, morte -1000
  - -1 por movimento, -10 por usar a seta
- Ambiente
  - Posições adjacentes a *pit* cheiram bem
  - Posições adjacentes ao *wumpus* cheiram mal
  - Brilho *sse* ouro está na mesma posição
  - Disparar gasta a única seta
  - Disparar mata o *wumpus* se estamos de frente para ele (e quando ele morre dá um grito)
  - Agarrar apanha o ouro que está na mesma posição
  - Largar liberta o ouro na posição
  - Agente morre na posição com *wumpus* (vivo) ou com *pit*



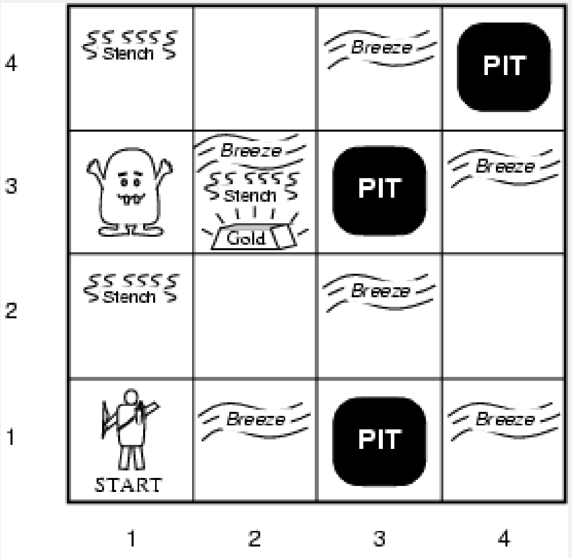
Caracterização do mundo do Wumpus

- Completamente observável? Não – só percepções locais
- Determinístico? Sim – acção resultante especificada com exactidão
- Episódico? Não – sequencial a nível das acções
- Estático? Sim – *Wumpus* e *Pits* não se movem
- Discreto? Sim

Perceções

- Sensores:

  [CheirarMal, CheirarBem, Brilhar, Chocar, Gritar]



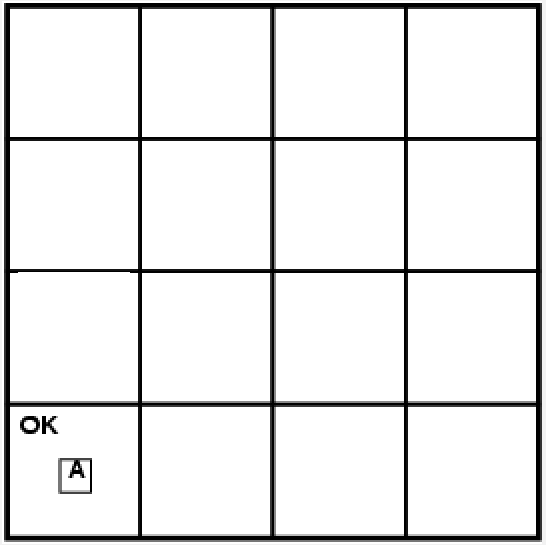
São dados ao agente como uma lista de 5 símbolos

Ex:

P1.2 - [CheiraMal, None, None, None, None]

P2.3 - [CheiraMal,CheiraBem, Brilha, None, None]

Exploração do mundo do wumpus: exemplo de um mundo desconhecido!

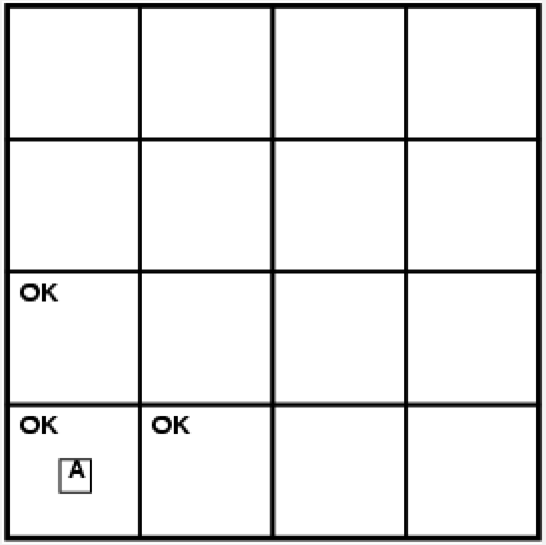


Perceçao:
[none, none, none, none, none]

A = Agente

OK = posição segura

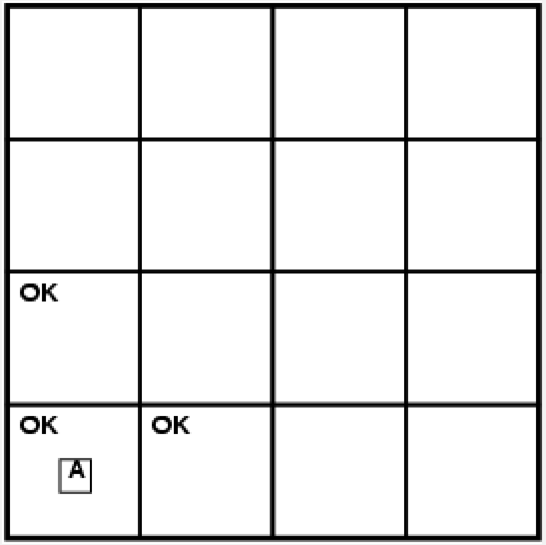
Exploração do mundo do wumpus



A = Agente

OK = posição segura

Exploração do mundo do wumpus

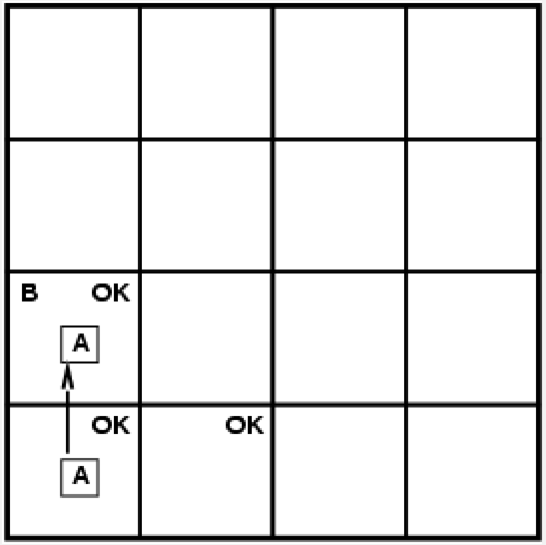


Agente decide andar para a posição 1.2

A = Agente

OK = posição segura

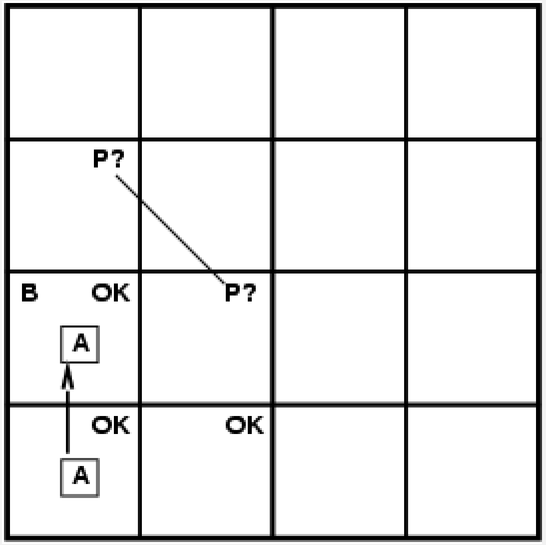
Exploração do mundo do wumpus



B = cheira bem

Perceçao:
[none, cheiraBem, none, none, none]

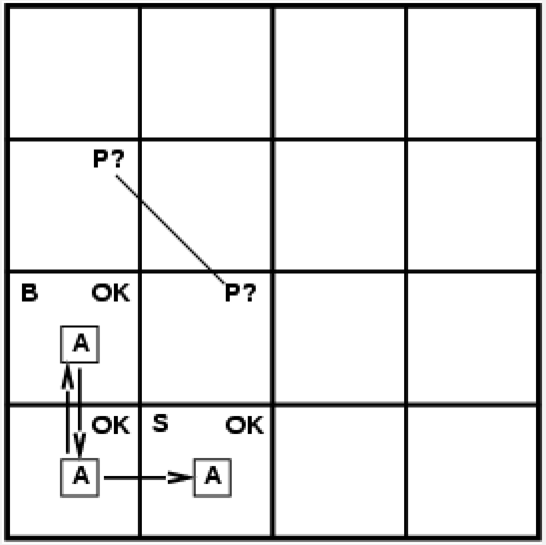
Exploração do mundo do wumpus



P = pit

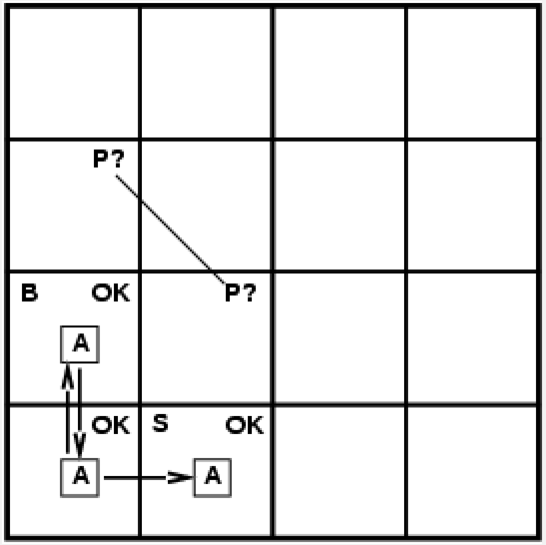
Agente “infere” que existe um buraco no 1.3 ou no 2.2

Exploração do mundo do wumpus



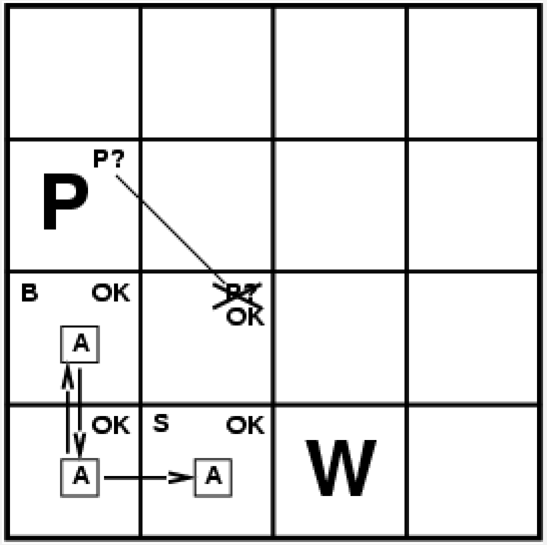
Agente volta para posição 1.1

Exploração do mundo do wumpus



S = cheira mal

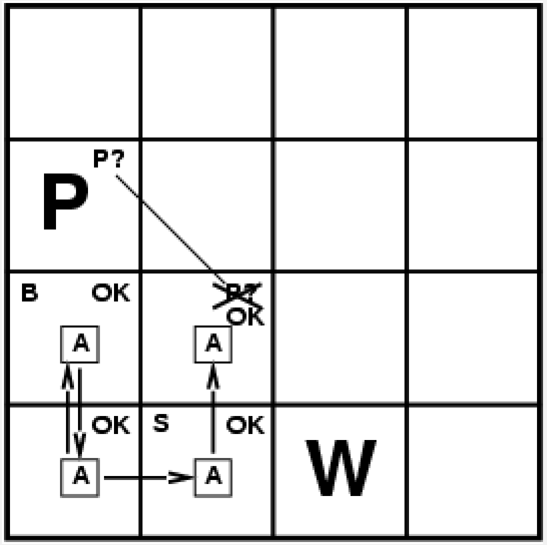
Exploração do mundo do wumpus



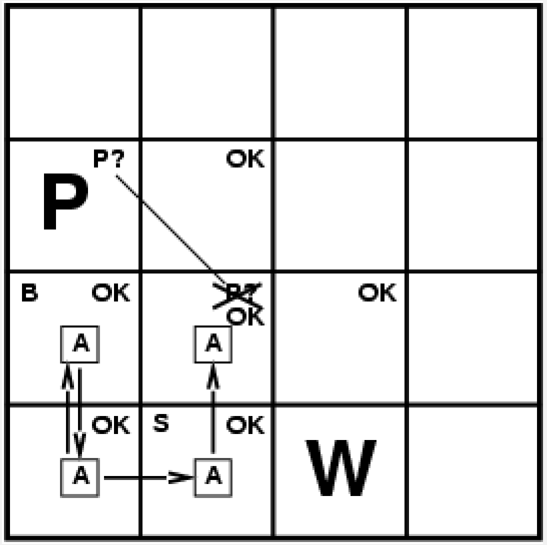
P = pit

W = wumpus

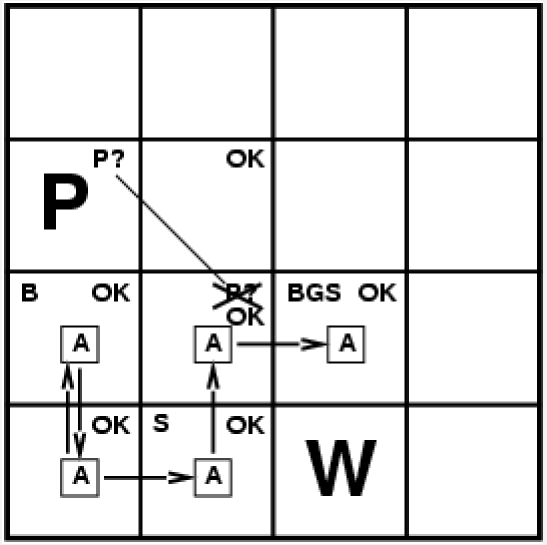
Exploração do mundo do wumpus



Exploração do mundo do wumpus



Exploração do mundo do wumpus



B = cheira bem

G = brilho

S = cheira mal

Lógica em geral

- Lógicas são linguagens formais para representar informação de tal modo que podem ser tiradas conclusões
- Sintaxe define as frases da linguagem
- Semântica define o “significado" das frases;
  - i.e., define se uma frase no mundo é verdadeira ou falsa

- E.g., linguagem aritmética
  - $x+2 \geq y$ é uma frase; $x2+y > \{\}$ não é uma frase
  - $x+2 \geq y$ é verdadeiro sse o número $x+2$ não é inferior ao número y
  - $x+2 \geq y$ é verdadeiro num mundo em que $x = 7$ e $y = 1$
  - $x+2 \geq y$ é falso num mundo em que $x = 0$ e $y = 6$

Consequência Lógica

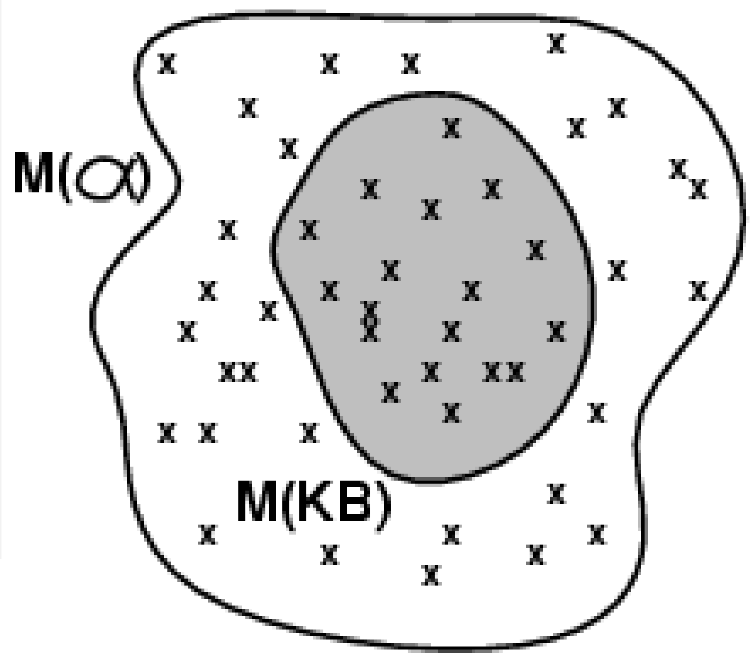
- **Consequência lógica (ou consequência semântica)** significa que uma coisa **resulta de outra:**

$$BC \models \alpha$$

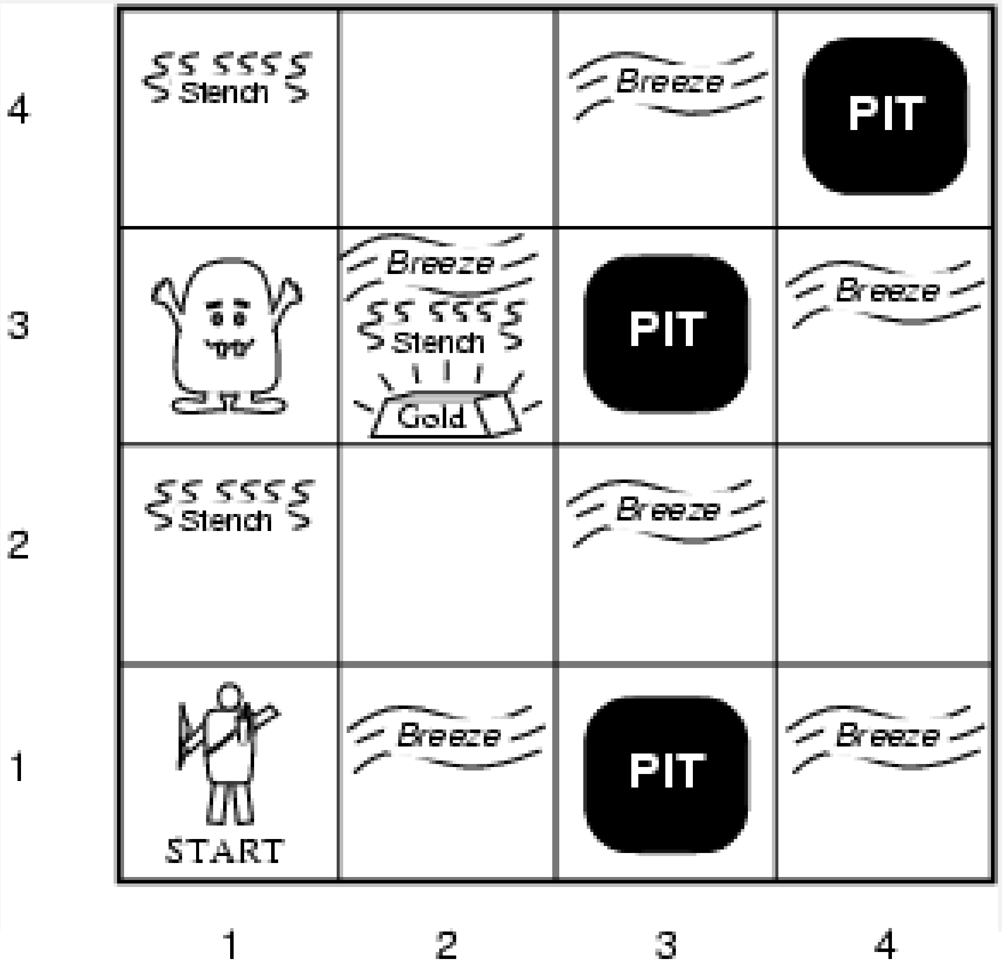
- A frase α é uma consequência lógica da base de conhecimento BC se e só se "α é verdadeiro em todos os mundos em que BC é verdadeiro"
  - E.g., a BC que contém "o Sporting venceu" e "o Benfica venceu" tem como consequência lógica "o Sporting venceu ou o Benfica venceu"
  - E.g., 4 = x+y é consequência lógica de x+y = 4
  - Consequência lógica é uma relação entre frases (i.e., sintaxe) que está baseada na semântica

Modelos

- Em lógica tipicamente pensamos em termos de modelos, que são mundos formalmente estruturados e em relação aos quais se pode determinar se são verdadeiros ou falsos
- Dizemos que *m* é modelo de uma frase α se α é verdadeiro em *m*
- *M(α)* é o conjunto de todos os modelos de α
- Então BC $\models$ α sse *M(BC)* $\subseteq$ *M*(α)
  - E.g. *BC* = Sporting venceu e Benfica venceu, α = Sporting venceu



Modelos do wumpus

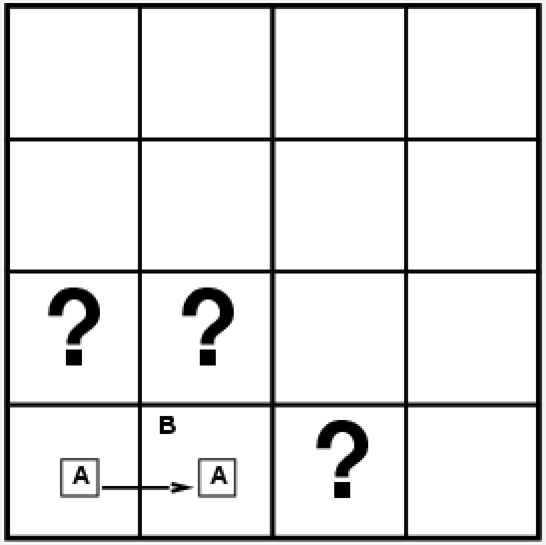


Consequências lógicas no mundo do wumpus

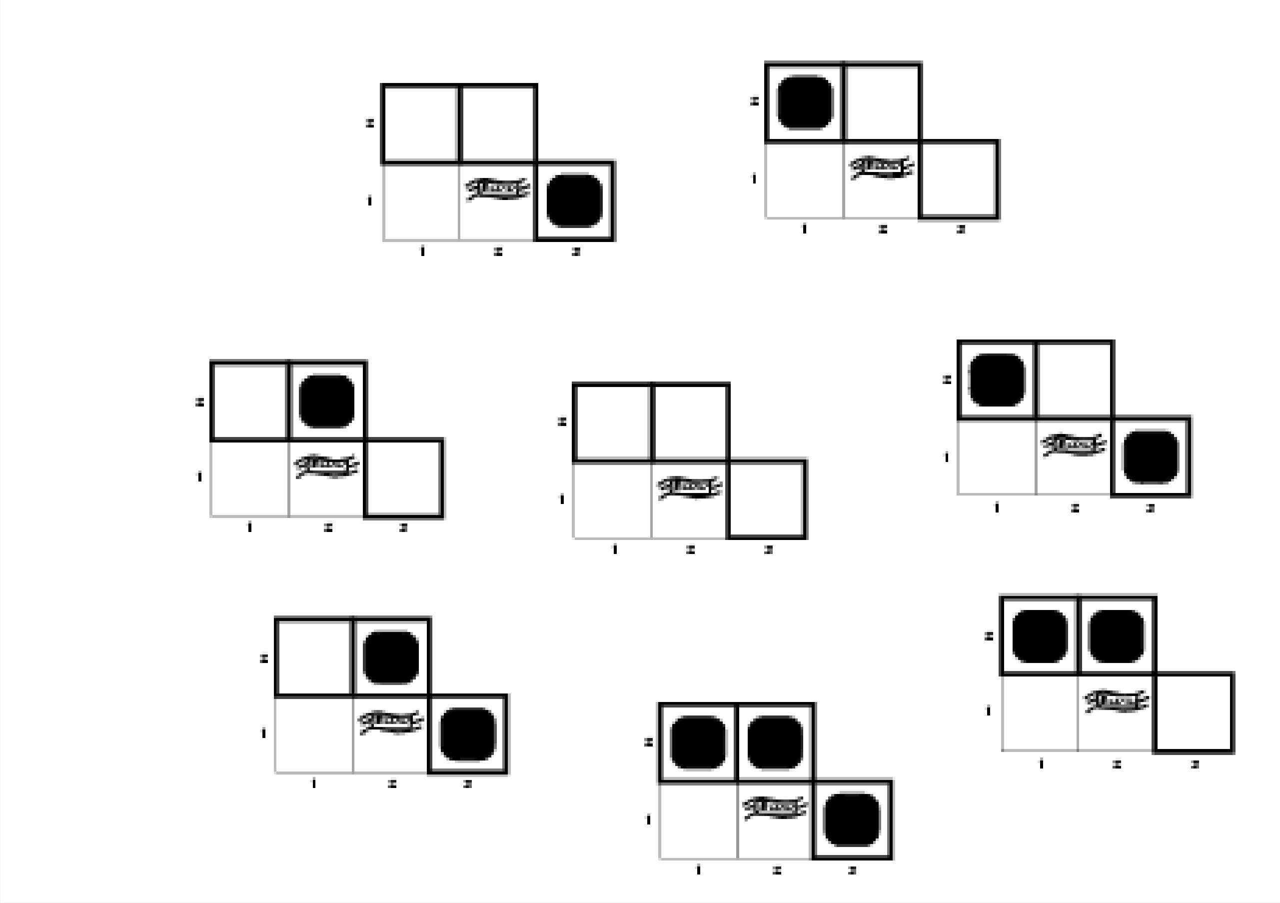
Situação depois de não detectar nada em [1,1], mover-se para a direita, bom cheiro em [2,1]

Considerar modelos possíveis para BC considerando apenas *pits*

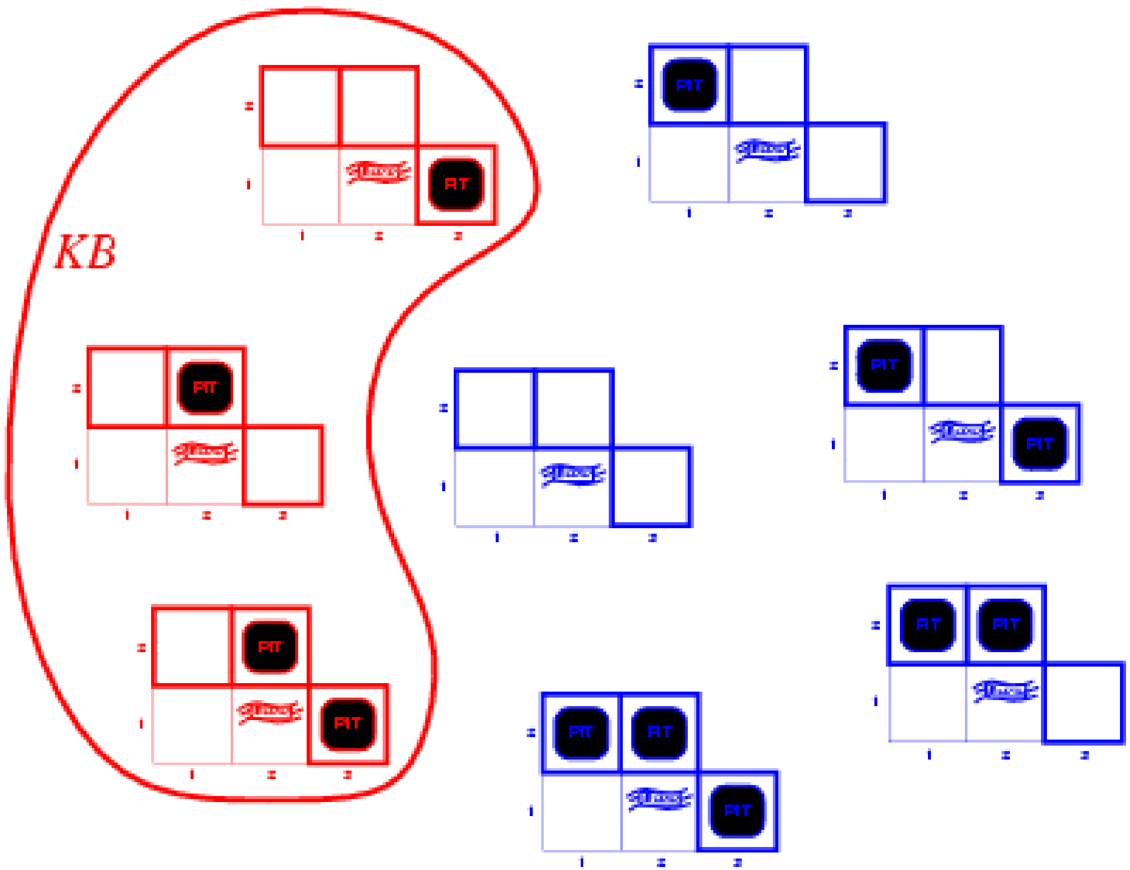
3 escolhas Booleanas $\Rightarrow$ 8 modelos possíveis



Modelos do wumpus

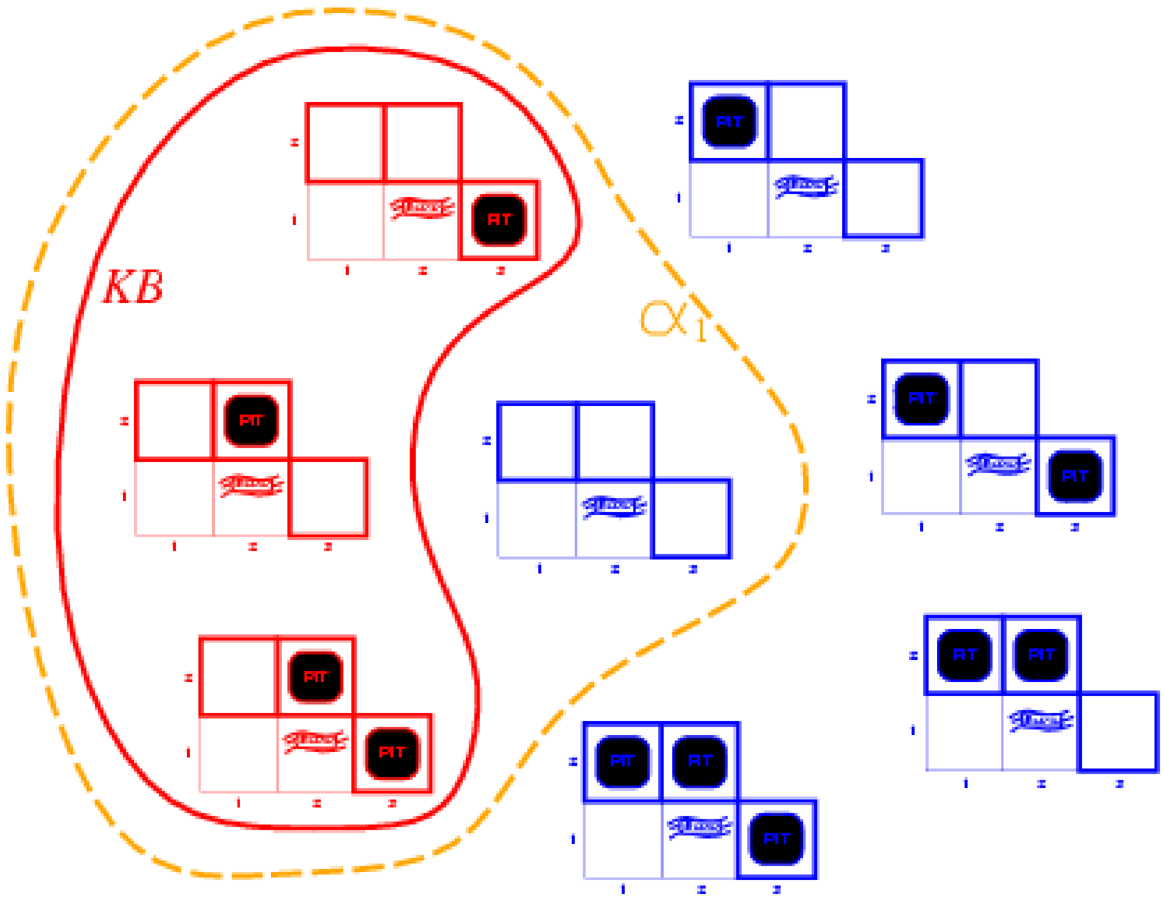


Modelos do wumpus



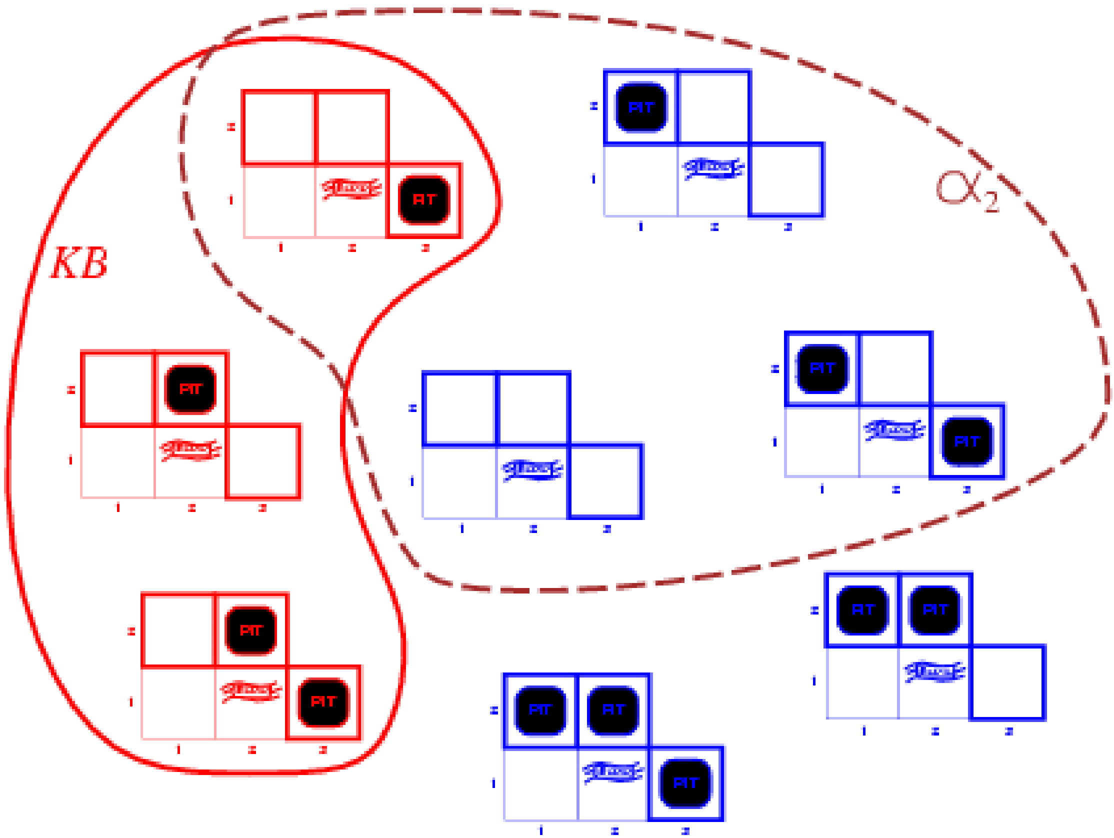
- *BC* = regras do mundo do wumpus + observações

Modelos do wumpus



- *BC* = regras do mundo do wumpus + observações
- α_1 = "não há buraco no [1,2] ", *BC* $\models \alpha_1$, pode ser provado por verificação de modelos

Modelos do wumpus



- *BC* = regras do mundo do wumpus + observações
- α_2 = ”não há buraco no [2,2] ", *BC* $\not\models \alpha_2$, o agente não pode concluir que que não há buraco no [2,2] por verificação de modelos

Inferência

- Inferência ou consequência sintáctica
- *BC* $\vdash_i$ α = frase α pode ser derivada a partir da BC usando o procedimento *i*
- Solidez: *i* é sólido se sempre que *BC* $\vdash_i$ α também é verdade que *BC* $\models$ α
- Completude: *i* é completo se sempre que *BC* $\models$ α também é verdade que *BC* $\vdash_i$ α

- Objectivo: definir uma lógica que é suficientemente expressiva para expressar quase toda a realidade, e para a qual existe um mecanismo de inferência que é sólido e completo.
- Ou seja, o procedimento responderá a qualquer questão cuja resposta seja obtida a partir do que é conhecido pela BC

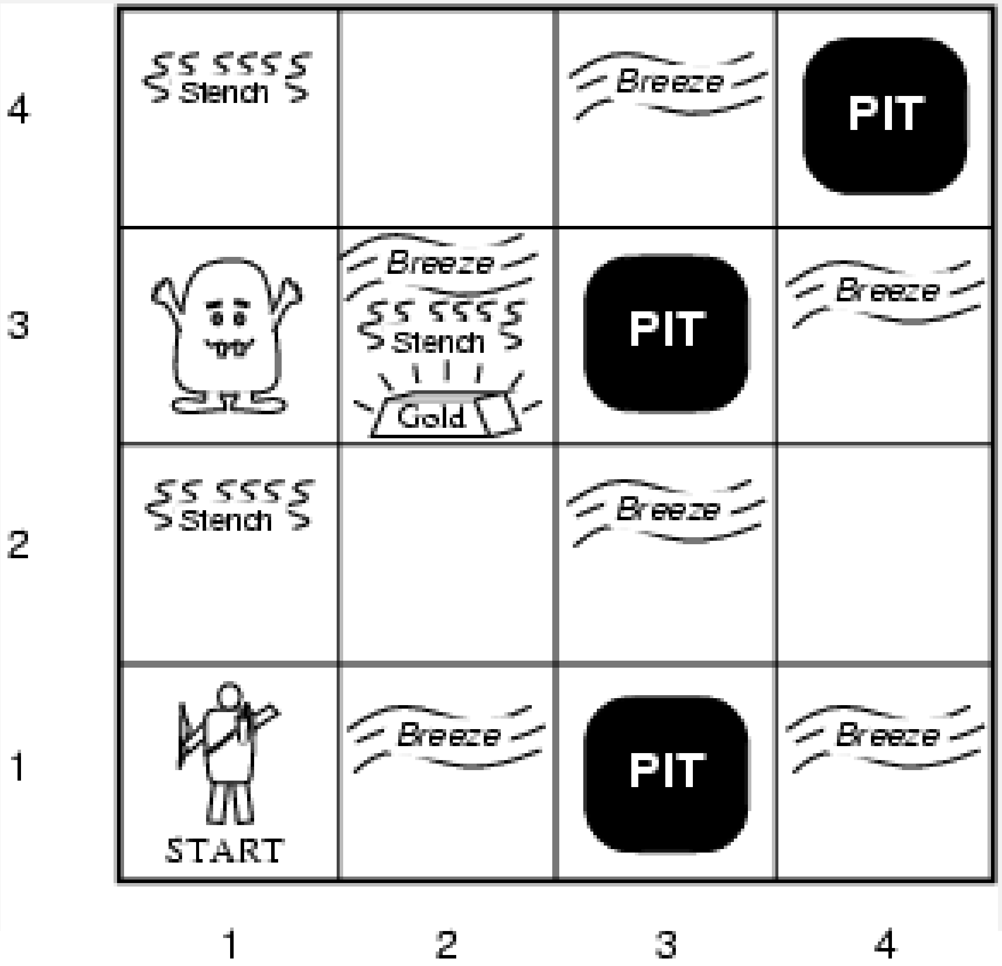
Lógica proposicional: sintaxe

- Lógica proposicional é uma lógica muito simples

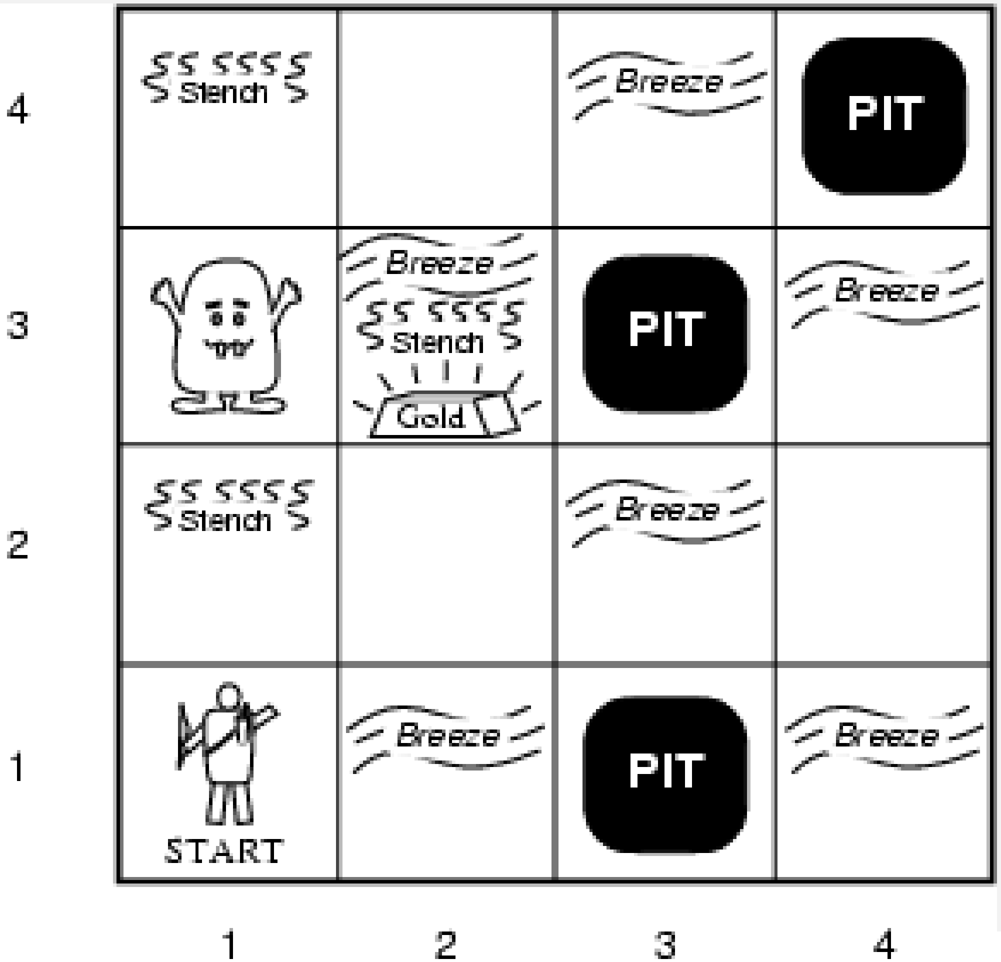
- Os símbolos proposicionais P_1, P_2 etc são frases

  - Se S é uma frase, $\neg$S é uma frase (negação)
  - Se S_1 e S_2 são frases, $S_1 \wedge S_2$ é uma frase (conjunção)
  - Se S_1 e S_2 são frases, $S_1 \vee S_2$ é uma frase (disjunção)
  - Se S_1 e S_2 são frases, $S_1 \Rightarrow S_2$ é uma frase (implicação)
  - Se S_1 e S_2 são frases, $S_1 \Leftrightarrow S_2$ é uma frase (equivalência)

Como representar o mundo do wumpus em lógica proposicional?



Como representar o mundo do wumpus em lógica proposicional?

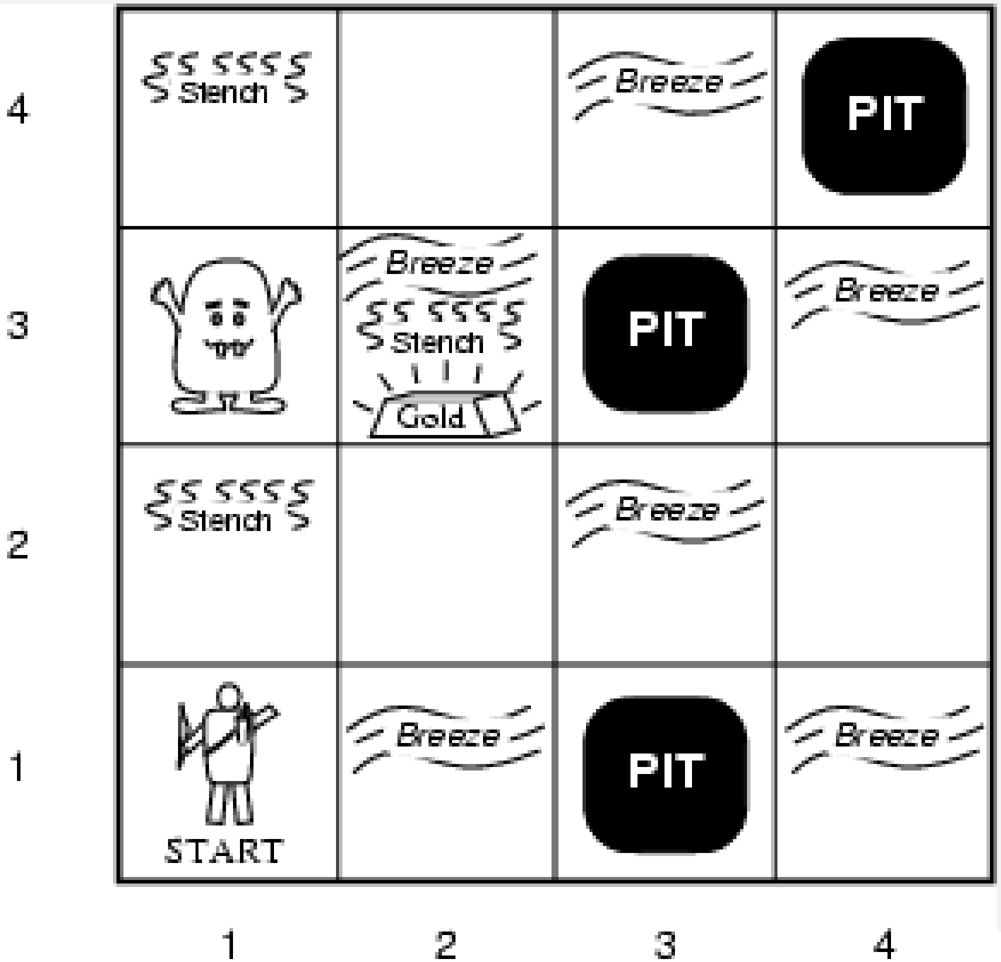


Considerar $P_{i,j}$ verdadeiro se existe um pit em [i, j].

Considerar $B_{i,j}$ verdadeiro se cheira bem em [i, j].

Então?

Como representar o mundo do wumpus em lógica proposicional?



Considerar $P_{i,j}$ verdadeiro se existe um pit em [i, j].

Considerar $B_{i,j}$ verdadeiro se cheira bem em [i, j].

Então?

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

Frases no mundo do Wumpus

Considerar $P_{i,j}$ verdadeiro se existe um pit em [i, j].

Considerar $B_{i,j}$ verdadeiro se cheira bem em [i, j].

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

- "Pits fazem com que cheire bem em posições adjacentes"

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

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

Lógica proposicional: semântica

Cada modelo atribui verdadeiro/falso a cada símbolo proposicional

E.g. $P_{1,2}$ $P_{2,2}$ $P_{3,1}$

falso verdadeiro falso

Com estes símbolos podem ser enumerados 8 modelos possíveis.

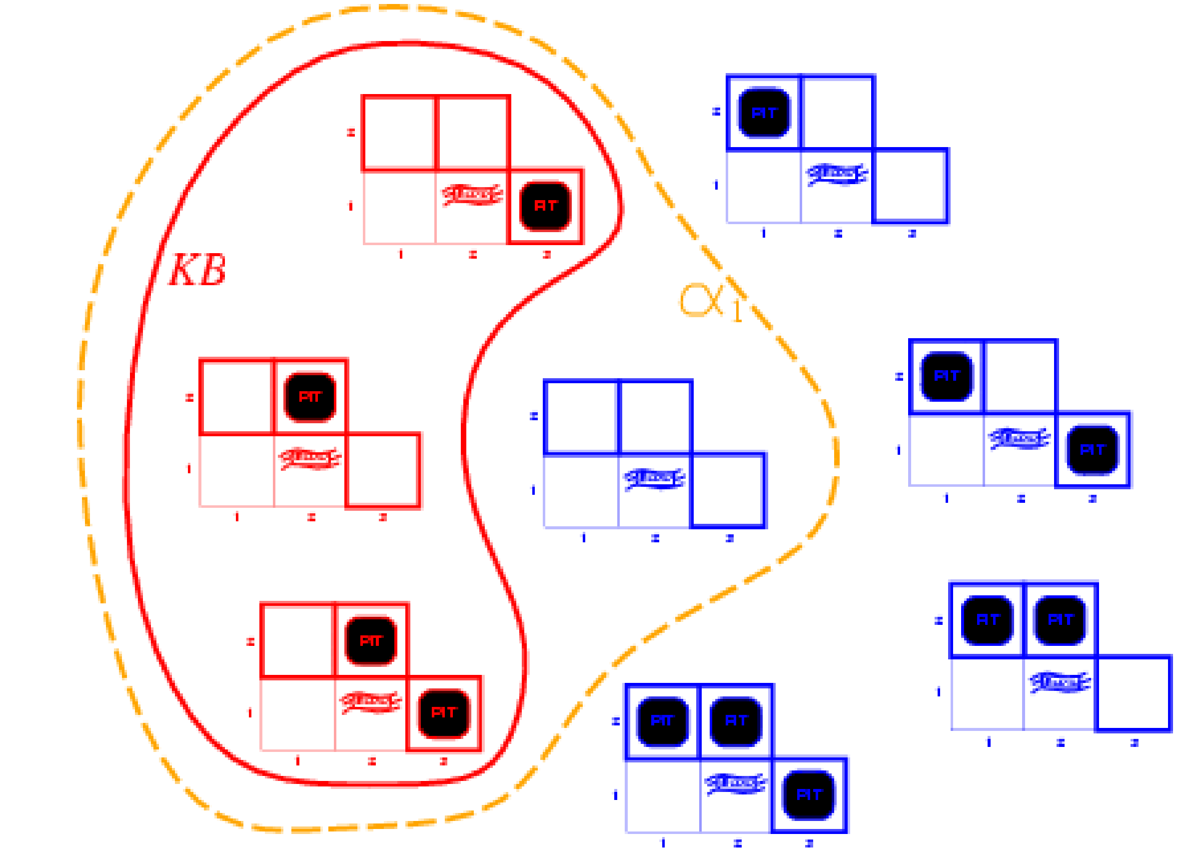
Regras para avaliar se um modelo *m é* verdadeiro ou falso:

$\neg S$	é verdadeiro sse	S é falso
$S_1 \wedge S_2$	é verdadeiro sse	S_1 é verdadeiro e S_2 é verdadeiro
$S_1 \vee S_2$	é verdadeiro sse	S_1 é verdadeiro ou S_2 é verdadeiro
$S_1 \Rightarrow S_2$	é verdadeiro sse	S_1 é falso ou S_2 é verdadeiro
i.e.,	é falso sse	S_1 é verdadeiro e S_2 é falso
$S_1 \Leftrightarrow S_2$	é verdadeiro sse	$S_1 \Rightarrow S_2$ é verdadeiro e $S_2 \Rightarrow S_1$ é verdadeiro

Processo recursivo simples é suficiente para avaliar qualquer frase, e.g.,

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = v \wedge (v \vee f) = v \wedge v = v$$

Tabelas de verdade para inferência



- α1 = "[1,2] é seguro", *BC* ╞ α1, pode ser provado por verificação de modelos
- Significa que α1 é verdade sempre que BC é verdade

Tabelas de verdade para inferência

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
$false$	$false$	$false$	$false$	$false$	$false$	$false$	$false$	$true$
$false$	$false$	$false$	$false$	$false$	$false$	$true$	$false$	$true$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$false$	$true$	$false$	$false$	$false$	$false$	$false$	$false$	$true$
$false$	$true$	$false$	$false$	$false$	$false$	$true$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$false$	$true$	$false$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$false$	$true$	$true$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$true$	$false$	$false$	$false$	$true$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$true$	$true$	$true$	$true$	$true$	$true$	$true$	$false$	$false$

Inferência por enumeração

- Construção implícita de uma tabela de verdade
- Enumeração em profundidade primeiro de todos os modelos é sólida e completa
- Para n símbolos, a complexidade temporal é $O(2^n)$ e a complexidade espacial é $O(n)$

TÉCNICO LISBOA

Métodos de prova

- Métodos de prova podem ser divididos em dois tipos:
  - Verificação de modelos
    - Tabela de verdade (sempre exponencial em *n*)
    - Procura com retrocesso optimizada, e.g., procura DPLL
    - Procura heurística no espaço de modelos (sólida mas incompleta)
      e.g., procura local com aleatorização e heurística do menor número de conflitos
  - Aplicação de regras de inferência
    - Geração (sólida) de novas frases a partir das já existentes
    - Prova = aplicação de uma sequência de regras de inferência; regras de inferência usadas como operadores um algoritmo de procura
    - Tipicamente requer a transformação das frases numa forma normal

Regras de Inferência: exemplos

- Modus Ponens

$$\frac{\alpha \qquad \alpha \Rightarrow \beta}{\beta}$$

Regras de Inferência: exemplos

- Modus Ponens

$$\frac{\alpha \qquad \alpha \Rightarrow \beta}{\beta}$$

- Eliminação do $\wedge$

$$\boxed{\frac{\alpha \wedge \beta}{\alpha}}$$

Regras de Inferência: exemplos

- Modus Ponens

$$\frac{\alpha \qquad \alpha \Rightarrow \beta}{\beta}$$

- Eliminação do $\wedge$

$$\frac{\alpha \wedge \beta}{\alpha}$$

- Eliminação/Introdução da $\Leftrightarrow$

$$\frac{\alpha \Leftrightarrow \beta}{\alpha \Rightarrow \beta \wedge \beta \Rightarrow \alpha}$$

$$\frac{\alpha \Rightarrow \beta \wedge \beta \Rightarrow \alpha}{\alpha \Leftrightarrow \beta}$$

Resolução

- Forma conjuntiva normal (CNF, do Inglês Conjunctive Normal Form)
  conjunção de disjunções de literais, i.e. conjunção de cláusulas
  E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

- **Resolução (para CNF):**

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

onde ℓ_i **e** m_j **são literais com sinais opostos.**

- Resolução é sólida e completa para lógica proposicional

Resolução

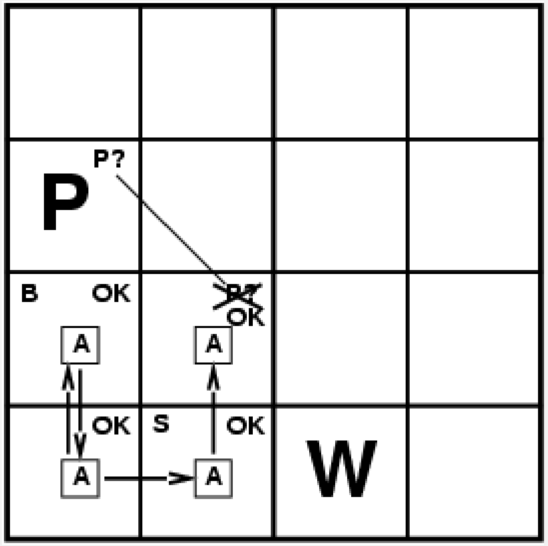
- Resolução com duas cláusulas

- **Resolução (para CNF):**

$$\frac{l_1 \vee l_2, \qquad \neg l_2 \vee l_3}{l_1 \vee l_3}$$

Resolução: exemplo

E.g., $\dfrac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$



Representação em CNF (Conjunctive Normal Form)

- Uma frase é representada por um conjunção de cláusulas
  - Ex: $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1})$
- Uma cláusula está na forma de Horn se for uma disjunção de literais os quais no máximo um é positivo.
  - Ex: $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$ não é
  - $(\neg B_{1,1} \vee P_{2,1})$ é

Conversão para CNF

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

1. Eliminar $\Leftrightarrow$, substituindo $\alpha \Leftrightarrow \beta$ por $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminar $\Rightarrow$, substituindo $\alpha \Rightarrow \beta$ por $\neg\alpha \vee \beta$.
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Mover $\neg$ para dentro usando as regras de De Morgan e da dupla negação:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Aplicar regra da distributividade ($\wedge$ sobre $\vee$):
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Algoritmo de Resolução

- Procedimentos de inferência baseados em resolução usam o principio de prova por contradição;
- Para mostrar que $BC \models \alpha$, então mostramos que $(BC \wedge \neg\alpha)$ é não satisfazível

Algoritmo de Resolução

- Prova por contradição, i.e., provar que $BC \wedge \neg\alpha$ é não satisfazível

Função Resolucao (BC, α) **devolve** *verdadeiro* ou *falso*
    cláusulas $\leftarrow$ conjº de cláusulas na representação CNF de $BC \wedge \neg\alpha$
    novas $\leftarrow$ { }
    ciclo
        para cada C_i, C_j em cláusulas
            resolventes $\leftarrow$ Resolucao(C_i, C_j)
            se cláusula vazia $\in$ *resolventes* **então devolve** *verdadeiro*
            novas $\leftarrow$ *novas* $\cup$ *resolventes*
        se *novas* $\subseteq$ *cláusulas* **então devolve** *falso*
        cláusulas $\leftarrow$ *cláusulas* $\cup$ *novas*

Exemplo de Resolução

- $BC = (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$

Como vimos anteriormente

$(B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1}))$ pode escrever-se

$(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

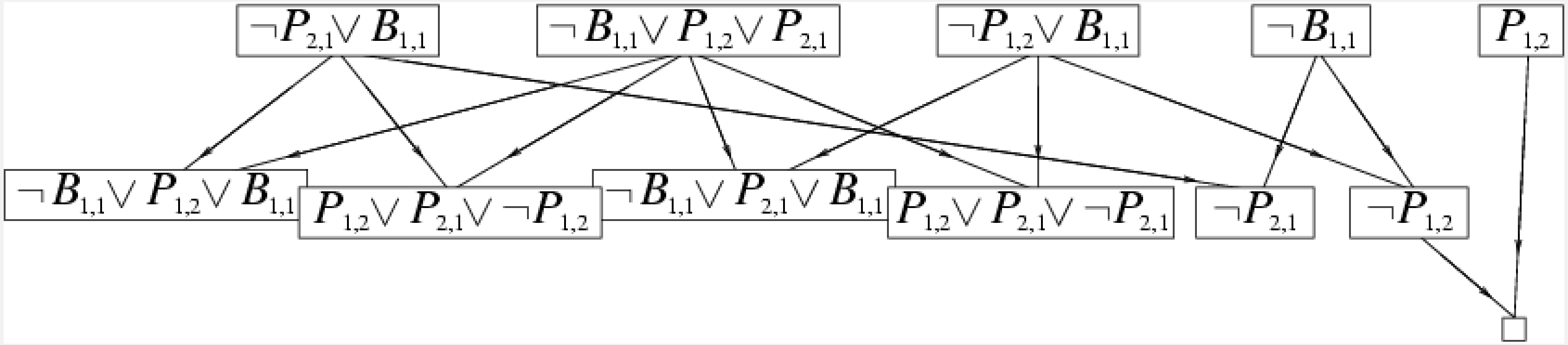
Logo:

$BC = (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1}) \land \neg B_{1,1}$

Exemplo de Resolução

$BC = (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1}) \land \neg B_{1,1}$

$\alpha = \neg P_{1,2}$



- Algoritmo de Resolução é completo
- No entanto não é muito eficiente
- Em muitos problemas do mundo real
  - Podemos usar formas mais simples
  - Algoritmos de Inferência mais eficientes (tempo linear)
    - Encadeamento para a frente
    - Encadeamento para trás

Encadeamento para a frente e para trás

- Forma de Horn

  Caso particular: BC = conjunção de cláusulas de Horn
  - Cláusula de Horn =
    - Símbolo proposicional; ou
    - (conjunção de símbolos) $\Rightarrow$ símbolo
  - E.g., $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$

- Modus Ponens (para Forma de Horn): completo para BCs de Horn

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

- Pode ser usado com encadeamento para a frente ou encadeamento para trás.
- Estes algoritmos são executados em tempo linear

Encadeamento para a frente: algoritmo

Função EF (*BC,q*) **devolve** *verdadeiro* ou *falso*
Variáveis locais: contador, tabela indexada por cláus., inicialmente nº de premissas
inferido, tabela indexada por símbolos, inicialmente a falso
agenda, lista de símbolos, inicialmente com símbolos verdadeiros

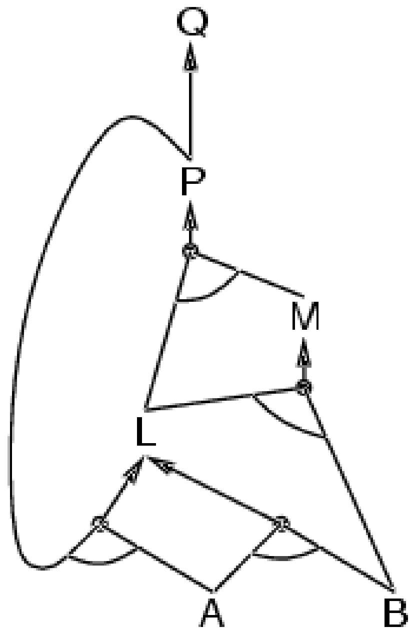
```
    enquanto agenda não está vazia
        P ← Pop(agenda)
        resto ← Rest(símbolos)
        a não ser que inferido[p]
            inferido[p] ← verdadeiro
            por cada cláusula de Horn c onde a premissa p aparece
                decrementar contador[c]
                se contador[c]=0 então
                    se cabeca[c]=q então devolve verdadeiro
                    Push(cabeca[c],agenda)
    devolve falso
```

- Encadeamento para a frente é sólido e completo para BC de Horn

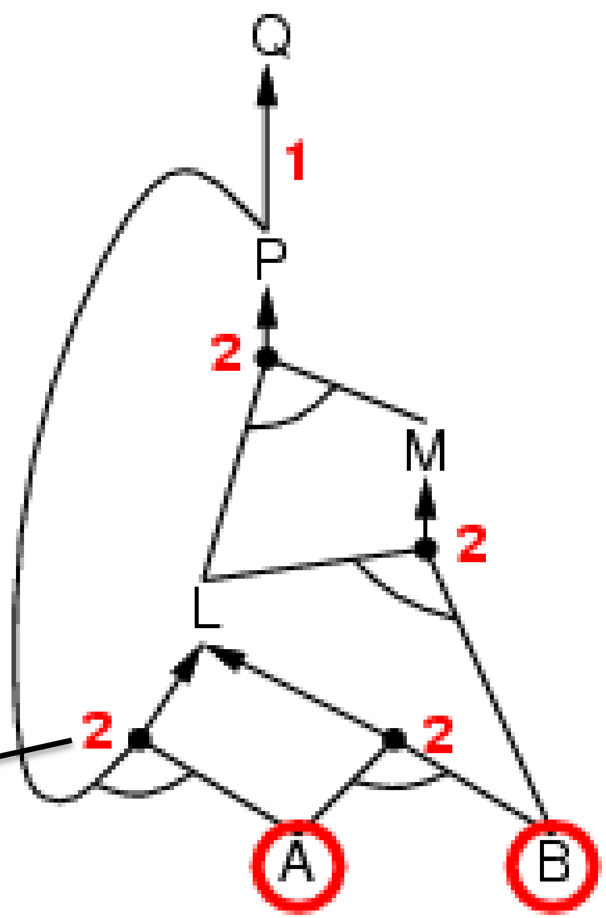
Encadeamento para a frente

- Ideia: usar qualquer regra cujas premissas sejam satisfeitas na BC,
  - Adicionar as conclusões da regra à BC, até que o objectivo seja encontrado

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



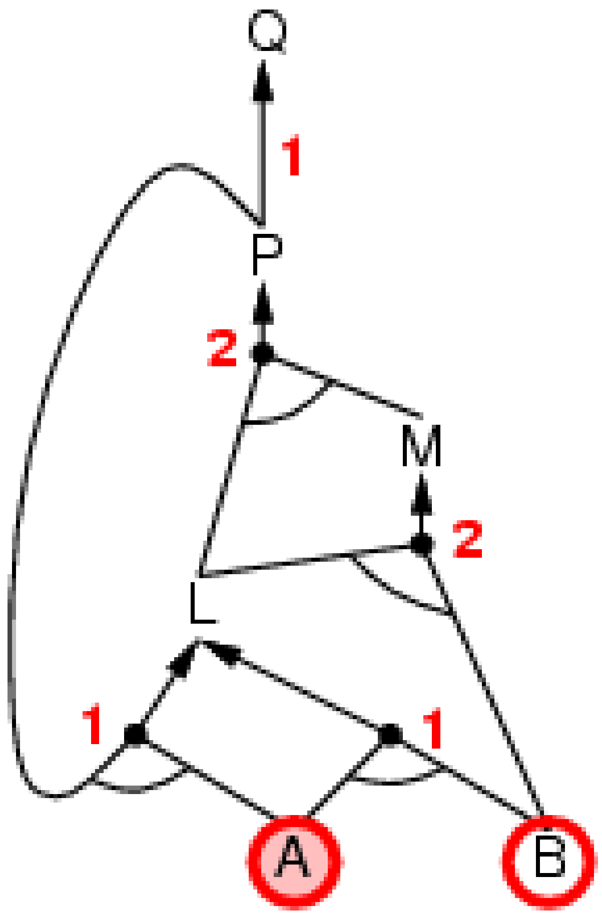
Encadeamento para a frente: exemplo



Quantas premissas de cada implicação são ainda desconhecidas

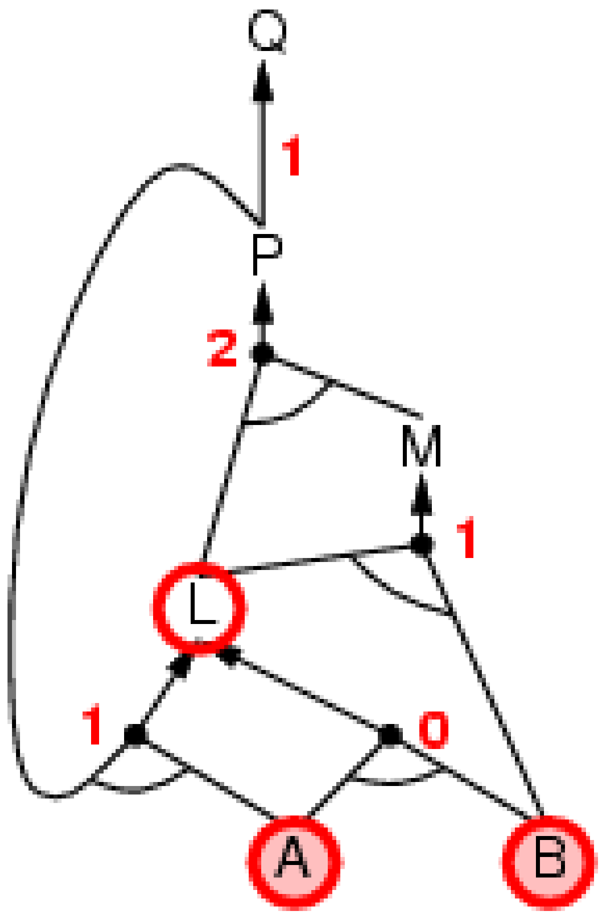
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Encadeamento para a frente: exemplo



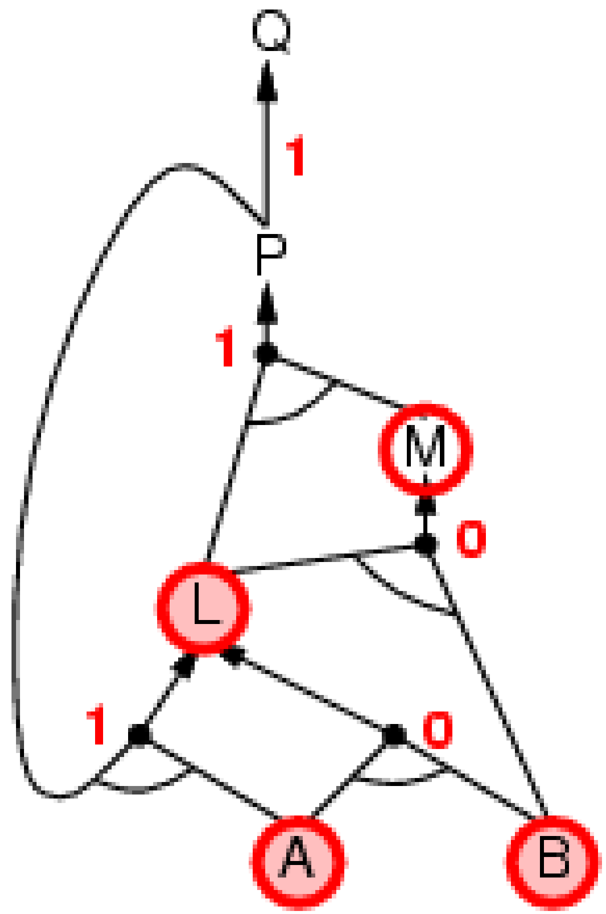
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Encadeamento para a frente: exemplo



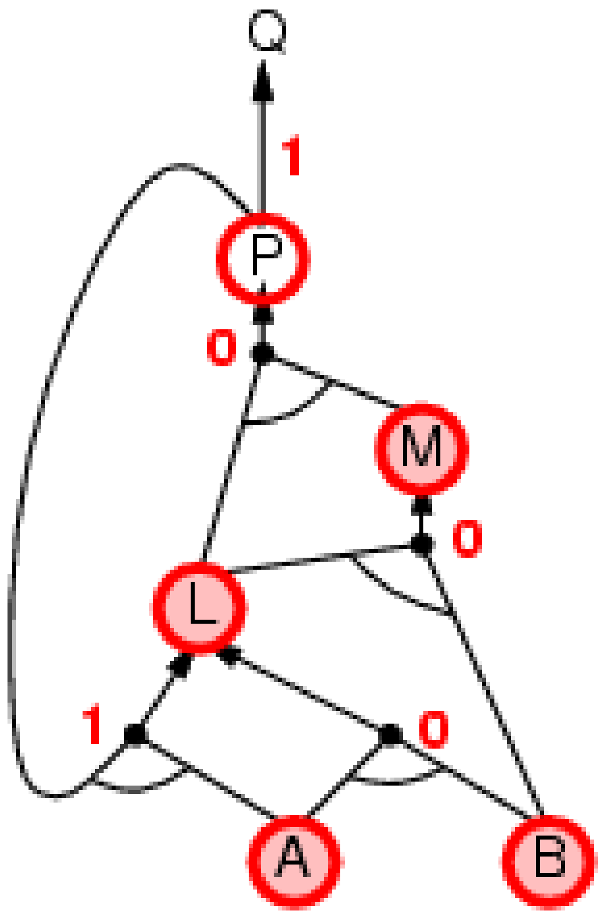
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Encadeamento para a frente: exemplo



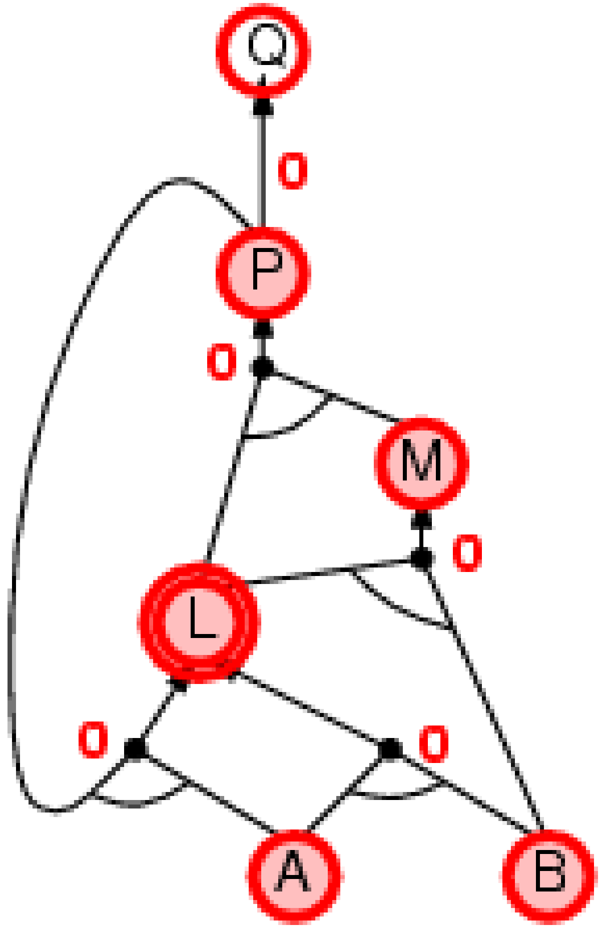
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Encadeamento para a frente: exemplo



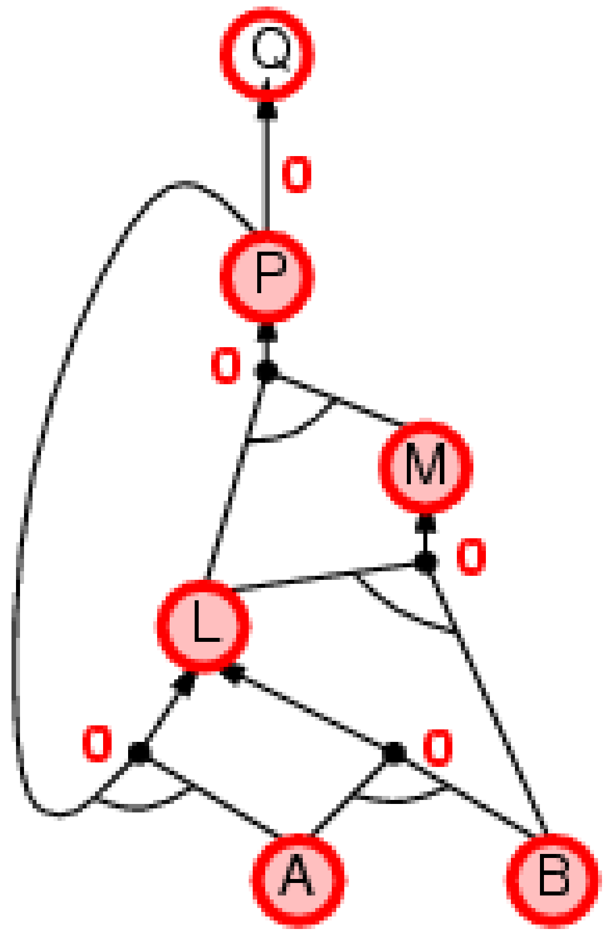
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Encadeamento para a frente: exemplo



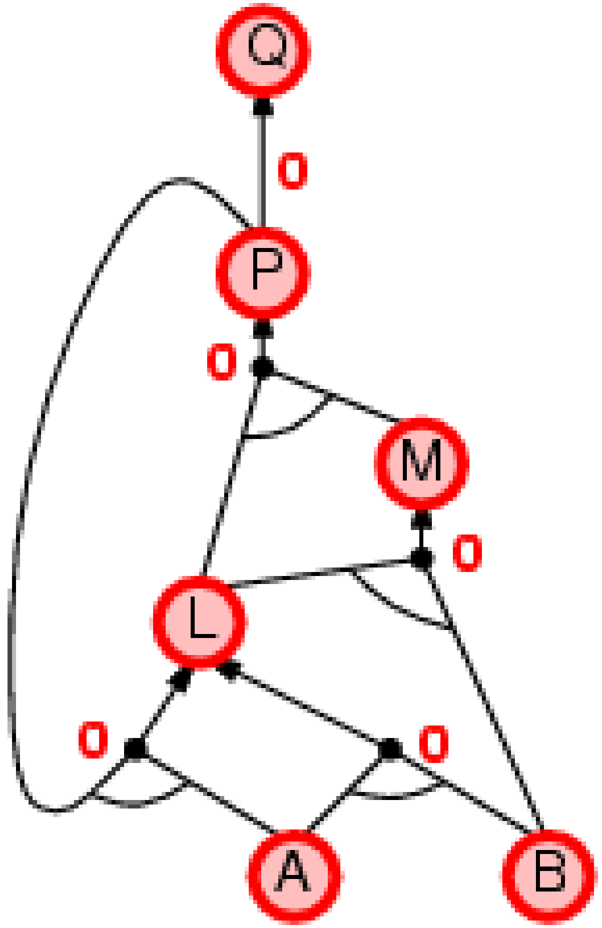
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Encadeamento para a frente: exemplo



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

Encadeamento para a frente: exemplo



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

Encadeamento para trás

Ideia: fazer o encadeamento para trás a partir da questão *q*:

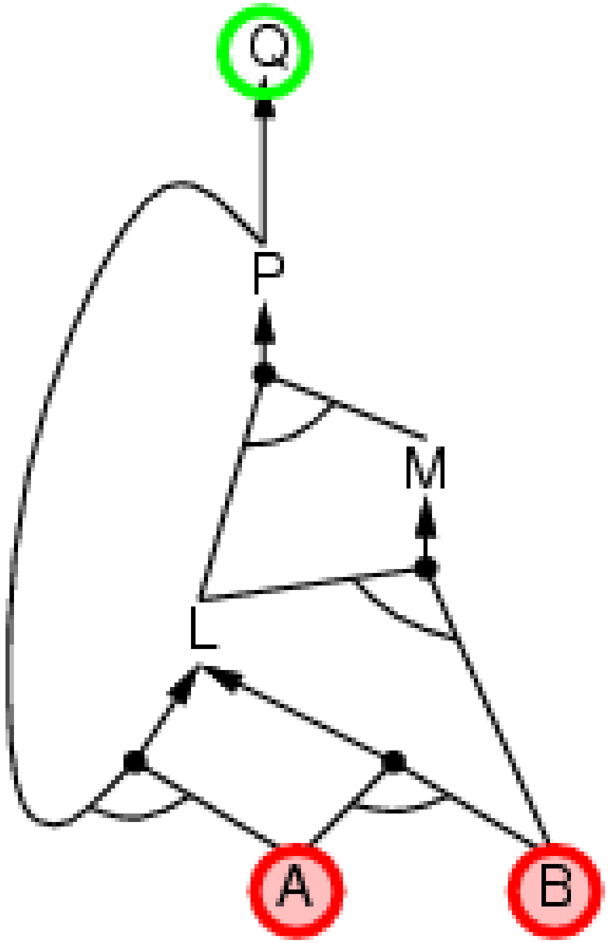
- Para provar *q* pela BC,
  - Verificar se *q* já é conhecido, ou
  - Provar pela BC todas as premissas de alguma regra que permite concluir *q*

Evitar ciclos: verificar se novo sub-objectivo já está no conjunto existente de objectivos

Evitar trabalho repetido: verificar se existe novo sub-objectivo

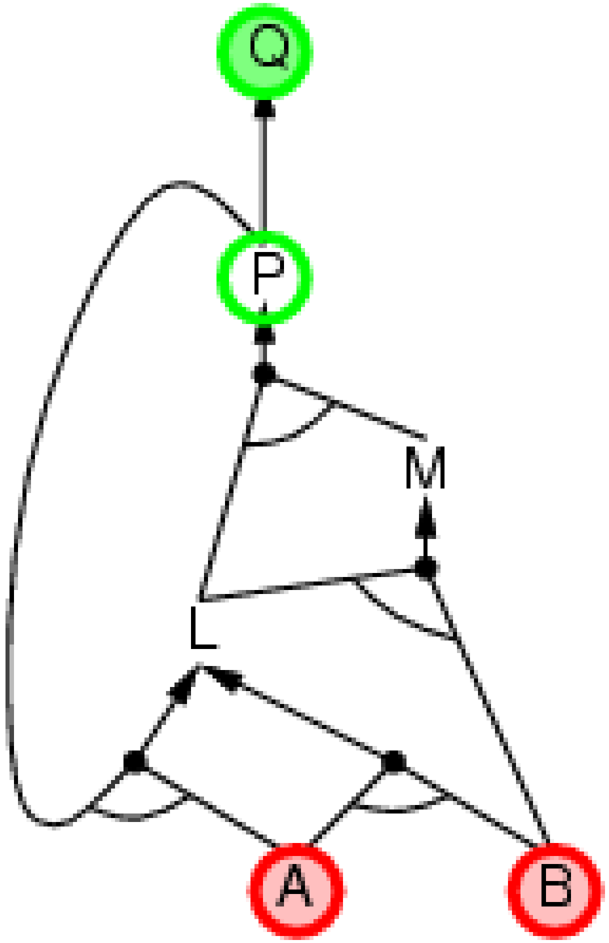
1. Se já foi provado como verdadeiro, ou
2. Se já foi provado como falso.

Encadeamento para trás: exemplo



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

Encadeamento para trás: exemplo



$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

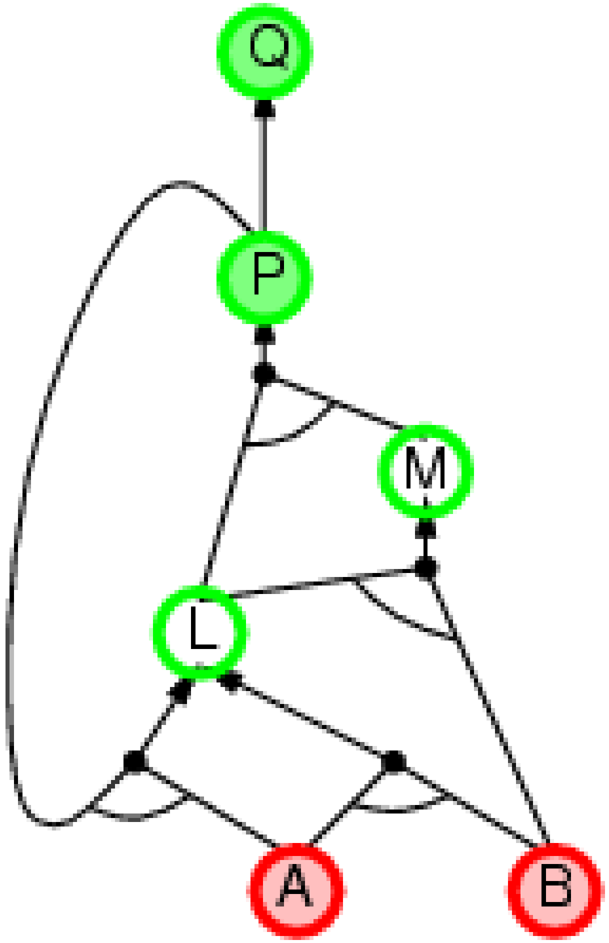
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

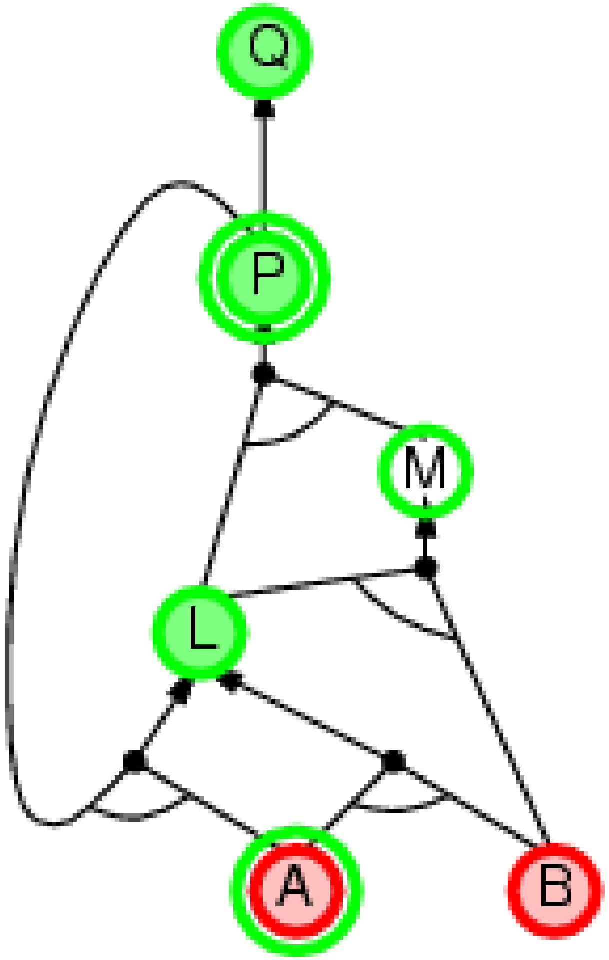
B

Encadeamento para trás: exemplo



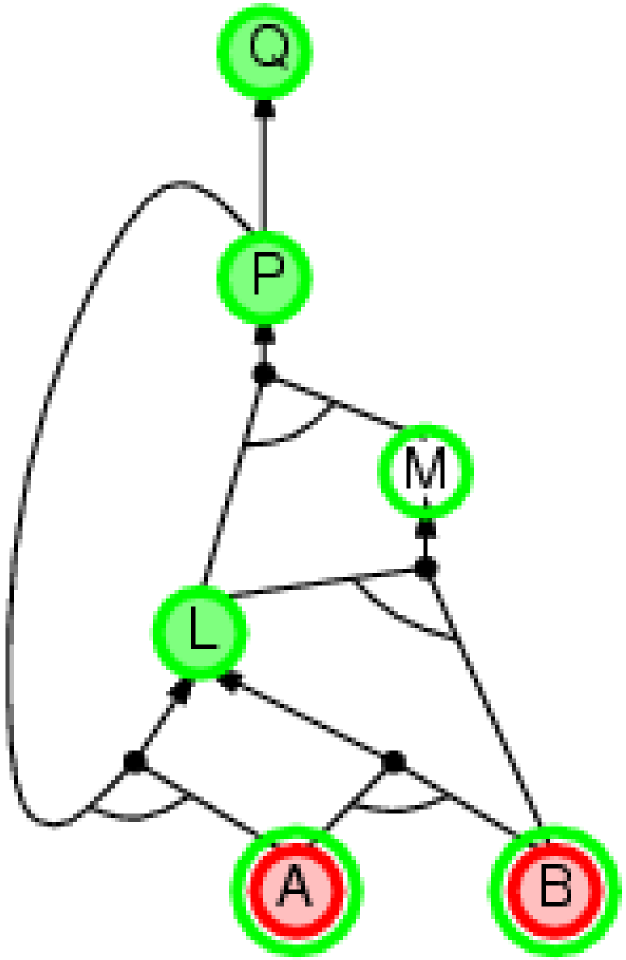
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Encadeamento para trás: exemplo



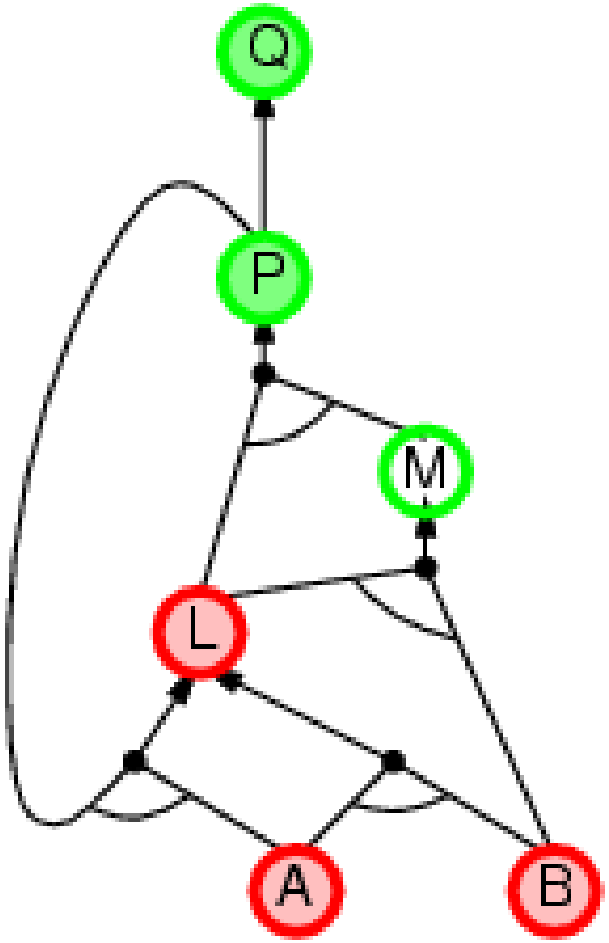
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Encadeamento para trás: exemplo



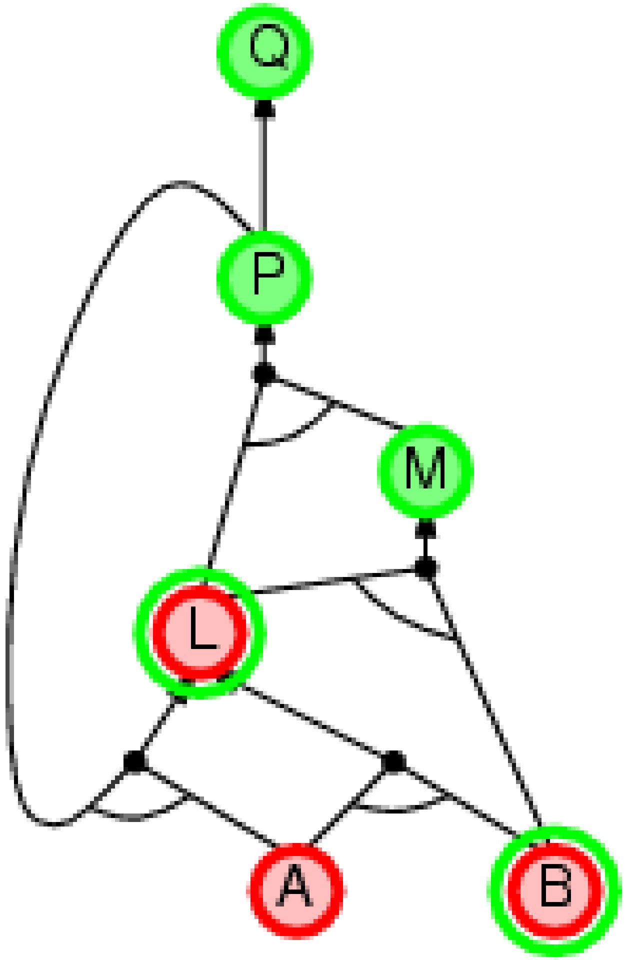
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Encadeamento para trás: exemplo



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

Encadeamento para trás: exemplo



$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

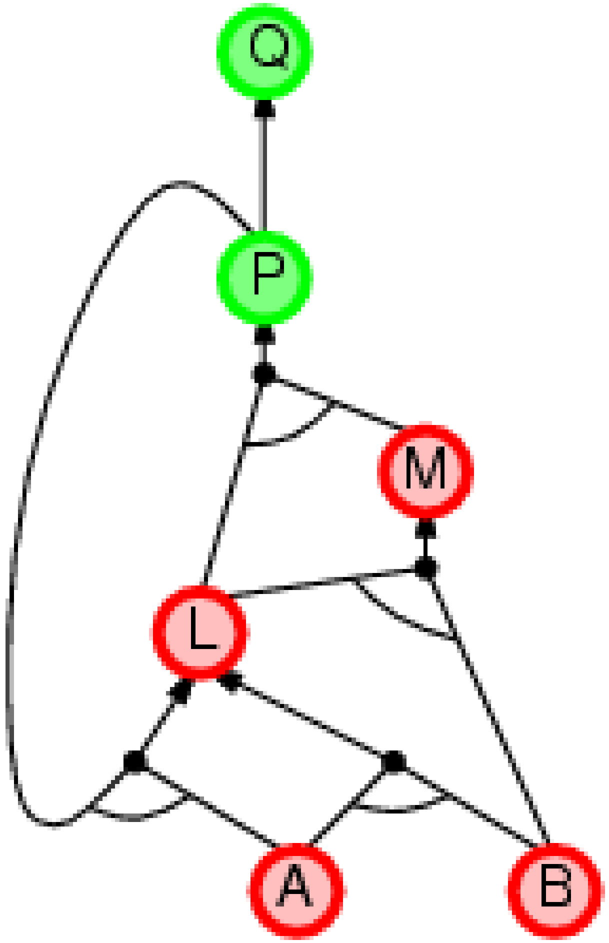
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

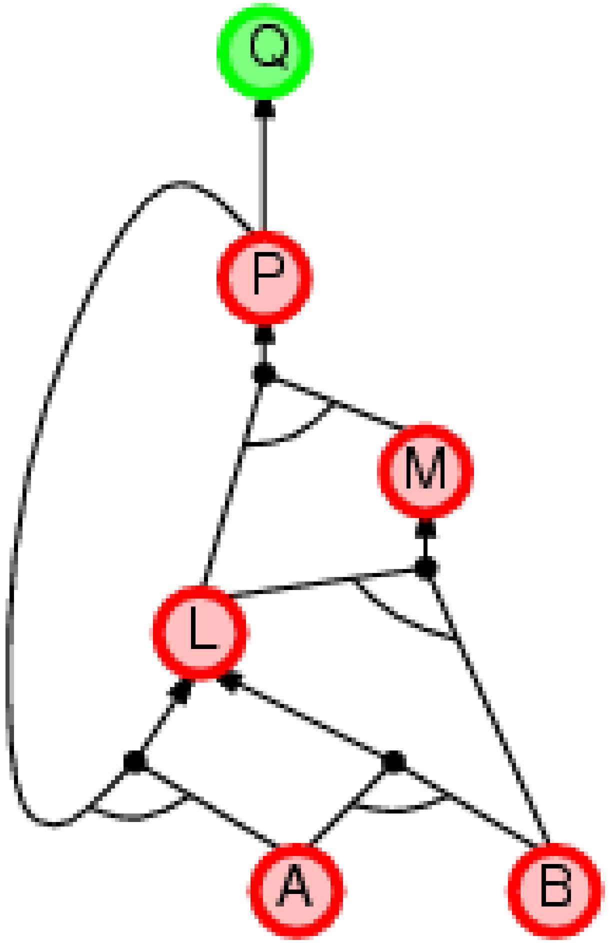
B

Encadeamento para trás: exemplo



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

Encadeamento para trás: exemplo



$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

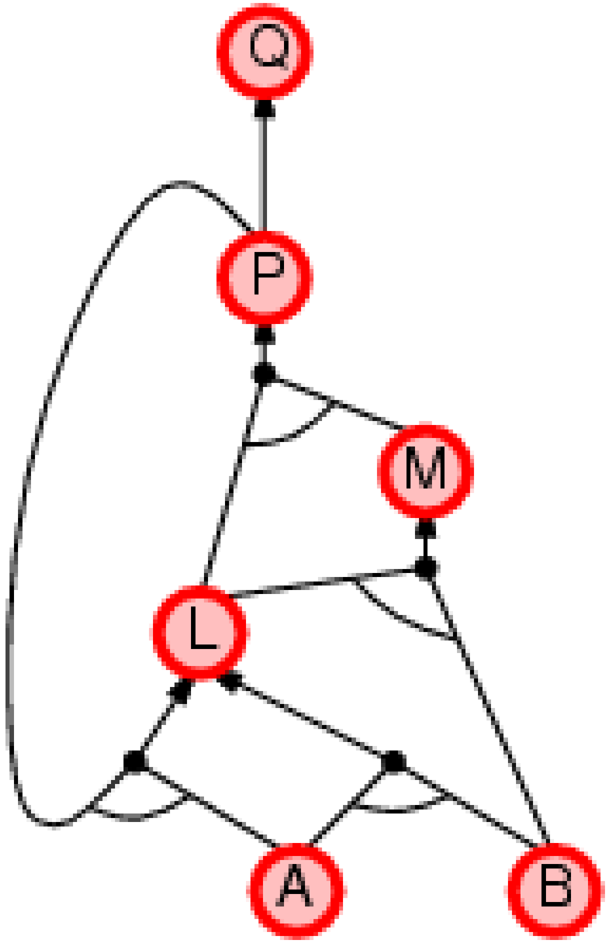
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Encadeamento para trás: exemplo



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

Encadeamento para a frente vs. para trás

- EF é guiado por dados, automático, faz um processamento *inconsciente*
  - e.g., Reconhecimento de objectos
- Pode ter muito trabalho que é irrelevante para o objectivo
- ET é guiado por objectivos, apropriado para a resolução de problemas
  - e.g., Onde estão as minhas chaves?
- Complexidade de ET pode ser muito menor do que linear na dimensão da BC

Função AgenteWumpus (*percepcao)* **devolve** *accao*

input: *percepcao* (lista *cheiramal, cheirabem, brillho)*

estatico: *BC,* inicialmente com posicao do agente

x,y,orient, posição do agente[x,y] e respectiva orientação

visitadas, registo das posições já visitas (inicialmente nada)

accao, acção mais recente do agente

plano, sequência de acções, inicialmente vazio

actualiza *x,y,orient,visitadas* em função de *accao*

se *cheiramal* **entao** DIZ$(BC,M_{x,y})$ **senao** DIZ$(BC,\neg M_{x,y})$

se *cheirabem* **entao** DIZ$(BC,B_{x,y})$ **senao** DIZ$(BC,\neg B_{x,y})$

se *brilho* **então** *accao* $\leftarrow$ *agarrar*

senão se *plano* não vazio **então** *accao* $\leftarrow$ Pop*(plano)*

senão se para posição adjacente *[i,j],* PERGUNTA$(BC,(\neg P_{i,j} \wedge \neg W_{i,j}))$ é *verdadeiro* **ou** para posição adjacente *[i,j],* PERGUNTA$(BC,(P_{i,j} \wedge W_{i,j}))$ é *falso* **então**

plano $\leftarrow$ A*-Grafo*(Problema([x,y],orient,[i,j],visitadas))*

accao $\leftarrow$ Pop*(plano)*

senão *accao* $\leftarrow$ movimento escolhido aleatoriamente

devolve *accao*

Lógica proposicional: limite de expressividade

- BC contém frases lógicas para cada posição

- Para cada instante de tempo *t* e cada posição [*x*,*y*],
  $L_{x,y}^{t} \wedge OlhaParaDireita^{t} \wedge Avança^{t} \Rightarrow L_{x+1,y}^{t+1}$

- Aumento muito rápido do número de cláusulas!

Sumário

- Agentes lógicos aplicam inferência a bases de conhecimento para derivar nova informação e tomar decisões
- Conceitos básicos de lógica:
  - sintaxe: estrutura das frases
  - semântica: validade das frases de acordo com modelos
  - consequência lógica: verdade de uma frase dada outra frase
  - inferência: derivação de frases a partir de outras frases
  - solidez: derivações só produzem consequências lógicas
  - completude: derivações podem produzir todas as consequências lógicas
- Mundo do wumpus requer a possibilidade de representar informação parcial e negada, raciocinar a partir de hipóteses, etc.
- Resolução é completa para lógica proposicional
  Encadeamento para a frente e para trás requer tempo polinomial e é completo para cláusulas de Horn
- Lógica proposicional tem falta de expressividade