

# *Inteligência Artificial*

---

## Agentes Lógicos



# Sumário

- O Mundo do Wumpus
  - Descrição PEAS
- Agentes Baseados em Conhecimento
- Lógica
  - Consequência Lógica
- Lógica Proposicional
- Raciocínio com Lógica Proposicional
- Agentes Baseados em Lógica Proposicional



# *Agentes Lógicos*

---

## O Mundo do Wumpus



# O Mundo do Wumpus

- O mundo do Wumpus é uma caverna escura com vários salões. O agente não possui fonte de luz e deve se guiar pelas pistas que recebe do lugar
  - Em algum lugar da caverna, está um Wumpus, um monstro que devora qualquer um que entre no mesmo salão que ele está
  - O Wumpus pode ser morto por uma flecha do agente, mas este possui somente uma flecha e deve usá-la com cuidado
  - Alguns salões possuem poços sem fundo, onde o agente pode cair e morrer uma morte dolorosa
  - No entanto, o mundo do Wumpus guarda um tesouro: Uma pilha de ouro que deixará o agente rico. Achar este tesouro é o objetivo do agente.



# Descrição PEAS

- Desempenho (P):
  - +1000 pts por pegar o ouro
  - -1000 pts por cair no poço
  - -1000 pts por ser comido pelo Wumpus
  - -1 pt por ação tomada
  - -10 pts por usar a flecha
- Ambiente (E):
  - Uma área de 4 x 4 salões. O agente sempre inicial no salão [1,1], voltado para a direita.
  - As posições do ouro e do Wumpus são determinadas aleatoriamente, com uma probabilidade uniforme entre os salões. O salão [1,1] não é considerado.
  - Cada salão, além do [1,1] pode conter um poço com probabilidade 0.2



# Descrição PEAS

## ■ Atuadores

- O agente pode mover para frente, ou girar um ângulo reto para direita ou para esquerda.
  - Mover para frente não tem nenhum efeito se há um muro na frente do agente
  - Entrar em um salão com um poço ou com o Wumpus é o fim da vida do agente
- O agente usar a ação “pegar” para coletar um objeto no mesmo quadrado que se encontra
- O agente pode usar uma ação de “atirar” para atirar uma flecha em uma linha reta na direção em que está olhando. A flecha continua até atingir um muro ou o Wumpus.
  - O agente somente tem uma flecha







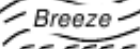
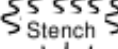









# Descrição PEAS

## ■ Sensores

- No salão contendo o Wumpus (mesmo morto) e nos salões imediatamente adjacentes (não na diagonal) o agente percebe um “fedor”
- Nos salões imediatamente adjacentes (não na diagonal) dos poços, o agente percebe uma “brisa”
- No salão onde se encontra o ouro, o agente perceberá um “brilho”
- Quando o agente bate em um muro, ele sente uma “trombada”
- Quando o Wumpus é morto, ele emite um “urro” horroroso que pode ser percebido pelo agente em qualquer lugar da caverna



# O Mundo do Wumpus

|   |   |  |   |  |
|---|---|--|---|--|
| 4 | <br>Stench |  | <br>Breeze |             |
| 3 |            | <br>Breeze<br><br>Stench<br><br>Gold |            | <br>Breeze  |
| 2 | <br>Stench |  | <br>Breeze |  |
| 1 | <br>START | <br>Breeze  |           | <br>Breeze |
|   | 1   | 2  | 3   | 4  |





# Caracterização do Ambiente

- Observável
  - Parcialmente
- Determinístico
  - Os resultados das ações são exatamente especificados
- Sequencial
  - As ações são executadas em sequencia para atingir o objetivo
- Estático
  - O Wumpus e os fossos não se movem
- Discreto
- Único Agente
  - O Wumpus é uma característica

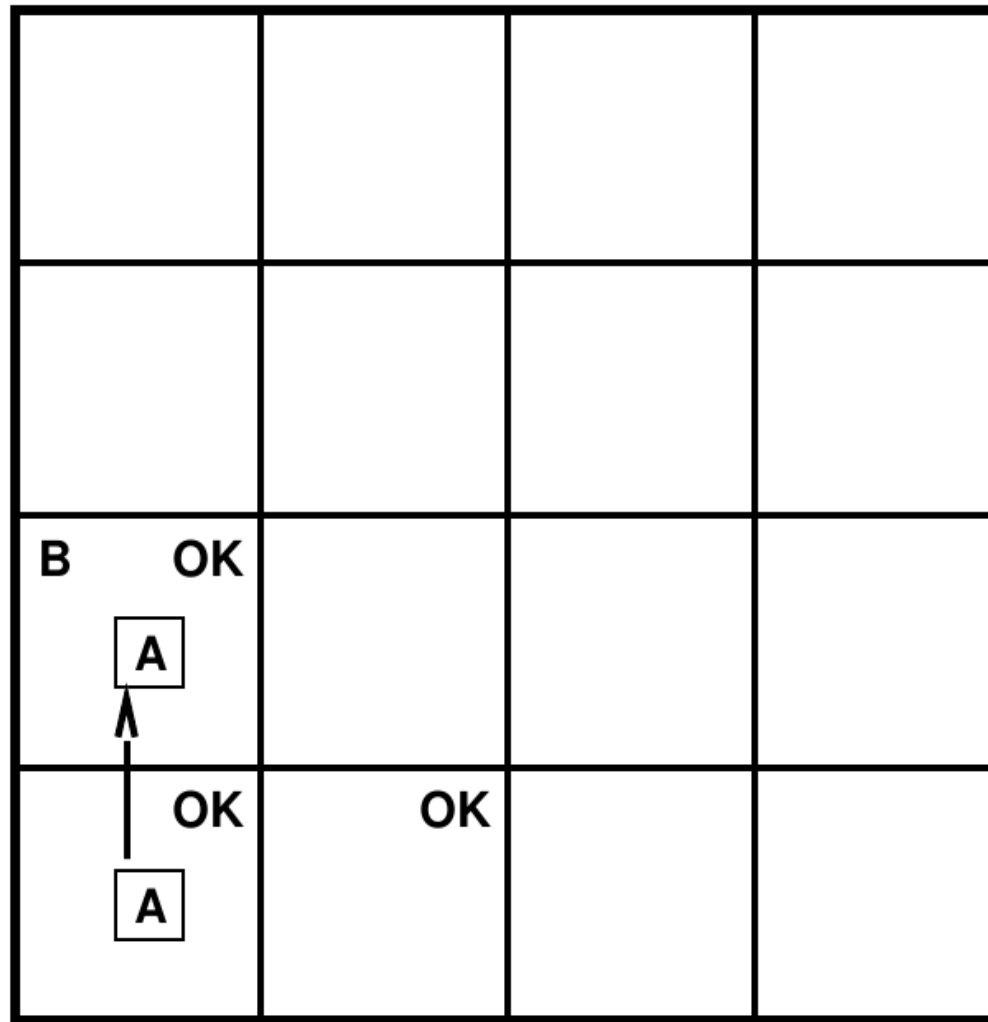


# *Explorando a Caverna*

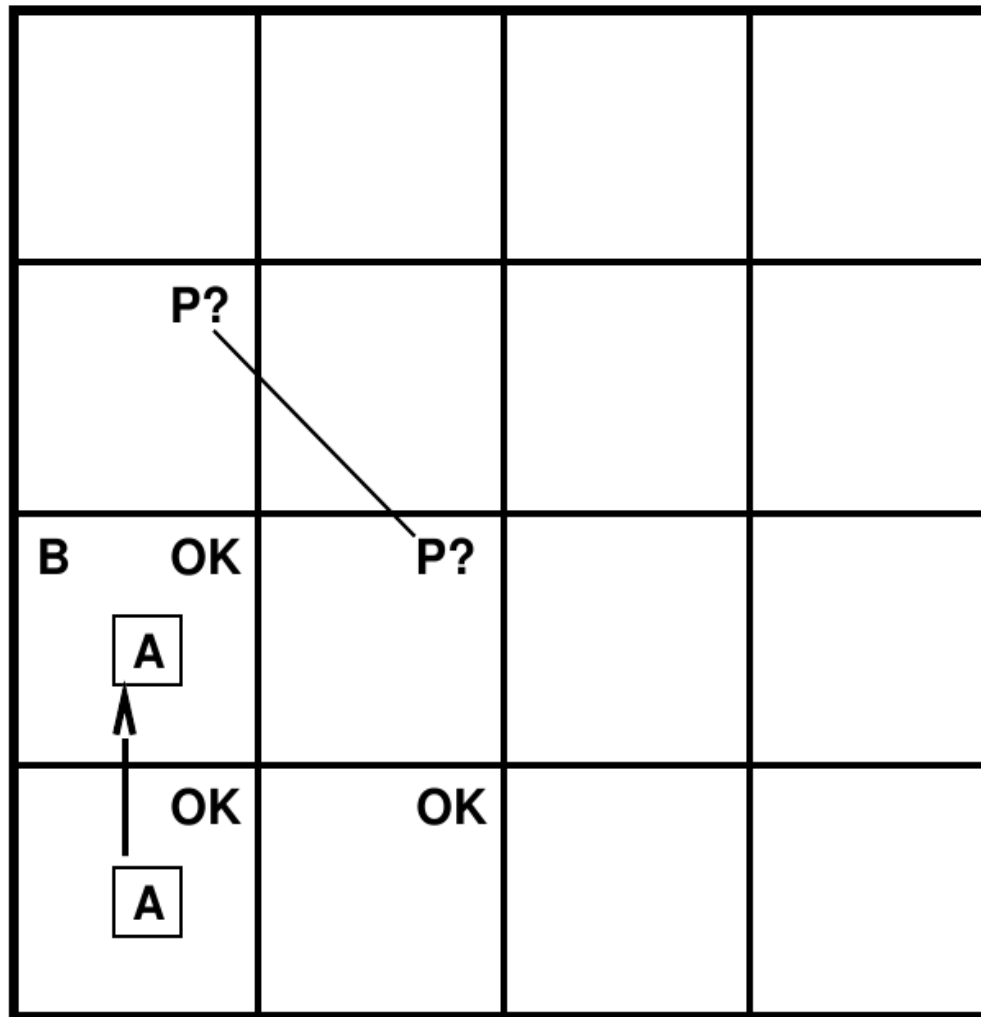
|         |    |  |  |
|---------|----|--|--|
|         |    |  |  |
|         |    |  |  |
| OK      |    |  |  |
| OK<br>A | OK |  |  |



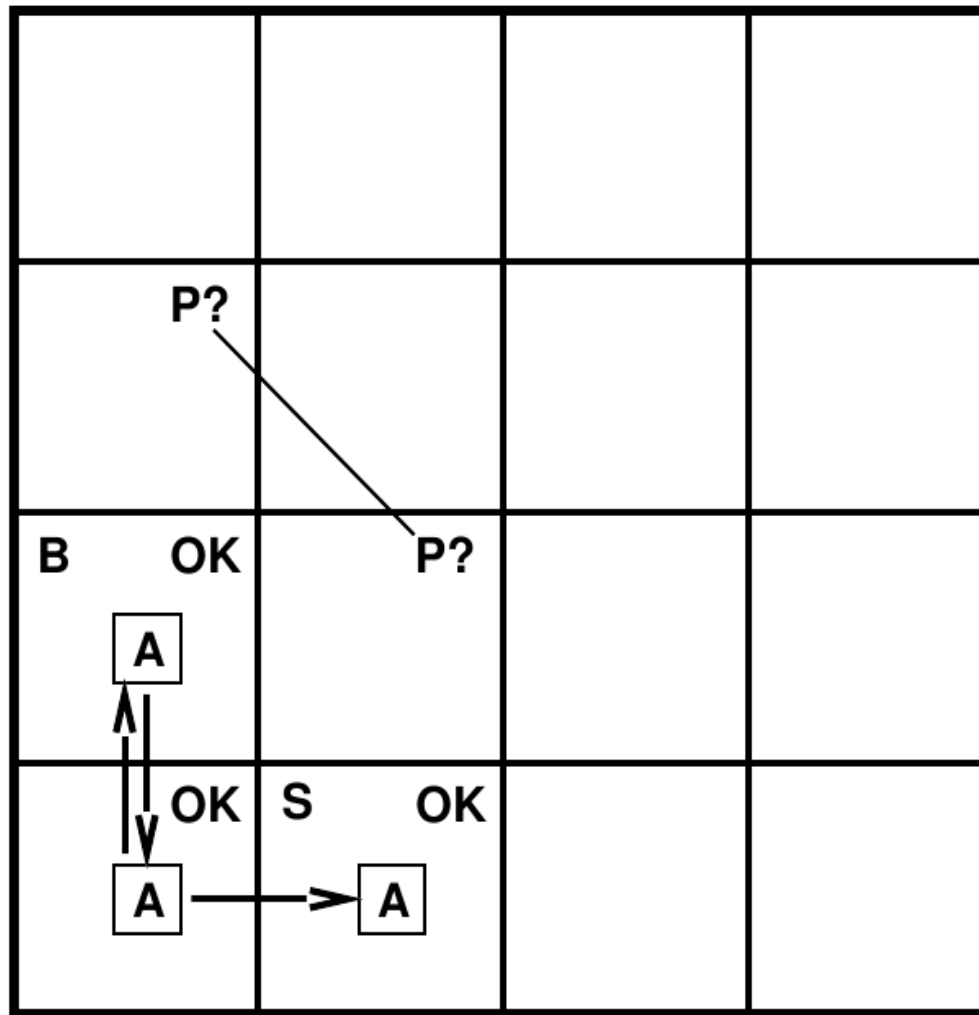
# *Explorando a Caverna*



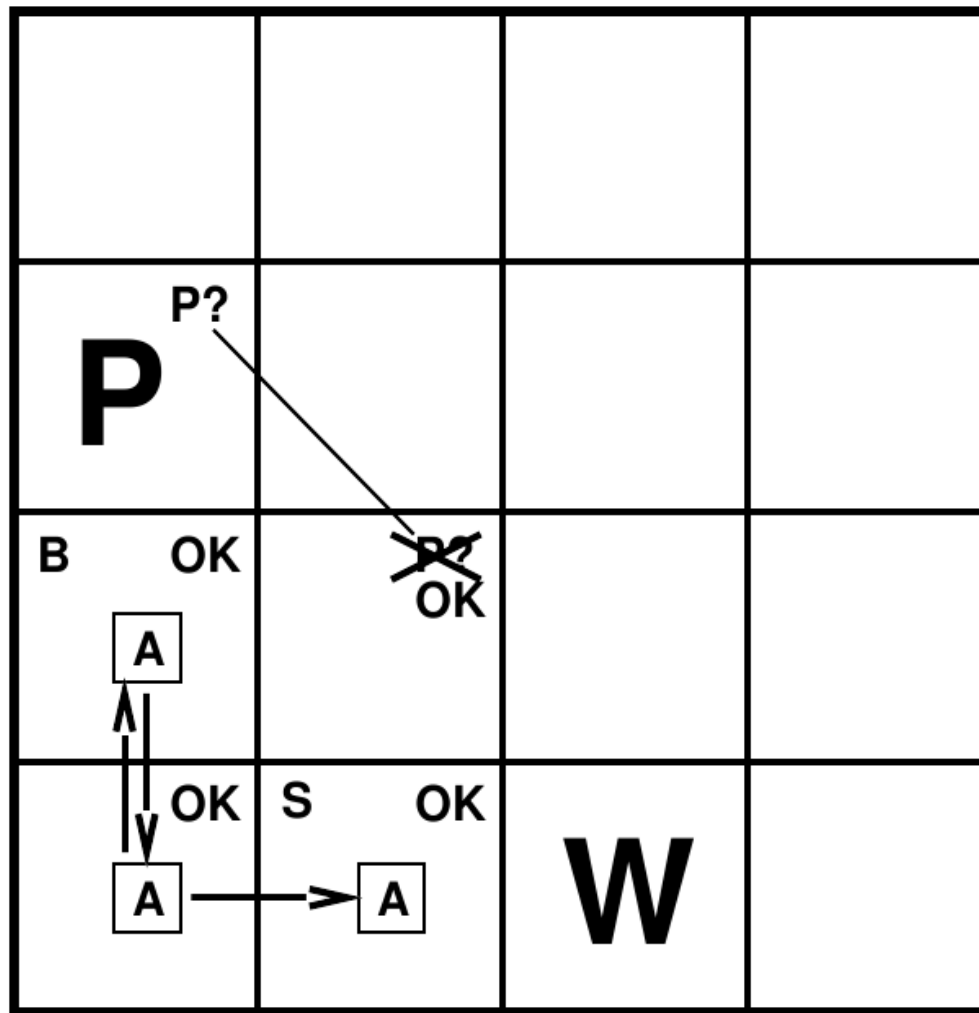
# *Explorando a Caverna*



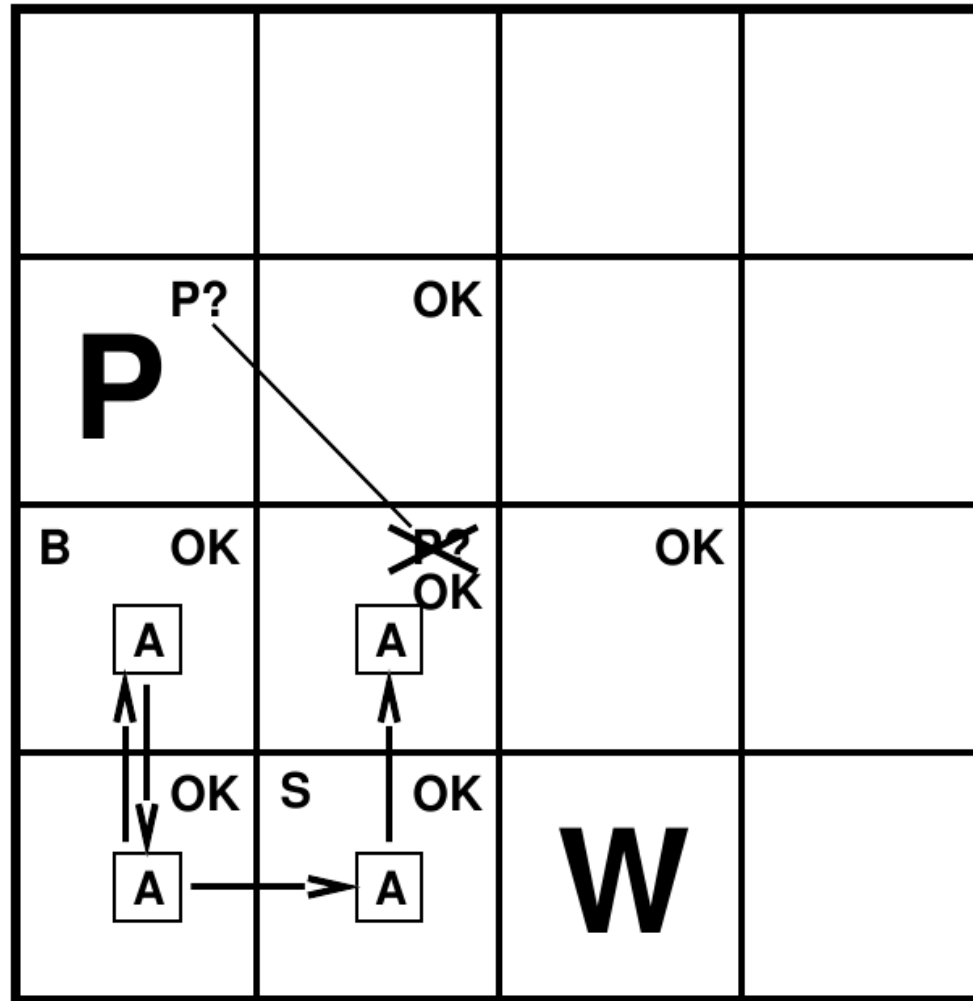
# *Explorando a Caverna*



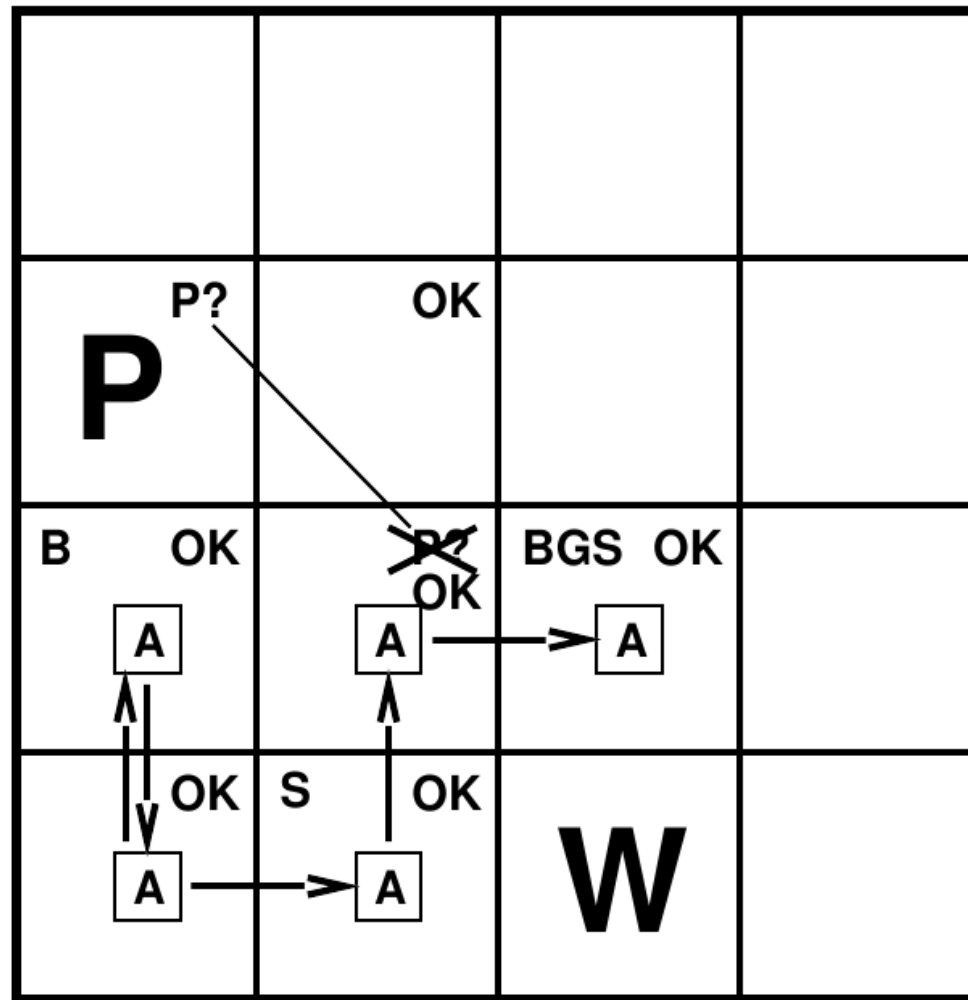
# *Explorando a Caverna*



# Explorando a Caverna

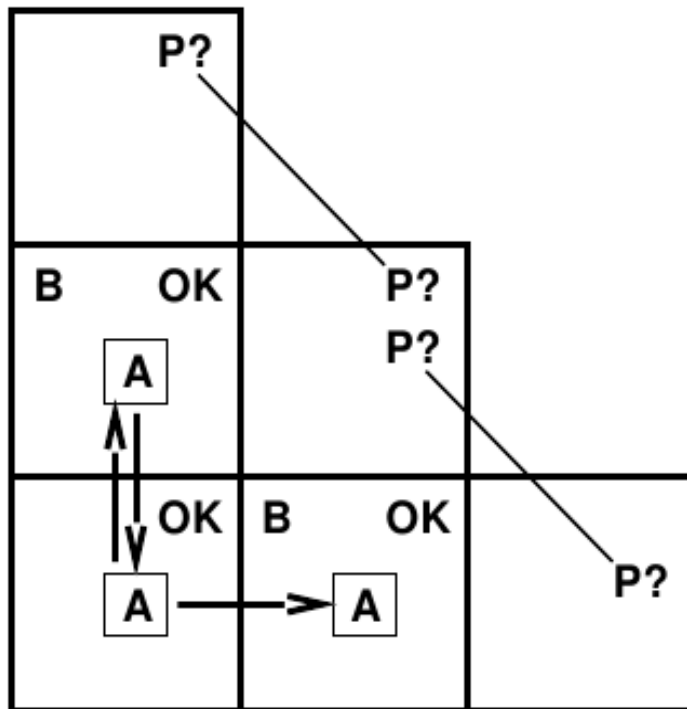


# Explorando a Caverna





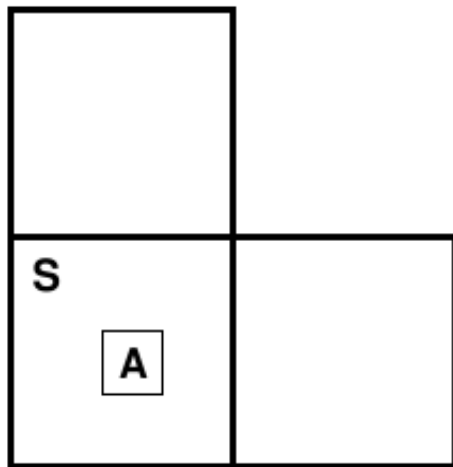
# Situações Complexas



- Não existe caminho seguro
- Análise de probabilidades
  - Ou desistir



# Situações Complexas



- Não pode se mover imediatamente
- Pode forçar o ambiente
  - Atirar para frente e marcar aquela direção como segura



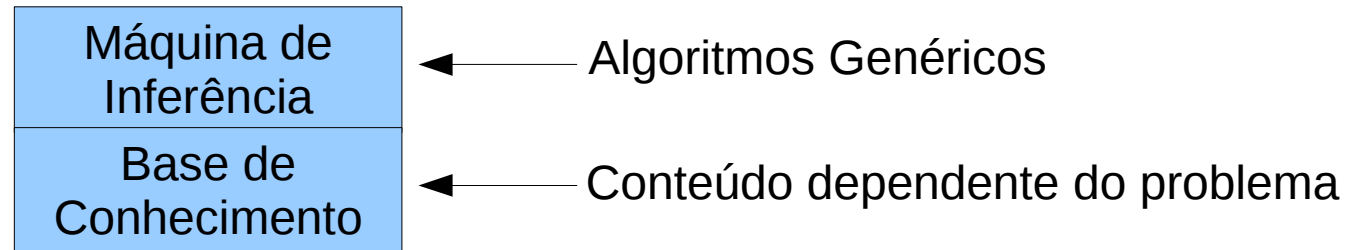
# *Agentes Lógicos*

---

## Agentes Baseados em Conhecimento



# Agentes Baseados em Conhecimento



- Base de Conhecimento
  - Conjunto de Sentenças em Linguagem Formal
- Funcionamento
  - “Informar” ao agente o que ele sabe
  - Ele pode se “Perguntar” o que fazer
- Agentes podem ser vistos
  - Nível de Conhecimento: O que eles sabem independentemente de como são implementados
  - Nível de Implementação: Quais estruturas de dados na BC e quais algoritmos empregados para manipular a BC



# Agente Simples Baseado em Co- nhhecimento

```
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 ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

- Os detalhes da linguagem formal utilizada na KB são abstraídos por
  - MAKE-PERCEPT-SENTENCE
  - MAKE-ACTION-QUERY
  - MAKE-ACTION-SENTENCE



# *Agentes Lógicos*

---

Lógica



# Lógica

- Lógica
  - Linguagem formal para representar informação de forma que possa-se chegar à conclusões
- Sintaxe
  - Define como se escrever sentenças na linguagem formal
- Semântica
  - Define o significado das sentenças
    - i.e. Define a “verdade” da sentença em um mundo
      - Em lógica clássica, as sentenças só podem ser verdadeiras ou falsas
    - Ex.  $x + y = 4$ , verdade na interpretação  $x=2$  e  $y=2$ , falso na interpretação  $x=1$  e  $y = 1$



# Consequência Lógica

- Consequência Lógica
  - Uma afirmativa lógica  $\alpha$  é consequência lógica de um conjunto de afirmativas  $\Gamma$  quando  $\alpha$  for verdadeira sempre que as afirmativas de  $\Gamma$  forem verdadeiras.

$$\Gamma \Rightarrow \alpha$$

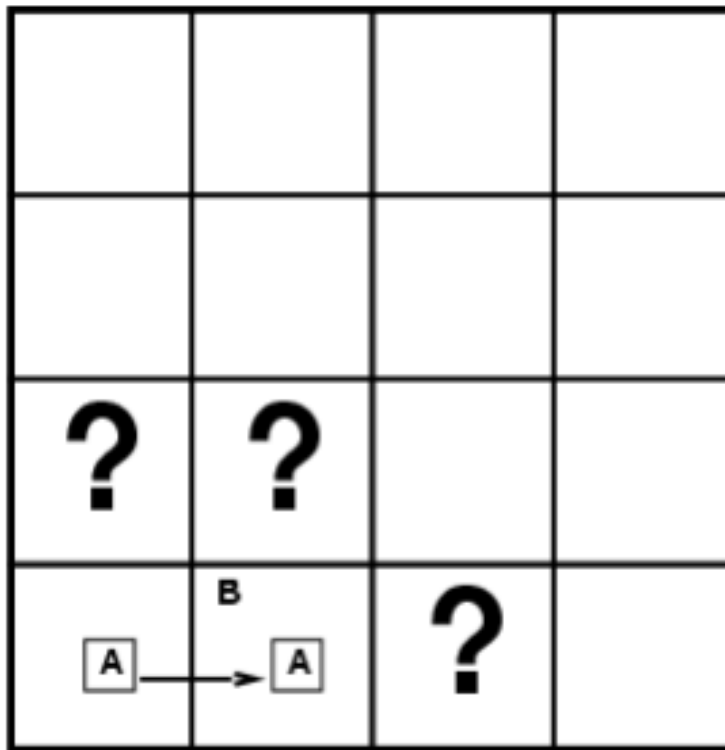
- Modelo
  - $m$  é um modelo de  $\alpha$  se, e somente se, a sentença  $\alpha$  é verdadeira para  $m$





# Consequência Lógica

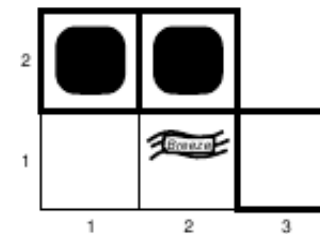
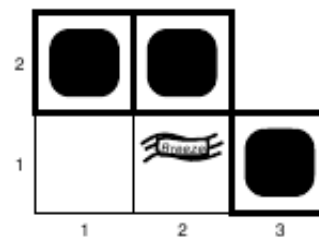
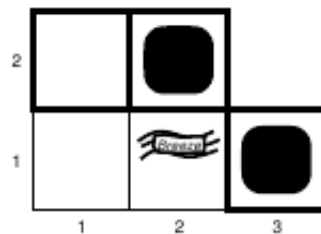
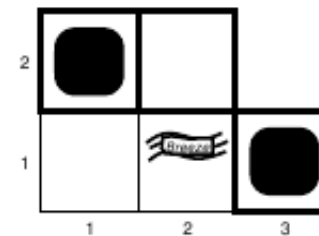
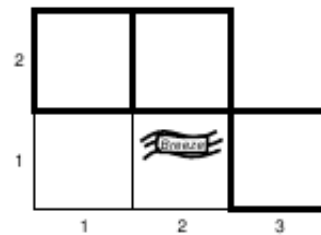
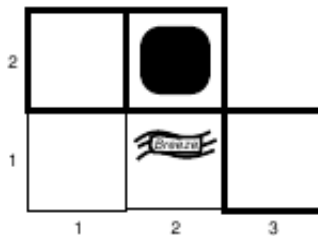
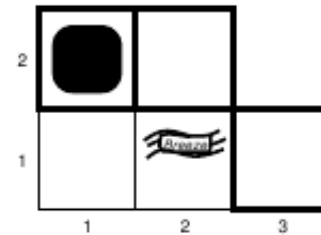
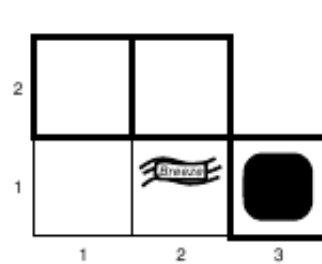
## Wumpus



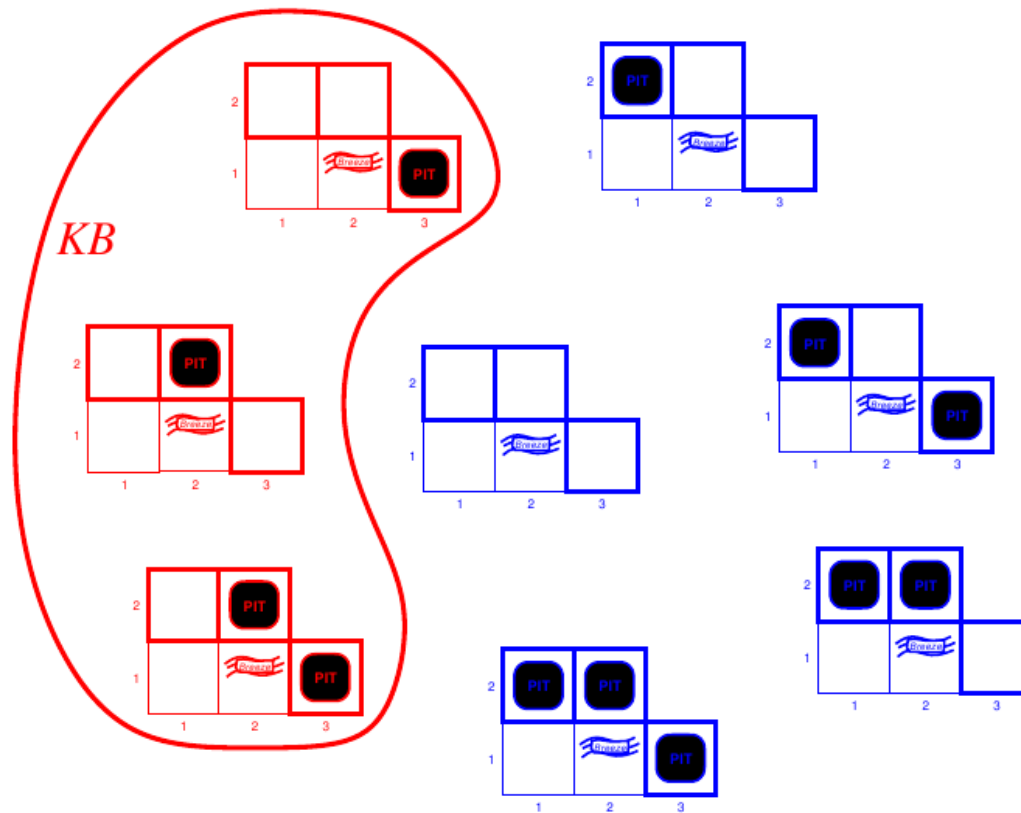
- Considerar os possíveis modelos para as ?s, levando-se em conta apenas os poços



# Modelos para as ?'s



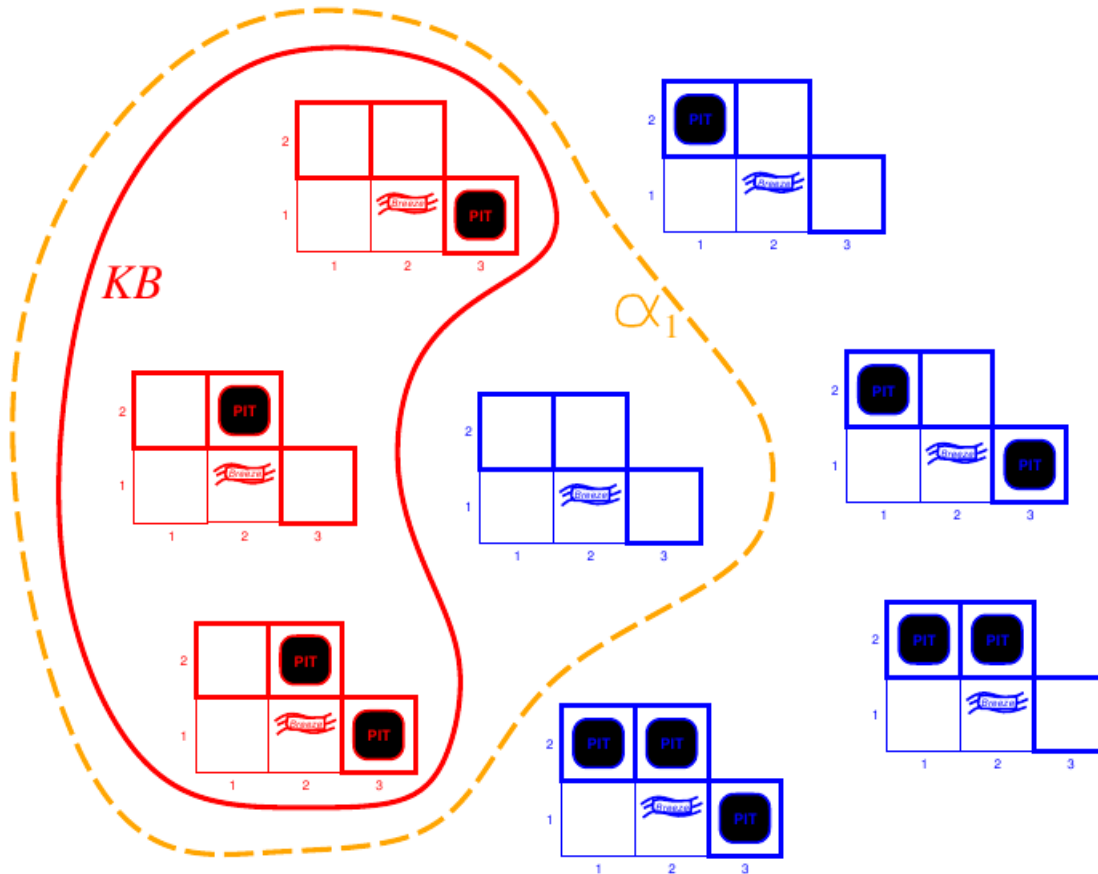
# Base de Conhecimento



- KB = Base de Conhecimento
  - Regras do Mundo do Wumpus + Percepções



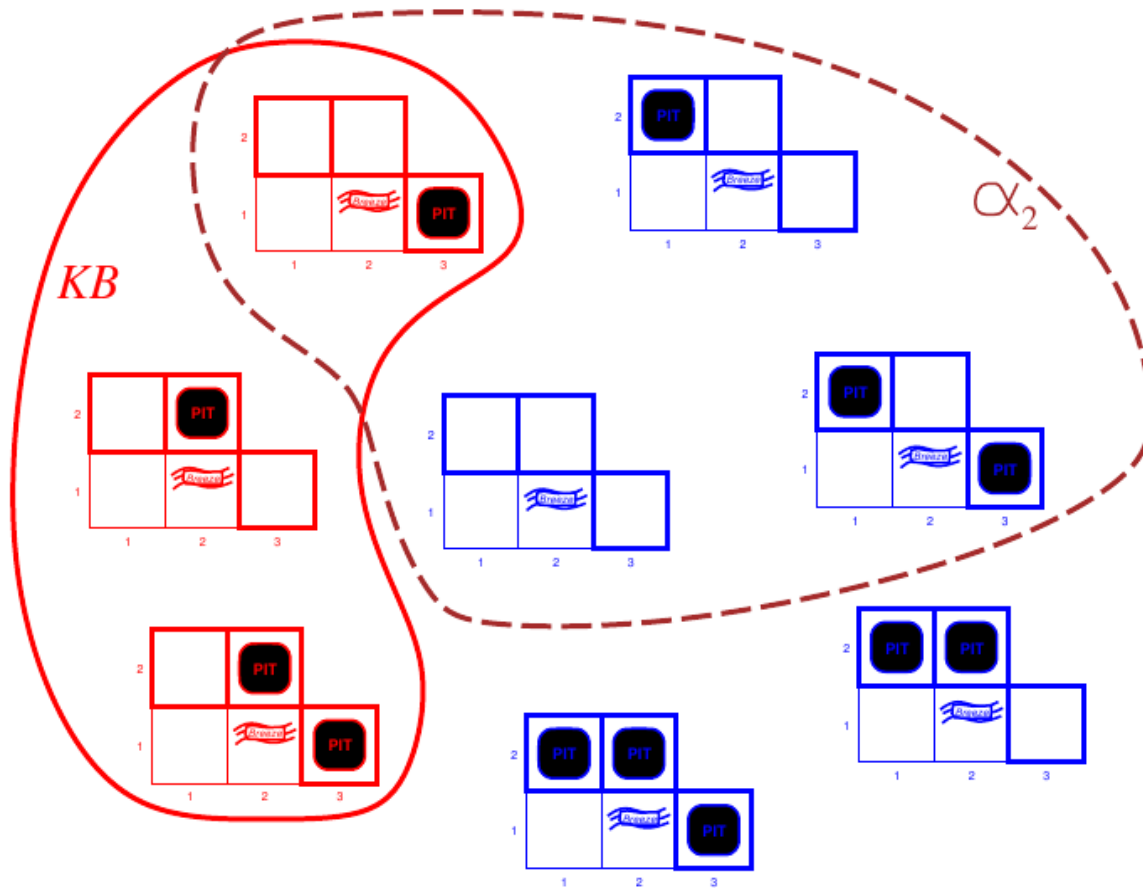
# Consequência Lógica



- Afirmativa  $\alpha_1$ :
  - Salão [1,2] é seguro
- $KB \Rightarrow \alpha_1$ 
  - Provado por verificação de modelos (*model checking*)



# Consequência Lógica



- Afirmativa  $\alpha_2$ :
  - Salão [2,2] é seguro
- $KB \not\models \alpha_2$ 
  - Provado por verificação de modelos (*model checking*)



# Derivação Lógica

- Se um algoritmo de inferência  $i$  pode derivar  $\alpha$  de KB, escreve-se:

$$KB \vdash_i \alpha$$

- Um algoritmo é **consistente** (*sound*) se somente derivar sentenças que são consequências lógicas
- Um algoritmo é **completo** se puder derivar alguma sentença que é consequência lógica



# *Hipótese Básica da IA*

- Se KB representa fatos do mundo real, qualquer sentença derivada de KB por um algoritmo de inferência consistente, também será verdadeira no mundo real
- Apesar do processo de inferência lidar apenas com a sintaxe da linguagem formal da KB



# *Conexão entre Sintaxe da KB e o Mundo Real*

- Como a conexão é criada
  - Pelos sensores do agente
    - Os sensores verificam o que é “verdade”
  - Pelo processo de aprendizagem
    - Modelos “pré-programados”
    - Experiência
- Assim, qualquer sentença na KB é considerada válida no mundo real





# *Agentes Lógicos*

---

## Lógica Proposicional



# Gramática BNF para Sentenças

*Sentença*  $\rightarrow$  *SentençaAtômica* | *SentençaComplexa*

*SentençaAtômica*  $\rightarrow$  **Verdadeiro** | **Falso** | *Símbolo*

*Símbolo*  $\rightarrow$  **P** | **Q** | **R** | ...

*SentençaComplexa*  $\rightarrow$   $\neg$ *Sentença*  
| (*Sentença*  $\wedge$  *Sentença*)  
| (*Sentença*  $\vee$  *Sentença*)  
| (*Sentença*  $\rightarrow$  *Sentença*)  
| (*Sentença*  $\leftrightarrow$  *Sentença*)



# Conectivos Lógicos

## Negação

| $\alpha$ | $\neg\alpha$ |
|----------|--------------|
| V        | F            |
| F        | V            |

## Conjunção

| $\alpha$ | $\beta$ | $\alpha \wedge \beta$ |
|----------|---------|-----------------------|
| V        | V       | V                     |
| V        | F       | F                     |
| F        | V       | F                     |
| F        | F       | F                     |

## Disjunção

| $\alpha$ | $\beta$ | $\alpha \vee \beta$ |
|----------|---------|---------------------|
| V        | V       | V                   |
| V        | F       | V                   |
| F        | V       | V                   |
| F        | F       | F                   |



# Conectivos Lógicos

## Condicional

| $\alpha$ | $\beta$ | $\alpha \rightarrow \beta$ |
|----------|---------|----------------------------|
| V        | V       | V                          |
| V        | F       | F                          |
| F        | V       | V                          |
| F        | F       | V                          |

## Bicondicional

| $\alpha$ | $\beta$ | $\alpha \leftrightarrow \beta$ |
|----------|---------|--------------------------------|
| V        | V       | V                              |
| V        | F       | F                              |
| F        | V       | F                              |
| F        | F       | V                              |

**CUIDADO**: Existe uma assimetria na tabela verdade da **condicional**. Inverter a ordem das expressões **altera** o resultado!!



# Base de Conhecimento

## Mundo do Wumpus

- Não existe poço em P(1,1)  
 $R_1 : \neg P_{1,1}$
- Existe brisa somente nos salões vizinhos aos poços (somente salões relevantes)

$$R_2 : B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})$$

$$R_3 : B_{2,1} \leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

- Percepções do Agente

$$R_4 : \neg B_{1,1}$$

$$R_5 : B_{2,1}$$



# Algoritmo de Inferência por Tabela Verdade

$$KB \equiv (R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5)$$

| $B_{1,1}$ | $B_{2,1}$ | $P_{1,1}$ | $P_{1,2}$ | $P_{2,1}$ | $P_{2,2}$ | $P_{3,1}$ | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ | $KB$  |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-------|-------|-------|-------|-------|-------|
| false     | false     | false     | false     | false     | false     | false     | true  | true  | true  | true  | false | false |
| false     | false     | false     | false     | false     | false     | true      | true  | true  | false | true  | false | false |
| ⋮         | ⋮         | ⋮         | ⋮         | ⋮         | ⋮         | ⋮         | ⋮     | ⋮     | ⋮     | ⋮     | ⋮     | ⋮     |
| false     | true      | false     | false     | false     | false     | false     | true  | true  | false | true  | true  | false |
| false     | true      | false     | false     | false     | false     | true      | true  | true  | true  | true  | true  | true  |
| false     | true      | false     | false     | false     | true      | false     | true  | true  | true  | true  | true  | true  |
| false     | true      | false     | false     | false     | true      | true      | true  | true  | true  | true  | true  | true  |
| false     | true      | false     | false     | true      | false     | false     | true  | false | false | true  | true  | false |
| ⋮         | ⋮         | ⋮         | ⋮         | ⋮         | ⋮         | ⋮         | ⋮     | ⋮     | ⋮     | ⋮     | ⋮     | ⋮     |
| true      | true      | true      | true      | true      | true      | true      | false | true  | true  | false | true  | false |



# Algoritmo de Enumeração em Profundidade para Inferência

**function** **TT-ENTAILS?**( $KB, \alpha$ ) **returns** *true* or *false*

**inputs:**  $KB$ , the knowledge base, a sentence in propositional logic

$\alpha$ , the query, a sentence in propositional logic

$symbols \leftarrow$  a list of the proposition symbols in  $KB$  and  $\alpha$

**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**

$P \leftarrow \text{FIRST}(symbols); rest \leftarrow \text{REST}(symbols)$

**return** **TT-CHECK-ALL**( $KB, \alpha, rest, \text{EXTEND}(P, true, model)$ ) **and**  
        **TT-CHECK-ALL**( $KB, \alpha, rest, \text{EXTEND}(P, false, model)$ )

- Consistente e Completo
- Co-NP-Completo



# Outra forma de Pensar

- Teorema da Dedução

$$(KB \Rightarrow \alpha) \leftrightarrow [(KB \rightarrow \alpha) \equiv \top]$$

- Afirmativa Válida

$\top \equiv$  **Verdadeiro** para todos modelos

- Contradição

$\perp \equiv$  **Falso** para todos modelos

- Satisfazível

- Uma afirmativa é satisfazível se é verdadeira para algum modelo





# *Algoritmo de Inferência*

## *Cascadeamento para Frente*

- No mundo real, as bases de conhecimento muitas vezes contêm cláusulas de Horn, ou podem ser transformadas em Cláusulas de Horn
  - Símbolo
  - (disjunção de símbolos)  $\rightarrow$  Símbolo

$$(L_{1,1} \wedge \text{Brisa}) \rightarrow B_{1,1}$$

$$P_{1,1} \rightarrow \text{falso}$$



# Regra de Inferência Utilizada

- Para provar  $\alpha \rightarrow \beta$ , supor  $\alpha$  e provar  $\beta$ .
  - *Modus Ponendo Ponens*

$$\frac{\begin{array}{c} \alpha \\ \alpha \rightarrow \beta \end{array}}{\beta}$$



# Cascadeamento para Frente

**KB  $\Rightarrow$  Q?**

$$P \rightarrow Q$$

$$L \wedge M \rightarrow P$$

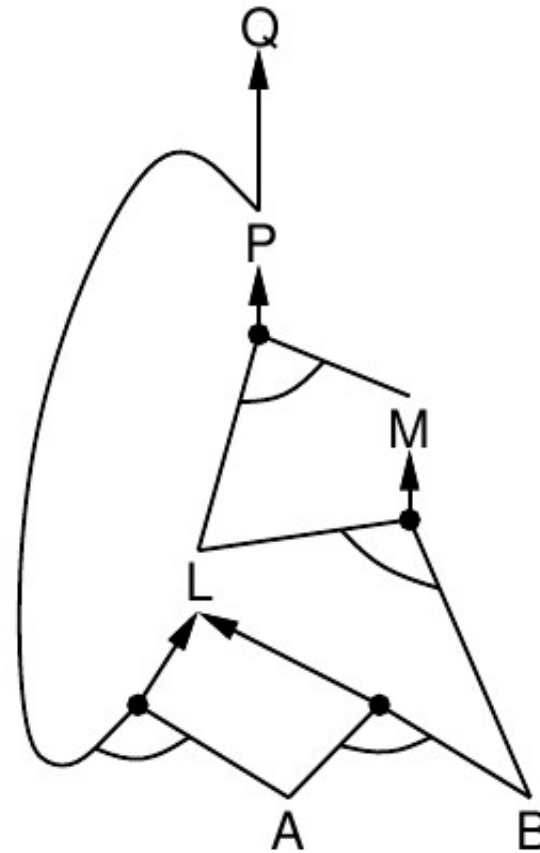
$$B \wedge L \rightarrow M$$

$$A \wedge P \rightarrow L$$

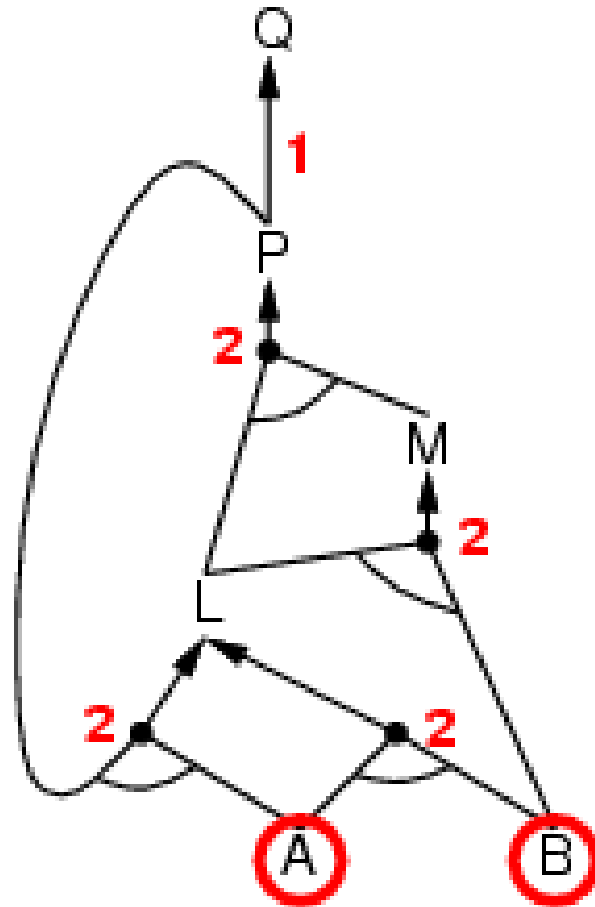
$$A \wedge B \rightarrow L$$

$A$

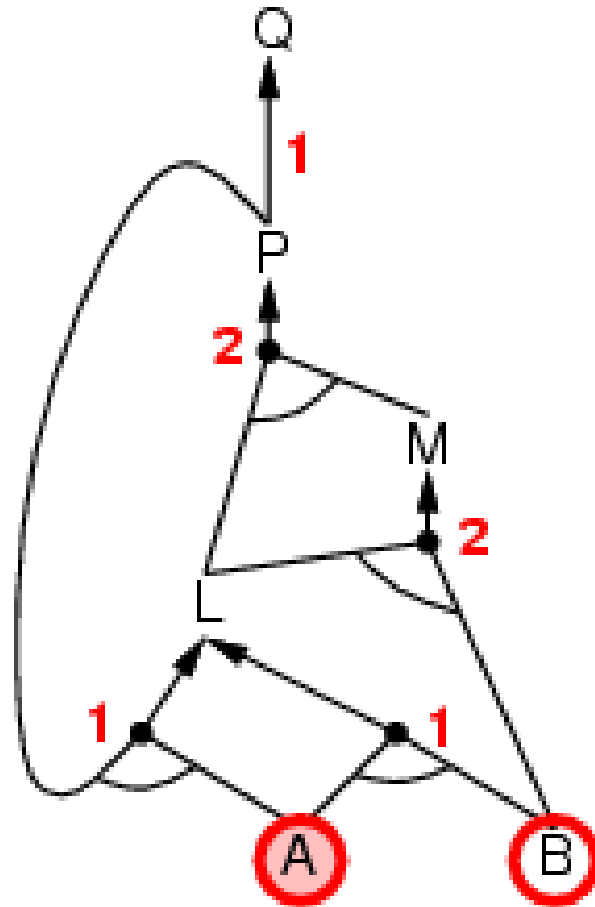
$B$



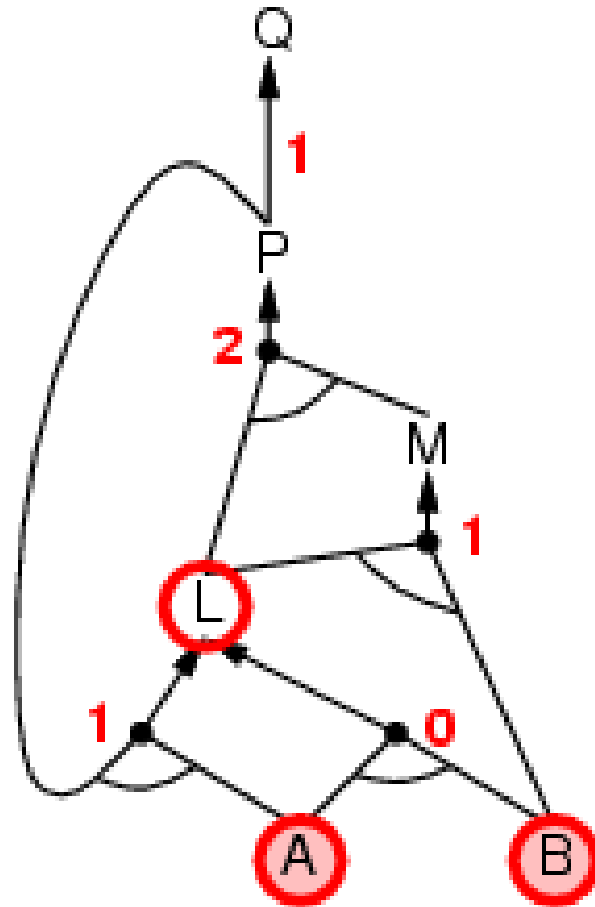
# *Cascadeamento para Frente*



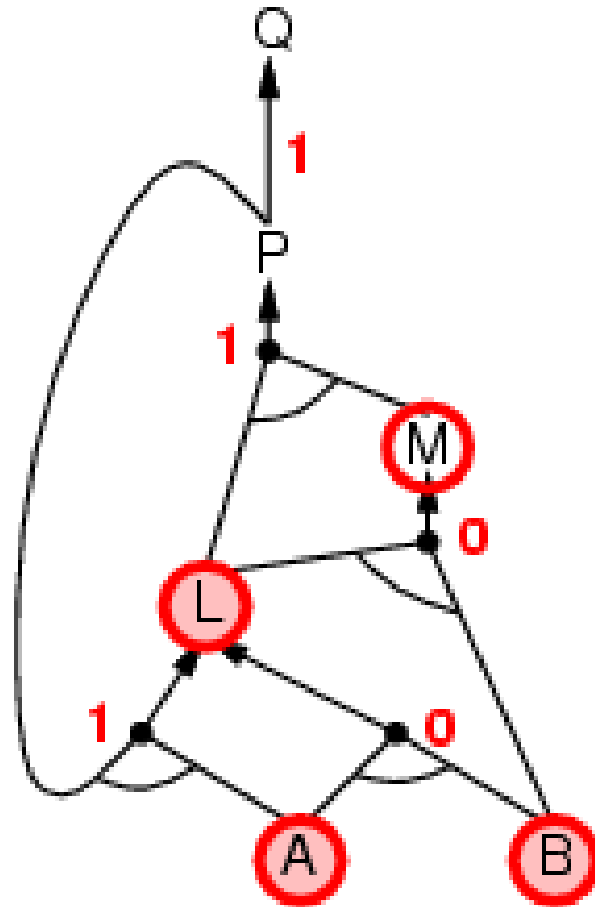
# *Cascadeamento para Frente*



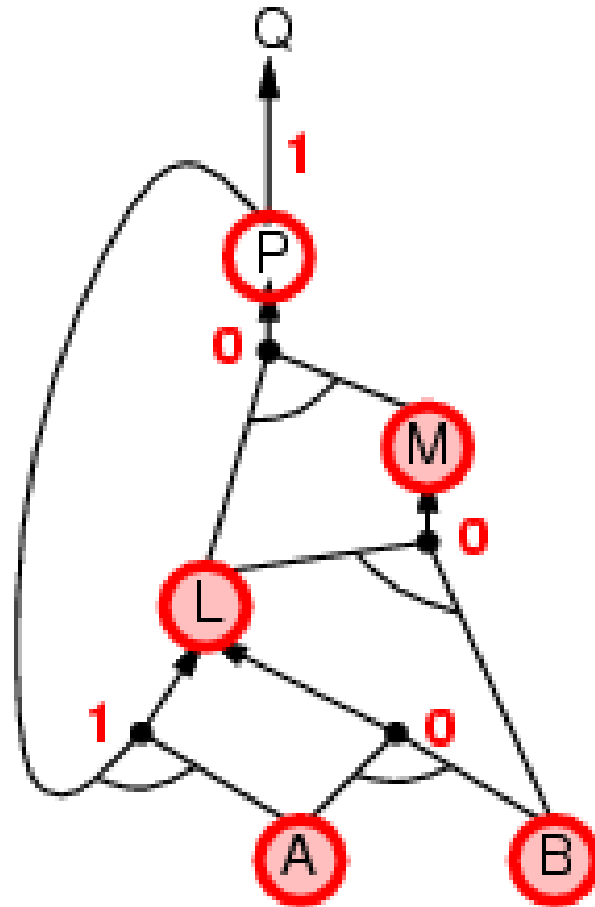
# Cascadeamento para Frente



# Cascadeamento para Frente

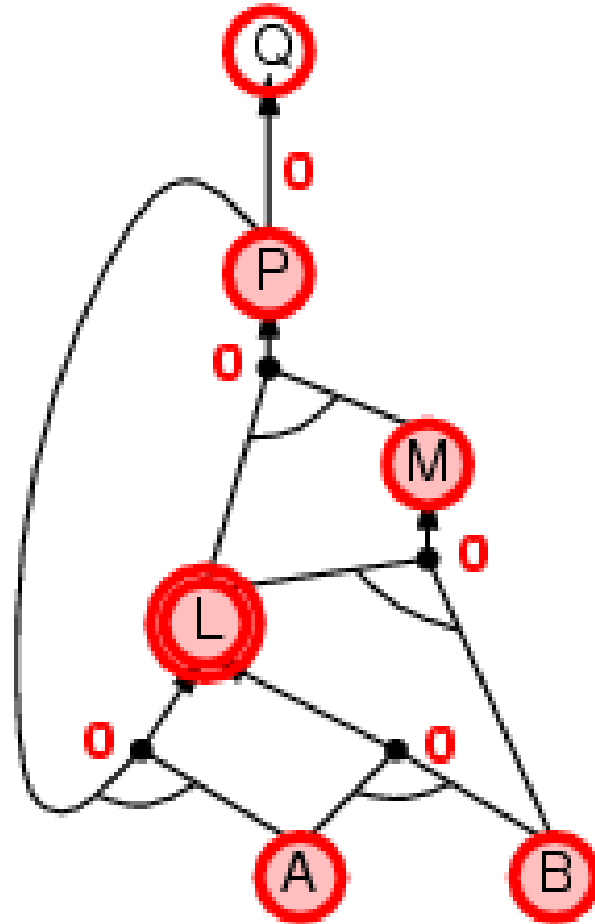


# Cascadeamento para Frente

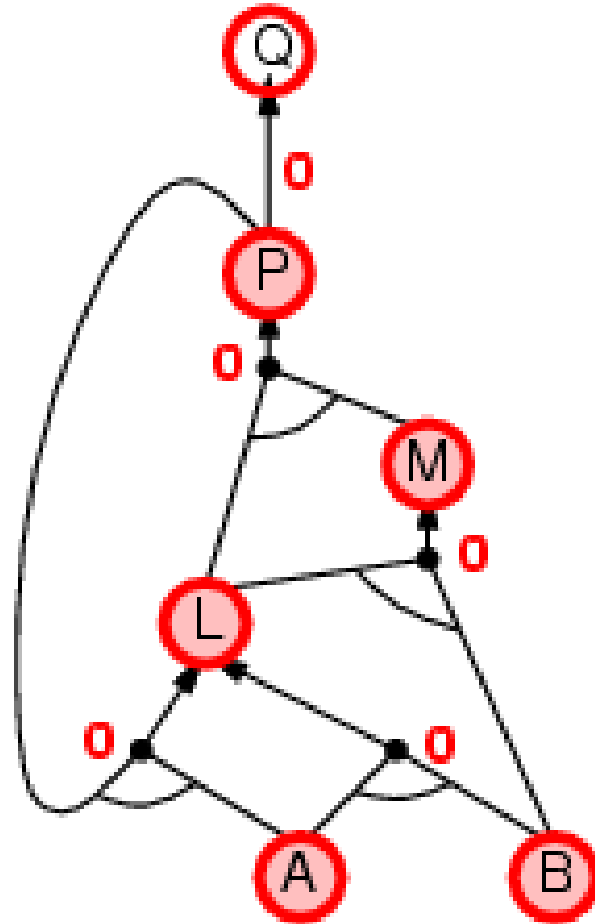




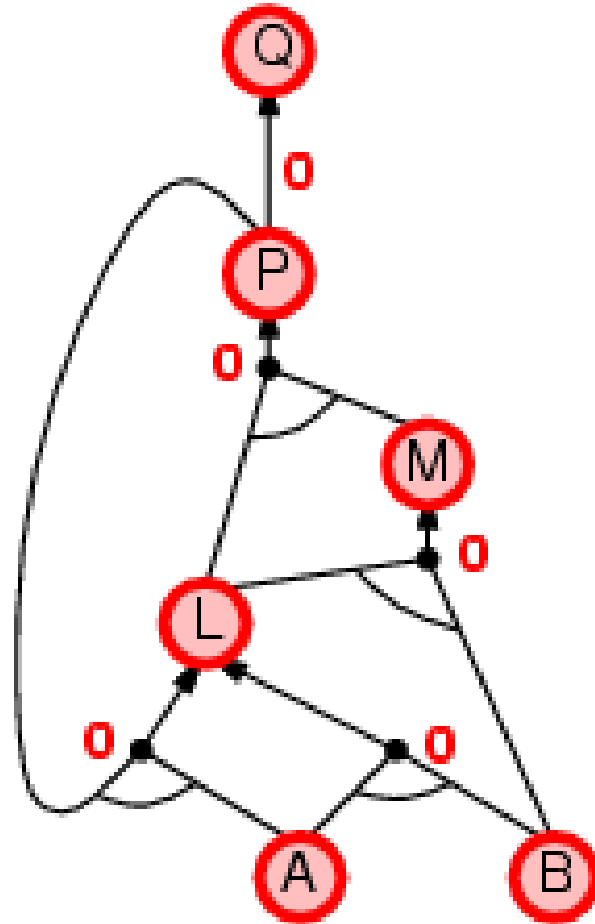
# Cascadeamento para Frente



# Cascadeamento para Frente



# Cascadeamento para Frente



# Algoritmo Cascadeamento para Frente

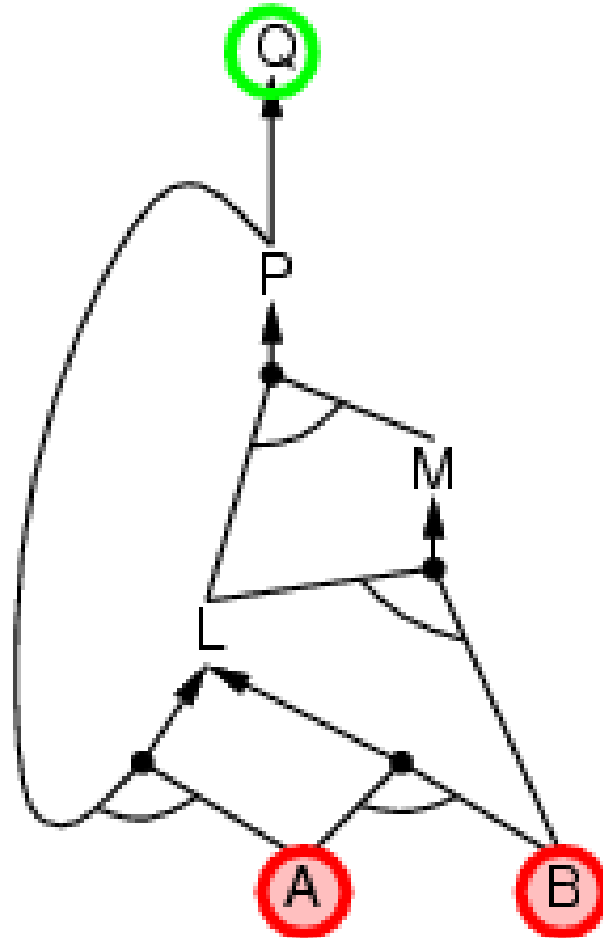
```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional Horn clauses
         q, the query, a proposition symbol
  local variables: count, a table, indexed by clause, initially the number of premises
                  inferred, a table, indexed by symbol, each entry initially false
                  agenda, a list of symbols, initially the symbols known in KB

  while agenda is not empty do
    p ← POP(agenda)
    unless inferred[p] do
      inferred[p] ← 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
```

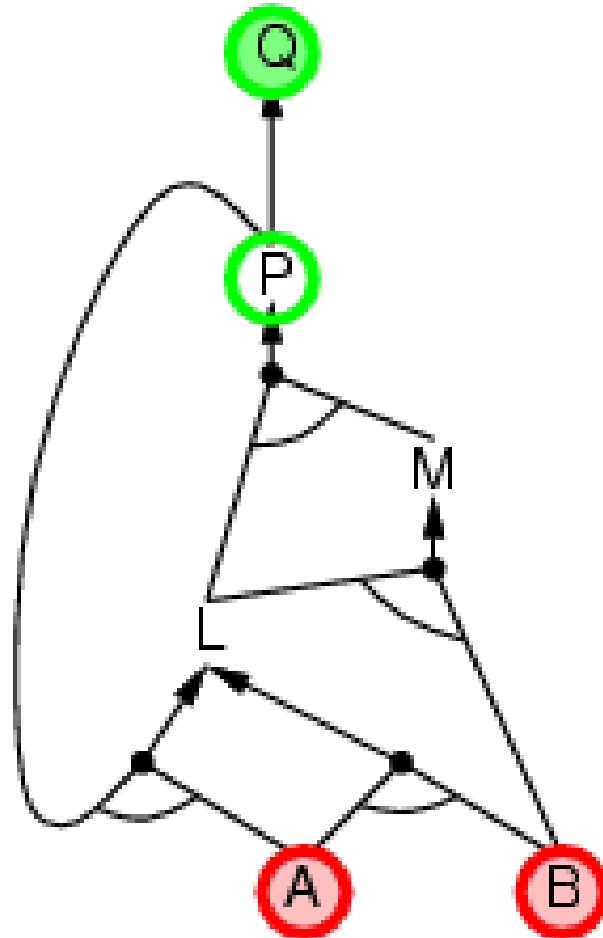
**Linear** em relação ao tamanho da KB!!!



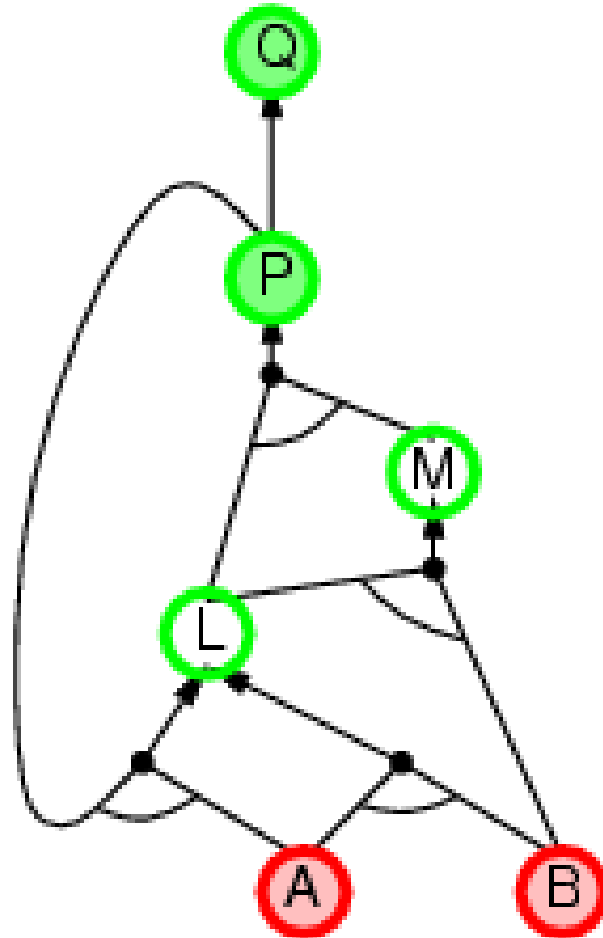
# Cascadeamento para Trás



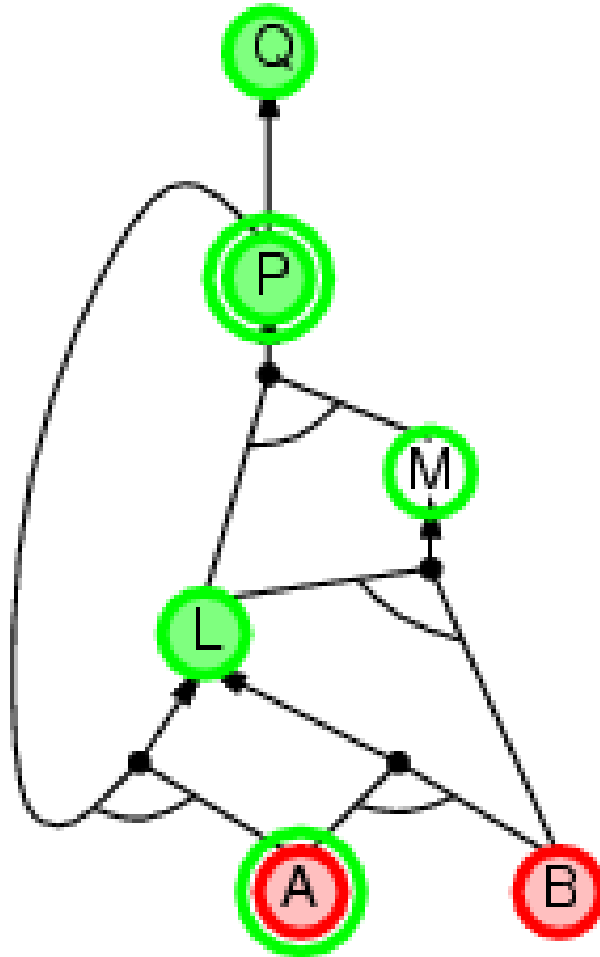
# Cascadeamento para Trás



# Cascadeamento para Trás

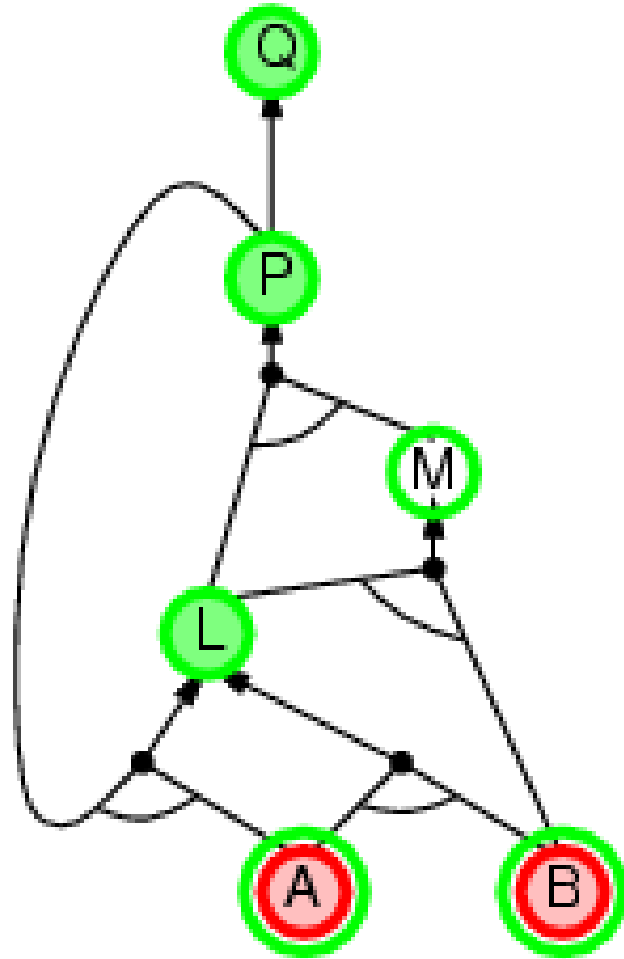


# Cascadeamento para Trás

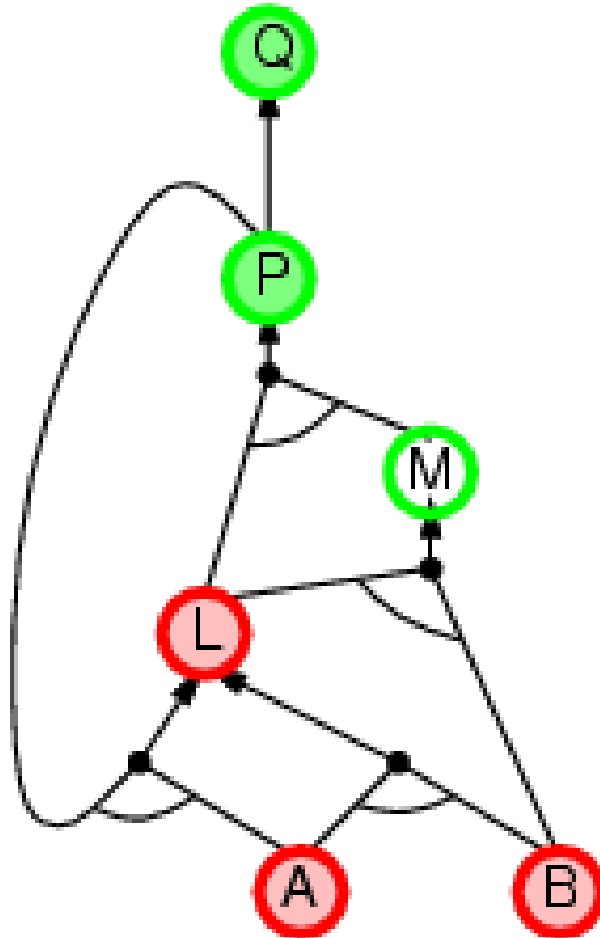




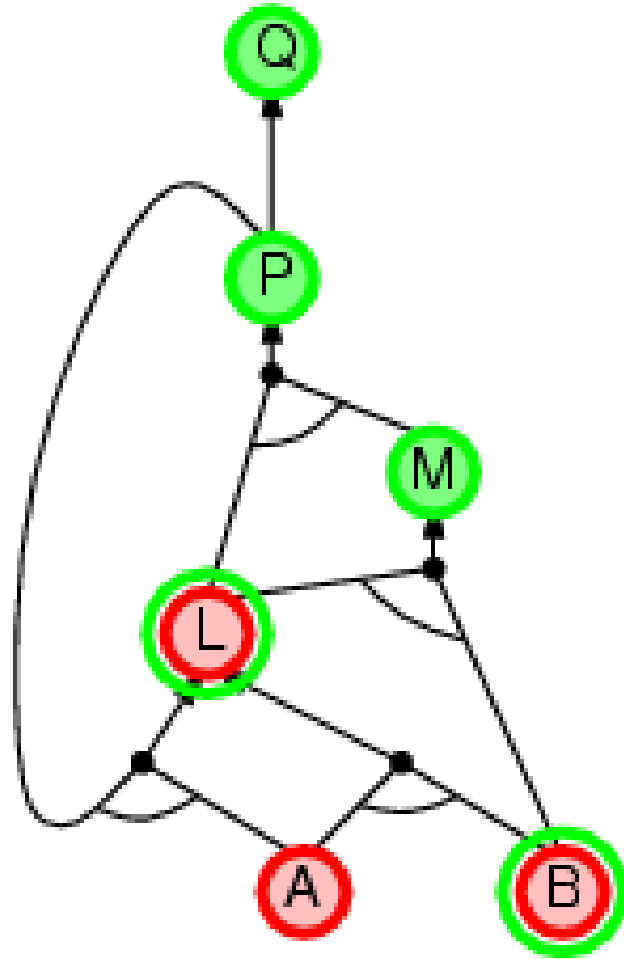
# Cascadeamento para Trás



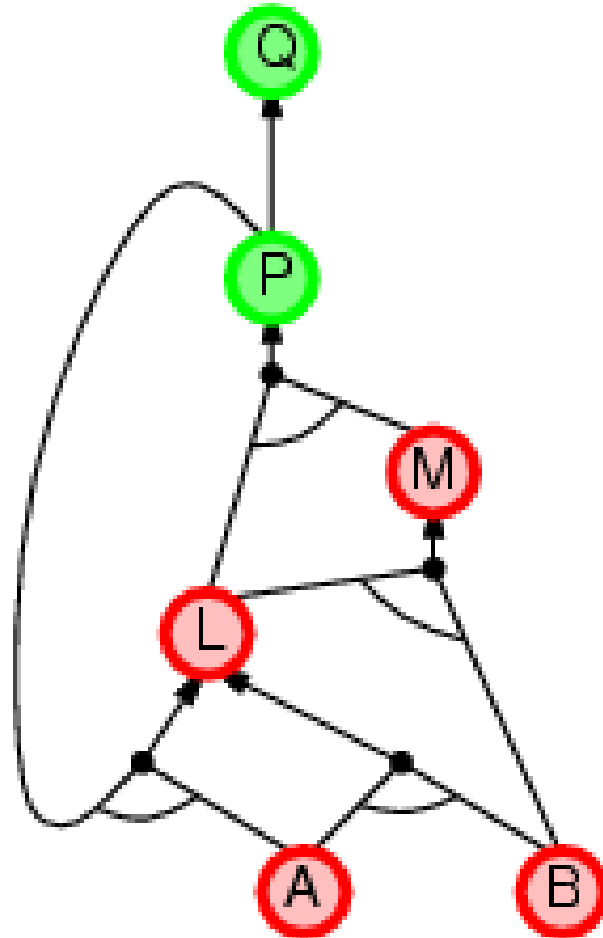
# Cascadeamento para Trás



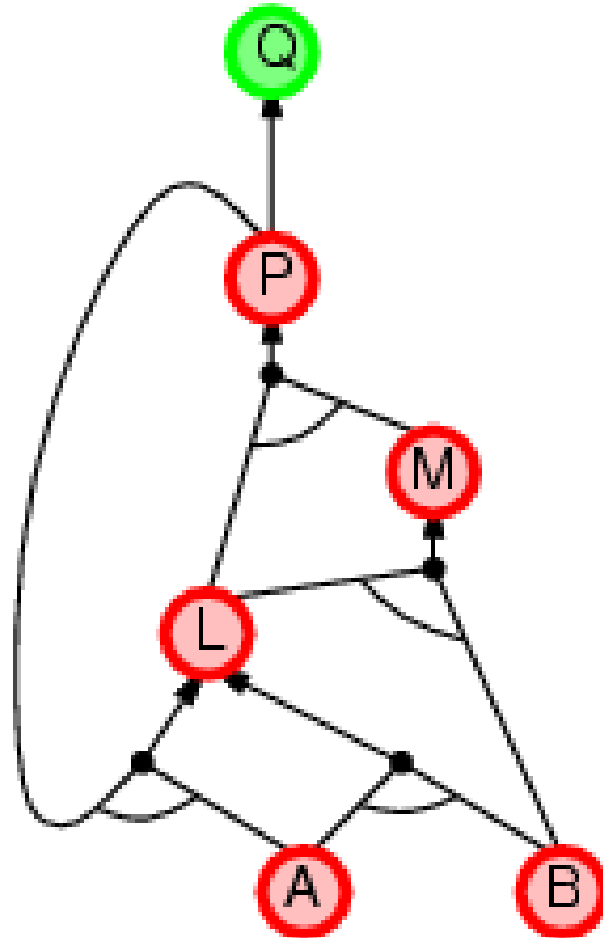
# Cascadeamento para Trás



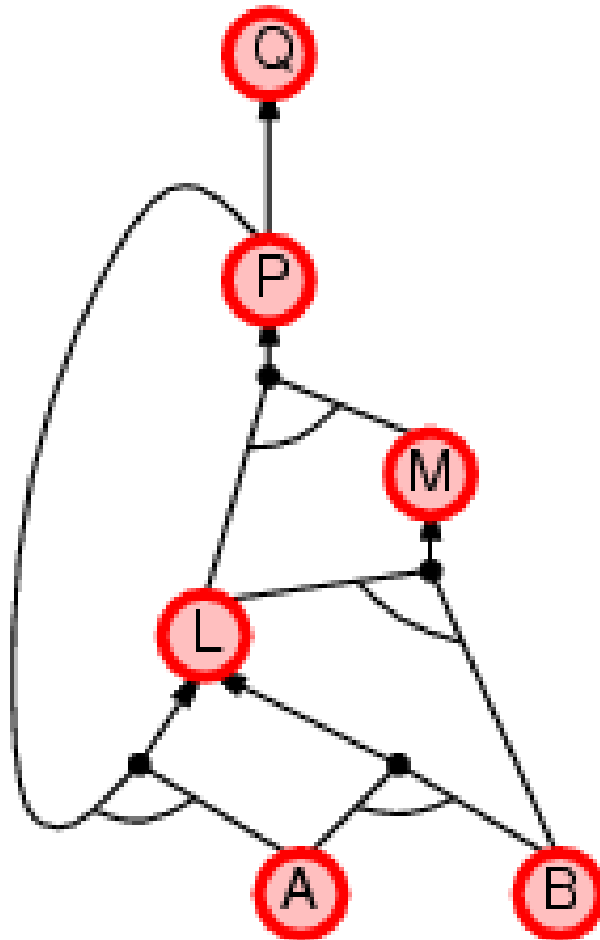
# Cascadeamento para Trás



# Cascadeamento para Trás



# *Cascadeamento para Trás*



# *Cascadeamento para Frente vs Cascadeamento para Trás*

- Cascadeamento para Frente
  - Direcionado pelos dados
    - Derivar conclusões a partir de percepções
    - Pode executar muito trabalho irrelevante para o objetivo
- Cascadeamento para Trás
  - Direcionado pelo Objetivo
    - Apropriado para Resolução de Problemas
    - Executa em tempo linear em relação à KB
      - Pode ser muito menor, pois só toca em informações relevantes

