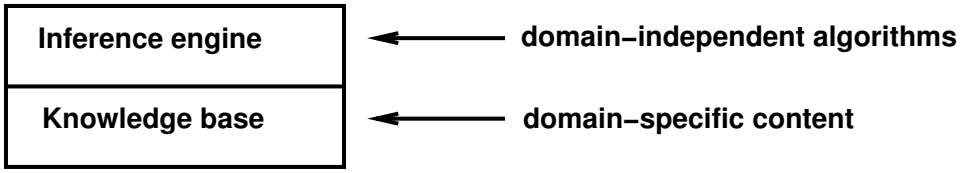
Agentes Lógicos

Ano Lectivo 2010/2011 - 1º Semestre – Adaptado de http://aima.eecs.berkeley.edu/slides-tex/

Resumo

- ♢ Agentes baseados em conhecimento
- ♢ O mundo Wumpus
- ♢ Lógica em geral—modelos e consequência
- ♢ Lógica Proposicional (Booleana)
- ♢ Equivalência, validade, satisfatibilidade
- ♢ Regras de Inferência e demonstração de teoremas
  - encadeamento para a frente (forward chaining)
  - encadeamento para trás (backward chaining)
  - resolução

Bases de Conhecimento(s)



Base de conhecimento = conjunto de frases numa linguagem formal

Aproximação declarativa na construção de um agente (ou outro sistema):

TELL ⇐ informar o sistema do que precisa de saber

Seguidamente, o sistema pode perguntar a si próprio (ASK) o que deve fazer—as respostas obtidas (implicitamente) a partir da KB

Os agentes podem ser analisados quanto ao seu nível de conhecimento

i.e., aquilo que sabem, independentemente da sua implementação

Ou quanto ao seu nível de implementação

i.e., estruturas de dados na KB e algoritmos que as manipulam

Um agente simples baseado em conhecimento

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*

O agente deve ser capaz de:
- Representar estados, acções, etc.
- Incorporar novas percepções
- Actualizar representações internas do mundo
- Deduzir propriedades escondidas do mundo
- Deduzir acções apropriadas

Descrição do mundo do Wumpus (PEAS)

Medida de desempenho

ouro +1000, morte -1000

-1 por passo, -10 por utilizar a seta

Ambiente

Casas adjacentes ao wumpus são malcheirosas

Casas adjacenctes a um poço são ventosas

Brilho sse ouro está na mesma casa

Disparo mata wumpus se estiver defronte dele

Disparar gasta a única seta

Agarrar apanha o ouro da casa

Largar deixa o ouro na mesma casa

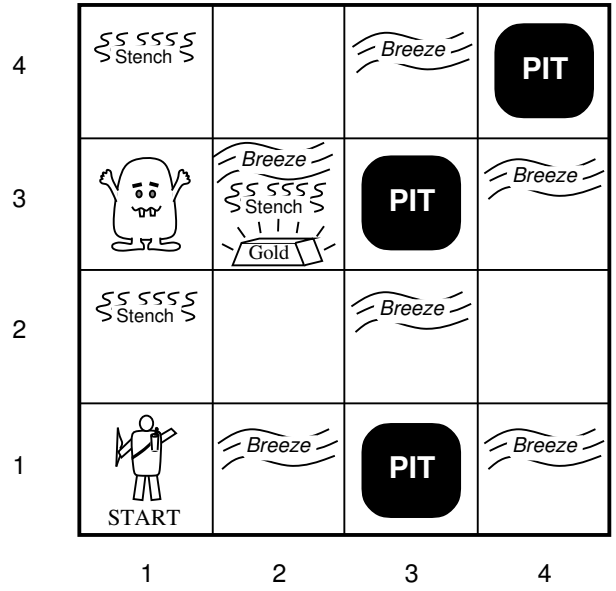
Sensores

Brisa, Brilho, Cheiro, Grito, Batida

Actuadores

Rodar Esquerda, Rodar Direita,

Avançar, Agarrar, Largar, Disparar, Subir



Caracterização do mundo do Wumpus

Observável??

Caracterização do mundo do Wumpus

Observável?? Não—apenas percepções locais

Determinista??

Caracterização do mundo do Wumpus

Observável?? Não—apenas percepções locais

Determinista?? Sim—os resultados especificados exactamente

Episódico??

Caracterização do mundo do Wumpus

Observável?? Não—apenas percepções locais

Determinista?? Sim—os resultados especificados exactamente

Episódico?? Não—sequencial ao nível das acções

Estático??

Caracterização do mundo do Wumpus

Observável?? Não—apenas percepções locais

Determinista?? Sim—os resultados especificados exactamente

Episódico?? Não—sequencial ao nível das acções

Estático?? Sim—Wumpus e poços não se movem

Discreto??

Caracterização do mundo do Wumpus

Observável?? Não—apenas percepções locais

Determinista?? Sim—os resultados especificados exactamente

Episódico?? Não—sequencial ao nível das acções

Estático?? Sim—Wumpus e poços não se movem

Discreto?? Sim

Agente único??

Caracterização do mundo do Wumpus

Observável?? Não—apenas percepções locais

Determinista?? Sim—os resultados especificados exactamente

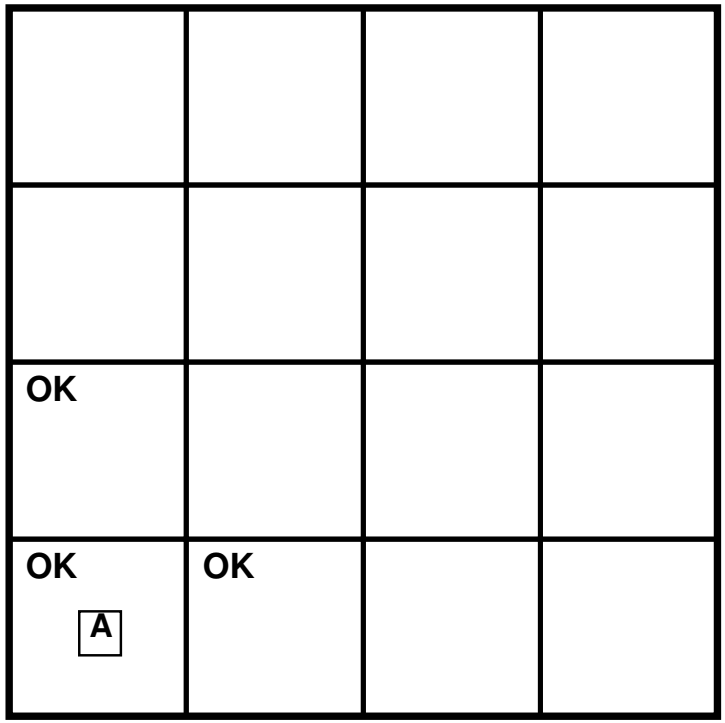
Episódico?? Não—sequencial ao nível das acções

Estático?? Sim—Wumpus e poços não se movem

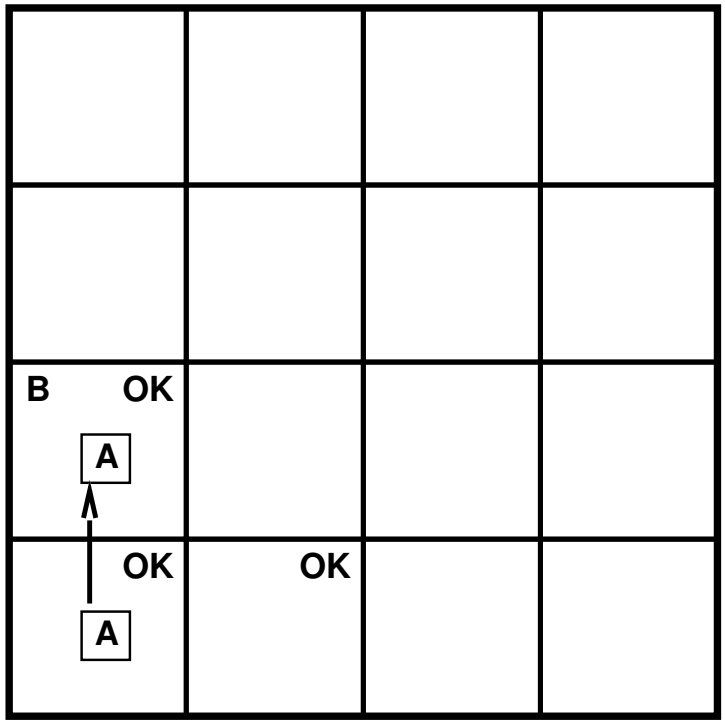
Discreto?? Sim

Agente único?? Sim—Wumpus é basicamente uma propriedade do ambiente

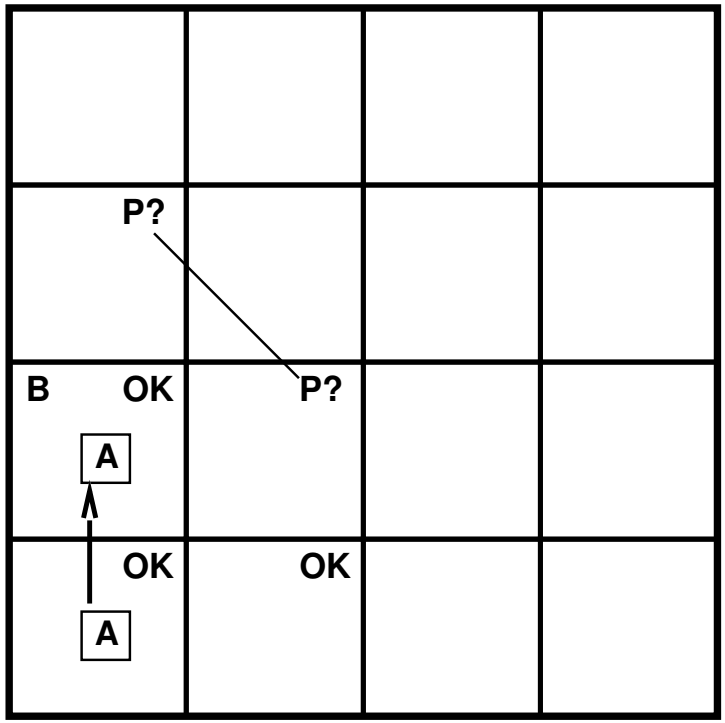
Explorando um mundo do Wumpus



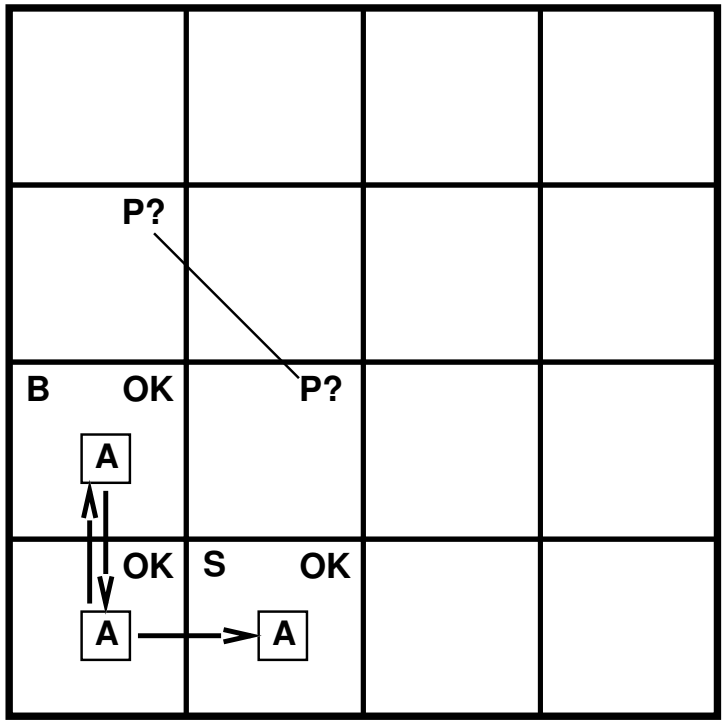
Explorando um mundo do Wumpus



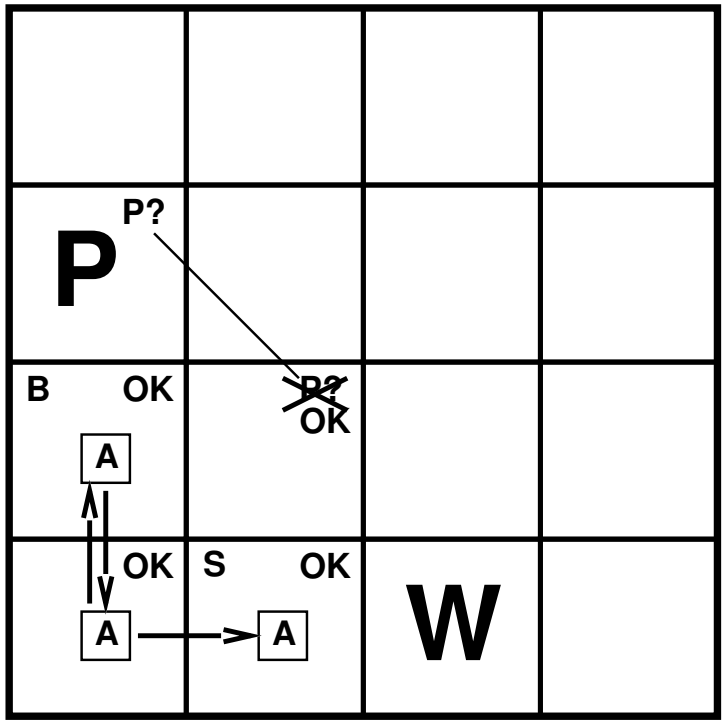
Explorando um mundo do Wumpus



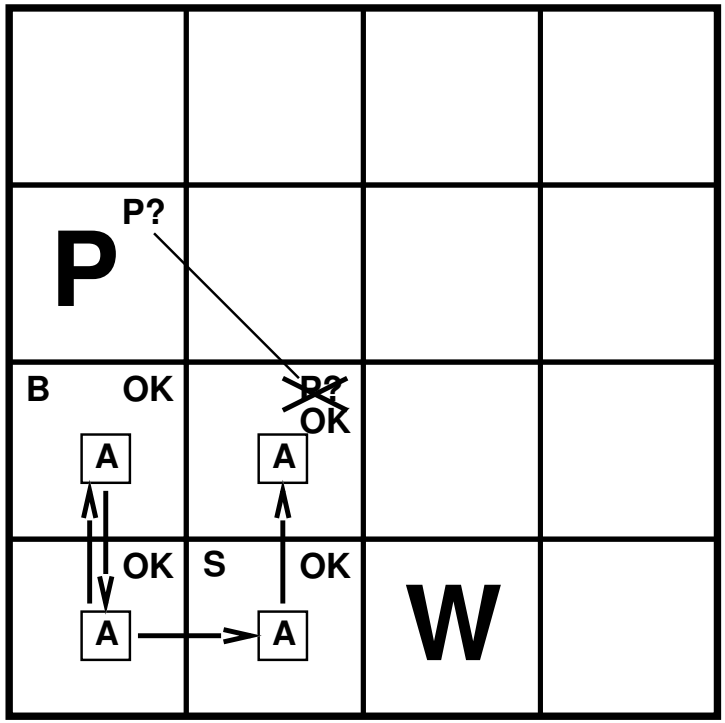
Explorando um mundo do Wumpus



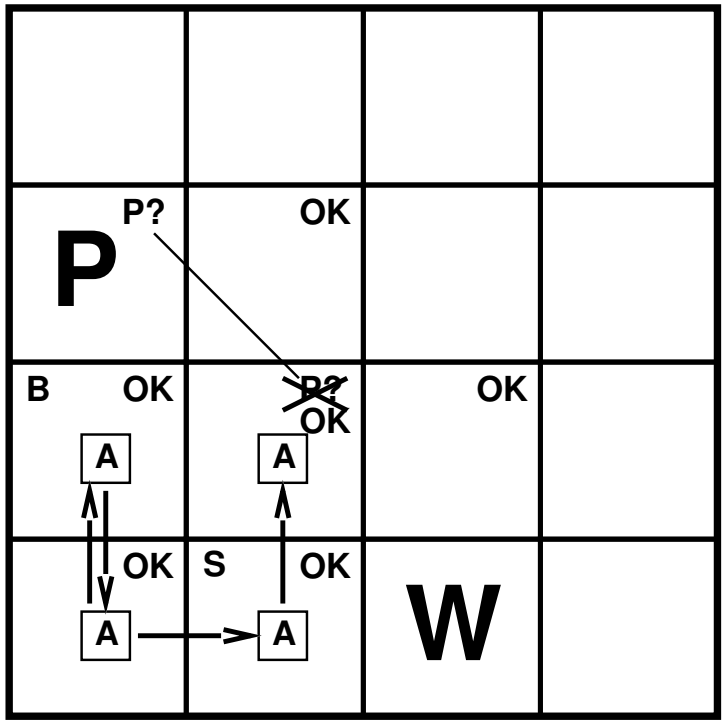
Explorando um mundo do Wumpus



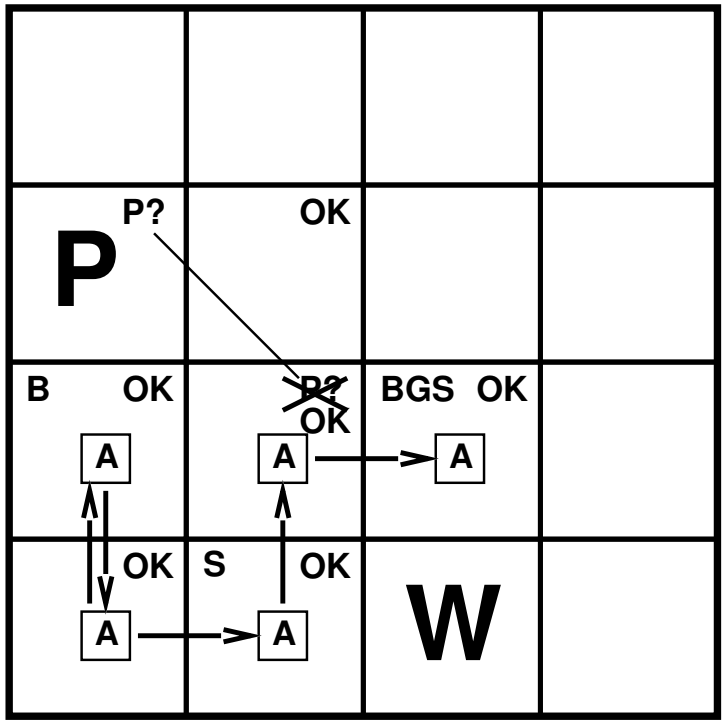
Explorando um mundo do Wumpus



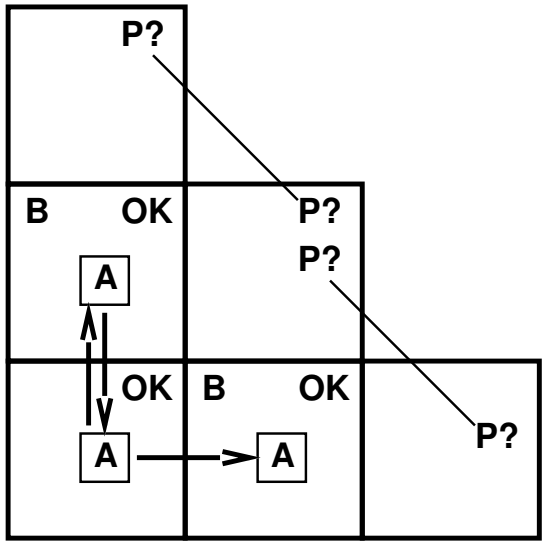
Explorando um mundo do Wumpus



Explorando um mundo do Wumpus



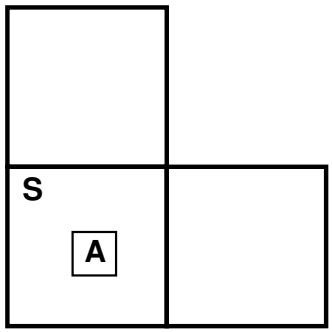
Outras situações difíceis



Vento em (1,2) e (2,1)

$\Rightarrow$ não existem acções seguras

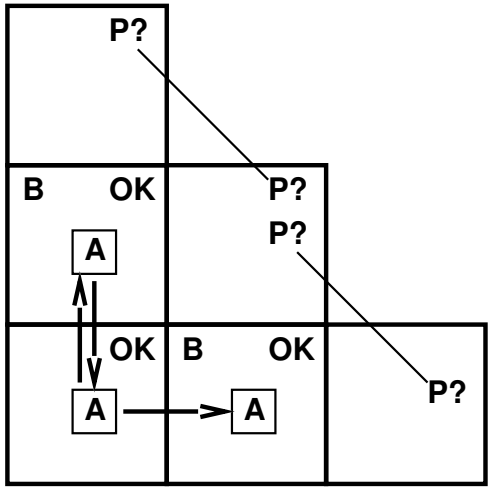
Assumindo poços uniformemente distribuídos, (2,2) tem poço c/ prob 0.86, vs. 0.31



Cheiro em (1,1)

$\Rightarrow$ não se pode mover

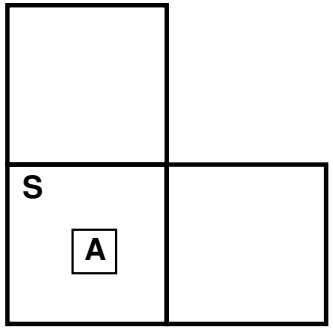
Outras situações difíceis



Vento em (1,2) e (2,1)

$\Rightarrow$ não existem acções seguras

Assumindo poços uniformemente distribuídos, (2,2) tem poço c/ prob 0.86, vs. 0.31



Cheiro em (1,1)

$\Rightarrow$ não se pode mover

Pode recorrer a estratégia de coerção:

disparar em frente

wumpus estava lá $\Rightarrow$ morto $\Rightarrow$ seguro

wumpus não estava lá $\Rightarrow$ seguro

Noções de Lógica

Lógicas são linguagens formais para de representação de informação
que permitem a extracção de conclusões

Sintaxe define as frases permitidas da linguagem

Semântica define o "significado" das frases;
i.e., define verdade de uma frase num mundo

E.g., a linguagem da aritmética

$x + 2 \geq y$ é uma frase (proposição); $x2 + y >$ não é uma frase

$x + 2 \geq y$ é verdade sse o número $x + 2$ não é menor do que o número y

$x + 2 \geq y$ é verdade num mundo em que $x = 7,\ y = 1$
$x + 2 \geq y$ é falso num mundo em que $x = 0,\ y = 6$

Conclusão Lógica

Conclusão significa que algo *segue* de outrem:

$$KB \models \alpha$$

Da base de conhecimento KB conclui-se a frase α α
se e somente se
α é verdade em todos os mundos em que KB é verdade

Da base de conhecimento KB contendo "o Boavista ganhou" e "o Porto ganhou" conclui-se (entails) "o Boavista ganhou ou o Porto ganhou"

E.g., De $x + y = 4$ conclui-se $4 = y + x$

Conclusão Lógica é uma relação entre frases (i.e., *sintaxe*)
que se encontra baseada na *semântica*

Nota: o cérebro processa *sintaxe* (de algum tipo)

Modelos

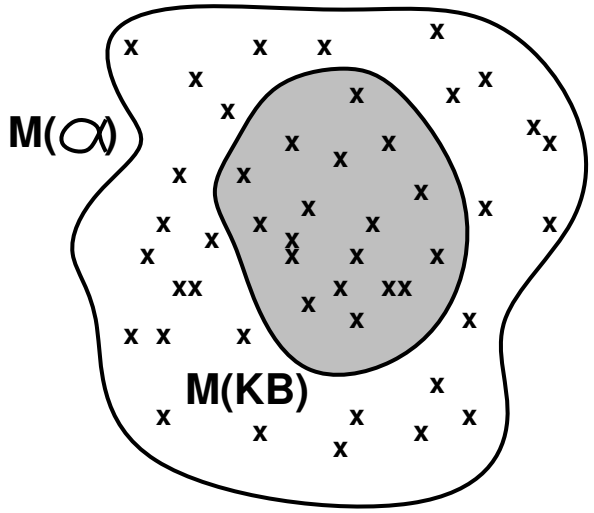
Os lógicos pensam normalmente em termos de modelos, que são mundos formalmente estruturadas relativamente aos quais se pode avaliar a veracidade

Diz-se que m é modelo de uma proposição α se α é verdade em m

$M(\alpha)$ é o conjunto de todos os modelos de α

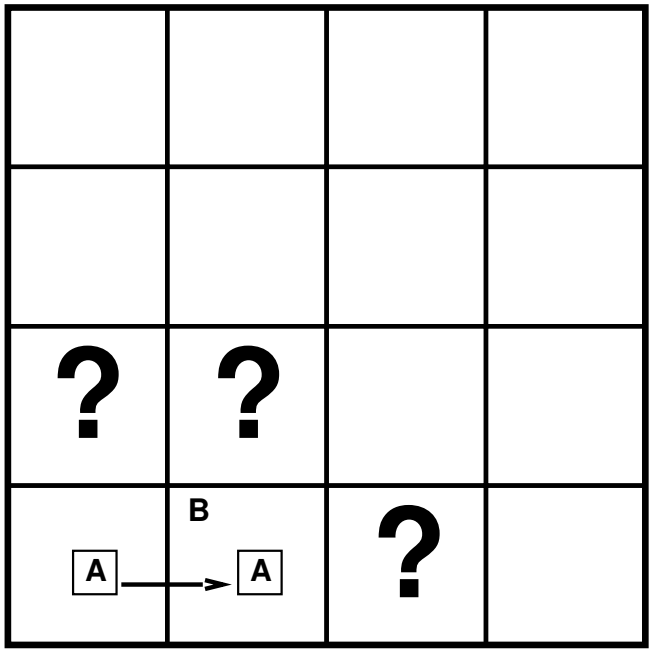
Logo $KB \models \alpha$ se e somente se $M(KB) \subseteq M(\alpha)$

E.g. $KB =$ Sporting ganhou e Benfica ganhou
$\alpha =$ Benfica ganhou



Conclusões no mundo do wumpus

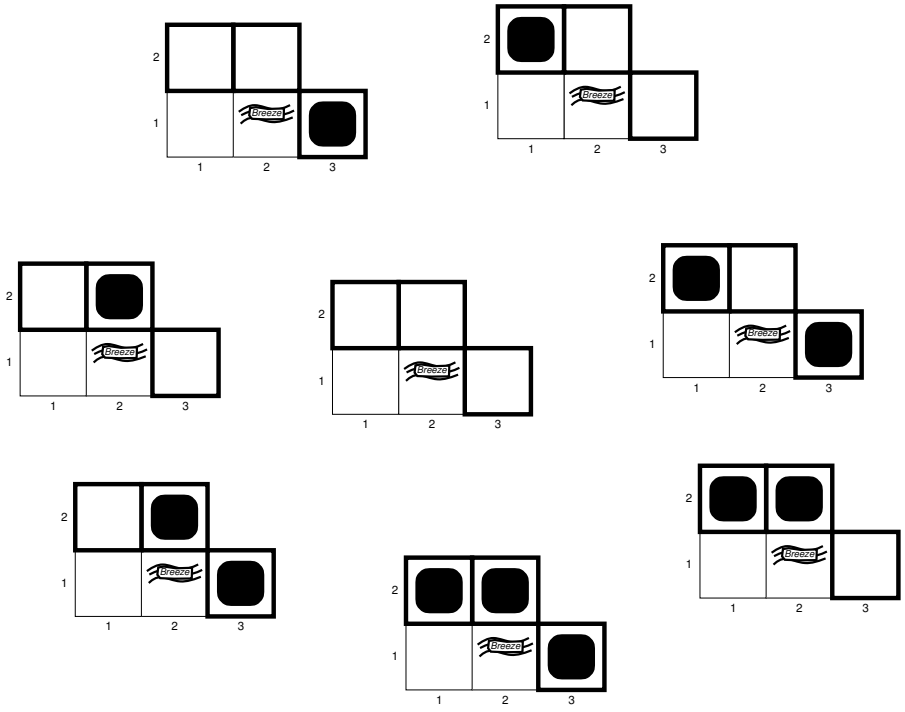
Situação após detectar nada em [1,1],
deslocação para a direita, brisa em [2,1]



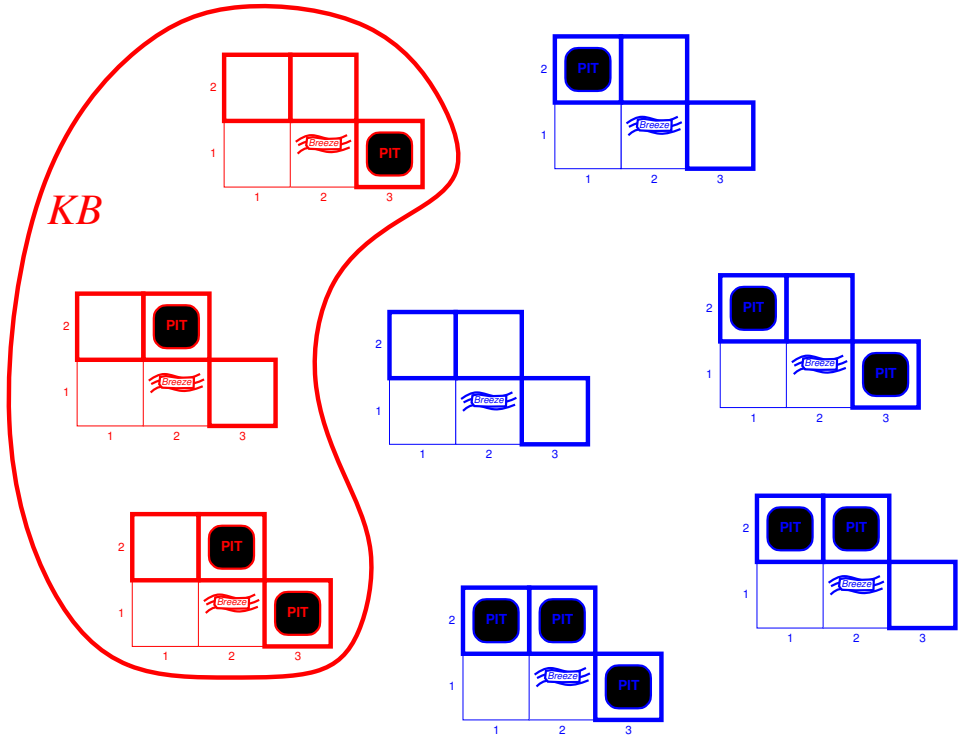
Considerar modelos possíveis para ?s
assumindo apenas poços

3 escolhas Booleanas $\Rightarrow$ 8 mundos possíveis

Modelos Wumpus

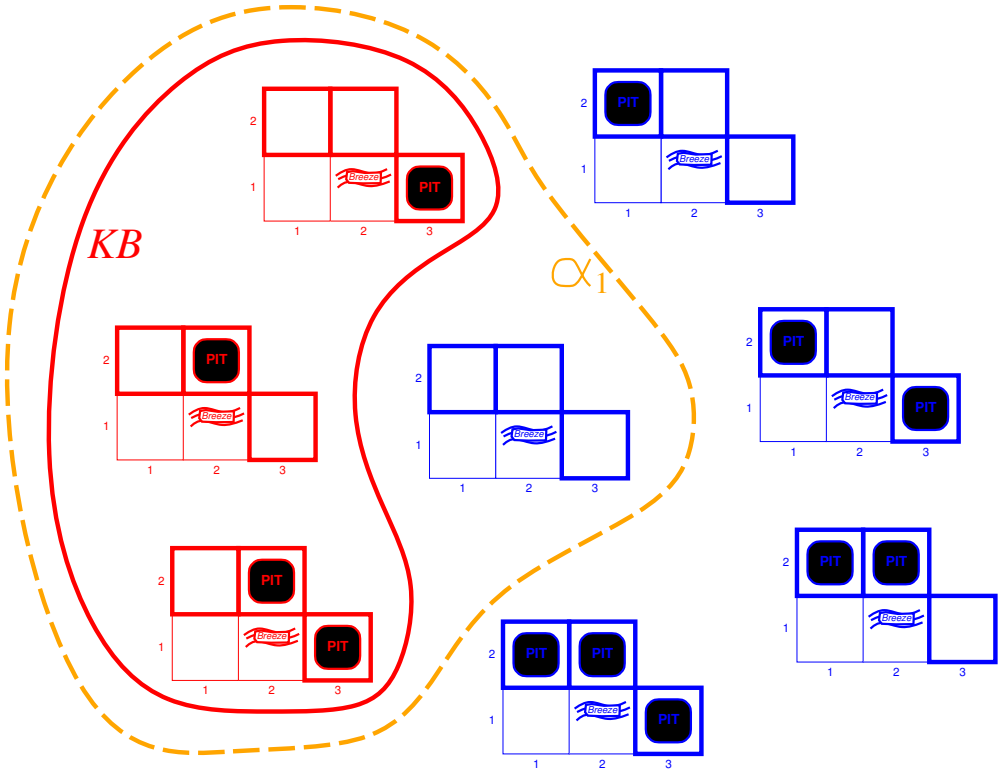


Modelos Wumpus



KB = regras do mundo-wumpus + observações

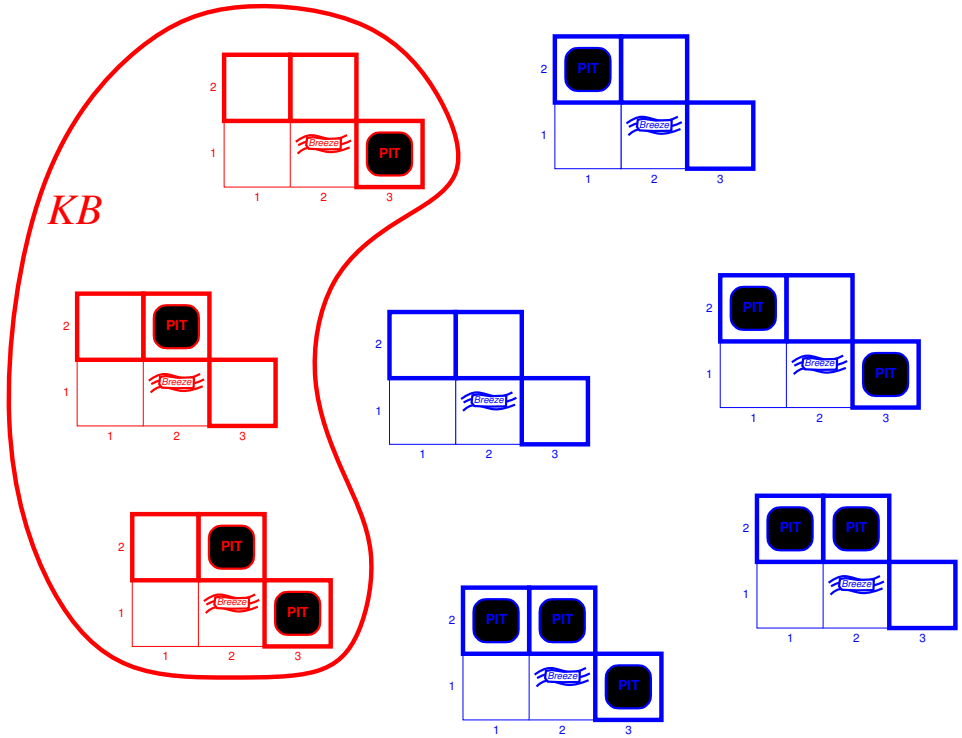
Modelos Wumpus



KB = regras do mundo-wumpus + observações

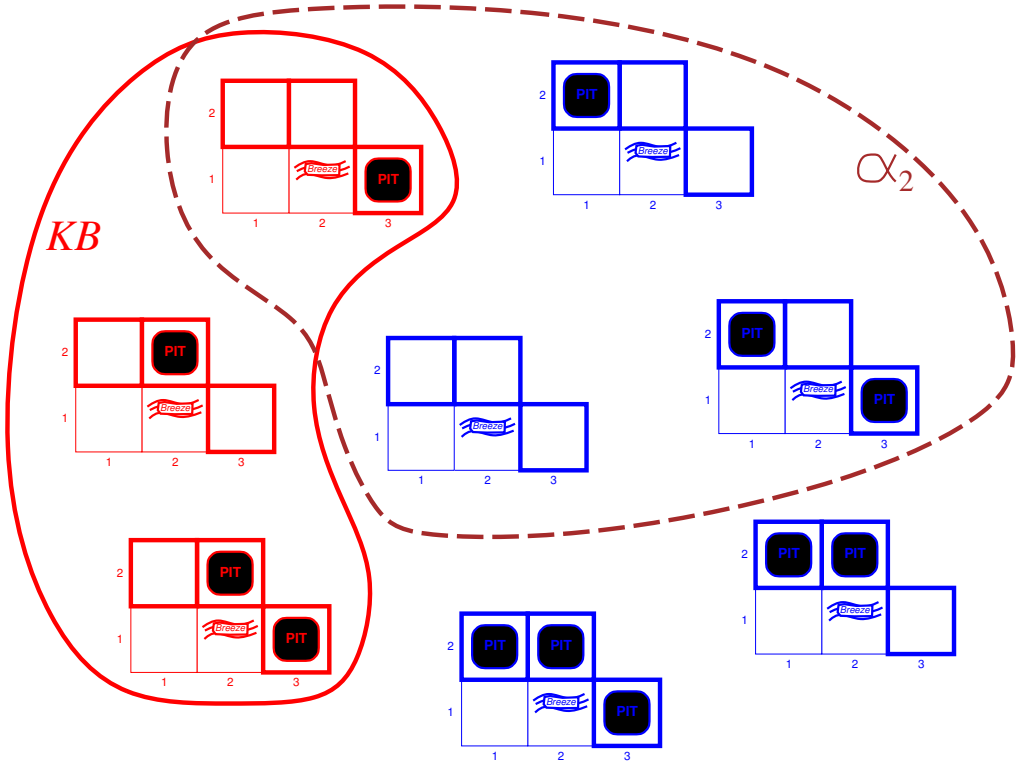
α_1 = "[1,2] é seguro", $KB \models \alpha_1$, demonstrado por verificação de modelos

Modelos Wumpus



KB = regras do mundo-wumpus + observações

Modelos Wumpus



$KB =$ regras do mundo-wumpus + observações

$\alpha_2 =$ "[2,2] é seguro", $KB \not\models \alpha_2$

Inferência

$KB \vdash_i \alpha =$ proposição α pode ser derivada de KB pelo procedimento i

Consequências de KB são o palheiro; α é a agulha.
Conclusão Lógica = agulha no palheiro; inferência = encontrá-la

Fidedigno: i é fidedigno (ou sólido) se
quando $KB \vdash_i \alpha$, então também é verdade que $KB \models \alpha$

Completo: i é completo se
quando $KB \models \alpha$, então também é verdade que $KB \vdash_i \alpha$

Antecipação: definiremos uma lógica (lógica de primeira ordem) que é suficientemente expressiva para dizer quase tudo o que é interessante, e para a qual existe um procedimento de inferência fidedigno e completo.

Ou seja, o procedimento responderá a qualquer questão que se segue daquilo que é conhecido pela KB.

Lógica Proposicional: Sintaxe

A lógica proposicional é a lógica mais simples—ilustra os conceitos básicos

Os símbolos (ou variáveis) proposicionais P_1, P_2 etc são proposições (frases)

Se S é uma proposição, $\neg S$ é uma proposição (negação)

Se S_1 e S_2 são proposições, $(S_1 \wedge S_2)$ é uma proposição (conjunção)

Se S_1 e S_2 são proposições, $(S_1 \vee S_2)$ é uma proposição (disjunção)

Se S_1 e S_2 são proposições, $(S_1 \Rightarrow S_2)$ é uma proposição (implicação)

Se S_1 e S_2 são proposições, $(S_1 \Leftrightarrow S_2)$ é uma proposição (bicondicional)

Lógica Proposicional: Semântica

Cada modelo atribui verdadeiro/falso a cada símbolo proposicional

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

(Com estes símbolos, 8 modelos possíveis, podem ser enumerados automaticamente.)

Regras para avaliar veracidade relativamente a um modelo m:

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

Um processo recursivo simples avalia uma proposição arbitrária, e.g.,
$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = verd \wedge (falso \vee verd) = verd \wedge verd = verd$

Tabelas de verdade para os conectivos

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

Proposições no mundo do Wumpus

Seja $P_{i,j}$ verdade se existir um poço em $[i, j]$.
Seja $B_{i,j}$ verdade se se sentir uma brisa em $[i, j]$.

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

"Poços causam brisa em casas adjacentes"

Proposições no mundo do Wumpus

Seja $P_{i,j}$ verdade se existir um poço em $[i, j]$.
Seja $B_{i,j}$ verdade se se sentir uma brisa em $[i, j]$.

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

"Poços causam brisa em casas adjacentes"

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

"Uma casa é ventosa *sse* existir um poço adjacente"

Inferência através de tabelas de verdade

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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

Enumeração em profundidade primeiro dos modelos todos é sólido e completo

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($P, false, model$)

Para n símbolos, complexidade temporal de $O(2^n)$ e espacial $O(n)$; problema é coNP-completo

Equivalência Lógica

Duas proposições são logicamente equivalentes sse forem verdadeiras nos mesmos modelos: $\alpha \equiv \beta$ sse $\alpha \models \beta$ e $\beta \models \alpha$

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{comutatividade de } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{comutatividade de } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associatividade de } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associatividade de } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{eliminação da dupla negação} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposição} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{eliminação da implicação} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{eliminação do bicondicional} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributividade de } \wedge \text{ sobre } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributividade de } \vee \text{ sobre } \wedge
\end{aligned}
$$

Validade e satisfatibilidade

Uma proposição é válida se for verdadeira em *todos* os modelos,
e.g., $True$, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

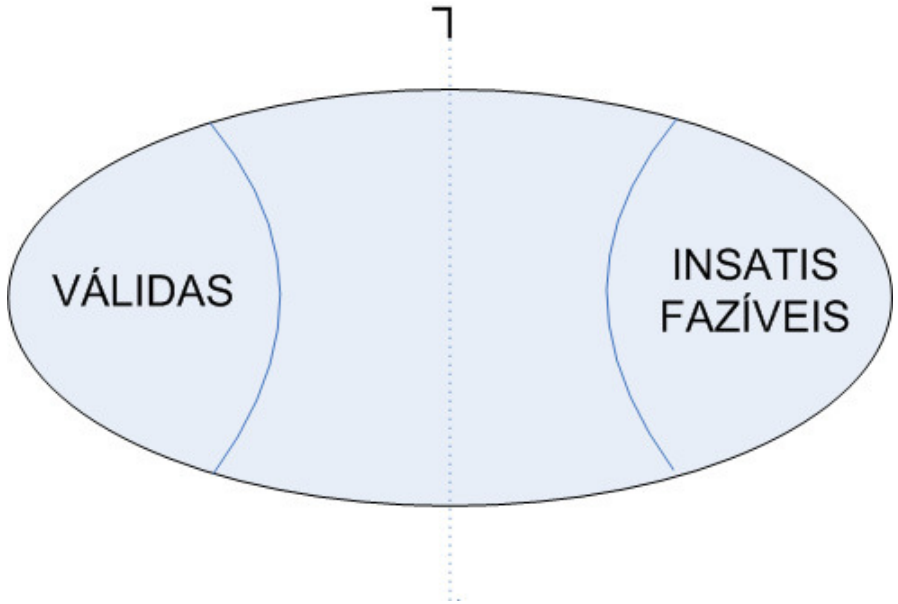
Validade está relacionado com consequência através do Teorema da Dedução:
$KB \models \alpha$ se e somente se $(KB \Rightarrow \alpha)$ é válida

Uma proposição é satisfazível se é verdadeira em algum modelo
e.g., $A \vee B$, C

Uma proposição é insatisfazível se for verdadeira em nenhum modelo
e.g., $A \wedge \neg A$

Insatisfatibilidade relaciona-se com consequência através de:
$KB \models \alpha$ se e somente se $(KB \wedge \neg\alpha)$ é insatisfazível
i.e., demonstrar α por *reductio ad absurdum*

Geografia das expressões Booleanas



Eixo de simetria = negação

Métodos de Prova

Os métodos de prova agrupam-se em dois tipos:

Aplicação de regras de inferência

- Geração legítima (sólida) de novas proposições a partir de antigas
- Prova = uma sequência de aplicação de regras de inferência

  Regras de inferência podem ser operadores em algoritmos de procura
- Habitualmente obrigam à tradução das frases para uma forma normal

Verificação de modelos

enumeração por tabelas de verdade (sempre exponencial em n)

melhoramentos ao retrocesso, e.g., Davis–Putnam–Logemann–Loveland

procura heurística em espaço de modelos (sólido mas incompleto)

e.g., algoritmos trepa-colinas com min-conflitos

Encadeamento para a frente e para trás

Forma de Horn (restrita)

KB = *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 a Forma de Horn): completa para KBs de Horn

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

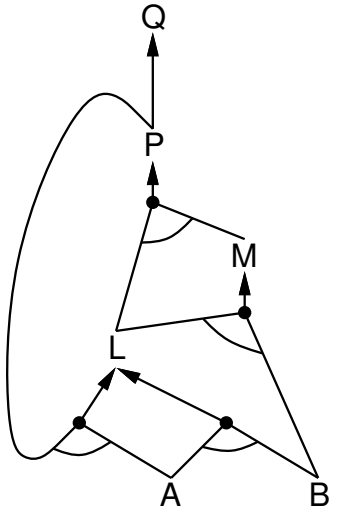
Pode ser utilizada com encadeamento para a frente ou para trás.

Estes algoritmos são muito naturais e executam em tempo *linear*

Encadeamento para a frente

Ideia: disparar toda a regra cujas premissas estão satisfeitas na KB, adicionar a sua conclusão à KB, até se chegar à pergunta

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



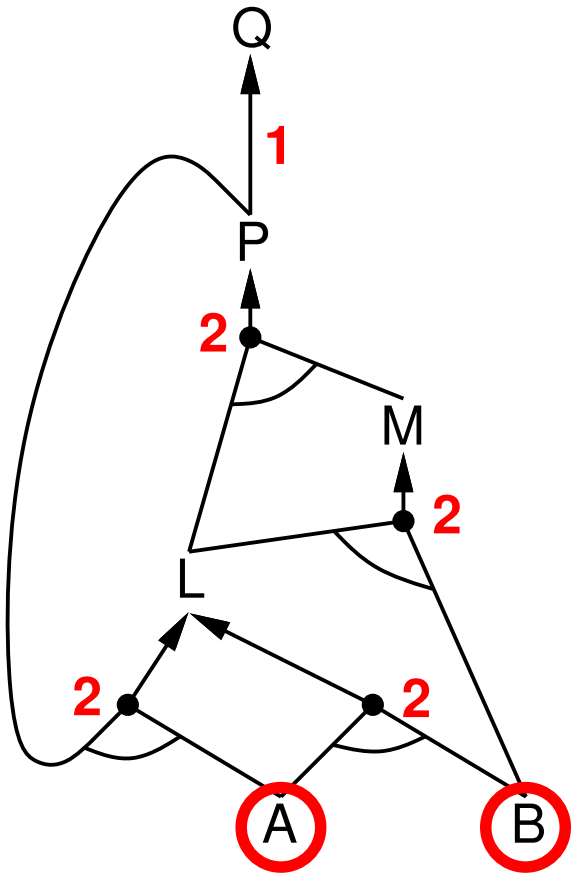
Algoritmo de encadeamento para a frente

function PL-FC-ENTAILS?(KB, q) **returns** *true* or *false*

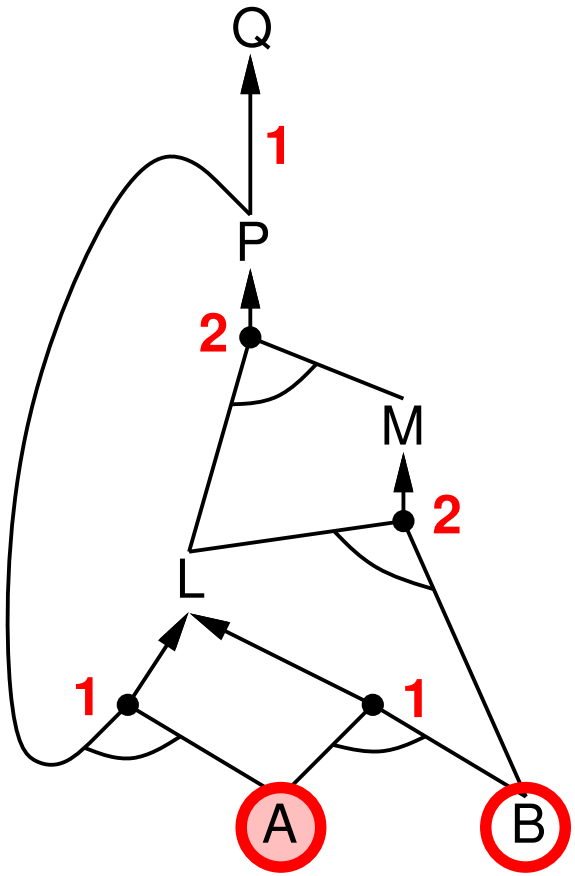
local variables: *count*, a table, indexed by clause, initially the number of premises
inferred, a table, indexed by symbol, each entry initially *false*
agenda, a list of symbols, initially the symbols known to be true

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*

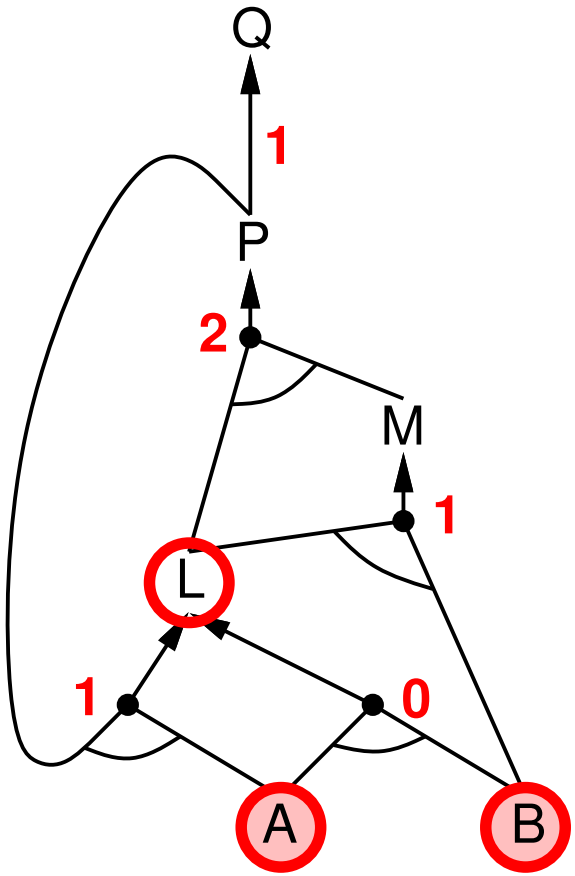
Exemplo de encadeamento para a frente



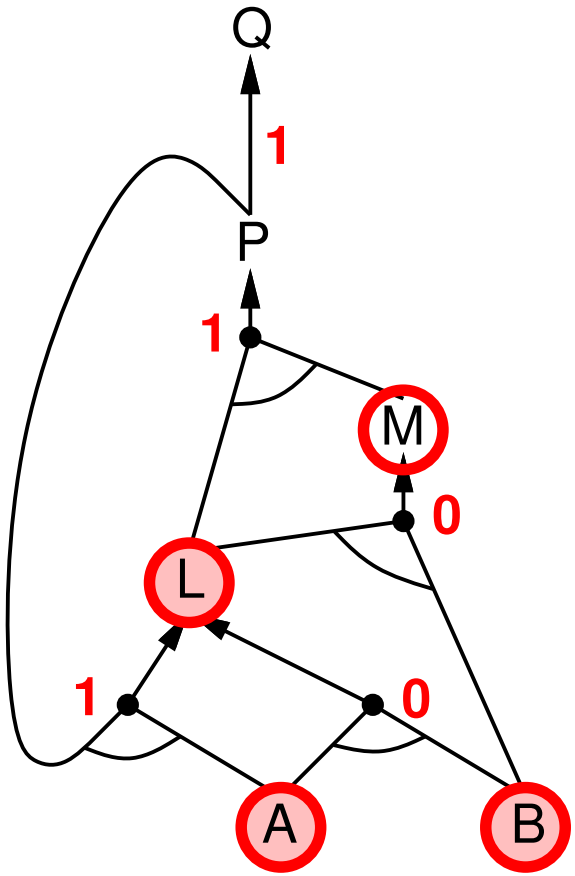
Exemplo de encadeamento para a frente



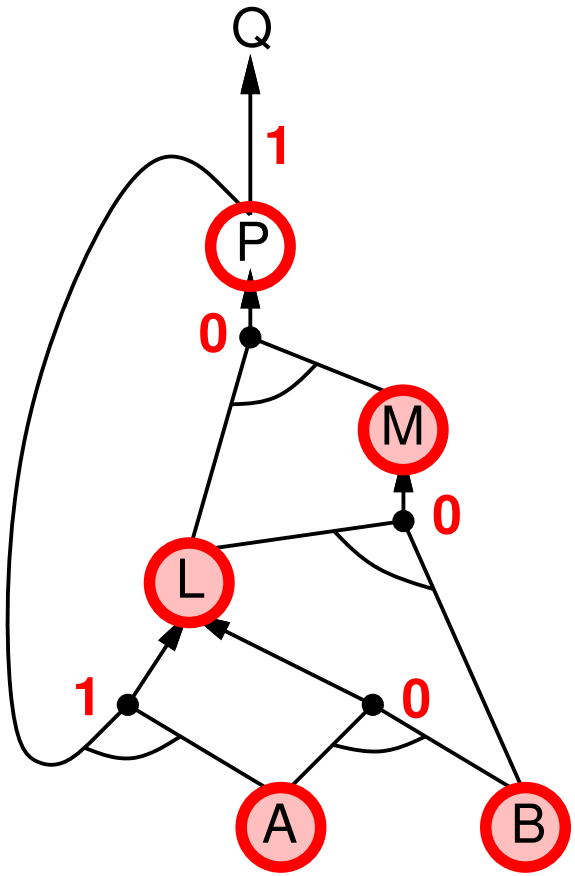
Exemplo de encadeamento para a frente



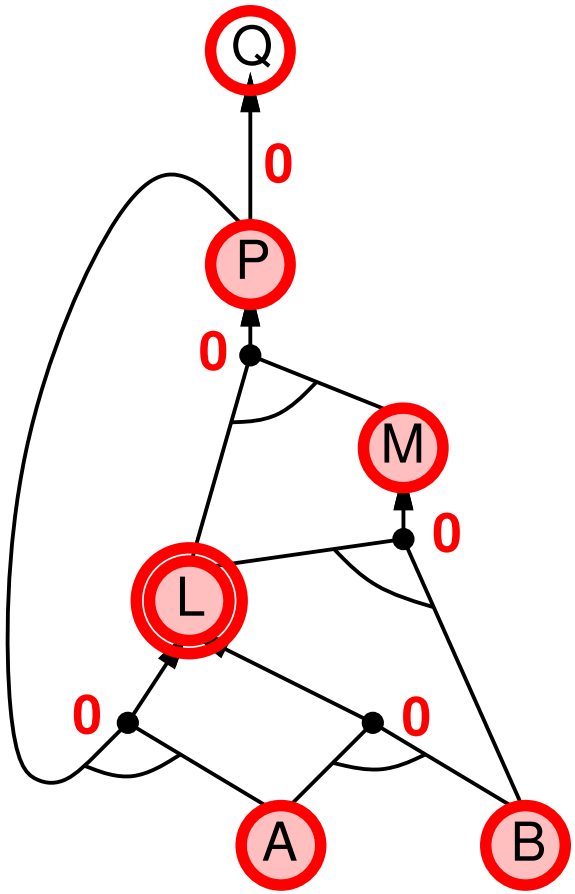
Exemplo de encadeamento para a frente



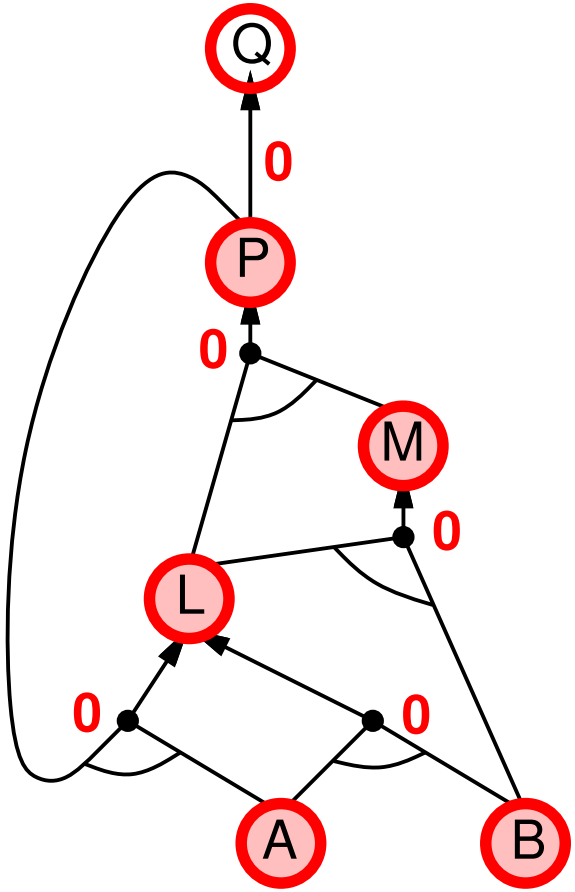
Exemplo de encadeamento para a frente



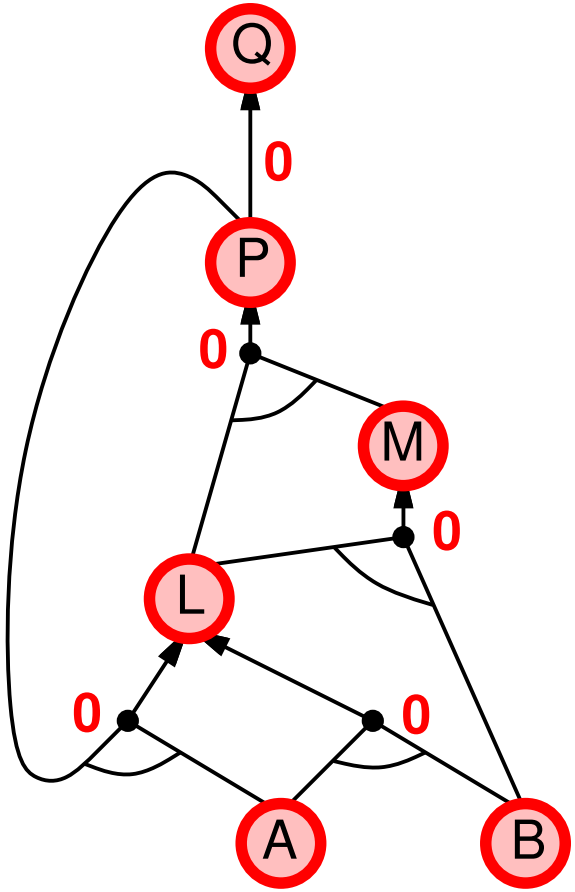
Exemplo de encadeamento para a frente



Exemplo de encadeamento para a frente



Exemplo de encadeamento para a frente



Demonstração de completude

EF deriva toda a proposição atómica que é concluída a partir de KB

1. EF atinge um ponto fixo em que não são derivadas novas proposições atómicas

2. Considere-se o estado final como um modelo m, atribuindo verdadeiro/falso aos símbolos

3. Toda a cláusula em KB inicial é verdadeira em m
   Demo: Suponha-se que a cláusula $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ é falsa em m
   Logo $a_1 \wedge \ldots \wedge a_k$ é verdadeiro em m e b é falso em m
   Logo o algoritmo não atingiu um ponto fixo!

4. Portanto m é modelo de KB

5. Se $KB \models q$, q é verdadeiro em *todo* o modelo de KB, incluindo m

Encadeamento para trás

Ideia: andar ao contrário a partir da pergunta q:

para provar q por ET,

verificar se q já é sabido, ou

provar por ET todas as premissas de alguma regra concluindo q

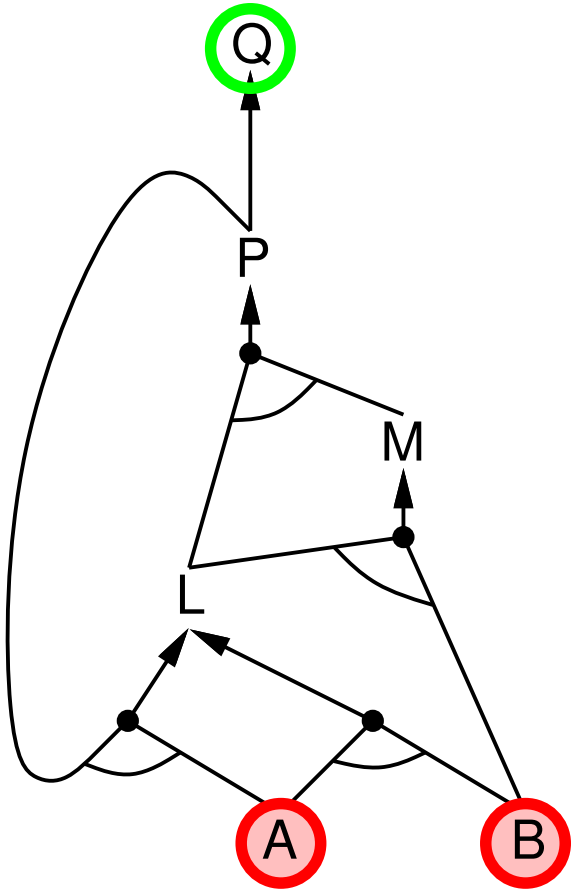
Evitar ciclos: verificar se um novo sub-objectivo já se encontra na pilha de objectivos

Evitar trabalho repetido: verificar se o novo sub-objectivo

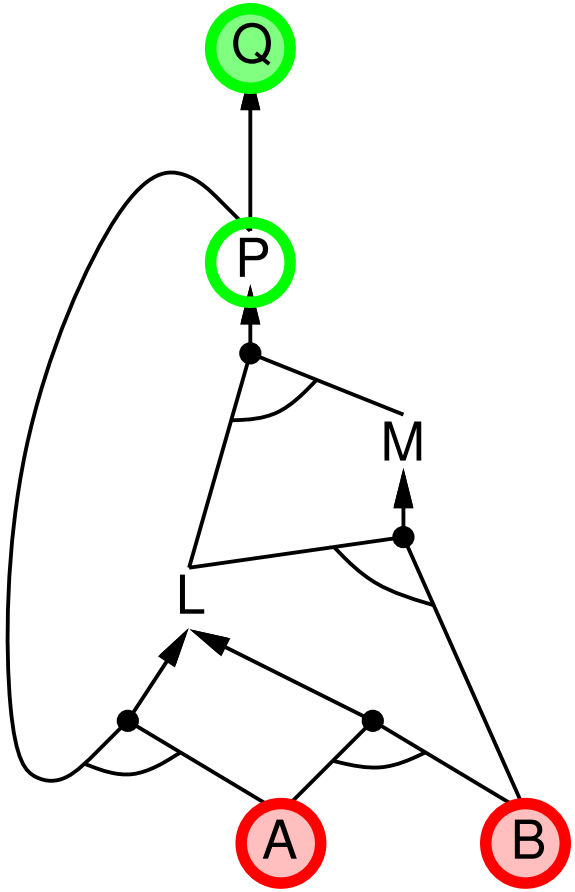
1) já foi demonstrado verdadeiro, ou

2) já falhou

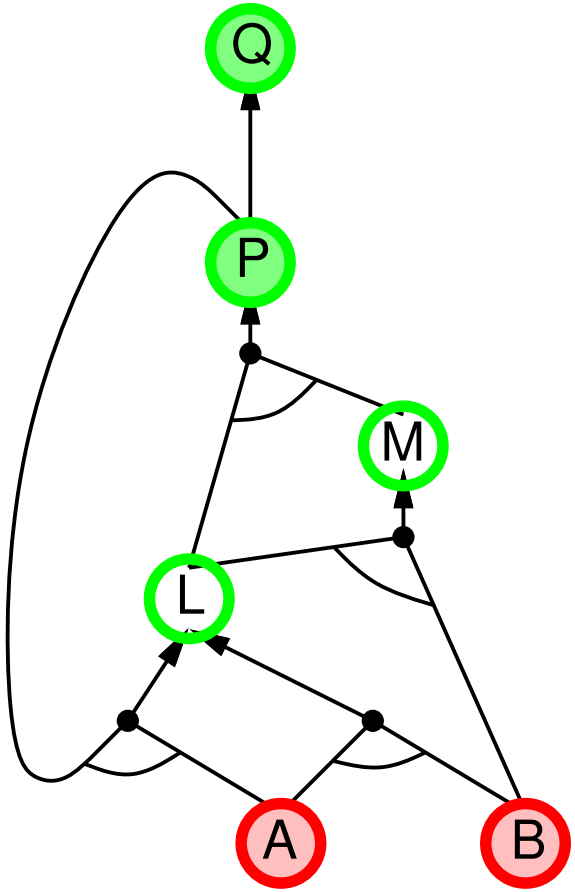
Exemplo de encadeamento para trás



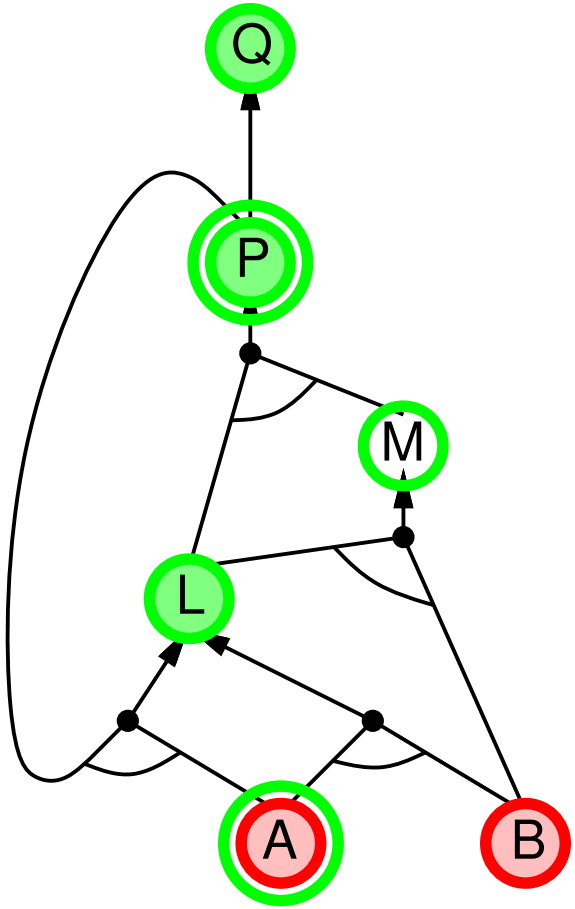
Exemplo de encadeamento para trás



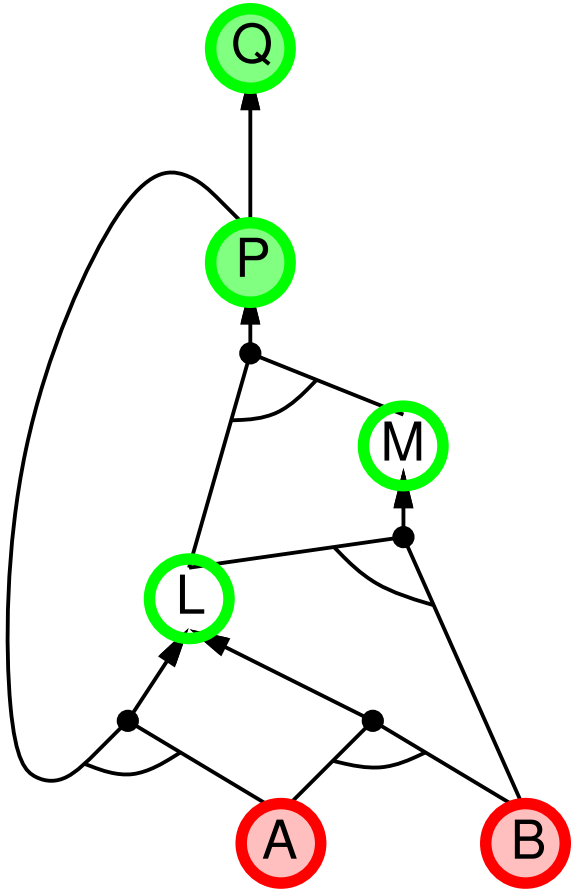
Exemplo de encadeamento para trás



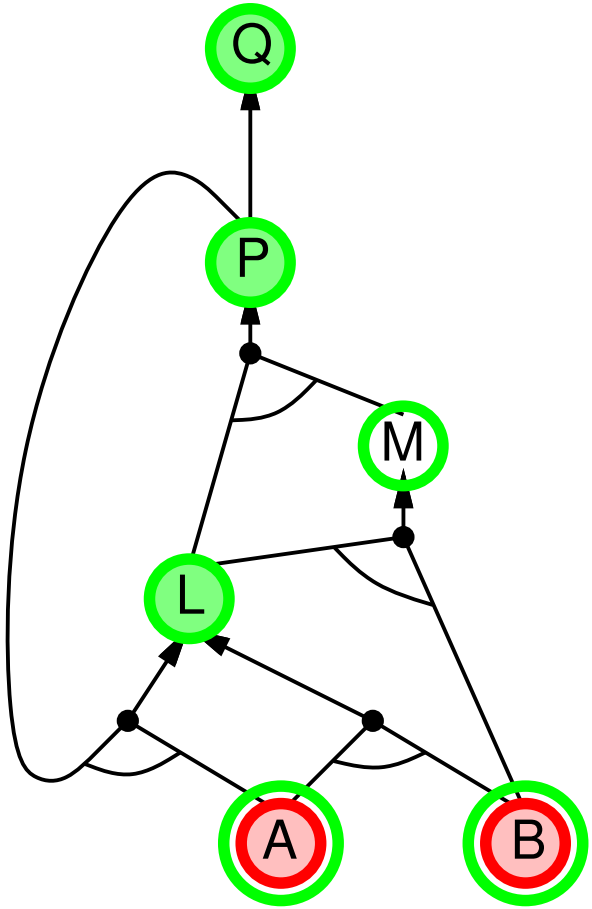
Exemplo de encadeamento para trás



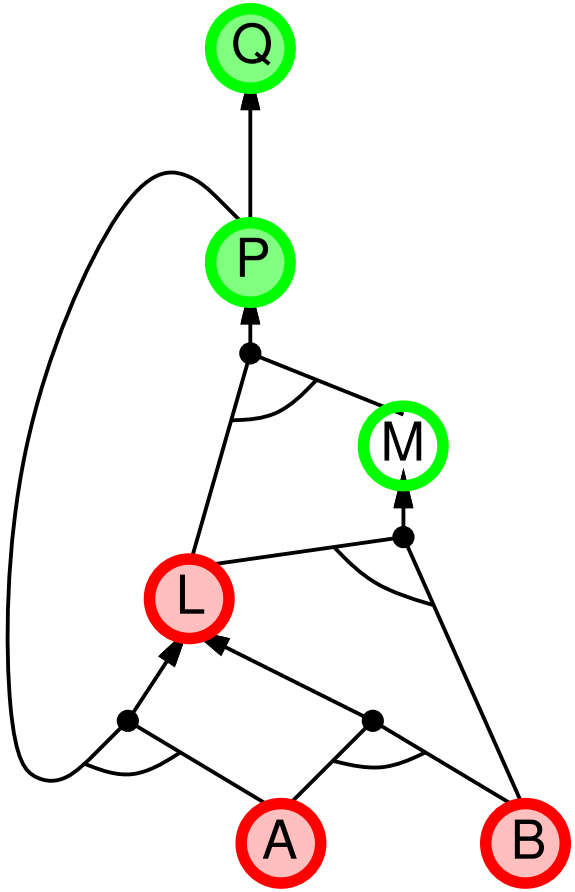
Exemplo de encadeamento para trás



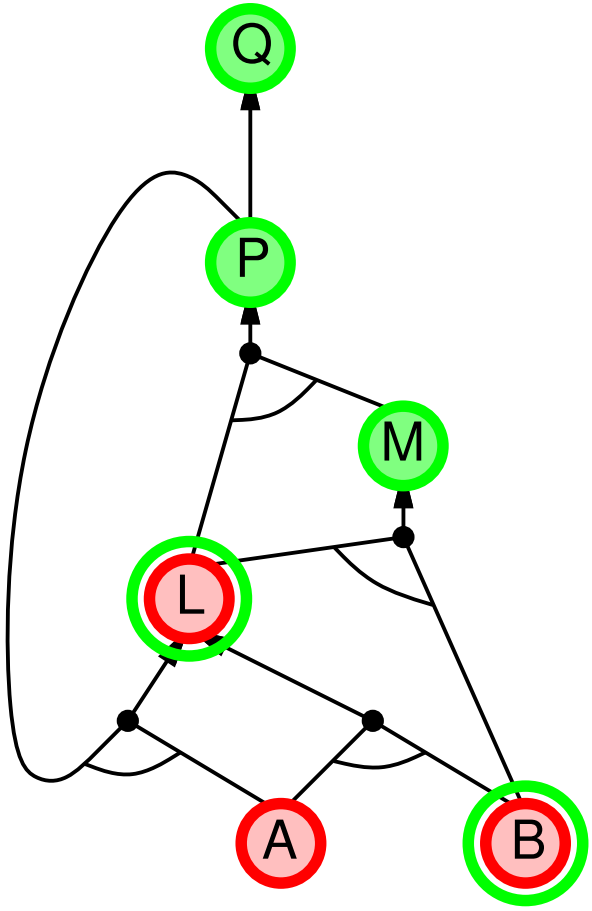
Exemplo de encadeamento para trás



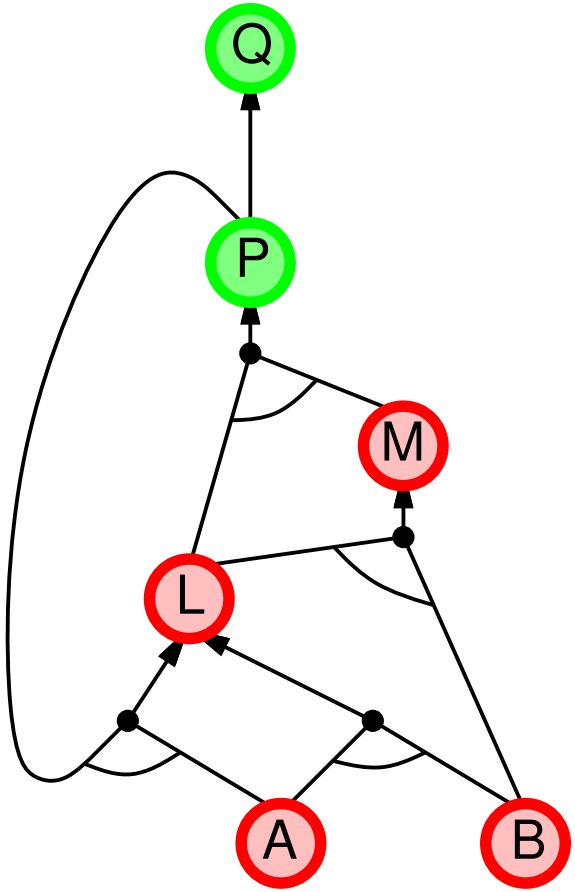
Exemplo de encadeamento para trás



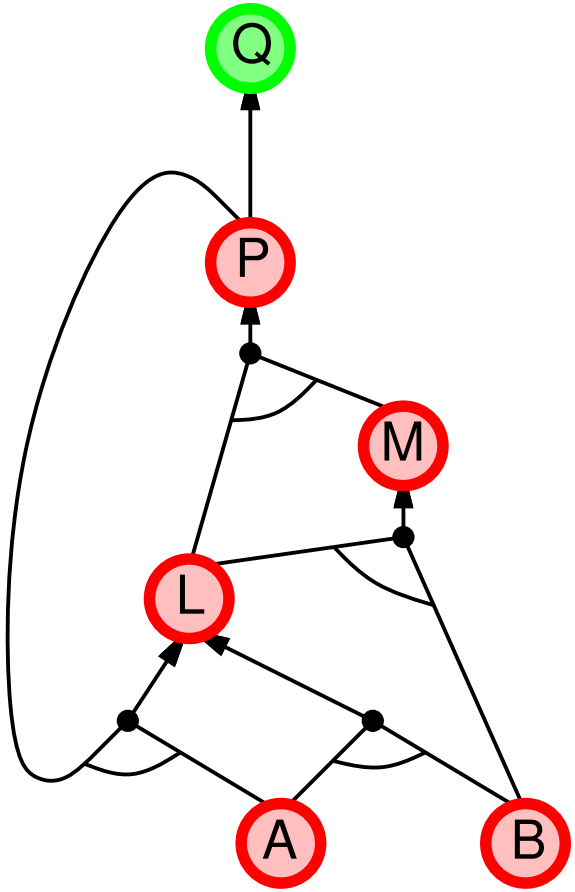
Exemplo de encadeamento para trás



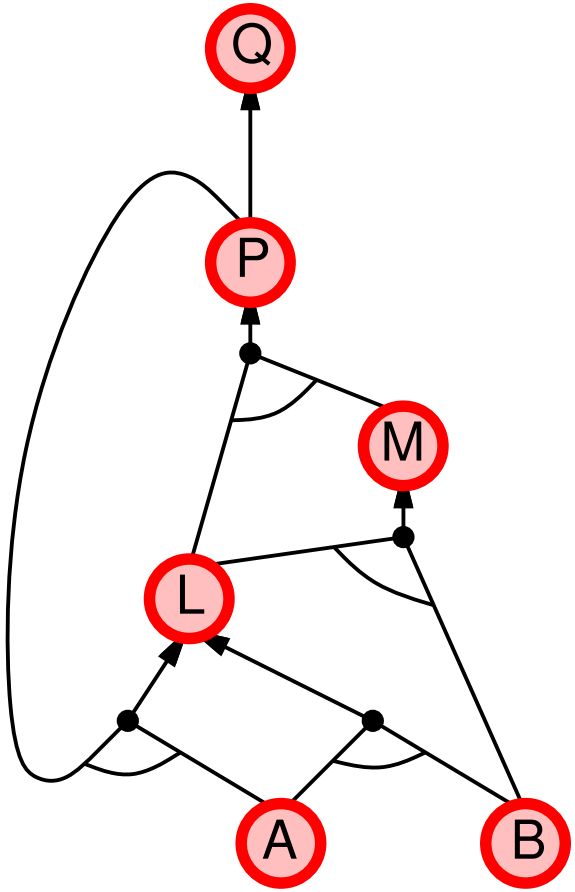
Exemplo de encadeamento para trás



Exemplo de encadeamento para trás



Exemplo de encadeamento para trás



Encadeamento para a frente vs. para trás

EF é guiado pelos dados, cf. processamento automático, inconsciente, e.g., reconhecimento de objectos, decisões rotineiras

Pode efectuar muito trabalho desnecessário que é irrelevante para o objectivo

ET é guiado pelo objectivo, adequadao para a resolução de problemas, e.g., Onde estão as minhas chaves? Como posso entrar para o 2^o ciclo?

Complexidade do ET pode ser *muito inferior* do que linear no tamanho da KB

Resolução

Forma Normal Conjuntiva (FNC—universal)

conjunção de $\underbrace{\textit{disjunções de literais}}_{\textit{cláusulas}}$

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Regra de inferência Resolução (para FNC): completa para a lógica proposicional

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

em que ℓ_i e m_j são literais complementares. E.g.,

$$\frac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$$

Resolução é sólida e completa para a lógica proposicional

Conversão para FNC

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

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. Deslocar $\neg$ para dentro recorrendo às leis de De Morgan e 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 distributividade ($\vee$ sobre $\wedge$):

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

Algoritmo de Resolução

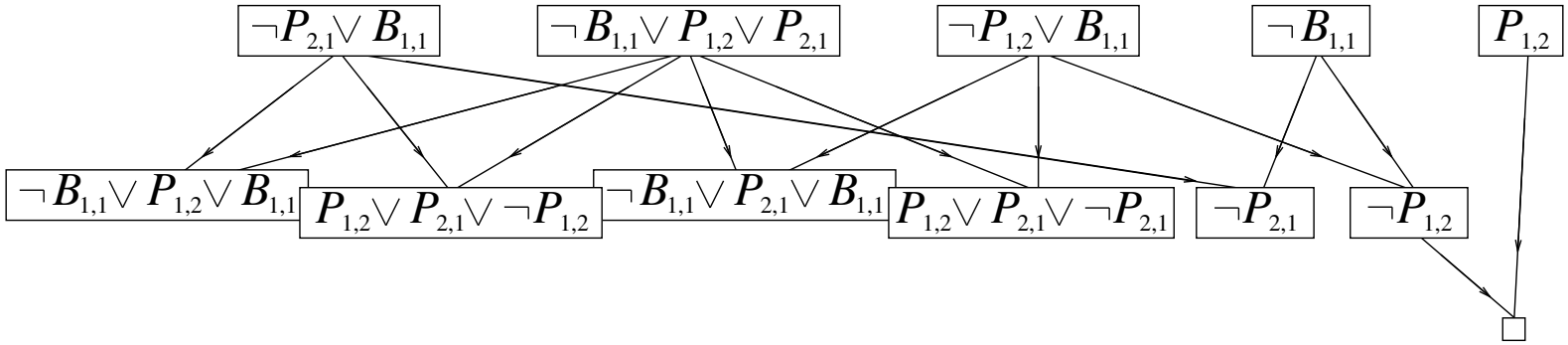
Prova por contradição, i.e., demonstrar que $KB \wedge \neg\alpha$ é insatisfazível

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*
  $clauses \leftarrow clauses \cup new$

Exemplo de Resolução

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



Um agente lógico no mundo do Wumpus

Formulação em lógica proposicional:

$$\neg P_{1,1}$$
$$\neg W_{1,1}$$
$$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$$
$$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$$
$$W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,4}$$
$$\neg W_{1,1} \vee \neg W_{1,2}$$
$$\neg W_{1,1} \vee \neg W_{1,3}$$
$$\vdots$$

São necessárias 64 variáveis proposicionais e 155 frases

function PL-WUMPUS-AGENT(*percept*) **returns** an *action*

inputs: *percept*, a list, [*stench*,*breeze*,*glitter*]

static: KB, a knowledge base, initially containing the "physics" of the world

$x, y, orientation$, the agent's position (initially [1,1]) and orientation (initially *right*)

visited, an array indicating which squares have been visited, initially *false*

action, the agent's most recent action, initially null

plan, an action sequence, initially empty

update x,y,$orientation$, *visited* based on *action*

if *stench* **then** TELL($KB, S_{x,y}$) **else** TELL($KB, \neg\ S_{x,y}$)

if *breeze* **then** TELL($KB, B_{x,y}$) **else** TELL($KB, \neg\ B_{x,y}$)

if *glitter* **then** *action* $\leftarrow$ *grab*

else if *plan* is nonempty then *action* $\leftarrow$ POP(*plan*)

else if for some fringe square [*i*,*j*], ASK($KB, (\neg\ P_{i,j} \wedge\ \neg\ W_{i,j})$) is *true* **or**

for some fringe square [*i*,*j*], ASK($KB, (P_{i,j} \vee\ W_{i,j})$) is *false* **then do**

plan $\leftarrow$ A*-GRAPH-SEARCH(ROUTE-PROBLEM([x,y], *orientation*, [*i*,*j*],*visited*))

action $\leftarrow$ POP(*plan*)

else *action* $\leftarrow$ a randomly chosen move

return *action*

Limites de expressividade da lógica proposicional

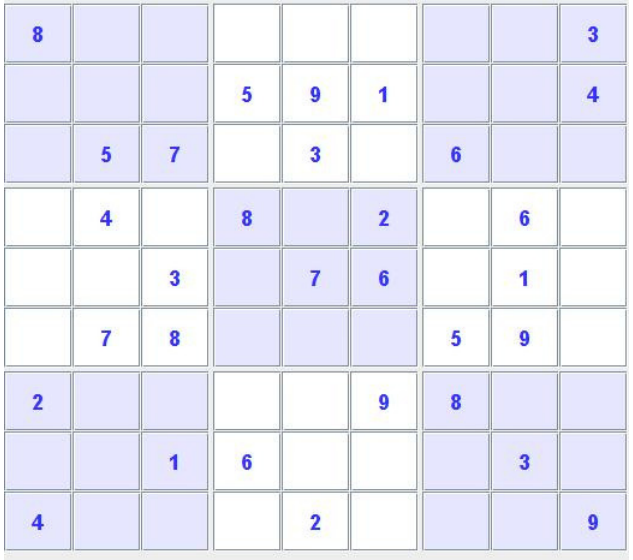
♢ KB contém cláusulas capturando as leis "física" para cada casa

Para todo o tempo t e casa $[x, y]$,

$$L^t_{x,y} \wedge FacingRight^t \wedge Forward^t \Rightarrow L^t_{x+1,y}$$

♢ Proliferação rápida do número de cláusulas

O SuDoKu* em lógica proposicional



♢ Cada célula contém um inteiro entre 1 e 9

♢ Nenhum par de células na mesma linha contém o mesmo valor

♢ Nenhum par de células na mesma coluna contém o mesmo valor

♢ Nenhum par de células num bloco 3x3 contém o mesmo valor

♢ Saber se existe ou não solução para um dado puzzle SuDoKu é um problema NP-completo, e como tal pode ser reduzido por uma transformação polinomial a um problema de satisfatibilidade booleana.

* - "suji wa dokushin ni kagiru"

Tradução do SuDoKu para lógica proposicional

♢ São necessárias $n \times n \times n = n^3$ variáveis para um SuDoKU $n \times n$.

Variável $c_{i,j,k}$ $(1 \leq i, j, k \leq n)$ é verdadeira quando $i \times j$ contém o valor k .

♢ Cláusulas de célula $\left(n^2 \times (1 + \frac{n \times (n-1)}{2})\right)$

$\vee_{1 \leq k \leq n} c_{i,j,k}$	1 cláusula para cada casa $1 \leq i, j \leq n$
$\neg c_{i,j,k} \vee \neg c_{i,j,l}$ $(1 \leq k < l \leq n)$	$\frac{n \times (n-1)}{2}$ cláusulas para cada casa $1 \leq i, j \leq n$

♢ Cláusulas de linha $\left(n^2 \times (1 + \frac{n \times (n-1)}{2})\right)$

$\vee_{1 \leq j \leq n} c_{i,j,k}$	1 cláusula para cada casa $1 \leq i, k \leq n$
$\neg c_{i,j,k} \vee \neg c_{i,l,k}$ $(1 \leq j < l \leq n)$	$\frac{n \times (n-1)}{2}$ cláusulas para cada casa $1 \leq i, k \leq n$

Adicionam-se cláusulas semelhantes para tratar as colunas e os blocos, obtendo um total de $2 \cdot n^4 - 2 \cdot n^3 + 4 \cdot n^2$ cláusulas. Para um puzzle 9x9 temos assim exactamente 11988 cláusulas.

♢ Junta-se a proposição $c_{i,j,k}$ para cada casa $i \times j$ ocupada com o valor k.

O nosso puzzle SuDoKu em lógica proposicional

8	1	4	2	6	7	9	5	3
3	6	2	5	9	1	7	8	4
9	5	7	4	3	8	6	2	1
1	4	9	8	5	2	3	6	7
5	2	3	9	7	6	4	1	8
6	7	8	1	4	3	5	9	2
2	3	5	7	1	9	8	4	6
7	9	1	6	8	4	2	3	5
4	8	6	3	2	5	1	7	9

$c_{1,1,1} \vee c_{1,1,2} \vee \ldots \vee c_{1,1,8} \vee c_{1,1,9}$ (células)

$\neg c_{1,1,1} \vee \neg c_{1,1,2}$

$\neg c_{1,1,1} \vee \neg c_{1,1,3}$

$\vdots$

$c_{1,1,1} \vee c_{1,2,1} \vee \ldots \vee c_{1,8,1} \vee c_{1,9,1}$ (linhas)

$\neg c_{1,1,1} \vee \neg c_{1,2,1}$

$\neg c_{1,1,1} \vee \neg c_{1,3,1}$

$\vdots$

$c_{1,1,1} \vee c_{2,1,1} \vee \ldots \vee c_{8,1,1} \vee c_{9,1,1}$ (colunas)

$\neg c_{1,1,1} \vee \neg c_{2,1,1}$

$\neg c_{1,1,1} \vee \neg c_{3,1,1}$

$\vdots$

$c_{1,1,1} \vee c_{1,2,1} \vee \ldots \vee c_{3,2,1} \vee c_{3,3,1}$ (blocos)

$\neg c_{1,1,1} \vee \neg c_{2,2,1}$

$\neg c_{1,1,1} \vee \neg c_{2,3,1}$

$\vdots$

$c_{1,1,8} \wedge c_{1,9,3} \wedge \ldots \wedge c_{9,9,9}$ (valores)

A solução do puzzle SuDoKu pode ser extraída das variáveis verdadeiras num modelo que satisfaça todas as cláusulas. Encontrar esse modelo é um problema da classe FNP (function NP problem, na terminologia inglesa).

Resumo das classes de complexidade (P)

♢ Um problema pertence à classe P, quando pode ser resolvido por uma máquina de Turing determinista em tempo polinomial.

Um problema de decisão é P-completo se pertence a P e qualquer problema na classe P pode ser reduzido a ele (em espaço logarítmico).

- Saber se um conjunto de cláusulas de Horn é satisfatível é um um problema P-completo (HORNSAT)
- Saber qual o valor de um circuito booleano dados os seus inputs é P-completo (CIRCUIT VALUE)

Saber se um número inteiro é primo pertence a P (resultado de 2002)!

Resumo das classes de complex. (NP e coNP)

♢ Um problema pertence à classe NP, quando pode ser resolvido por uma máquina de Turing não determinista em tempo polinomial.

♢ Um problema de decisão é NP-completo quando a solução pode ser verificada em tempo polinomial. Formalmente, um problema é NP-completo

1. se pertence a NP
2. qualquer problema na classe NP pode ser reduzido a ele por uma transformação polinomial

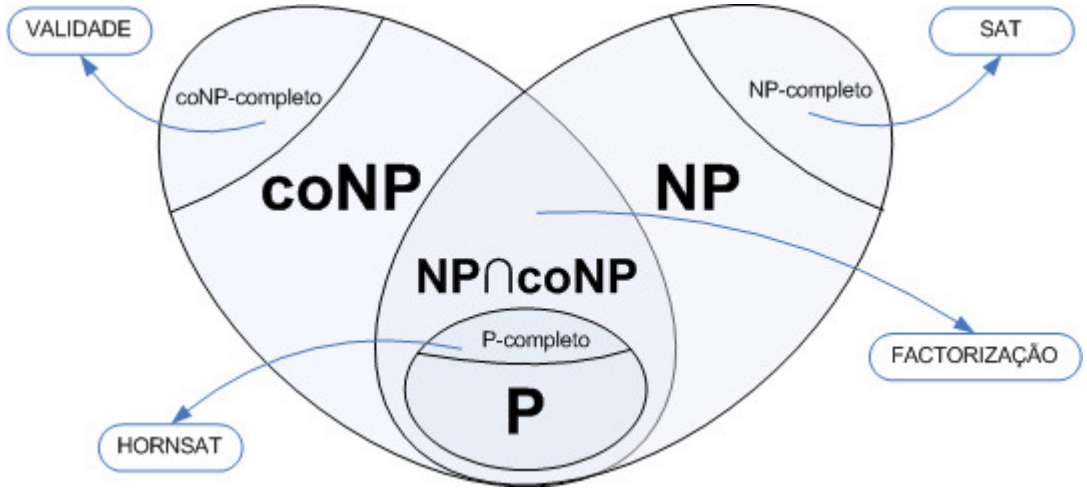
♢ Caso um problema só obedeça ao critério (2) diz-se NP-difícil.

♢ Um problema pertence à classe coNP quando o seu complementar pertence à classe NP (pode-se verificar que não é solução em tempo polinomial).

Presume-se que $P \neq NP$ e que $NP \neq coNP$.

Complexidade e lógica proposicional

Teorema de Cook-Levin (1971): O problema SAT é NP-completo.



NOTA: Testar a validade de um conjunto de fórmulas booleanas é um problema coNP-completo (VALIDITY).

Outros problemas NP-completos (Karp 1972)

SAT

- 0-1 INTEGER PROGRAMMING
- CLIQUE
  - SET PACKING
  - VERTEX COVER
  - SET COVERING
  - FEEDBACK ARC SET
  - FEEDBACK NODE SET
  - DIRECTED HAMILTONIAN CIRCUIT
    - UNDIRECTED HAMILTONIAN CIRCUIT
- 3-SAT
  - CHROMATIC NUMBER
  - CLIQUE COVER
  - EXACT COVER
    - 3D MATCHING
    - STEINER TREE
    - HITTING SET
    - KNAPSACK
      - JOB SEQUENCING
      - PARTITION
      - MAX-CUT

Verificação eficiente de satisfatibilidade

♢ A existência de algoritmos eficientes de satisfatibilidade permite-nos lidar com problemas combinatórios complexos

♢ Todos os problemas NP-completos podem ser reduzidos ao problema da satisfatibilidade de cláusulas de lógica proposicional

♢ Duas famílias de algoritmos:
- ⇒ Baseados em retrocesso, e.g. Davis-Putnam-Logemann-Loveland
- ⇒ Por melhoramento iterativo (procura local), e.g. WalkSAT

Algoritmo de Davis-Putnam

Enumeração recursiva em profundidade primeiro de todos os modelos possíveis para proposições na FNC, com as seguintes melhorias:

♢ Terminação com modelos parciais: uma cláusula é verdadeira quando pelo menos um dos literais é verdadeiro. Logo, os restantes valores dos símbolos proposicionais são irrelevantes.

♢ Heurística dos símbolos puros: um símbolo é puro quando ocorre sempre com o mesmo sinal em todas as cláusulas. Nas proposições abaixo, A e $\neg B$ são puros:

$$(A \vee \neg B) \quad \wedge \quad (\neg B \vee \neg C) \quad \wedge \quad (C \vee A)$$

♢ Heurística da cláusula unitária: Quando todos os literais são falsos à excepção de um, o valor desse fica automaticamente definido. Pode originar propagações unitárias em cascata.

Algoritmo de Davis-Putnam

function DPLL-SATISFIABLE?(s) **returns** *true* or *false*

inputs: s, a sentence in propositional logic

$clauses \leftarrow$ the set of clauses in the CNF representation of s

$symbols \leftarrow$ a list of the proposition symbols in s

return DPLL($clauses$, $symbols$, [ ])

---

function DPLL($clauses$, $symbols$, $model$) **returns** *true* or *false*

if every clause in $clauses$ is true in $model$ **then return** *true*

if some clause in $clauses$ is false in $model$ **then return** *false*

P, $value \leftarrow$ FIND-PURE-SYMBOL($symbols$, $clauses$, $model$)

if P is non-null **then return** DPLL($clauses$, $symbols$–P, $[P = value|model]$)

P, $value \leftarrow$ FIND-UNIT-CLAUSE($clauses$, $model$)

if P is non-null **then return** DPLL($clauses$, $symbols$–P, $[P = value|model]$)

$P \leftarrow$ FIRST($symbols$); $rest \leftarrow$ REST($symbols$)

return DPLL($clauses$, $rest$, $[P = true|model]$) **or** DPLL($clauses$, $rest$, $[P = false|model]$)

Exemplo de aplicação

$$a \vee \neg b \vee c$$
$$\neg b \vee \neg d$$
$$\neg a \vee \neg c$$
$$a \vee d$$
$$c \vee e$$
$$\neg e \vee f$$
$$\neg d \vee c \vee \neg f$$

Propriedades do algoritmo de Davis-Putnam

♢ Completo

♢ Eficaz na prática podendo resolver problemas de verificação de Hardware com 1 milhão de variáveis

WalkSAT

♢ Trepa-colinas no espaço de atribuições completas

♢ Em cada iteração o algoritmo escolhe uma cláusula não satisfeita e um símbolo dessa cláusula para trocar. A forma de escolha do símbolo a trocar de valor é ela própria aleatória, podendo ser:

⇒ Utilizando a heurística "min-conflitos" minimizando o número de cláusulas insatisfeitas no passo seguinte

⇒ Escolha aleatória do símbolo a trocar na cláusula ("passeio aleatório")

WalkSAT

function WALKSAT(*clauses*, p, *max-flips*) **returns** a satisfying model or *failure*

inputs: *clauses*, a set of clauses in propositional logic

p, the probability of choosing to do a "random walk" move, typically around 0.5

max-flips, number of flips allowed before giving up

model $\leftarrow$ a random assignment of *true*/*false* to the symbols in *clauses*

for $i = 1$ **to** *max-flips* **do**

if *model* satisfies *clauses* **then return** *model*

clause $\leftarrow$ a randomly selected clause from *clauses* that is false in *model*

with probability p flip the value in *model* of a randomly selected symbol from *clause*

else flip whichever symbol in *clause* maximizes the number of satisfied clauses

return *failure*

Propriedades do WalkSAT

♢ Incompleto

♢ Se uma proposição é insatisfazível então o algoritmo não termina: limita-se max_flips...

♢ Logo, procura local não serve em geral para resolver o problema da consequência lógica

♢ Algoritmos locais como o WalkSAT são mais eficazes quando se espera que uma solução exista

♢ Muito eficiente na prática...

Problemas de satisfatibilidade difíceis

Considere cláusulas 3-CNF geradas aleatoriamente, e.g:

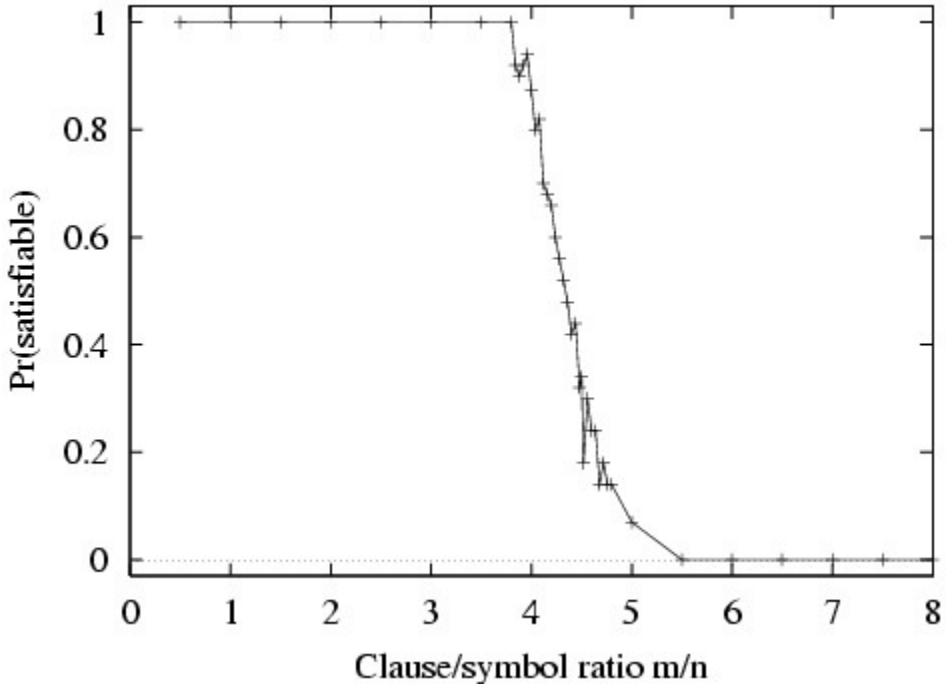
$$(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$$

Seja

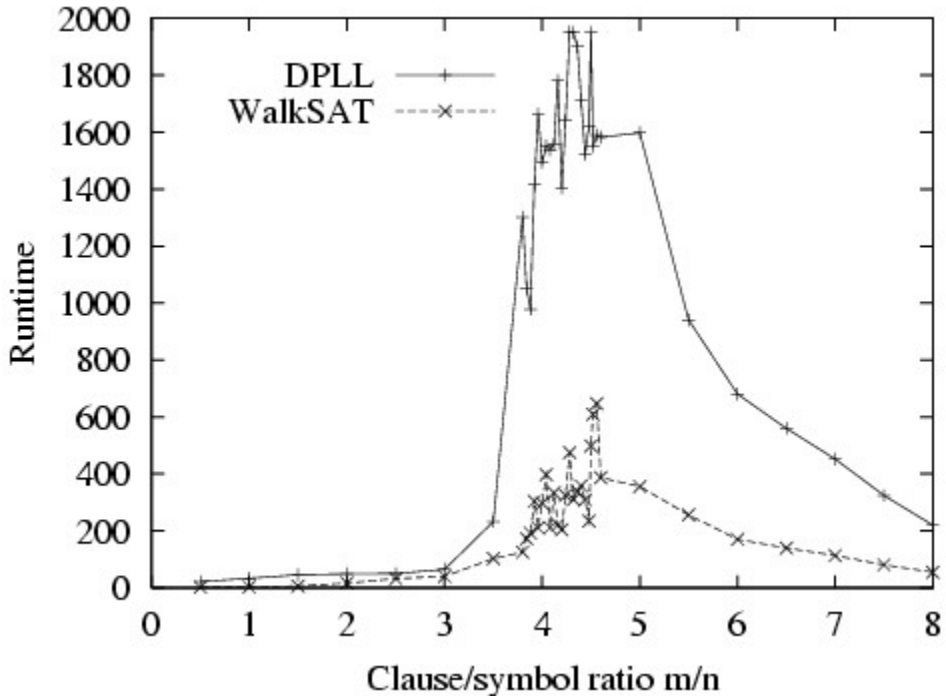
- $m =$ número de cláusulas
- $n =$ número de símbolos

Os problemas mais difíceis parecem concentrar-se perto do valor do rácio $m/n = 4.3$ (ponto crítico)

Problemas de satisfatibilidade difíceis



Problemas de satisfatibilidade difíceis



Tempo de execução médio para 100 fórmulas aleatórias 3-CNF satisfazíveis (com $n = 50$).

Sumário

Agentes lógicos aplicam inferência a bases de conhecimento para derivar nova informação e tomar decisões

Conceitos básicos de lógica:

- sintaxe: estrutura formal das frases declarativas
- semântica: veracidade das frases relativamente a modelos
- conclusão: verdade necessária de uma frase dado outra
- inferência: derivação de frases a partir de outras frases
- sólido: derivações produzem apenas frases que são conclusões lógicas
- completo: derivações conseguem produzir todas as frases que são consequência lógicas

O mundo do Wumpus requer a capacidade de lidar com informação partial e negativa, raciocínio por casos, etc.

Encadeamento para a frente e para trás têm complexidade temporal linear na dimensão da KB, completos para cláusulas de Horn. Resolução é completa para a lógica proposicional

A lógica proposicional não tem poder expressivo suficiente