



EA 072 Inteligência Artificial em Aplicações Industriais

5-Lógica Matemática

Representação e Inferência

Introdução

- Objetivo

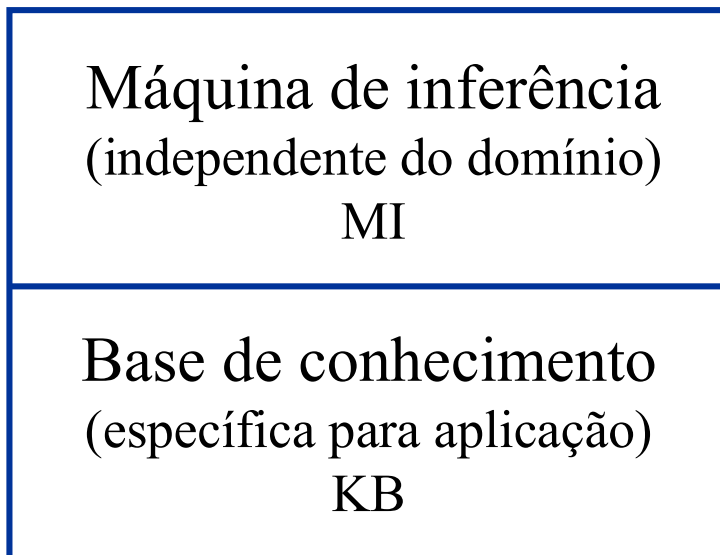
- projetar agentes que podem criar representações do mundo
- inferência para obter novas representações sobre o mundo
- usar estas novas representações para deduzir o que fazer

- Elementos básicos de projeto do agente:

- linguagem formal para expressar conhecimento
- mecanismo para raciocinar com esta linguagem
- linguagem formal \leftrightarrow lógica matemática

Agente baseado em conhecimento

Componentes



Base de conhecimento (KB)

- conjunto de representações sobre o mundo
- cada elemento do conjunto é uma sentença
 - sentença: expressão em uma linguagem formal
- conhecimento inicial: *background knowledge*

Operações de inserção e consulta (*query*)

- TELL
- ASK
- ambas envolvem inferência

O que o agente lógico faz:

TELLs à KB o que ele percebe

ASKs à KB qual ação a tomar: inferência sobre

- estado corrente
- resultados de sequências factíveis de ações

TELLs à KB qual ação foi escolhida e executa ação

Interface entre sensores e atuadores, MI e KB

MAKE_PERCEPT_SENTENCE (*percept*, *t*)

MAKE_ACTION_QUERY (*t*)

MAKE_ACTION_SENTENCE (*action*, *t*)

Agente baseado em conhecimento

function KB_AGENT (*percept*) **returns** an *action*

persistent: *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*

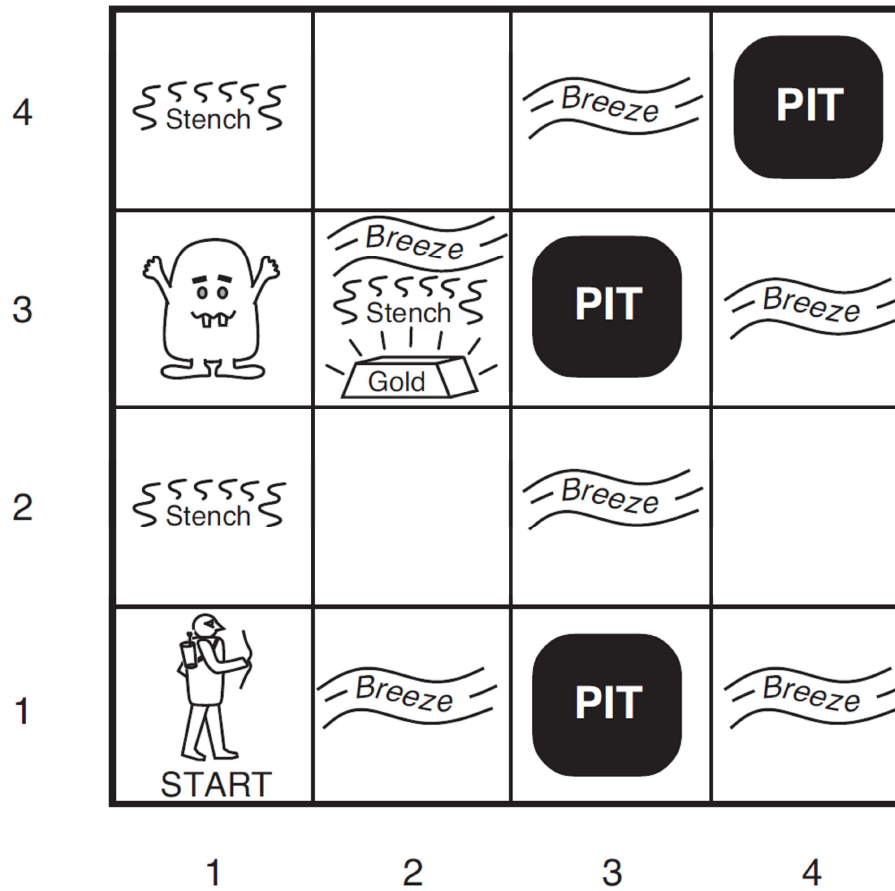
Níveis de descrição de ABC

- nível epistemológico: descreve o agente dizendo o que ele sabe
Ponte Rio–Niteroi conecta Rio e Niteroi
- nível lógico: codificação do conhecimento em sentenças
Conecta (Ponte_R_N, Rio, Niteroi)
- nível implementação: representação física
lista
Conecta (Ponte_R_N, Rio, Niteroi)
tabelas

Construção de ABC

- abordagem declarativa
 - descreve o que o agente precisa conhecer
- abordagem procedimental
 - codifica comportamento desejado diretamente
- aprendizagem

O mundo do Wumpus



PEAS

■ Performance

- ouro +1000, morte -1000
- cada ação - 1, disparar flexa - 10
- término: agente morre ou escapa com ouro

■ Ambiente

- grid 4×4
- posição inicial agente [1, 1], voltado para a direita
- posição Wumpus e ouro aleatória (distribuição uniforme)
- probabilidade de um local ter poço é 0.2

■ Ações

- *forward*, *turn right 90°*, *turn left 90°*, *grab*, *shoot*
- *shoot* somente uma vez, na direção do agente
- *climb* somente se estiver na posição [1,1]
- agente morre se entra local com Wumpus ou poço
- pega o ouro só se estiverem no mesmo local

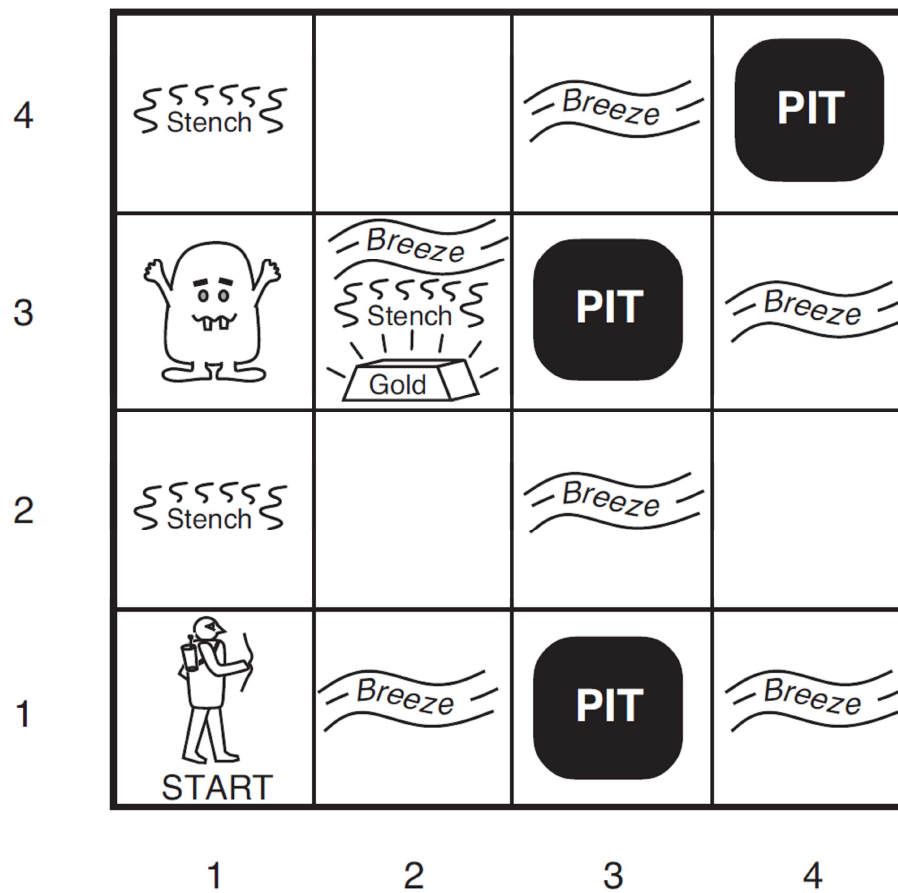
■ Sensores

- lista de símbolos [*Stench*, *Breeze*, *Glitter*, *Bump*, *Scream*]
- odor no local com Wumpus e adjacentes (*stench*)
- locais adjacentes a poços ventam (*breeze*)
- local onde tem ouro brilha (*glitter*)
- *bump* se move em direção às paredes
- grito quando Wumpus morre (*scream*)

- Caracterização do mundo do Wumpus
 - parcialmente observável
 - posição agente, saúde Wumpus e disponibilidade da flecha não são conhecidos diretamente
 - determinístico: Wumpus, poços, ouro imutáveis (não observáveis)
 - sequencial
 - estático: Wumpus, poços e pilha de ouro não se movem
 - discreto
 - único agente

- Desafio: decorrente da ignorância da configuração inicial do ambiente
- Superação: eliminação da ignorância requer raciocínio lógico

Exemplo: explorando o mundo do Wumpus

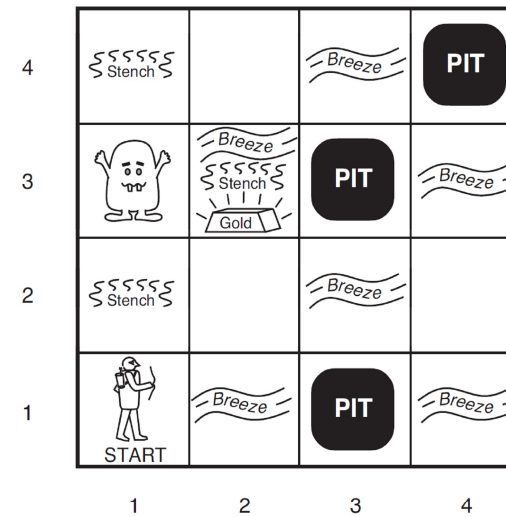


ok			
A ok	ok		

4	Stench		Breeze	PIT
3	Ghost	Breeze Stench Gold	PIT	Breeze
2	Stench		Breeze	
1	START Player	Breeze	PIT	Breeze
	1	2	3	4

[None, None, None, None, None]

ok	P?		
ok v ok	A B ok	P?	



[None, Breeze, None, None, None]

W!			
A S ok	ok		
v ok	B v ok	P!	

[*Stench, None, None, None, None*]

4	Stench		Breeze	PIT
3	Stench	Breeze	PIT	Breeze
2	Stench		Breeze	
1	START	Breeze	PIT	Breeze
	1	2	3	4

	P?		
W!	A	P?	
s v ok	G B		
v ok	v ok	P!	

4	Stench		Breeze	PIT
3	Stench	Breeze	PIT	Breeze
2	Stench		Breeze	
1	START	Breeze	PIT	Breeze
	1	2	3	4

[Stench, Breeze, Glitter, None, None]

Representação, raciocínio e lógica

- Representação de conhecimento
 - descrição de conhecimento em linguagem computacionalmente tratável
 - base de conhecimento: conjunto de sentenças na linguagem
- Linguagem de representação de conhecimento
 - sintaxe: sentenças bem formadas (corretas)
 - semântica: significado das sentenças (a que se referem)

■ Exemplo

– sintaxe

- $x \geq y$ sentença linguagem da aritmética
- $x^2y+ =$ não é uma sentença da aritmética

– semântica

- verdade de sentenças com relação a mundo possíveis
- linguagem da aritmética: sentenças sobre números
- se x é maior que ou igual à y , então sentença verdadeira; senão falsa

Lógica = sintaxe + semântica (+ *proof theory*)

Sintaxe e semântica bem definidas

- permitem desenvolver mecanismos de inferência

Semântica em lógica

- define a veracidade de cada sentença
- veracidade em cada mundo possível

Exemplo

- sentença $x + y = 4$
 - verdadeira no mundo onde $x = 2$ e $y = 2$
 - falsa no mundo onde $x = 1, y = 1$
- modelos possíveis: todas atribuições para x e y

Mundo possível \leftrightarrow modelo

- notação: m é um modelo para sentença α
- α é verdadeira no modelo m (m satisfaz α)
- $M(\alpha)$: conjunto de todos modelos de α
- fixa veracidade (*true*, *false*) de cada sentença

Raciocínio (inferência) em lógica

$\alpha \models \beta$ sentença α *entails* sentença β

sentença β segue logicamente da sentença α

se α é verdadeira em m , então β também é verdadeira em m

sentença α *mais forte* que sentença β

$\alpha \models \beta$ se e somente se $M(\alpha) \subseteq M(\beta)$

Exemplo: $x = 0$ *entails* $x y = 0$

Modelos em lógica

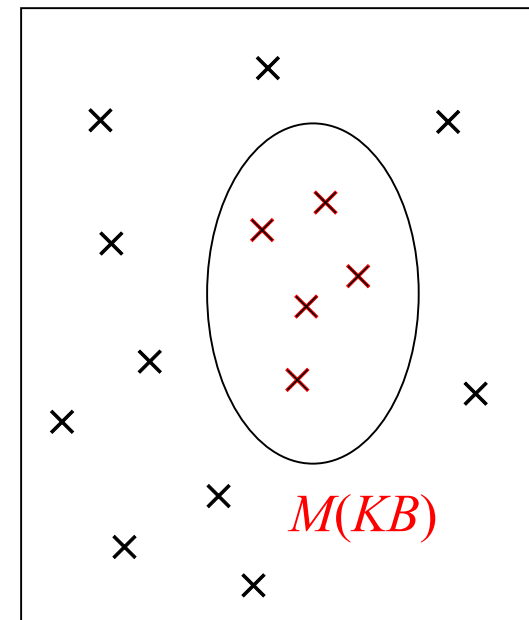
$M(\alpha)$ conjunto todos modelos de α

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

Exemplo: $KB = \text{homen, mortal}$
 $\alpha = \text{mortal}$

KB : conjunto de sentenças

Inferência: KB *entailing* outras sentenças (fatos)



$M(\alpha)$

Exemplo: explorando o mundo do Wumpus

ok	P?		
v ok	A B ok	P?	

4	Stench	Breeze	PIT	
3	Wumpus	Breeze, Stench, Gold	PIT, Breeze	
2	Stench	Breeze		
1	START	Breeze	PIT, Breeze	
	1	2	3	4

$KB = \text{conhecimento agente mundo Wumpus (PEAS)} + [1, 1] \text{ ok} + [2, 1] \text{ B}$

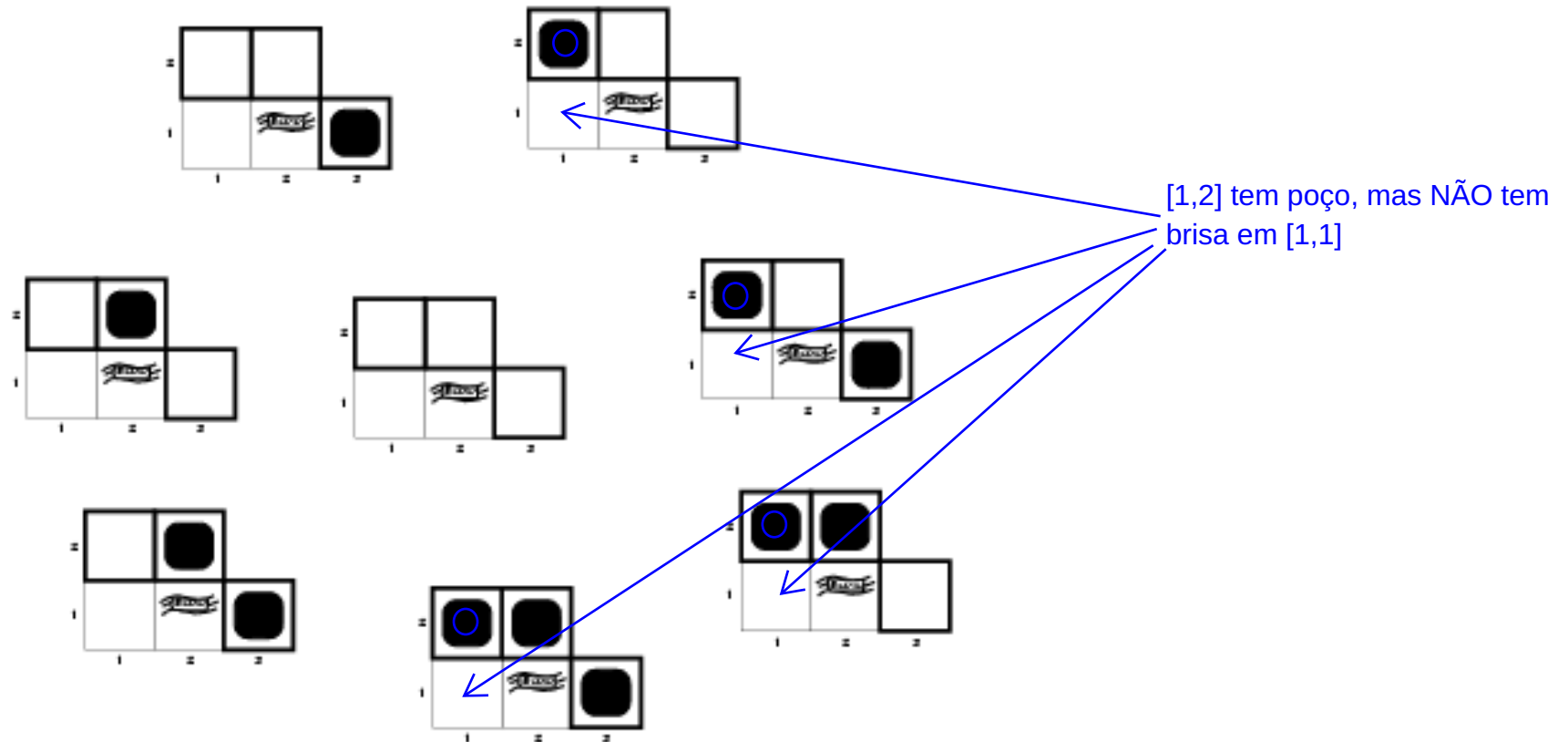
Interesse do agente: existe poço em [1, 2], [2, 2], [3, 1] ?

P?	P?		
v ok	A B ok	P?	

4	Stench	Breeze	PIT	
3	Stench	Breeze	PIT	
2	Stench	Breeze		
1	START	Breeze	PIT	
	1	2	3	4

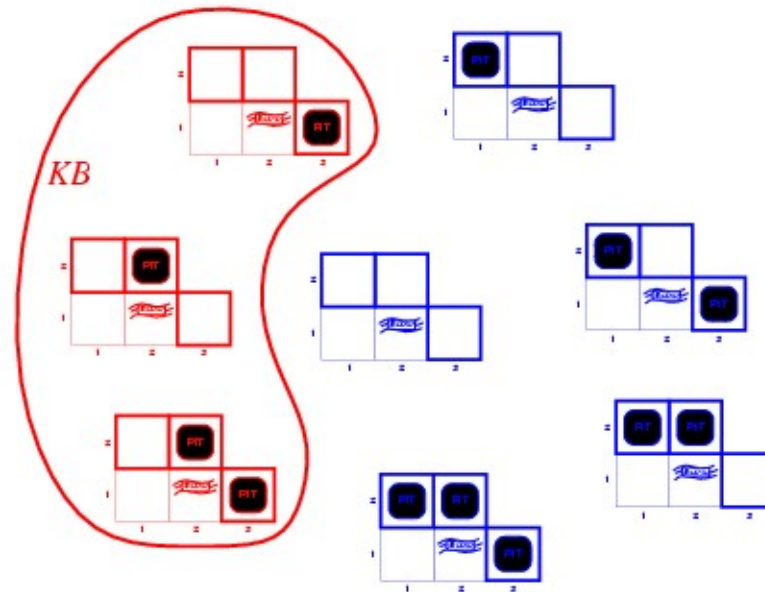
Existem $2^3 = 8$ modelos

Modelos



KB é falsa em todo modelo que contradiz o que o agente conhece
Exemplo: modelos onde [1, 2] têm poço (não venta em [1, 1])

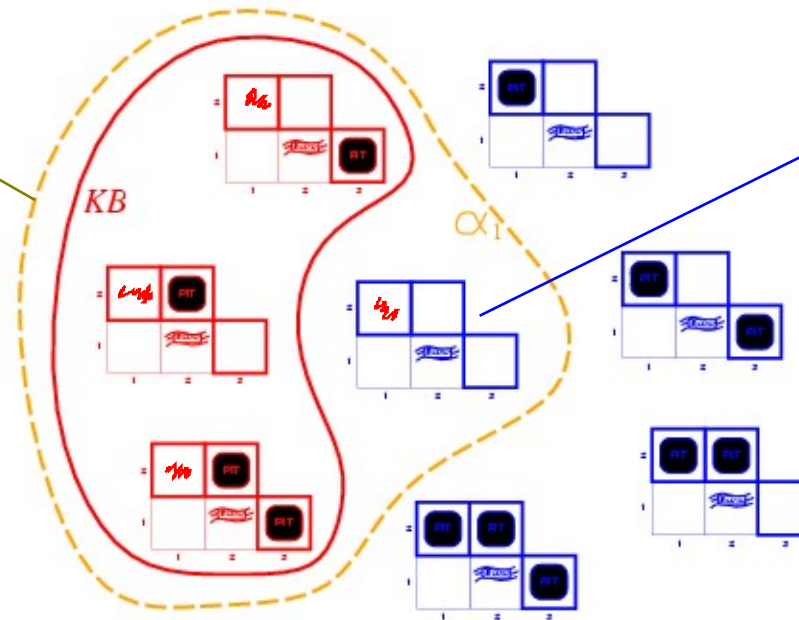
Base de conhecimento (*KB*)



Pergunta:
 Será que a *KB*
 tem por consequência
 lógica $\sim P12$?
 Isto é, $\sim P12$ é
 consequência lógica da
KB?

$$KB = \text{regras mundo Wumpus (PEAS)} + \underbrace{[1, 1] \text{ ok} + [2, 1] \text{ B}}_{\text{observações (fatos)}}$$

Todos os modelos onde $\sim P12$

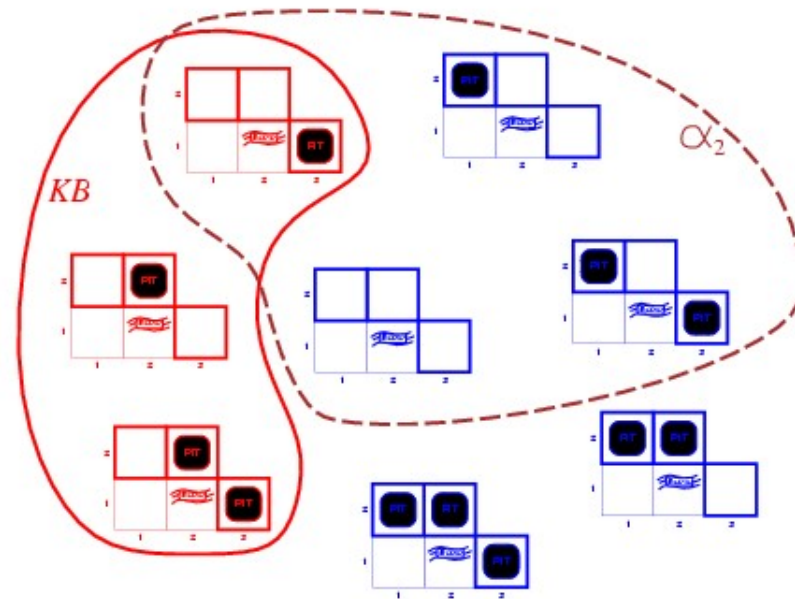


Outros casos
que não estão
na KB

$KB =$ regras mundo Wumpus + observações

$\alpha_1 =$ “não tem poço em [1,2]”

$KB \models \alpha_1$ provado verificando o modelo



Alguns modelos de KB não pertencem a alpha 2

$KB = \text{regras mundo Wumpus} + \text{observações}$

$\alpha_2 = \text{“não tem poço em [2,2]”}$

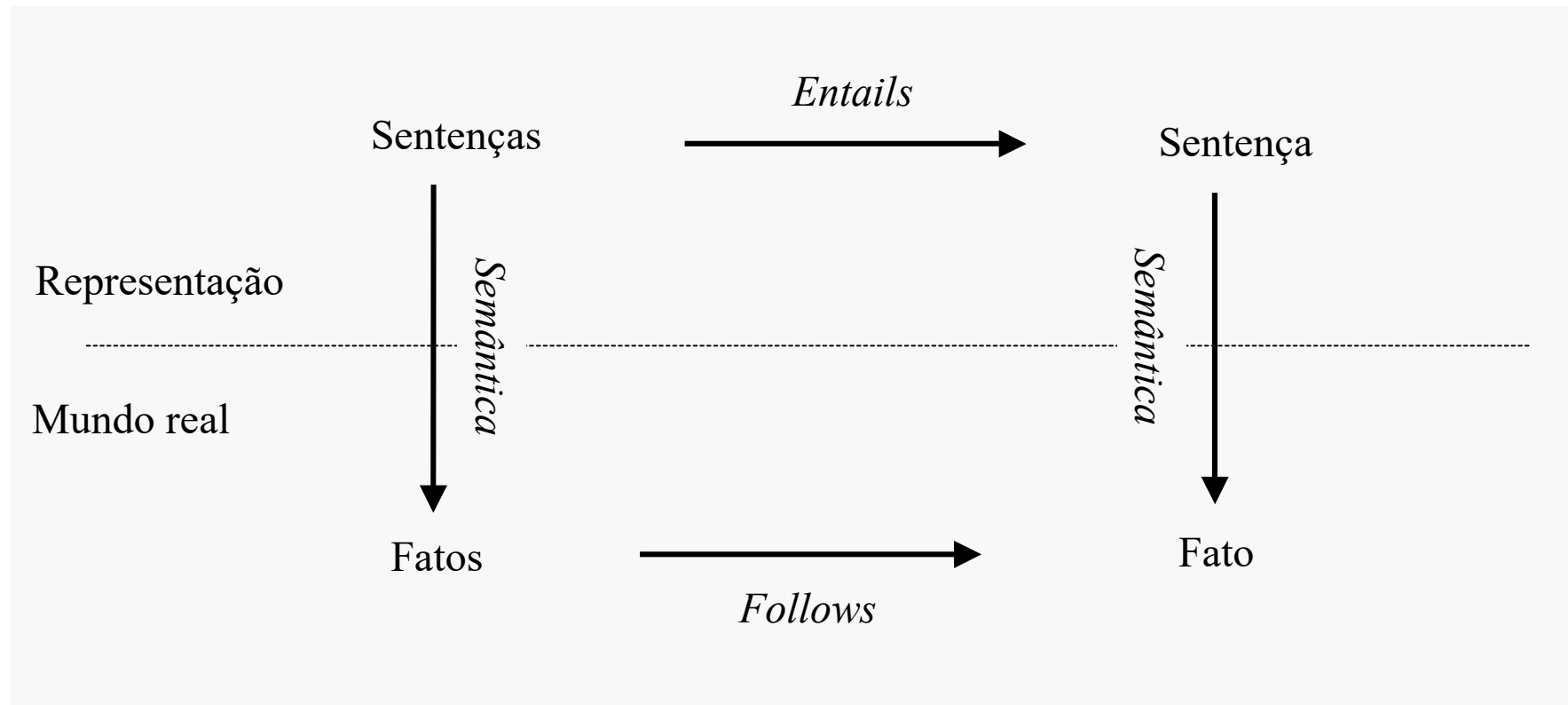
$KB \not\subseteq \alpha_2$

- Procedimento de inferência: alternativas
 - dada KB , gerar novas sentenças α que sejam conseqüências lógicas de KB
 - notação: $KB \vdash_i \alpha$ se procedimento de inferência i deriva α de KB
 - dada KB e uma sentença α , verificar se α é conseqüência lógica de KB

- Procedimento de inferência **consistente** (*sound*)
 - produz somente sentenças que são conseqüências lógicas de KB
 - sempre que $KB \vdash_i \alpha$, então $KB \models \alpha$ também é verdade

- Procedimento de inferência **completo**
 - encontra sentenças que são conseqüências lógicas
 - sempre que $KB \models \alpha$, então $KB \vdash_i \alpha$ também é verdade

Mundo real e sua representação



Instanciação (*grounding*): conexão raciocínio lógico e o ambiente do agente

Representação: linguagens formais \times linguagens naturais

Linguagens de programação: não são expressivas o suficiente

Linguagem natural: comunicação ao invés de representação;
ambiguidade

Linguagem de representação:

ideal: combinar vantagens das formais e naturais

inferências

Em IA: linguagem da lógica matemática

Lógica proposicional (Cálculo proposicional)

símbolos representam proposições

conectivos Booleanos

Lógica de primeira ordem (Cálculo de predicados)

objetos e predicados

quantificadores

Lógica temporal

mundo ordenado por um conjunto de pontos ou intervalos (tempo)

Lógica fuzzy

Lógica proposicional

Sentença \rightarrow *SentençaAtomica* | *SentençaComplexa*

SentençaAtomica \rightarrow *True* | *False* | *Símbolo*

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

SentençaComplexa \rightarrow (*Sentença*) | [*Sentença*]

| *Sentença Conectivo* *Sentença*

| \neg *Sentença*

Conectivo \rightarrow \wedge | \vee | \Rightarrow | \Leftrightarrow

sintaxe
(BNF)

P, Q, R, S, \dots símbolos proposicionais

$\neg P \vee Q \wedge R \Rightarrow S$ sentenças (ou fórmulas)

Precedência: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

Exemplo: $\neg P \vee Q \wedge R \Rightarrow S \equiv ((\neg P) \vee (Q \wedge R)) \Rightarrow S$

Literal

- sentença atômica (literal positivo)
- sentença atômica negada (literal negativo)

■ Semântica de sentenças

- regras para definir veracidade de sentenças com relação a um modelo
- modelo fixa o valor verdade (*true*, *false*) de todo símbolo proposicional
 - Exemplo: $m_1 = \{P_{1,2} = \textit{false}, P_{2,2} = \textit{false}, P_{3,1} = \textit{true}\}$
- modelos são objetos puramente matemáticos
- instanciação do significado é feita pelo projetista/analista

■ Regras semânticas

- especifica como computar valor verdade de toda sentença
- lógica proposicional é composicional
- sentenças = sentenças atômicas + conectivos
- valor verdade de sentenças requer
 - valor verdade de sentenças atômicas
- valor verdade de sentenças complexas

■ Semântica sentenças atômicas

- *True* é verdadeira (*true*) em todos modelos
- *False* é falso (*false*) em todos os modelos
- valor verdade de qualquer outro átomo deve ser especificado no modelo

Exemplo: $P_{1,2}$ é falsa em $m_1 = \{P_{1,2} = false, P_{2,2} = false, P_{3,1} = true\}$

■ Semântica sentenças complexas

- sentença s e modelo m : $\neg s$ é verdadeira em m se somente s é falsa em m
- veracidade sentença reduz a veracidade de sentenças mais simples
- regras para sentenças com conectivos: tabela verdade

Tabela verdade para conectivos lógicos

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>
<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>

- Exemplo

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

$$m_1 = \{P_{1,2} = \text{false}, P_{2,2} = \text{false}, P_{3,1} = \text{true}\}$$

$$\text{true} \wedge (\text{false} \vee \text{true}) = \text{true} \wedge \text{true} = \text{true}$$

- Exemplo: validade

P	H	$\neg H$	$P \vee H$	$(P \vee H) \wedge \neg H$	$(P \vee H) \wedge \neg H \Rightarrow P$
<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>
<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>
<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>

Base de conhecimento

conjunção de sentenças

início com KB vazia

$TELL (KB, S_1), TELL (KB, S_2), \dots, TELL (KB, S_n)$

$$KB = S_1 \wedge S_2 \wedge \dots \wedge S_n$$

$KB \leftrightarrow$ sentença

Poço [1, 2], [2, 2], [3, 1] ?

P?	P?		
v ok	A B ok	P?	

4	Stench	Breeze	PIT	
3	Ghost Gold	Breeze Stench	PIT Breeze	
2	Stench	Breeze		
1	START	Breeze	PIT Breeze	
	1	2	3	4

Base conhecimento Wumpus

$P_{x,y}$ verdade se tem poço em $[x,y]$

$B_{x,y}$ verdade se venta em $[x, y]$

$R_1: \neg P_{1,1}$ não tem poço em $[1, 1]$

Poços provocam ventania em posições adjacentes

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

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

$R_4: \neg B_{1,1}$ não venta em $[1, 1]$

$R_5: B_{2,1}$ venta em $[2, 1]$

$KB = R_1 \wedge R_2 \wedge \dots \wedge R_5$

Inferência I

$KB \models \alpha$

exemplo: $\alpha = \neg P_{1,2}$ é consequência lógica de KB (KB entails α) ?

procedimento de inferência = prova

- enumerar modelos
- verificar se α é verdadeira em todos modelos onde KB é verdadeira

algoritmo de enumeração

- consistente: deriva da definição de *entailment*
- completo: funciona para qq KB e α , e termina pois KB é finita
- complexidade: n símbolos \rightarrow temporal $O(2^n)$, espacial $O(n)$

Algoritmo de enumeração

function TT_ENTAILS? (KB, α) **returns** *true* or *false*

inputs: KB , a knowledge base, a sentence in propositional logic

α , a query, a sentence in propositional logic

$symbols \leftarrow$ a list of propositional symbols in KB and α

return TT_CHECK_ALL($KB, \alpha, symbols, \{\}$)

function TT_CHECK_ALL($KB, symbols, model$) **returns** *true* or *false*

if EMPTY?($symbols$) **then**

if PL_TRUE?($KB, model$) **then return** PL_TRUE?($\alpha, model$)

else return true // when KB is false always return true

else do

$P \leftarrow$ FIRST($symbols$)

$rest \leftarrow$ REST($symbols$)

return (TT_CHECK_ALL($KB, \alpha, rest, model \cup \{P = true\}$)

and

TT_CHECK_ALL($KB, \alpha, rest, model \cup \{P = false\}$))

Base conhecimento Wumpus

7 símbolos proposicionais ($B_{1,1}$, $B_{2,1}$, $P_{1,1}$, $P_{1,2}$, $P_{2,1}$, $P_{2,2}$ e $P_{3,1}$)

$2^7 = 128$ modelos

Algoritmo:

enumerar modelos

verificar se α é verdadeira em todos modelos onde KB é verdadeira

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

Modelos para o exemplo do Wumpus

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>
<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<u><i>true</i></u>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<u><i>true</i></u>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>false</i>

$P_{1,2}$ é falsa \rightarrow não tem poço em $[1, 2] \equiv \neg P_{1,2}$ é verdadeira $\rightarrow KB \models \alpha$

$P_{2,2}, P_{3,1} \rightarrow$ pode ou não pode ter poço $[2, 2]$ e $[3, 1]$

Equivalência

sentenças logicamente equivalentes \leftrightarrow verdadeiras mesmo modelo

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$	comutatividade \wedge
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$	comutatividade \vee
$(\alpha \wedge \beta) \wedge \gamma \equiv (\alpha \wedge (\beta \wedge \gamma))$	associatividade \wedge
$(\alpha \vee \beta) \vee \gamma \equiv (\alpha \vee (\beta \vee \gamma))$	associatividade \vee
$\neg(\neg \alpha) \equiv \alpha$	dupla negação
$(\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha)$	contraposição
$(\alpha \Rightarrow \beta) \equiv (\neg \alpha \vee \beta)$	eliminação implicação
$(\alpha \Leftrightarrow \beta) \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$	eliminação bicondicional
$\neg(\alpha \wedge \beta) \equiv (\neg \alpha \vee \neg \beta)$	de Morgan
$\neg(\alpha \vee \beta) \equiv (\neg \alpha \wedge \neg \beta)$	de Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$	distributividade de \wedge sobre \vee
$(\alpha \vee (\beta \wedge \gamma)) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$	distributividade de \vee sobre \wedge

$\alpha \equiv \beta$ se e somente se $\alpha \models \beta$ e $\beta \models \alpha$

Validade

sentença válida é verdadeira em todos modelos (tautologia)

- Exemplos: $True$, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

validade e Teorema da dedução

- $KB \models \alpha$ se e somente se $(KB \Rightarrow \alpha)$ é válida

P	H	$\neg H$	$P \vee H$	$(P \vee H) \wedge \neg H$	$(P \vee H) \wedge \neg H \Rightarrow P$
<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>
<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>
<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>

Satisfatibilidade

sentença é satisfatível se é verdade em algum modelo

– Exemplo: $A \vee B, C$

sentença é insatisfatível se não é verdade em nenhum modelo

– Exemplo: $A \wedge \neg A$

satisfatibilidade e validade

– α é válida se e somente se $\neg\alpha$ é insatisfatível

– α é satisfatível se e somente se $\neg\alpha$ não é válida

Teorema (redução ao absurdo, refutação, prova por contradição):

$KB \models \alpha$ se e somente se $(KB \wedge \neg\alpha)$ é insatisfatível

Inferência II

Regras de inferência em lógica proposicional

- modus ponens: de α e $(\alpha \Rightarrow \beta)$ inferimos β
 - Exemplo: $(WumpusAhead \wedge WumpusAlive) \Rightarrow Shoot$
 $(WumpusAhead \wedge WumpusAlive)$
 $Shoot$
- And–Elimination: de $(\alpha \wedge \beta)$ inferimos α (β)
 - Exemplo: $(WumpusAhead \wedge WumpusAlive)$
 $WumpusAlive$
- equivalências lógicas também podem ser usadas como regras inferência
- regras produzem inferências consistentes (*sound*) sem enumerar modelos

Regras de inferência

Modus ponens	$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$
And-elimination	$\frac{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}{\alpha_i}$
And-introduction	$\frac{\alpha_1, \alpha_2, \dots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}$
Or-introduction	$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \dots \vee \alpha_n}$
Double negation elimination	$\frac{\neg \neg \alpha}{\alpha}$
Resolution	$\frac{\alpha \vee \beta, \neg \beta \vee \gamma}{\alpha \vee \gamma}$

Exemplo inferência no mundo do Wumpus

$$\left. \begin{array}{l}
 R_1: \neg P_{1,1} \text{ não tem poço em } [1, 1] \\
 R_2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}) \\
 R_3: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1}) \\
 R_4: \neg B_{1,1} \text{ não venta em } [1, 1] \\
 R_5: B_{2,1} \text{ venta em } [2, 1]
 \end{array} \right\} KB$$

ok	P?		
v ok	A B-ok	P?	

Provar $\neg P_{1,2}$ ($KB \models \neg P_{1,2}$)

$$\begin{array}{ll}
 R_6: (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}) & \text{eliminação bicondicional} \\
 R_7: (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1} & \text{and-elimination em } R_6 \\
 R_8: (\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1})) & \text{contraposição} \\
 R_9: \neg(P_{1,2} \vee P_{2,1}) & R_4 \text{ e modus ponens} \\
 R_{10}: \neg P_{1,2} \wedge \neg P_{2,1} & \text{de Morgan}
 \end{array}$$

Inferência em lógica \equiv prova

- prova = sequência de sentenças constituída por
 - sentenças da KB (hipóteses)
 - sentenças derivadas por regras de inferência
 - sentença que se quer provar (tese)
- algoritmo de busca onde *problem* é:
 - INITIAL STATE: KB inicial
 - ACTIONS: todas regras de inferência
 - RESULT: adicionar sentença inferida
 - GOAL: estado com sentença que se quer provar
- monotonicidade: inferência “aumenta” KB
 - se $KB \models \alpha$ então $KB \wedge \beta \models \alpha$

Métodos de prova: resumo

dois tipos básicos

- verificação de modelo (I)
 - enumeração via tabela verdade (exponencial em n)
 - backtracking, e.g., Davis--Putnam-Logemann-Loveland (DPLL)
 - busca heurística no espaço de modelos (*sound* mas incompleto)
Exemplo: min-conflicts-like hill-climbing algorithms
- aplicação de regras de inferência (II)
 - geração consistente (*sound*) de novas sentenças
 - prova = sequência de sentenças e aplicação de regras inferência
regras inferência como operadores em algoritmos de busca
 - tipicamente requer transformação sentenças em uma forma normal

Resolução

algoritmo de inferência

- consistente (*sound*)
- completo (teorema da resolução: $RC(S) \supset \beta$), decide $\alpha \models \beta$
- $RC(S)$ *resolution closure* de S
- baseia-se em cláusulas

regra da resolução

$$\frac{l_i \vee \dots \vee l_k, \quad m_1 \vee \dots \vee m_n}{l_i \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n}$$

l_i e m_j são literais complementares

$$\text{Exemplo: } \frac{P_{1,1} \vee P_{3,1}, \quad \neg P_{1,1} \vee \neg P_{2,2}}{P_{3,1} \vee \neg P_{2,2}}$$

Exemplo resolução no mundo do Wumpus

R_1, \dots, R_{10}

$R_{11}: \neg B_{1,2}$

$R_{12}: B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$

$R_{13}: \neg P_{2,2}$

$R_{14}: \neg P_{1,3}$

$R_{15}: P_{1,1} \vee P_{2,2} \vee P_{3,1}$

$R_{16}: P_{1,1} \vee P_{3,1}$

$R_{17}: P_{3,1}$

W?			
A S ok			
v ok	B v ok	P?	

4	Stench	Breeze	PIT	
3	Wumpus	Breeze, Stench, Gold	PIT, Breeze	
2	Stench	Breeze		
1	START	Breeze	PIT, Breeze	
	1	2	3	4

Forma normal conjuntiva

FNC (CNF)

- conjunção de disjunções de literais
- disjunção de literais é uma cláusula

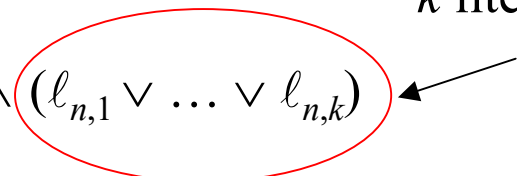
Teorema

qualquer sentença de uma linguagem proposicional é logicamente equivalente a uma conjunção de disjunções de literais

k -CNF

$$(\ell_{1,1} \vee \dots \vee \ell_{1,k}) \wedge \dots \wedge (\ell_{n,1} \vee \dots \vee \ell_{n,k})$$

k literais/cláusula



Conversão para forma normal conjuntiva

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

1. eliminar \Leftrightarrow substituir $\alpha \Leftrightarrow \beta$ 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 substituir $\alpha \Rightarrow \beta$ 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 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 (\wedge sobre \vee) e organizar

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

Algoritmo de resolução

Algoritmo de resolução

1. iniciar convertendo $(KB \wedge \neg\alpha)$ para FNC
2. aplicar regra resolução às cláusulas resultantes
3. resolver literais para produzir novas cláusulas
4. continuar processo até que
 - 4.1 se novas cláusulas não são obtidas, então $KB \not\models \alpha$
 - 4.2 se resolução produz cláusula vazia, então $KB \models \alpha$

característica do algoritmo de resolução

- prova por refutação
- $KB \models \alpha$ se e somente se $(KB \wedge \neg\alpha)$ é insatisfável

Algoritmo de resolução

function PL_RESOLUTION (KB, α) **returns** *true* or *false*

inputs: KB , a knowledge base, a sentence in propositional logic
 α , a query, a sentence in propositional logic

$clauses \leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg \alpha$

$new \leftarrow \{ \}$

loop do

for each pair of clauses C_i, C_j **in** $clauses$ **do**

$resolvents \leftarrow$ PL_RESOLVE(C_i, C_j)

if $resolvents$ contains the empty clause **then return** *true*

$new \leftarrow new \cup resolvents$

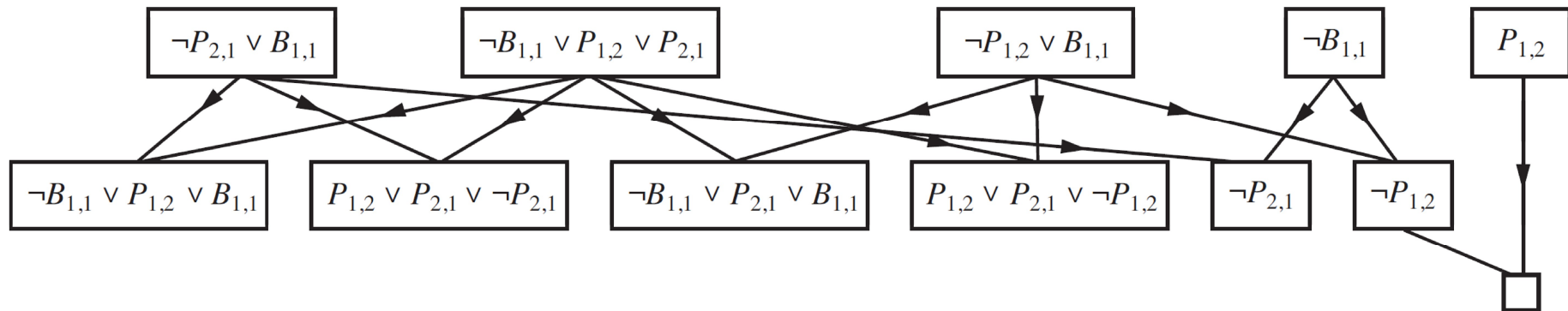
if $new \subseteq clauses$ **then return** *false*

$clauses \leftarrow clauses \cup new$

Exemplo algoritmo de resolução

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

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



resolução é consistente e completo

Encadeamento direto e reverso

forma de Horn

- KB = conjunção de cláusulas de Horn

cláusulas de Horn

- disjunção de literais onde no máximo um é positivo

exemplo:



regra de inferência: modus ponens

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

Observações

1. $(\neg B_{1,1} \vee P_{1,2}, \vee P_{2,1})$ não é cláusula de Horn
2. $(\neg L_{1,1} \vee \neg Breeze \vee B_{1,1}) \equiv (L_{1,1} \wedge Breeze) \Rightarrow B_{1,1}$
3. cláusula definida: se contém **exatamente** um literal positivo
4. $(\neg W_{1,1} \vee W_{1,2}) \equiv (W_{1,1} \wedge W_{1,2}) \Rightarrow False$ (restrição integridade)
5. inferência com cláusulas de Horn feitas via encadeamento
6. decidir consequência lógica com cláusulas de Horn é linear

Encadeamento direto (*Forward Chaining*)

Algoritmo de encadeamento direto

1. “disparar regras” sempre que premissas são satisfeitas na *KB*
2. adicionar conclusão na *KB* até
 - 2.1 encontrar consulta *q* ou
 - 2.2 todas regras foram *disparadas*

consistente e completo

Encadeamento direto (*forward chaining*)

function PL_FC_ENTAILS? (*KB*, *q*) **returns** *true* or *false*

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

q, a query, a sentence in propositional logic

count \leftarrow a table, where *count*[*c*] is the number of symbols in *c*'s premise

inferred \leftarrow a table, where *inferred*[*s*] is initially *false* for all symbols

agenda \leftarrow a queue of symbols, initially symbols known to be true in *KB*

while *agenda* is not empty **do**

p \leftarrow POP(*agenda*)

if *p* = *q* **then return** *true*

if *inferred*[*p*] = *false* **then**

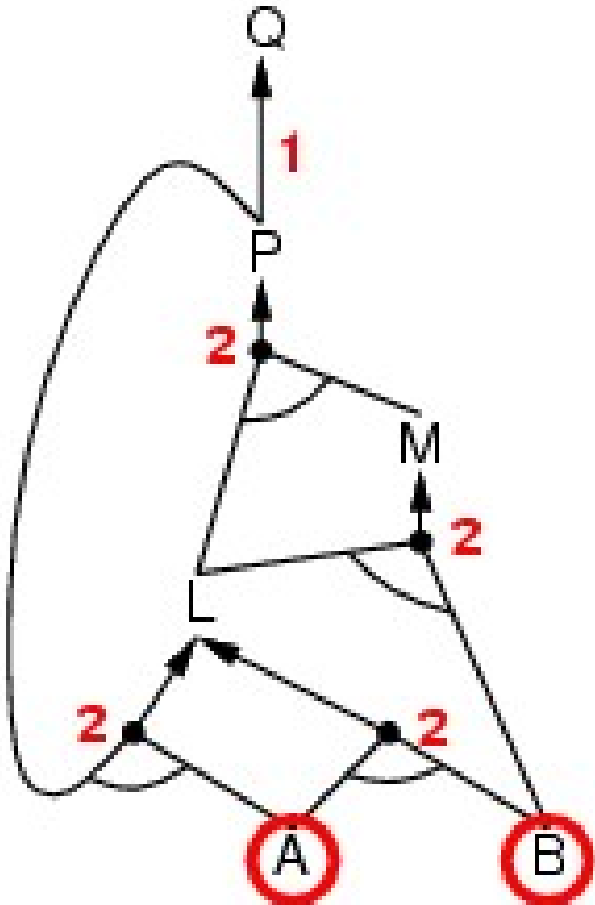
inferred[*p*] \leftarrow *true*

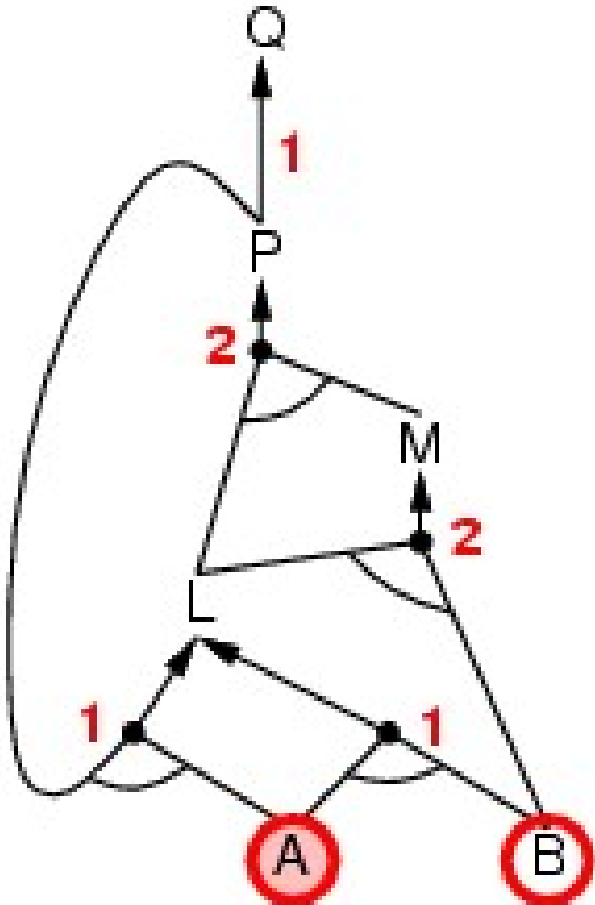
for each clause *c* in *KB* where *p* is in *c*.PREMISE **do**

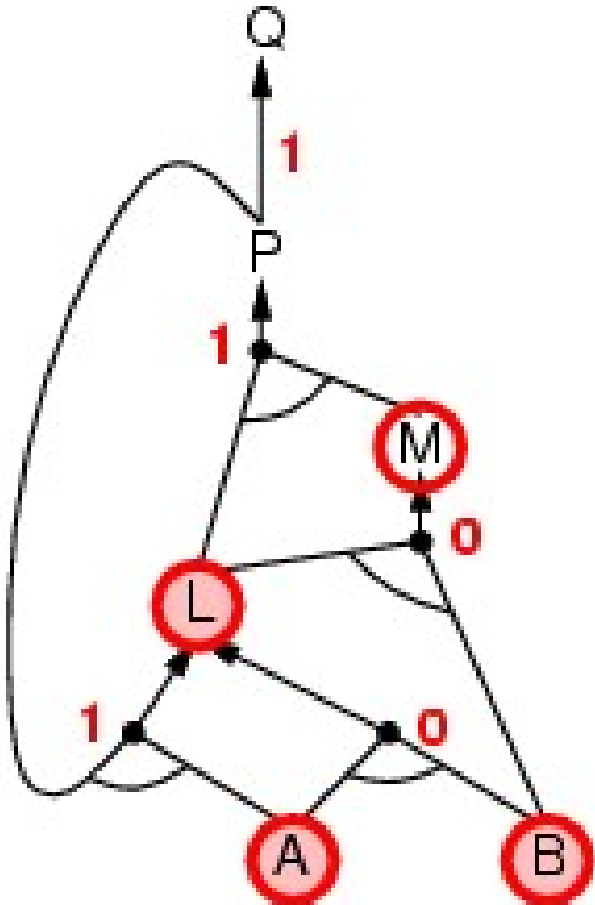
decrement *count*[*c*]

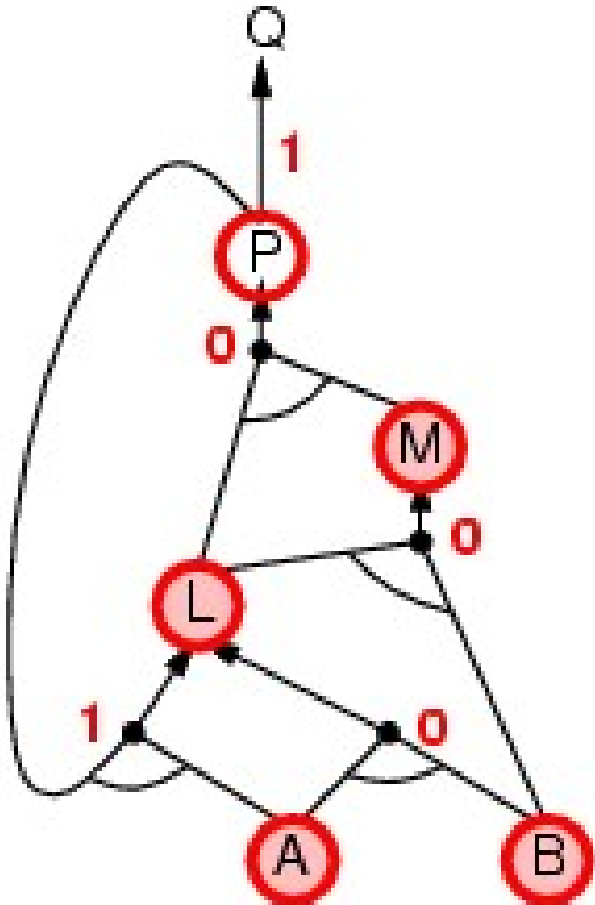
if *count*[*c*] = 0 **then** add *c*.CONCLUSION to *agenda*

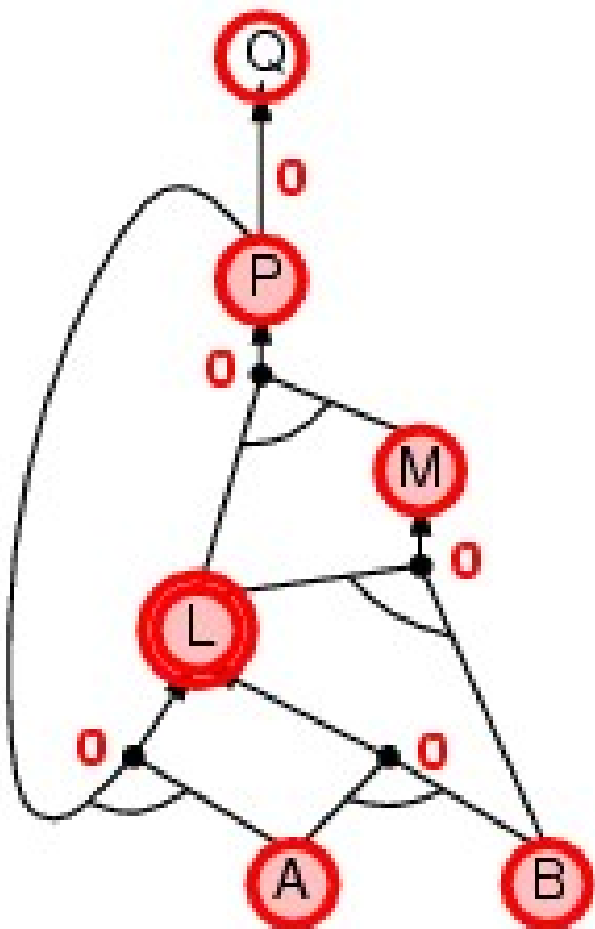
return *false*

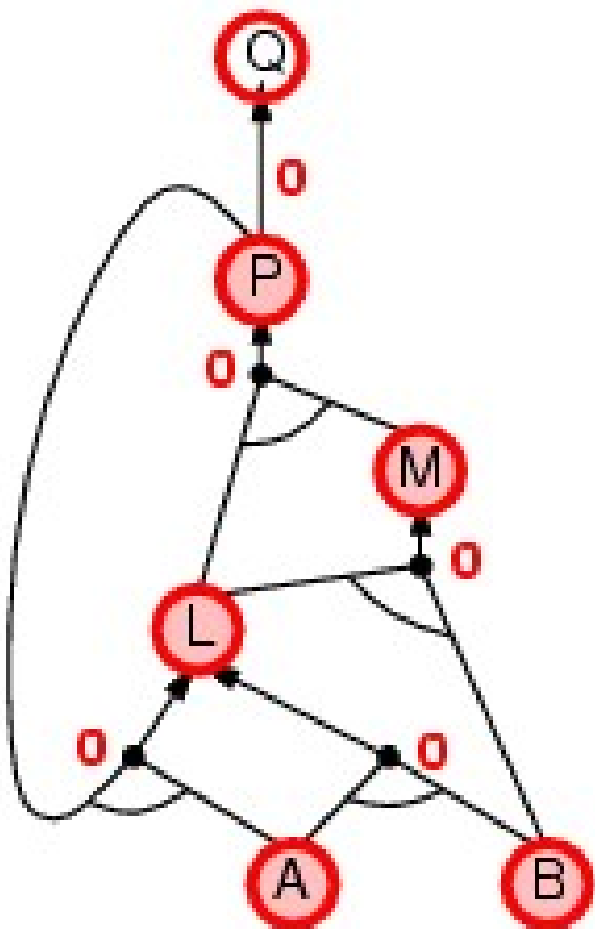


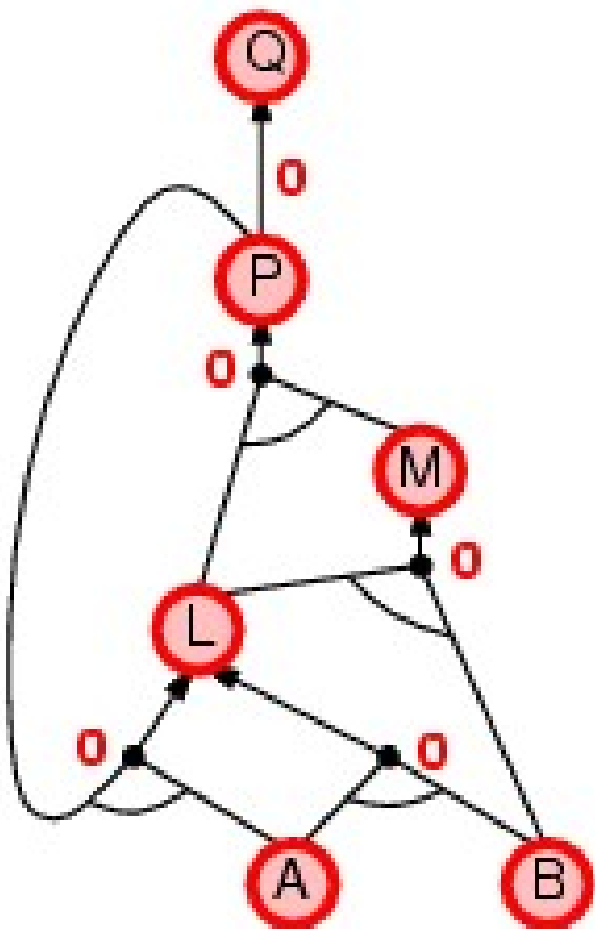












Encadeamento reverso (*backward chaining*)

Algoritmo de encadeamento reverso

1. disparar regras sempre que consequentes são satisfeitos na *KB*
2. adicionar antecedentes na *KB* até
 - 2.1 encontrar consulta *q* ou
 - 2.2 todas premissas forem provadas por fatos conhecidos

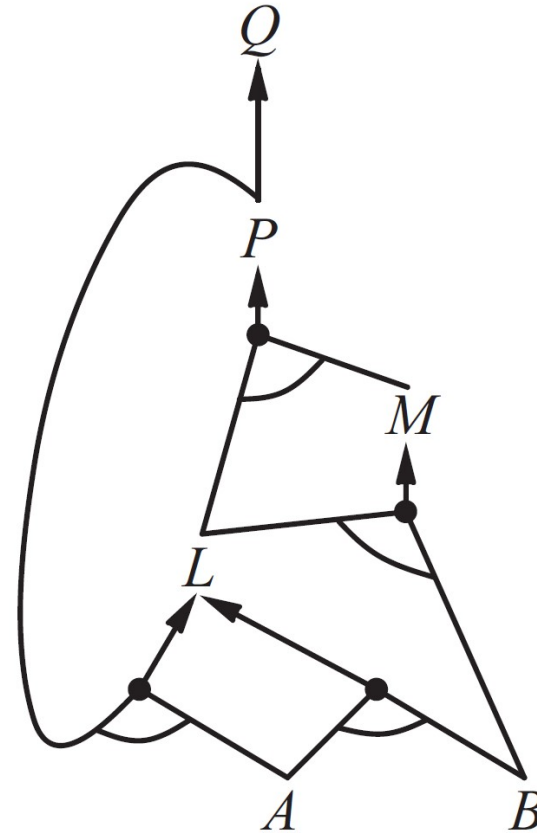
Exemplo

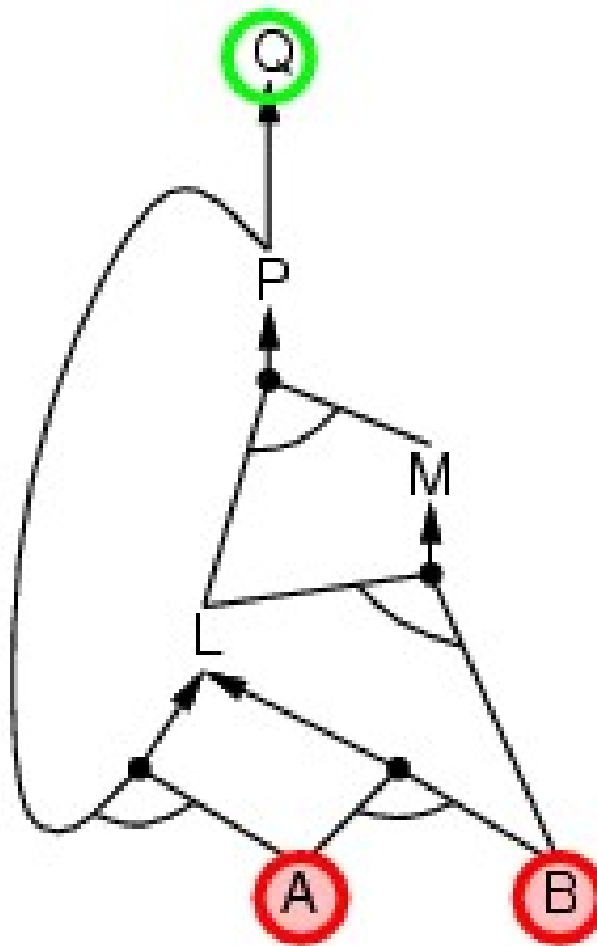
KB

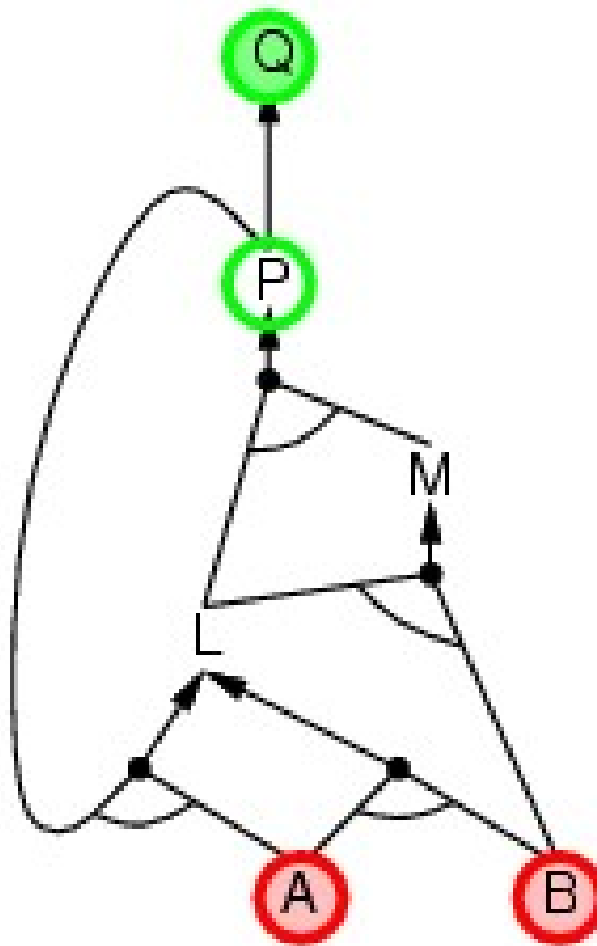
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$

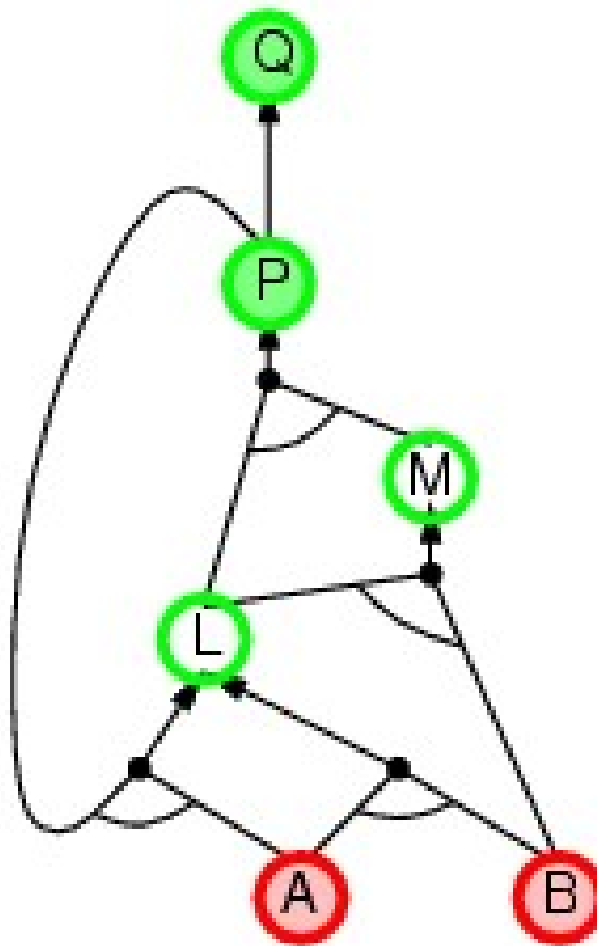
A

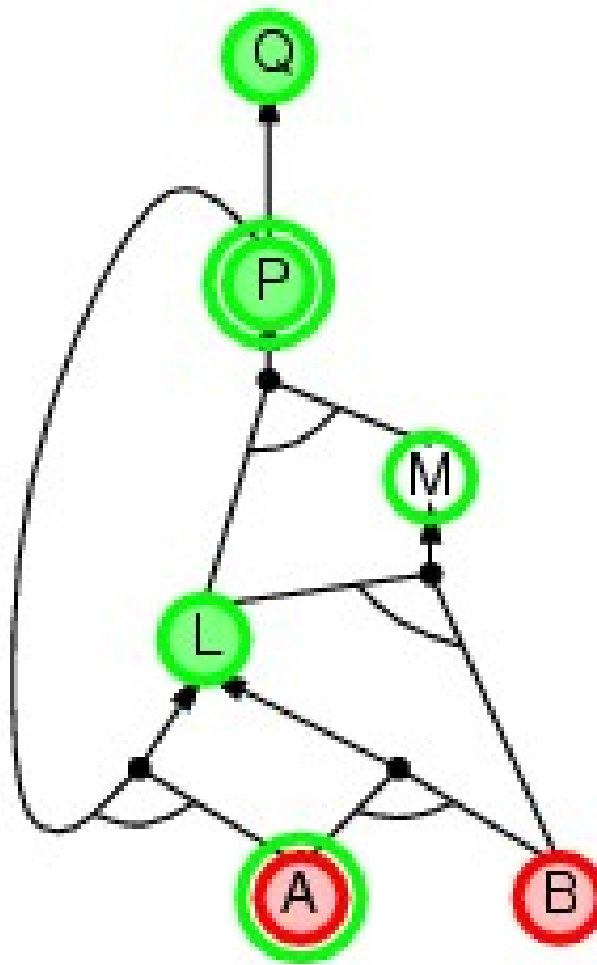
B

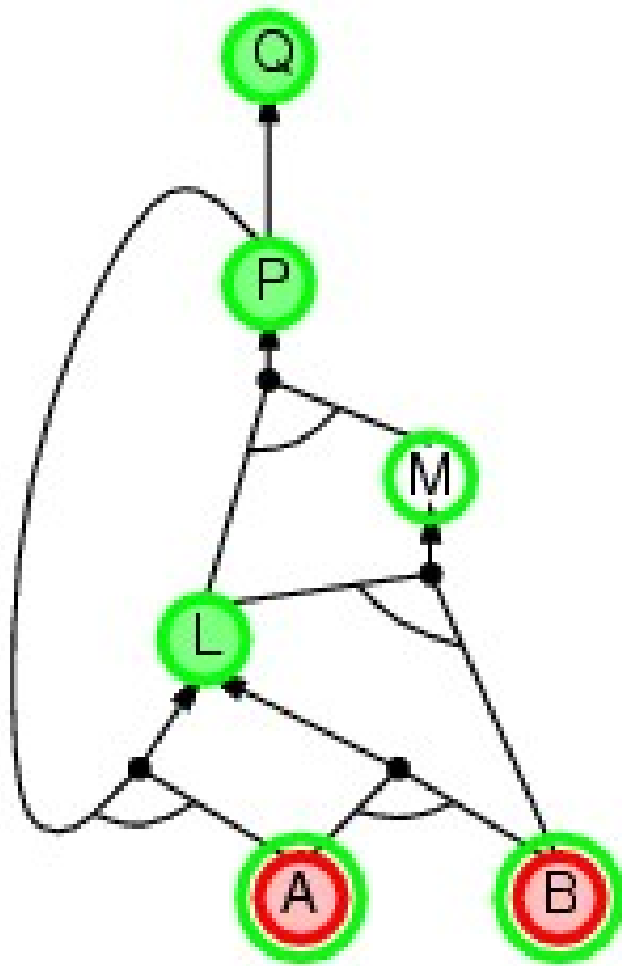


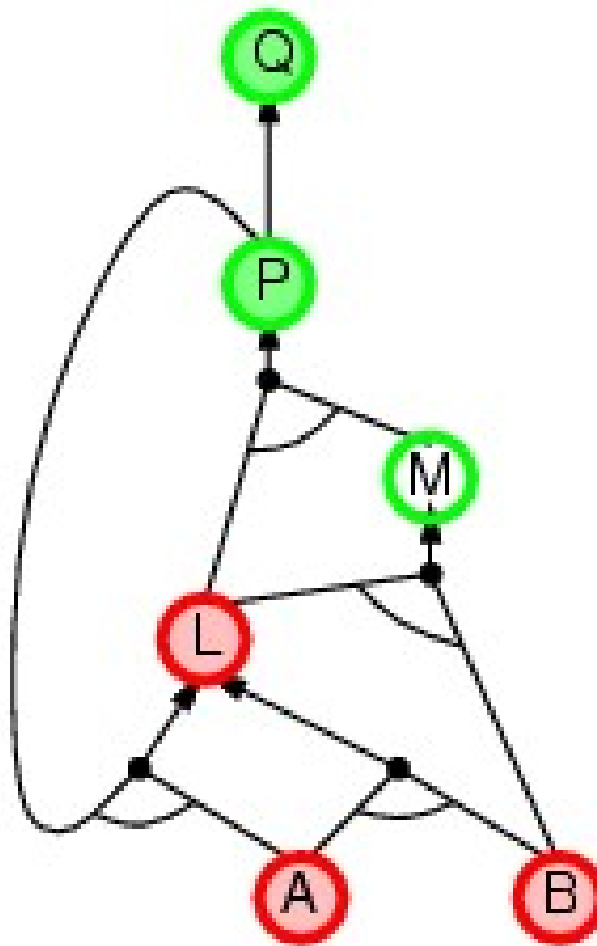


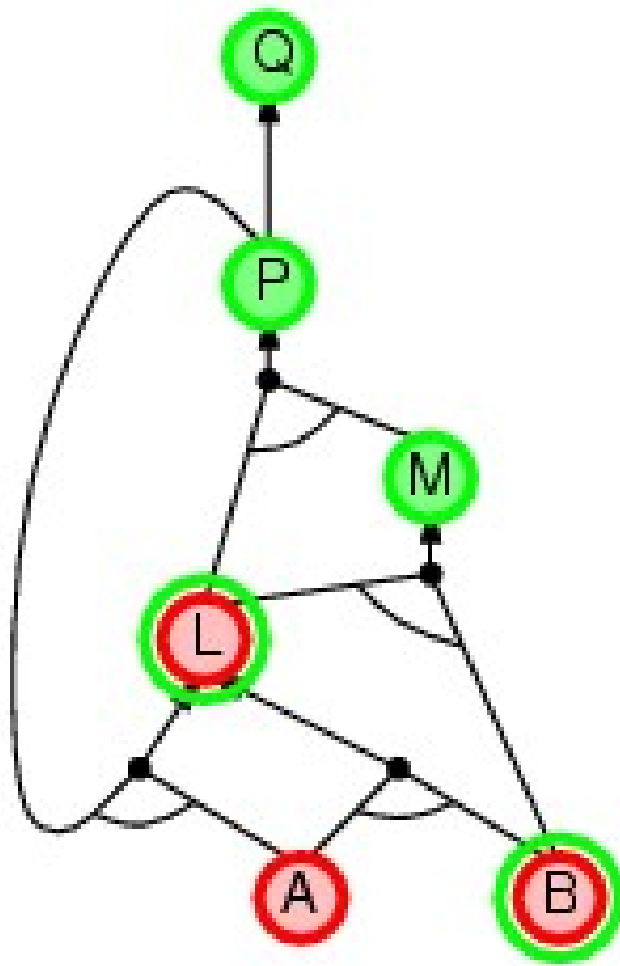


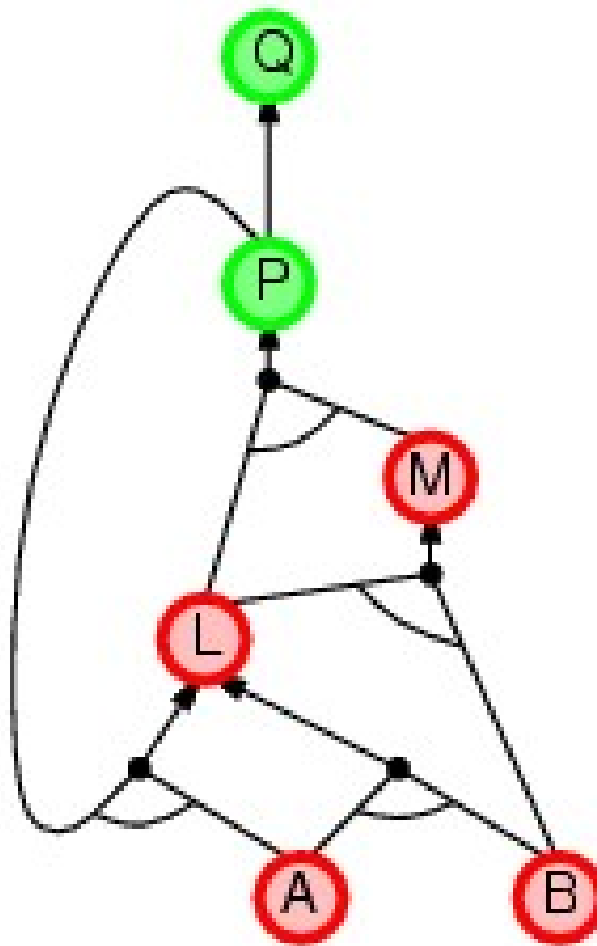


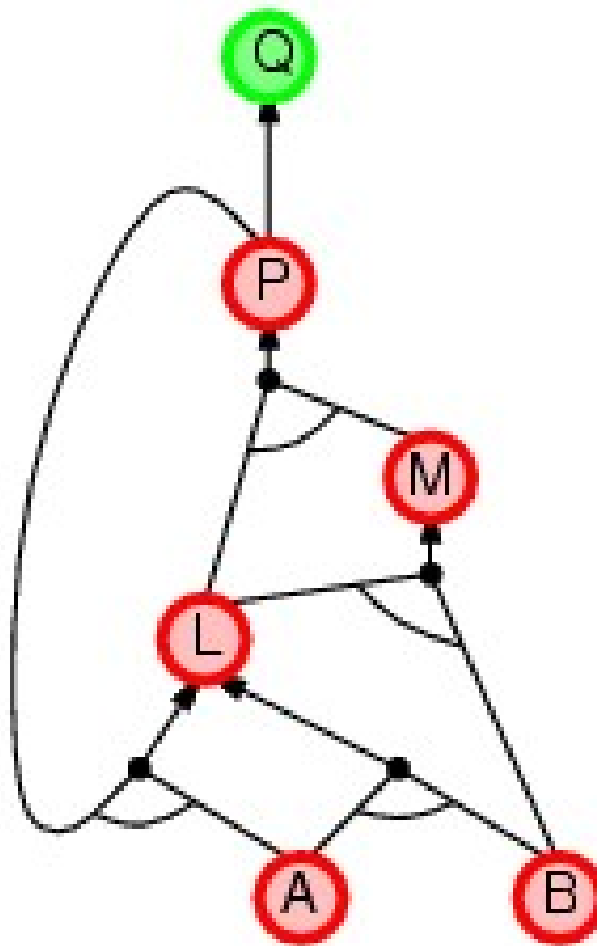


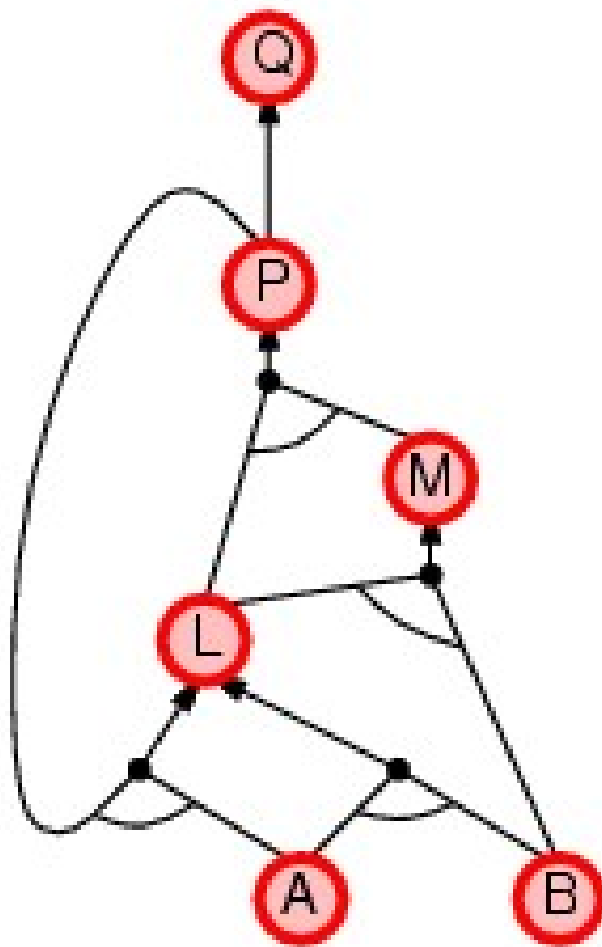












Encadeamento direto × reverso

ED é *data-driven*, automático, processamento “inconsciente”

Exemplo: reconhecimento objetos, projeto

ED pode executar passos irrelevantes para encontrar objetivo

ER é *goal-driven*, apropriado para solução de problemas

Exemplo: diagnóstico, como se tornar um pesquisador?

complexidade do ER pode ser menor que linear no tamanho da *KB*

Agente proposicional no mundo Wumpus

sentenças (PEAS)

$$\begin{aligned} &\neg P_{1,1} \\ &\neg W_{1,1} \\ &B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y}) \\ &S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y}) \\ &W_{1,1} \vee W_{1,2} \vee \dots \vee W_{4,4} \\ &\neg W_{1,1} \vee \neg W_{1,2} \\ &\neg W_{1,1} \vee \neg W_{1,3} \\ &\dots \end{aligned}$$

64 símbolos proposicionais distintos, 155 sentenças p/ mundo (4×4)

Limitações da lógica proposicional

poder de expressão limitado

KB contém as sentenças do "estado" de toda posição

para todo instante t e para toda posição $[x,y]$

$$L_{x,y}^t \wedge FacingEast^t \wedge Forward^t \Rightarrow L_{x+1,y}^t \wedge \neg L_{x,y}^t$$

proliferação rápida de cláusulas

alternativa: agente híbrido

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

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

persistent: *KB*, a knowledge base, initially the atemporal “wumpus physics”
t, a counter, initially 0, indicating time
plan, an action sequence, initially empty

TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
TELL the *KB* the temporal “physics” sentences for time *t*
 $safe \leftarrow \{[x, y] : \text{ASK}(KB, OK_{x,y}^t) = true\}$
if ASK(*KB*, $Glitter^t$) = true **then**
 $plan \leftarrow [Grab] + \text{PLAN-ROUTE}(current, \{[1,1]\}, safe) + [Climb]$
if *plan* is empty **then**
 $unvisited \leftarrow \{[x, y] : \text{ASK}(KB, L_{x,y}^{t'}) = false \text{ for all } t' \leq t\}$
 $plan \leftarrow \text{PLAN-ROUTE}(current, unvisited \cap safe, safe)$
if *plan* is empty and ASK(*KB*, $HaveArrow^t$) = true **then**
 $possible_wumpus \leftarrow \{[x, y] : \text{ASK}(KB, \neg W_{x,y}) = false\}$
 $plan \leftarrow \text{PLAN-SHOT}(current, possible_wumpus, safe)$
if *plan* is empty **then** // no choice but to take a risk
 $not_unsafe \leftarrow \{[x, y] : \text{ASK}(KB, \neg OK_{x,y}^t) = false\}$
 $plan \leftarrow \text{PLAN-ROUTE}(current, unvisited \cap not_unsafe, safe)$
if *plan* is empty **then**
 $plan \leftarrow \text{PLAN-ROUTE}(current, \{[1, 1]\}, safe) + [Climb]$
action $\leftarrow \text{POP}(plan)$
TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
t $\leftarrow t + 1$
return *action*

function PLAN-ROUTE(*current*,*goals*,*allowed*) **returns** an action sequence

inputs: *current*, the agent’s current position
goals, a set of squares; try to plan a route to one of them
allowed, a set of squares that can form part of the route

problem $\leftarrow \text{ROUTE-PROBLEM}(current, goals, allowed)$
return A*-GRAPH-SEARCH(*problem*)

Observação

Este material refere-se às notas de aula do curso EA 072 Inteligência Artificial em Aplicações Industriais da Faculdade de Engenharia Elétrica e de Computação da Unicamp. Não substitui o livro texto, as referências recomendadas e nem as aulas expositivas. Este material não pode ser reproduzido sem autorização prévia dos autores. Quando autorizado, seu uso é exclusivo para atividades de ensino e pesquisa em instituições sem fins lucrativos.