Agentes

Prof. Eric A. Antonelo

DAS-5341: Inteligência Artificial
UFSC

Agente aspirador

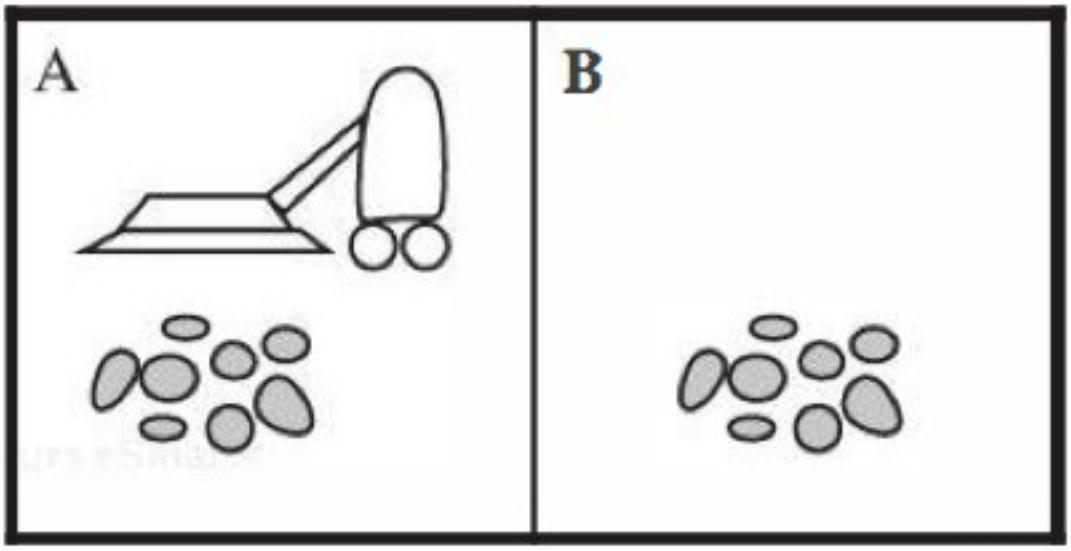


Tabela parcial para **função de agente**

Percept sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean], [A, Clean]	Right
[A, Clean], [A, Dirty]	Suck
⋮	⋮
[A, Clean], [A, Clean], [A, Clean]	Right
[A, Clean], [A, Clean], [A, Dirty]	Suck
⋮	⋮

Agente racional

- Medidas de desempenho
  - Eletricidade consumida
  - Ruído gerado

Definição
Para cada sequência de percepções possível, um agente racional deve selecionar uma ação que se espera que venha a **maximizar sua medida de desempenho**, dada a evidência fornecida pela sequência de percepções e por qualquer conhecimento interno do agente

Características desejáveis

- Exploração do ambiente (coleta de dados) *
- Aprendizagem *
- Autonomia *

video

Propriedade de ambientes de tarefa

- Completamente observável X Parcialmente observável
  - Depende se os sensores fornecem todos dados relevantes (para maximizar a função de desempenho) para uma ação.
- Determinístico X Estocástico *
  - Depende do ponto de vista (ex.: do agente)
- Episódico X Sequencial *
  - Problemas de classificação; jogar xadrez; navegação robô
- Estático X Dinâmico *
  - Ambiente (não) muda entre uma ação e outra do agente
- Discreto X Contínuo *
  - Estado do ambiente; percepções; e ações
- Agente Único X Multiagente
  - Multiagente: competitivo (jogos), cooperativo (swarm intelligence)

A estrutura de agentes

Agente = arquitetura + programa

Programa do Agente dirigido por tabela

Percept sequence	Action
[*A*, *Clean*]	*Right*
[*A*, *Dirty*]	*Suck*
[*B*, *Clean*]	*Left*
[*B*, *Dirty*]	*Suck*
[*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮

função Agente-Tabela(*percepção*) **retorna** *ação*
estática:*sequência*, inicialmente uma sequência vazia
tabela, uma tabela indexada por sequência de *percepção*, inicialmente vazia

insira *percepção* final *sequência*
ação <- procura_tabela(*sequência*, *tabela*)
retorna *ação*

Tipos de Agentes

- Agentes reativos simples
- Agentes reativos baseados em modelos (ou com estado interno)
- Agentes baseados em objetivos
- Agentes baseados na utilidade

Tipos de Agentes

- **Agentes reativos simples**
  - Regra Condição-Ação
  - Leva em conta somente percepção atual

```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

Ambientes completamente observáveis

Funciona?

?

Ambientes parcialmente observáveis

Funciona?

Tipos de Agentes

- **Agentes reativos simples**
  - Regra Condição-Ação
  - Leva em conta somente percepção atual

```
function SIMPLE-REFLEX-AGENT(percept) returns an action
  persistent: rules, a set of condition–action rules

  state ← INTERPRET-INPUT(percept)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

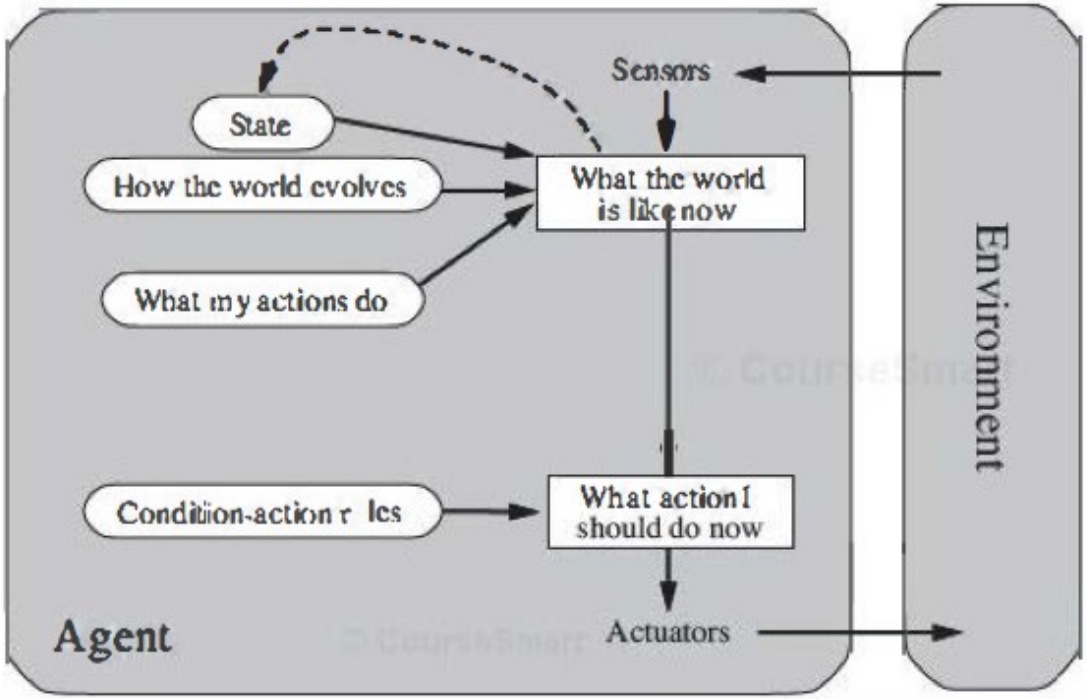
Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

Ambientes parcialmente observáveis → **Loops** (ex.: sem sensor de posição)

Ações aleatórias resolve? → **Agente Racional?**

Tipos de Agentes

- Agentes reativos simples
- **Agentes reativos baseados em modelos (ou com estado interno)**



Controlar a parte do mundo que não pode ver agora

Ambientes parcialmente observáveis

Ex:
Motorista de táxi (frear)
Oclusão

Tipos de Agentes

- Agentes reativos simples
- **Agentes reativos baseados em modelos (ou com estado interno)**

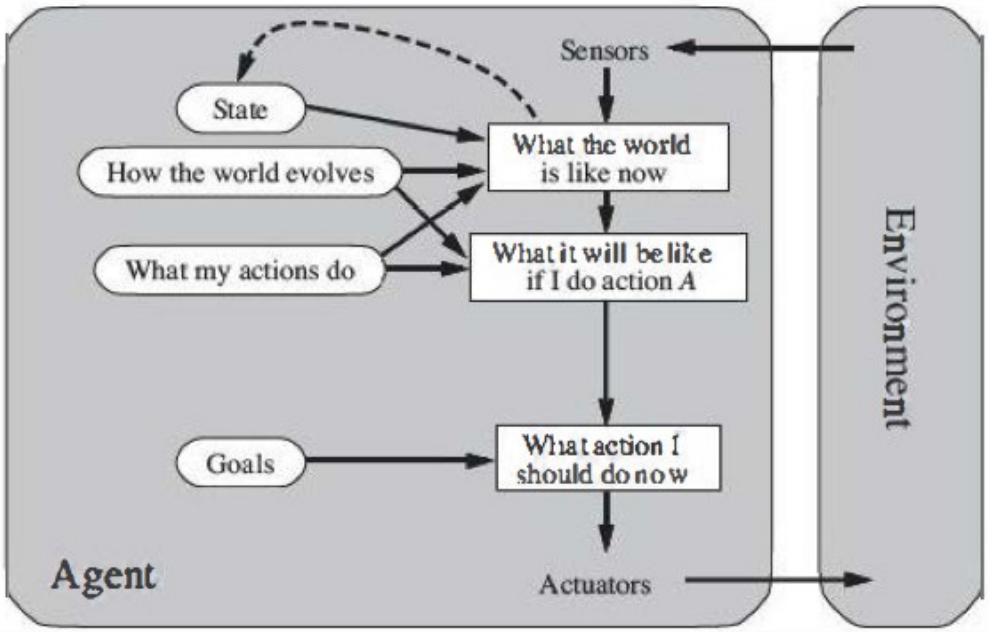
```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              model, a description of how the next state depends on current state and action
              rules, a set of condition–action rules
              action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept, model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

Ambientes parcialmente observáveis

Tipos de Agentes

- Agentes reativos simples
- Agentes reativos baseados em modelos (ou com estado interno)
- **Agentes baseados em objetivos**



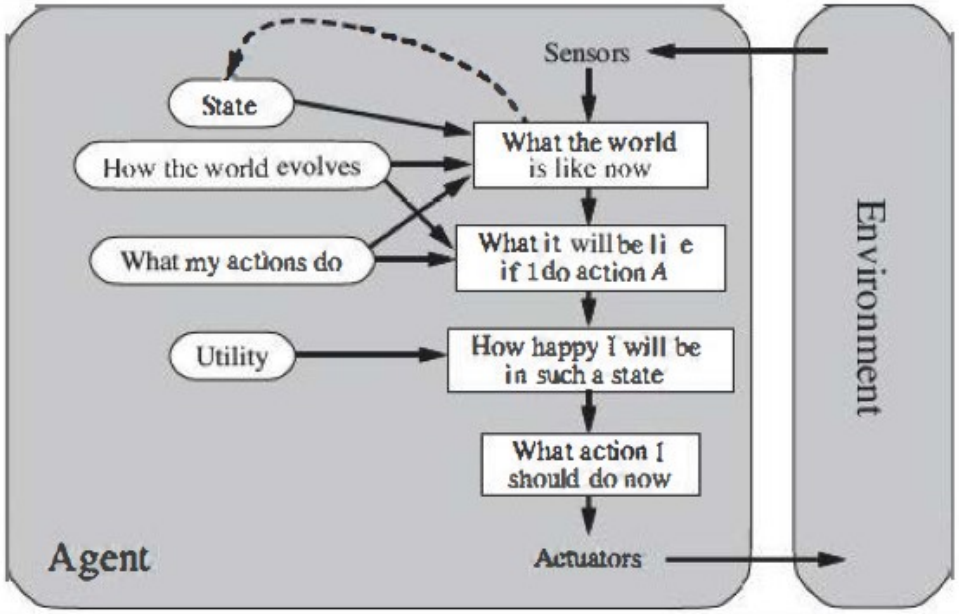
Diferentes de regras **se-então**

Busca
e
Planejamento

- **Busca por estados futuros**
- **Mais flexível**

Tipos de Agentes

- Agentes reativos simples
- Agentes reativos baseados em modelos (ou com estado interno)
- Agentes baseados em objetivos
- **Agentes baseados na utilidade**
  - Teoria da probabilidade + utilidade

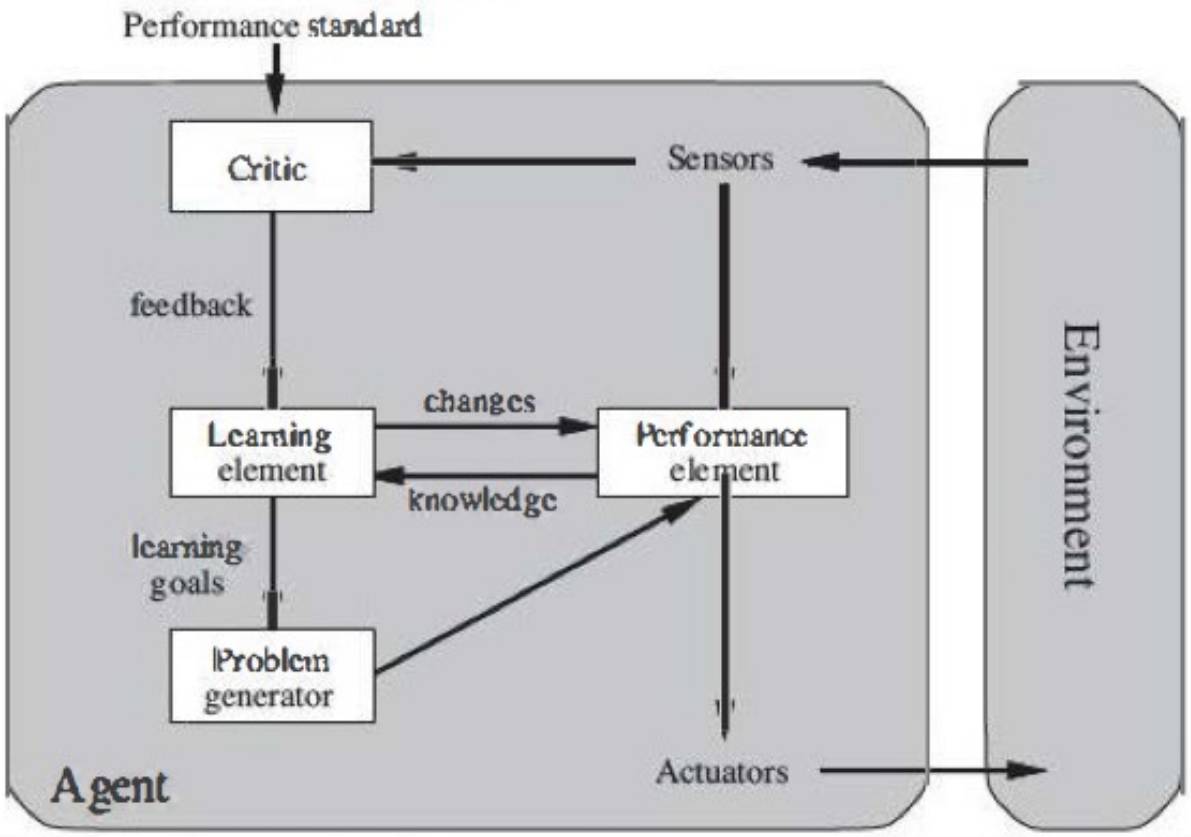


Teoria da Decisão

Ambientes estocásticos e parcialmente observáveis

Função de utilidade
- Ex.: Compromisso entre velocidade e segurança

Agente com aprendizagem



Exemplos de tipos de agentes

Tipo	Percepção	Ações	Objetivos	Ambiente
Diagnóstico médico	Sintomas, descobertas, respostas do paciente.	Questões, exames, e tratamentos	Saúde do paciente, minimização de custos	Paciente, Hospital
Análise de imagens	Cor, resolução	Classificação da cena	Classificação correta	Imagens de satélite
Robô manipulador	Pontos de imagem	Partes a serem manipuladas	Movimento correto	Esteira
Controlador de refinaria	Temperatura, pressão	Válvulas, ajuste de temp.	Segurança, nível de pureza	Refinaria
Tutor de Inglês inteligente	Palavras digitadas	Sugestões de exercícios	Maximizar acertos aluno	Grupos de alunos

Questões

- Defina com suas próprias palavras os termos a seguir: agente, função de agente, programa de agente, agente reativo, agente baseado em modelo, agente baseado em objetivos, agente baseado em utilidade, agente com aprendizagem.

Agentes baseados em conhecimento

Agentes baseados em conhecimento

Agentes que pode formar **representações do mundo**, usar um **processo de inferência** para **derivar novas representações** sobre o mundo e utilizar essas novas representações para deduzir o que fazer.

Quadro **

Esqueleto de agente baseado em conhecimento

função KB-Agente(*percepção*) **retorna** *ação*

estático: KB, uma base de conhecimento

t, um contador que indica o tempo

Tell(KB,constrói-sentença-percepção(*percepção*))

ação <- **ASK**(KB,constrói-pergunta-ação(*t*))

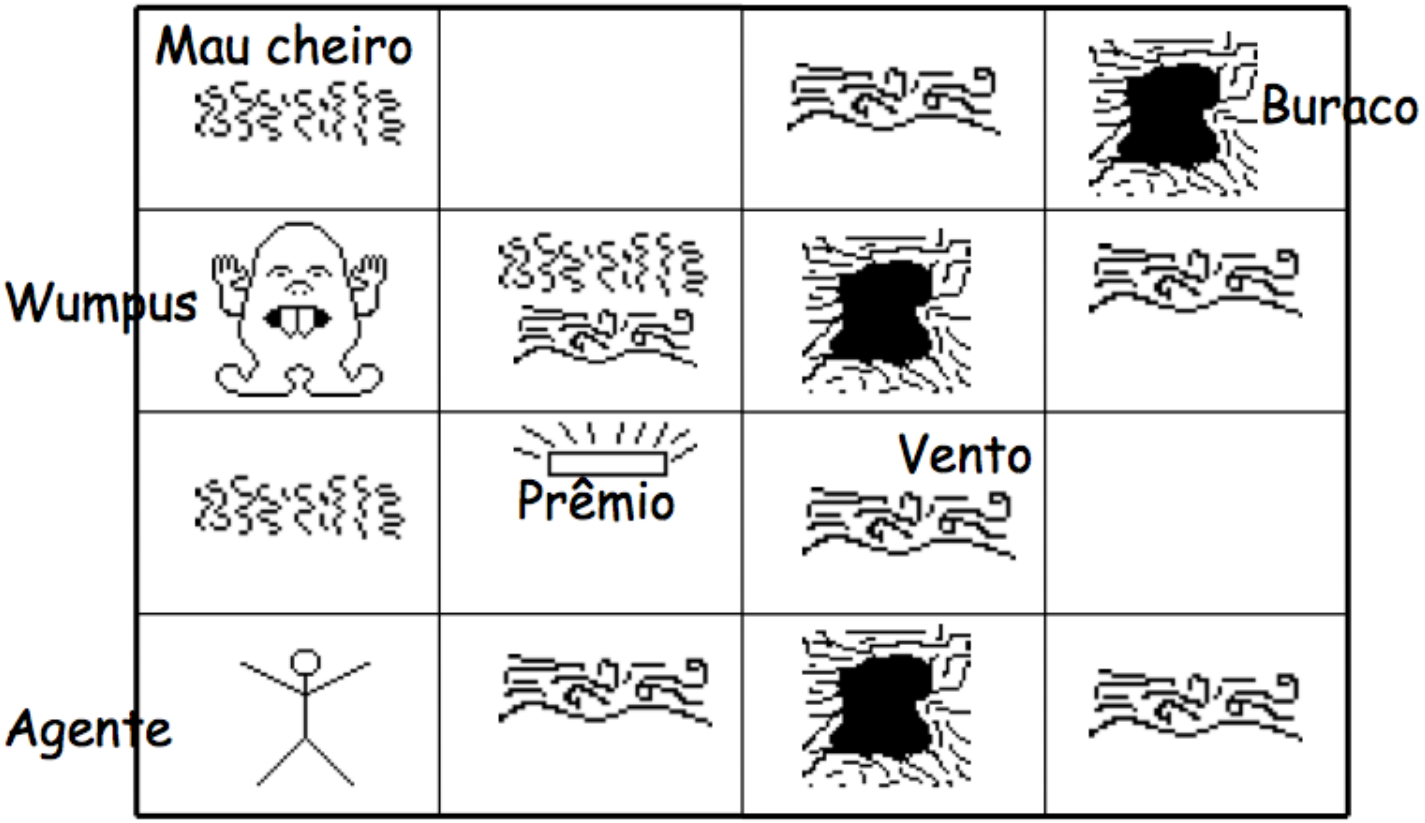
Tell(KB,constrói-sentença-ação(*ação*,*t*))

t <- *t* + 1

retorna *ação*

Paradigma Declarativo VS Paradigma Procedural

Mundo de Wumpus **



Lógica

- **Sintaxe** e Semântica
  - Sentenças são construídas de acordo com a **sintaxe** da linguagem de representação

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} &\rightarrow \textit{True} \mid \textit{False} \mid P \mid Q \mid R \mid \ldots \\
\textit{ComplexSentence} &\rightarrow (\textit{Sentence}) \mid [\textit{Sentence}] \\
&\mid \neg \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \\
&\mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\
&\mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
\text{OPERATOR PRECEDENCE} &: \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{aligned}
$$

Página 195 do livro
Russell & Norvig, Inteligência Artificial

Lógica

- Sintaxe e **Semântica**
  - Sentenças são construídas de acordo com a sintaxe da linguagem de representação
  - A **semântica** define a **verdade** de cada sentença com relação a cada **mundo possível**
- **Mundo possível = Modelo**
  - Uma atribuição de valores (A=F,B=V,C=V,D=F) para cada proposição na seguinte sentença:
    A ^ B ^ C => D

Página 195 do livro
Russell & Norvig, Inteligência Artificial

Lógica

- Consequência Lógica $\alpha \models \beta$
  - Ex.: x+y = 4 tem como consequência lógica 4 = x + y
  - Para todo modelo em que x+y = 4 (ex.: x=2, y=2), 4 = x + y é Verdadeiro.

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

Que modelo satisfaz?

$\alpha = A \wedge B$

Quadro **

Lógica

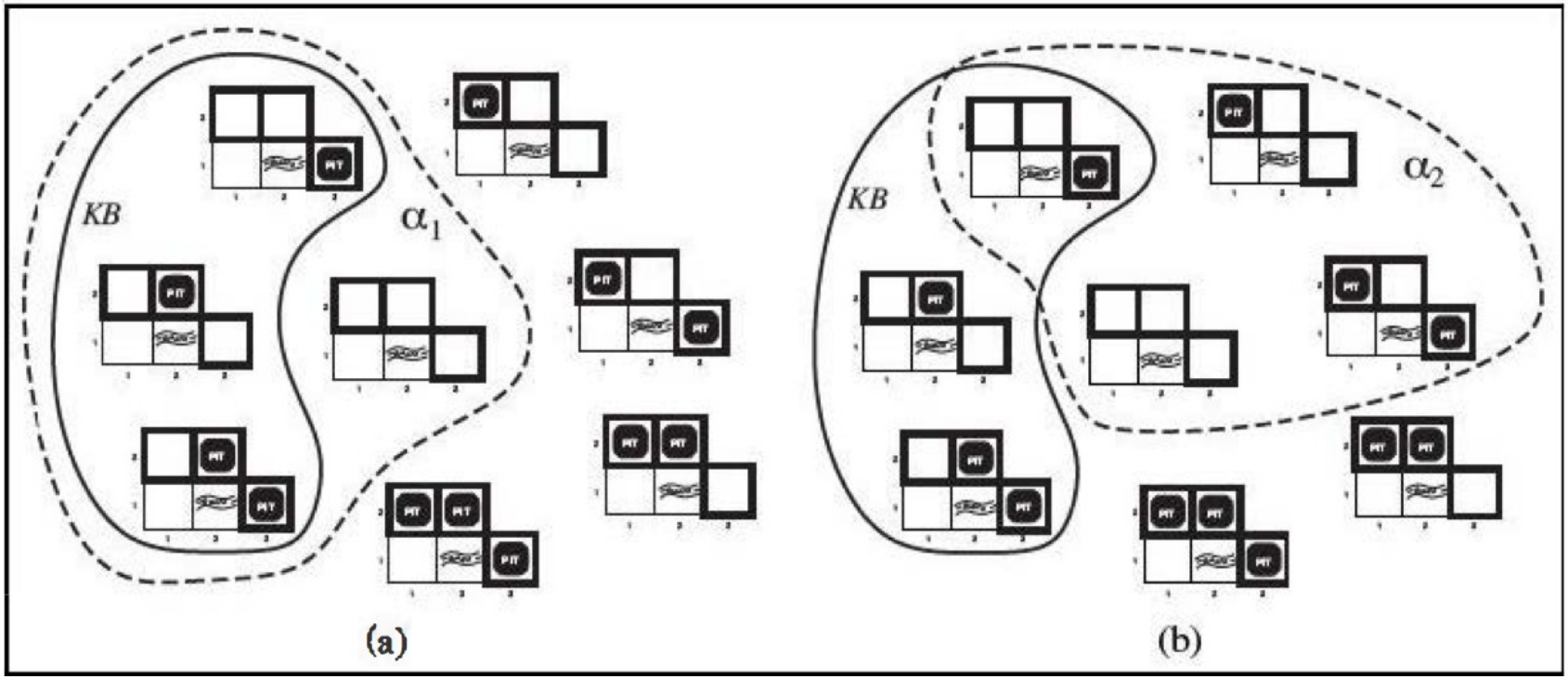
- Consequência Lógica

  Ex.:
  - O agente detectou nada em [1,1] e uma brisa em [2,1].
  - BC (base de conhecimento): percepções do ambiente + regras do mundo de Wumpus
  - Quadrados [1,2], [2,2] e [3,1] contém poços?

Quadro **

Consequência Lógica e Verificação de Modelos

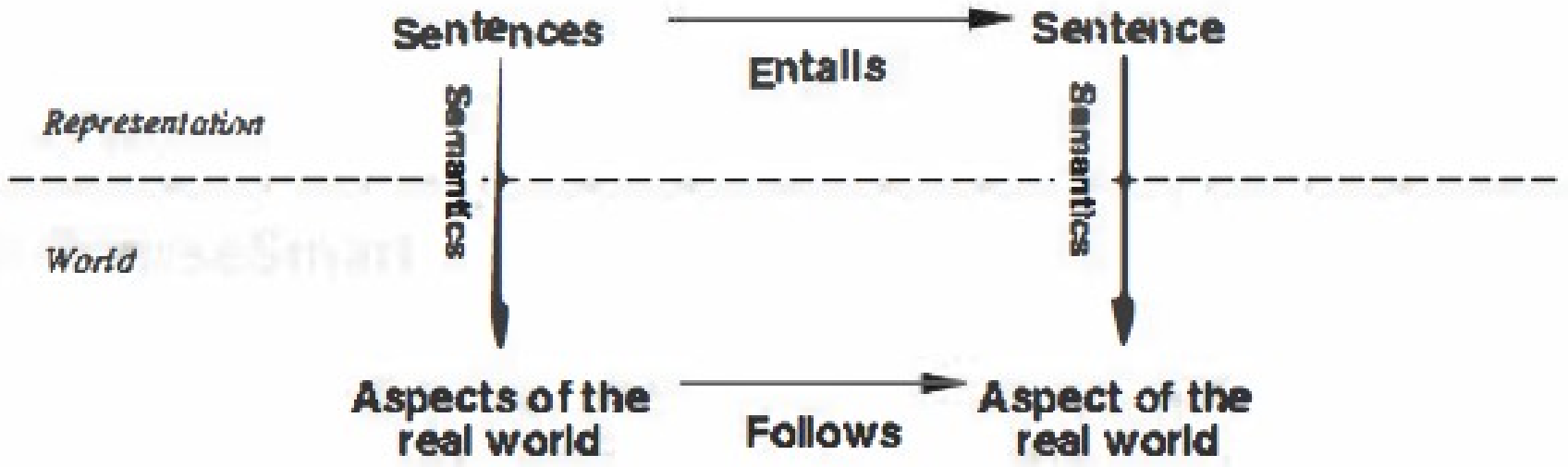
- Geramos os modelos possíveis para os quadrados [1,2], [2,2] e [3,1]
- Marcamos α_1=”não existe poço em [1,2]”, α_2=”não existe poço em [2,2]” na Figura:



Consequência Lógica e Verificação de Modelos

- Consequência Lógica
  - Pode ser usada para **inferência lógica**: derivar conclusões
  - O algoritmo de inferência mostrado na figura anterior é chamado de **verificação de modelos:**
    - **Pois enumera todos possíveis modelos para verificar que uma sentença é verdadeira em todos os modelos em que BC é verdadeira.**

Semântica e dedução lógica



Raciocionio lógico deve garantir
a validade no mundo real

Problema de lógica

"Se o unicórnio é mítico, então é imortal; porém, se ele não é mítico, então é um mamífero mortal. Se o unicórnio é imortal ou um mamífero, então ele tem chifre. O unicórnio é mágico se tem chifre.”

Descobrir, a partir das sentenças anteriores, se (a) o unicórnio é mítico, (b) o unicórnio é mágico, e (c) o unicórnio tem chifre, dado o seguinte conjunto de sentenças: