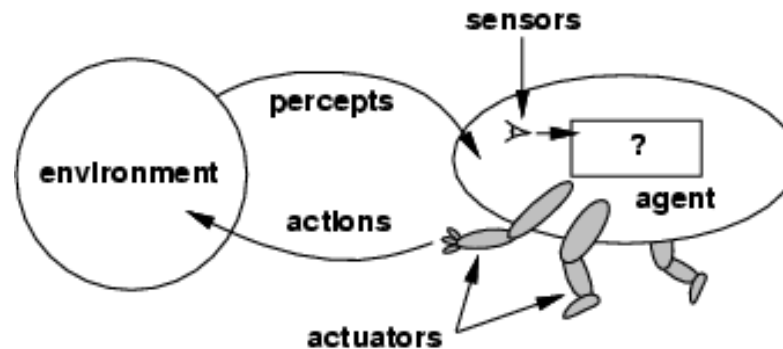


Agir Racionalmente - Agentes

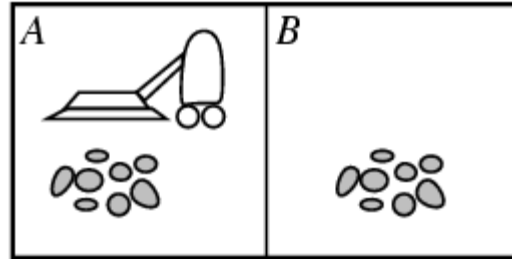
- Um agente pode ser considerado uma entidade que **percebe o ambiente** através de seus sensores e **age** através de atuadores
- Exemplos:
 - Seres Humanos
 - Visão, tato, etc... / Mãos, pernas, etc...
 - Robôs
 - Câmera, laser, etc... / Motores para atuação
 - Agente de Software
 - Input (teclado, mouse, rede) / Output (tela, rede, etc...)

Agentes



- **Percepts:** entrada em um determinado momento
- **Actions:** normalmente depende da seqüência de percepts até aquele momento
- Existe uma função que faz esse “mapeamento”
- Essa função vai ser implementada por um programa
 - Função: abstrato, modelo matemático do comportamento
 - Programa: concreto, roda na arquitetura específica

Exemplo: Vacuum-Cleaner World



- Características:
 - ❑ Quadrados A e B que podem estar sujos ou não
 - ❑ Percepção: **A, B** e **sujo, limpo** – [p1, p2]
 - ❑ Ações: **aspira, direita, esquerda, nop**
- Função simples:
 - ❑ Se o quadrado está sujo, limpe-o, senão mova para o outro quadrado

Exemplo: Vacuum-Cleaner World

Percepção	Ação
[A, limpo]	Direita
[A, sujo]	Aspira
[B, limpo]	Esquerda
[B, Sujo]	Aspira

- Essa é a forma correta de preencher a tabela?
- Esse agente está sendo **racional**?

Agentes Racionais

- O agente deve tentar fazer a “coisa certa”
- O que significa fazer a “coisa certa”?
 - Aquilo que faz o agente ter o maior grau de sucesso
- Medida de Desempenho
 - Algum critério para medir o sucesso do agente
 - Normalmente tem que ser algo objetivo
 - Exemplo:
 - Maximizar a quantidade de sujeira removida por dia, reduzir o consumo de eletricidade, manter o chão limpo por mais tempo.
 - Regra Geral: a medida de desempenho deve considerar o que se deseja no ambiente, e não o que se deseja que o agente faça.

Agentes Racionais

Para cada seqüência de percepções, um agente racional deve escolher a ação que espera maximizar a sua medida de desempenho, considerando a percepção corrente do ambiente e algum eventual conhecimento prévio

- Isso envolve:
 - A seqüência de percepções até o momento
 - As ações que ele pode fazer
 - A medida de desempenho
 - Algum conhecimento prévio do ambiente

Agentes Racionais

- Racionalidade \neq Onisciência
 - Racionalidade não necessariamente leva a perfeição
 - Racionalidade maximiza o desempenho esperado enquanto perfeição é a maximização do desempenho real
 - Mas isso não significa dizer que o agente pode ser burro
- O agente pode fazer ações de forma a melhorar a sua percepção
 - *Information Gathering, Exploration, etc*
- Aprendizado: incorporar percepções
- Autonomia*: o agente depende mais da sua própria “experiência” do que em conhecimento prévio

* Na verdade o conceito de autonomia é muito mais complexo

Ambiente: PEAS

- **Performance,
Environment,
Actuators,
Sensors**
- **Exemplo: Taxi autônomo**
 - P: rápido, seguro, confortável,
 - E: ruas, sinais, pedestres, passageiros
 - A: direção, acelerador, freio, buzina
 - S: Cameras, Sonar, GPS, IMU

PEAS

- **Agente: Sistema de Diagnóstico Médico**
- P: Paciente saudável, minimizar custos
- E: Paciente, Hospital, Funcionários
- A: Tela do Computador (questões, testes, receitas, etc)
- S: Teclado (entrada de dados sobre os sintomas, respostas dos pacientes)

PEAS

- **Agente: Robô separador de peças**
- P: Peças nas caixas corretas
- E: Esteira com as peças, caixas
- A: Braço robótico, garra
- S: Câmera, sensores de juntas

Propriedades dos Ambientes

- Completamente Observável (vs. Parcialmente)
 - Os sensores dão acesso ao estado completo do ambiente em qualquer ponto do tempo
- Determinísticos (vs. Estocásticos)
 - O próximo estado do ambiente é completamente determinado pelo estado corrente em conjunto com a ação executada pelo agente
 - Estratégico: determinístico, a não ser pelas ações de outros agentes
- Episódico (vs. Seqüencial)
 - A experiência do agente é dividida em episódios que consistem em perceber e executar uma ação. As ações não dependem de episódios anteriores (nem afetam futuros)

Tipos de Ambientes

- Estáticos (vs. Dinâmicos)

- O estado do ambiente não se altera enquanto o agente decide o que fazer
 - Semidinâmico: o ambiente não muda com o tempo mas o valor do desempenho do agente muda

- Discreto (vs. Contínuo)

- Pode ser aplicado ao estado do ambiente, tempo, e percepções / ações

- Agente único (vs. Múltiplos agentes)

- O agente esta sozinho no ambiente
 - Múltiplos agentes: Competitivo x Colaborativo

Tipos de Ambientes

Aplicação	Rest 1	Xadrez	Robô
Observável			
Determinístico			
Episódico			
Estático			
Discreto			
Agente Único			

Agentes: funções e programas


- Funções e Programas
 - Função modela o comportamento
 - O programa é a implementação da função
- Arquitetura: “dispositivo” com sensores e atuadores aonde o programa vai executar
- Agente = Arquitetura + Programa
 - O programa e a arquitetura são relacionados

Esqueleto Básico de um programa

- O programa recebe um *percept* dos sensores e retorna uma ação para os atuadores
- O *percept* é a informação corrente com base no estado do ambiente
 - Se for necessário informações antigas, o programa deve guardá-las na memória

Tipos Básicos de Programas

Mais
Sofisticados

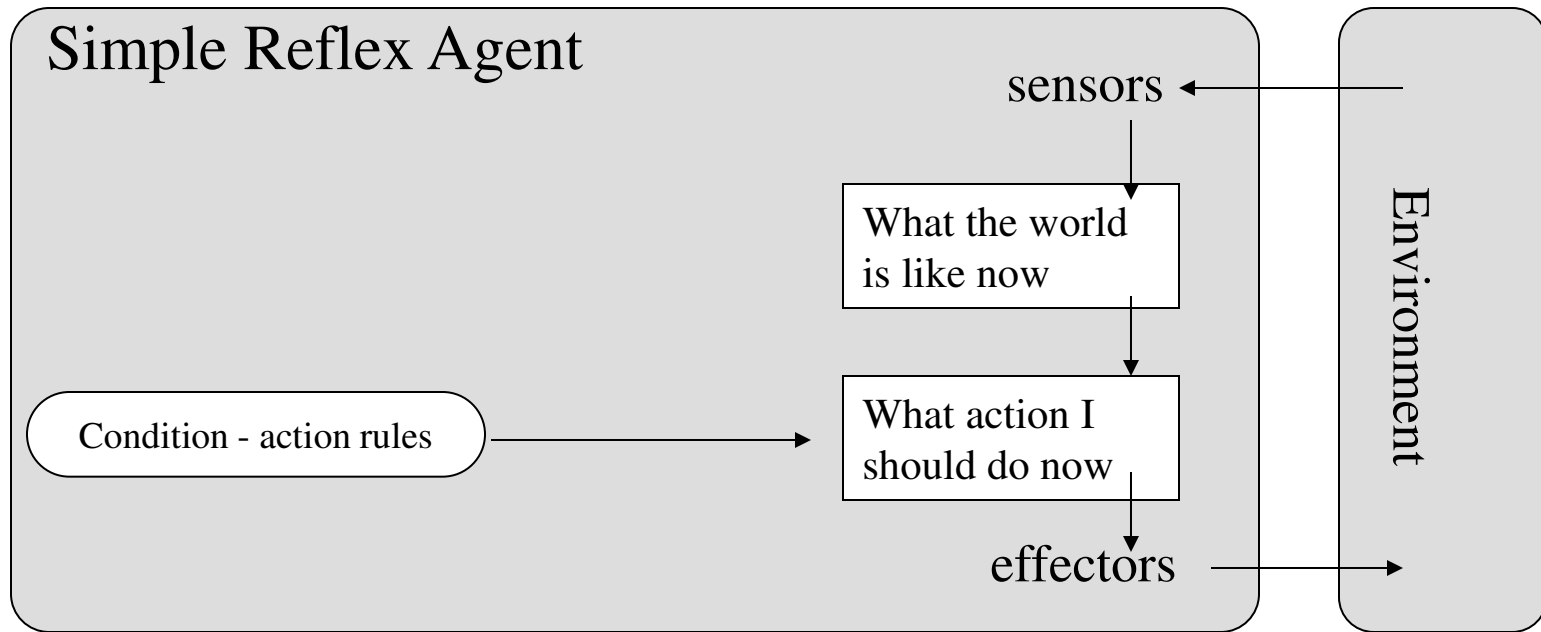
- 
1. Simple reflex agent
 2. Reflex agent with internal state
 3. Agent with explicit goals
 4. Utility-based agent

Simple Reflex-Agent

- Decide o que fazer com base na percepção corrente, ignorando a história
- *Lookup Table*: Regras de Condição – Ação
- Exemplo - Aspirador de Pó

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

Simple Reflex-Agent



function SIMPLE-REFLEX-AGENT(*percept*) **returns** action

static: *rules*, a set of condition-action rules

state \leftarrow INTERPRET-INPUT (*percept*)

rule \leftarrow RULE-MATCH (*state*, *rules*)

action \leftarrow RULE-ACTION [*rule*]

return *action*

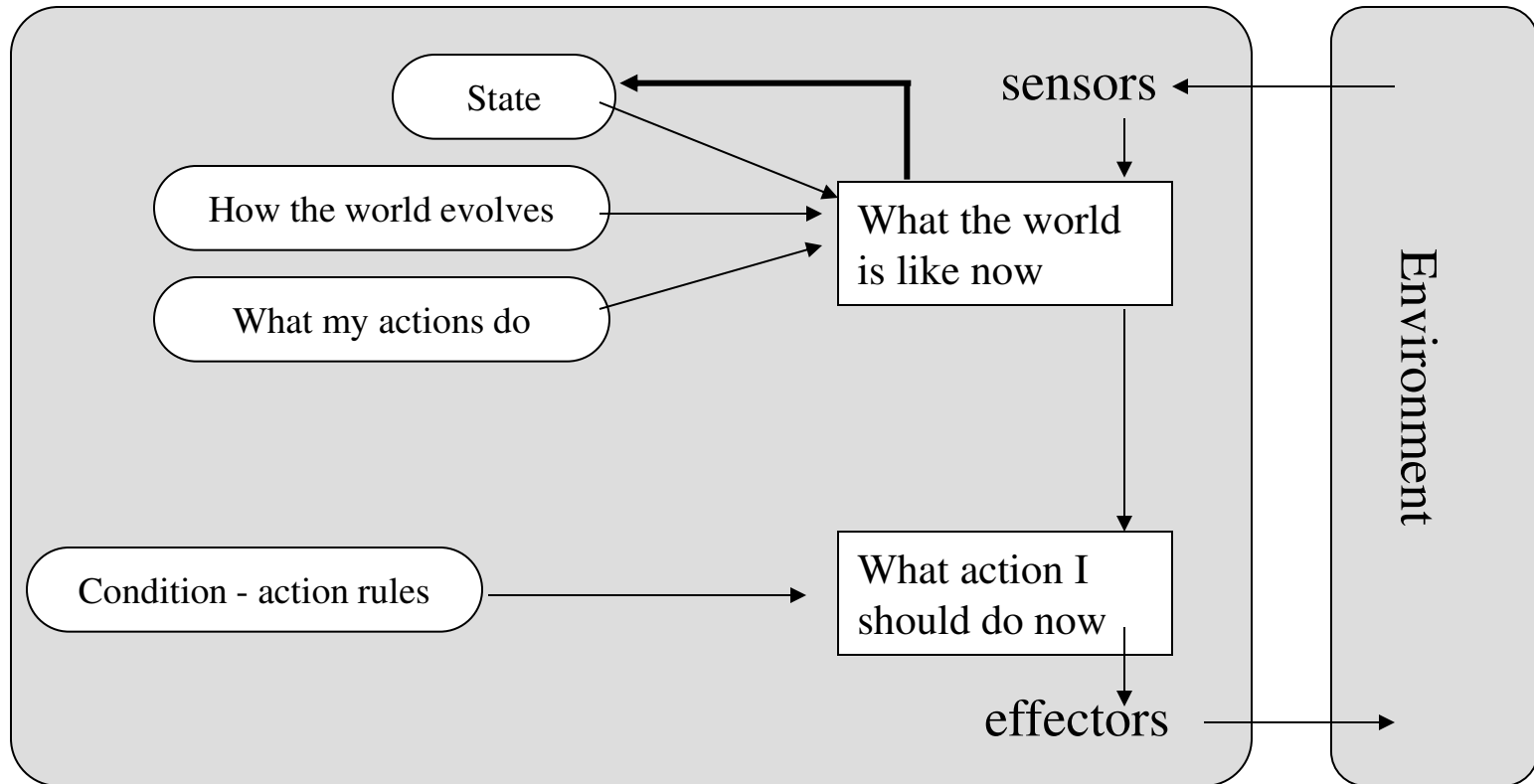
Simple Reflex-Agent

- Vantagem: são simples!
- Desvantagens:
 - ❑ Inteligência limitada
 - ❑ Tabela pode ficar grande
 - ❑ Só funcionam bem se o ambiente for completamente observável e a decisão correta puder ser tomada com base na percepção corrente
 - ❑ Problema comum em ambientes parcialmente observáveis: *loops infinitos*
 - Soluções randômicas

Model-Based Reflex Agents

- Mantém informação sobre o estado do sistema com base na seqüência de percepções e ações
- Além da sequência de percepções / ações, é necessário um modelo do mundo e modelos de como as ações interferem no mundo.

Model-Based Reflex Agents



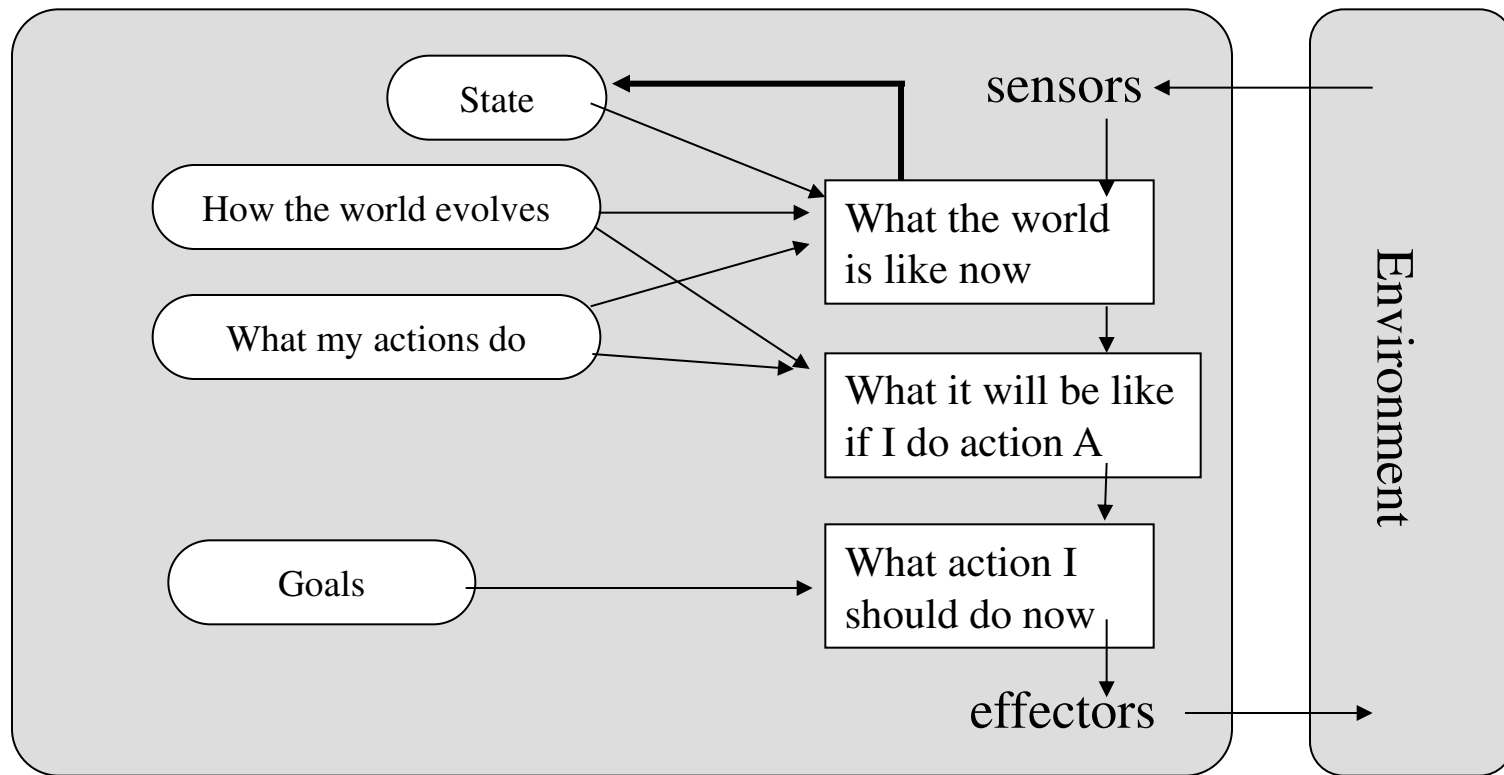
Model-Based Reflex Agents

```
function REFLEX-AGENT-WITH-STATE (percept) returns action
  static: state, a description of the current world state
         rules, a set of condition-action rules
         action, the most recent action, initially none

  state  $\leftarrow$  UPDATE-STATE (state, percept, action)
  rule  $\leftarrow$  RULE-MATCH (state, rules)
  action  $\leftarrow$  RULE-ACTION [rule]
  return action
```

Goal Based Agents

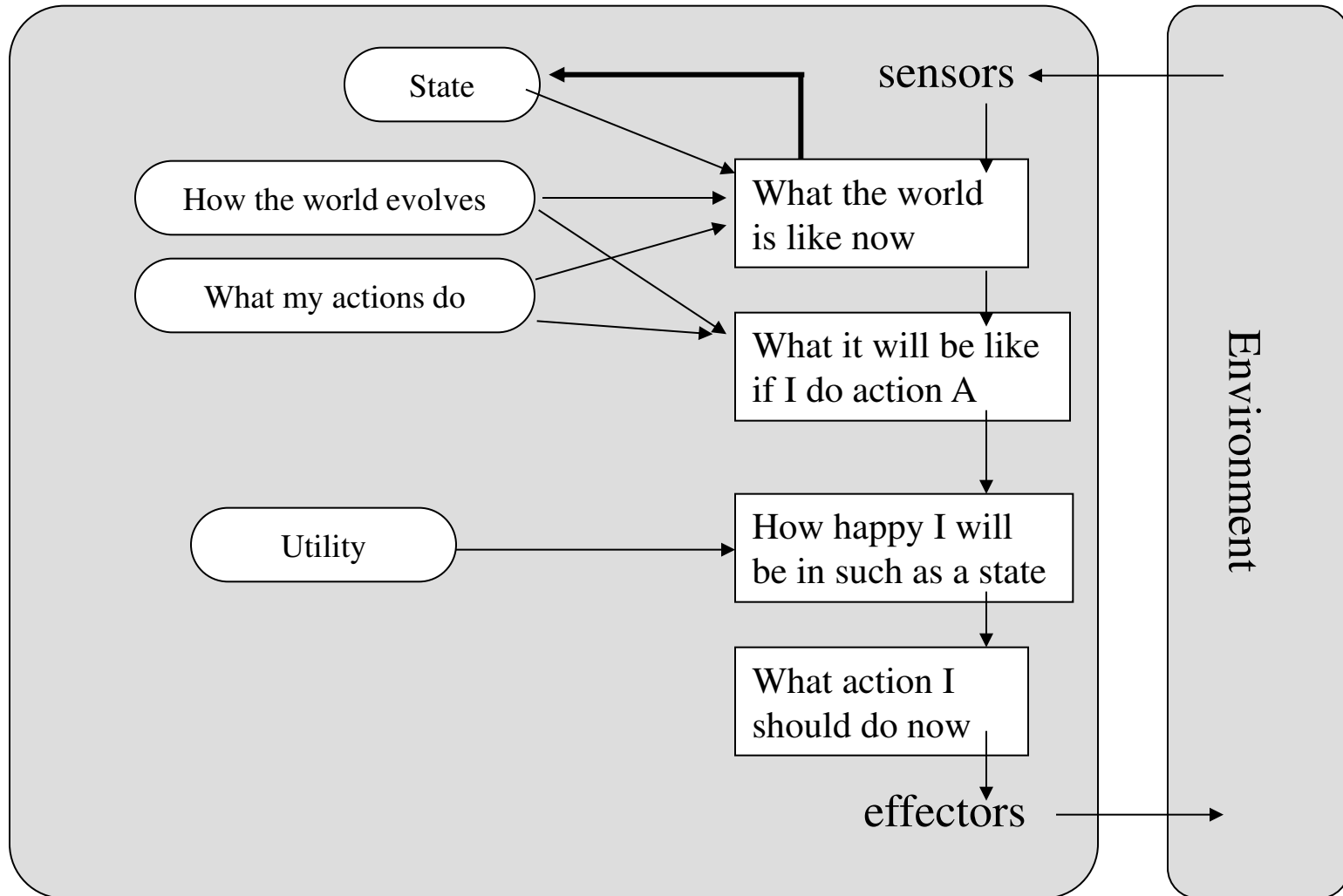
- Apenas o estado atual pode não ser suficiente
 - Considera informação sobre o gol
 - Considera seqüências de ações (deliberativo)



Utility Based Agents

- Quando existem várias alternativas... Qual escolher?
- Função de Utilidade: mapeia um estado ou uma seqüência de estados em um número
- Interessante quando
 - Múltiplos gols conflitantes
 - Custo benefício, principalmente em ambientes estocásticos: *expected utility*

Utility Based Agents



Learning agents

