Inteligência artificial

Aula 3 - Busca Gulosa e A(*)

Inteligência artificial

Prof. RELTON ALVES DA SILVA

Aula 3 - Busca Gulosa e A(*)

Comparação de Algoritmos

Comparação de algoritmos

- Dado um determinado problema, pode haver diferentes soluções algorítmicas para ele
- Como determinar qual solução é mais eficiente?
  - O que é um algoritmo eficiente?
  - Como “medir” a eficiência de um algoritmo?

Eficiência de algoritmos

- Eficiência de espaço
  - Refere-se à quantidade de memória requerida pelo algoritmo, além do espaço necessário para entradas e saídas
- Eficiência de tempo
  - Indica quão rápido um algoritmo é executado

Eficiência de algoritmos

- ❑ Eficiência de espaço
  - ❑ Refere-se à quantidade de memória requerida

Tamanho da entrada

- Um algoritmo vai demorar mais para executar entradas maiores
  - Exemplo: percorrer um vetor de 10 ou 1000 elementos
- Em geral, a eficiência do algoritmo é analisada em função do tamanho da sua entrada. Exemplo:
  - tamanho do vetor para problemas que trabalham com vetores: ordenação, buscar, encontrar o menor elemento, etc.
  - polinômio: grau do polinômio

Como medir o tempo?

- Unidades de tempo (minutos, segundos...)
  - depende do computador
  - depende do compilador
- Contar quantas vezes cada operação é executada
  - muito difícil
  - desnecessário
- Identificar a operação mais importante: **operação básica**

Como medir o tempo?

- ❑ Operação básica
  - ❑ operação que impacta mais no tempo de execução
  - ❑ em geral, é a operação no laço mais interno do algoritmo
  - ❑ contar quantas vezes ela é executada
- ❑ Qual é a operação básica da busca sequencial?

Pior caso, melhor caso e caso médio

- A eficiência de alguns algoritmos dependem não apenas do tamanho da entrada, mas também de uma entrada específica
- Exemplo: busca sequencial

Pior caso

- É uma entrada de tamanho *n* para a qual o algoritmo gastará o **maior** tempo possível entre todas as entradas do mesmo tamanho
- Como calcular a eficiência no pior caso:
  - verificar o tipo de entrada(s) que resulta no maior valor de vezes que a operação básica vai ser executada
- Qual é o pior caso da busca sequencial?

Melhor caso

- É uma entrada de tamanho *n* para a qual o algoritmo gastará o **menor** tempo possível entre todas as entradas do mesmo tamanho
- Como calcular a eficiência no melhor caso:
  - verificar o tipo de entrada(s) que resulta no menor valor de vezes que a operação básica vai ser executada
- Qual é o melhor caso da busca sequencial?

Caso médio

- Representa uma entrada comum ou aleatória
- Mais complexa de calcular
- Divide todas as instâncias de tamanho *n* em várias classes, sendo que para cada instância da classe, o número de vezes que a operação básica do algoritmo é executada é a mesma

Ordem de crescimento

- Avalia o quanto o tempo de execução de um algoritmo aumenta, de acordo com o aumento do tamanho da entrada n
- Considera valores grandes de n
- Para comparar a ordem de crescimento dos algoritmos, há três notações: O, Ω e θ
- Vamos ver a definição da notação O

Notação O

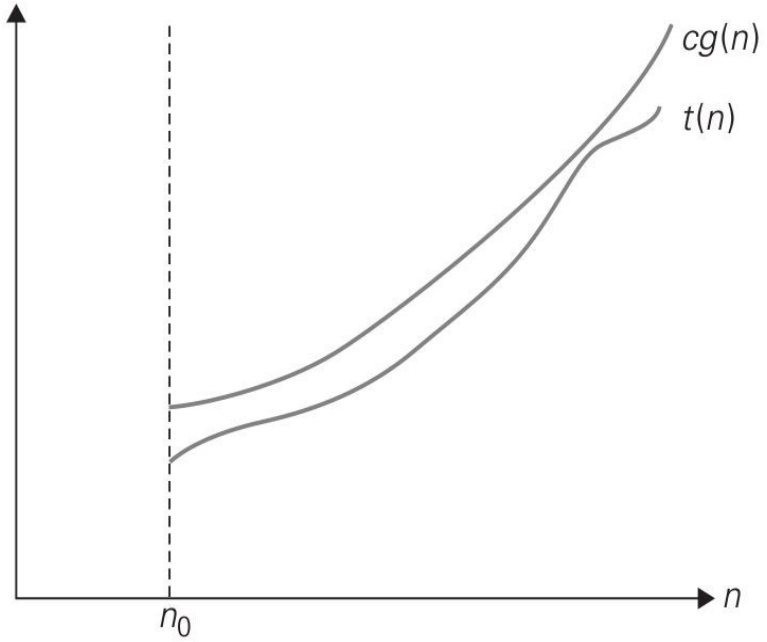
- Considere uma função $g(n)$ que representa um algoritmo
- Informalmente, $O(g(n))$ é o conjunto de todas as funções com ordem de crescimento menor ou igual a $g(n)$.
- Exemplo:
  - $n \in O(n^2)$
  - $100n+5 \in O(n^2)$
  - $n(n-1) \in O(n^2)$
  - $n^3 \notin O(n^2)$

Notação o

- Definição: Uma $t(n)$ pertence a $O(g(n))$, $t(n) \in O(g(n))$, se t(n) é limitada acima por alguma constante múltipla de g(n), se existe alguma constante positiva c e algum inteiro não-negativo n_0, tal que:

$$t(n) \leq c(g(n)), \text{ para todo } n \geq n_0$$

Notação o



Classes de algoritmo

n	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n	$n!$
10	3.3	10^1	$3.3 \cdot 10^1$	10^2	10^3	10^3	$3.6 \cdot 10^6$
10^2	6.6	10^2	$6.6 \cdot 10^2$	10^4	10^6	$1.3 \cdot 10^{30}$	$9.3 \cdot 10^{157}$
10^3	10	10^3	$1.0 \cdot 10^4$	10^6	10^9		
10^4	13	10^4	$1.3 \cdot 10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7 \cdot 10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0 \cdot 10^7$	10^{12}	10^{18}		

Exercício 1

- Considere a função que calcula o maior elemento de um vetor.

```
bool maior(int v[], int n){
      int maior = v[0];
      for(int i=1; i<n; i++){
            if(v[i]>maior)
                  maior = v[i];
      }
      return maior;
}
```

- Qual é a operação básica?
- Quantas vezes a operação básica é executada?

Exercício 2

- Considere a operação de atribuição destacada.

```
bool maior(int v[], int n){
      int maior = v[0];
      for(int i=1; i<n; i++){
            if(v[i]>maior)
                  maior = v[i];
      }
      return maior;
}
```

- Em que situação ele será executada o menor número de vezes?
- Em que situação ele será executada o maior número de vezes?

Finalizando Buscas

Métodos não informados de busca

BUSCA CEGA

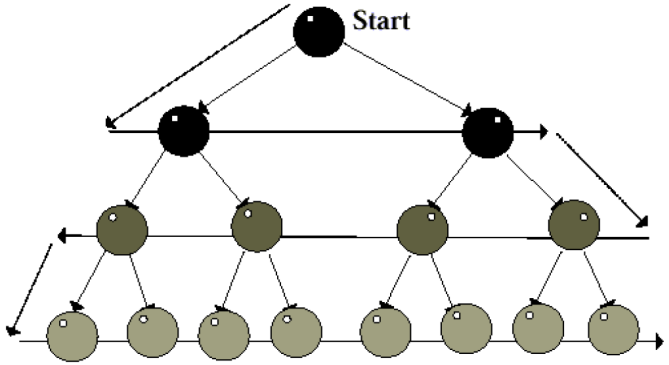
- Não utilizam qualquer conhecimento específico do problema para determinar a prioridade com que os nós serão expandidos, por isso são chamados de busca cega

Busca de Custo Uniforme

- Expande nós de acordo com o custo
- Se (custo=profundidade do nó) temos a Busca em Largura

Busca em Largura

- Expande os nós do nível d antes dos nós do nível d+1
- Encontra a solução ótima
- Complexidades de tempo e espaço exponenciais



Busca em Profundidade

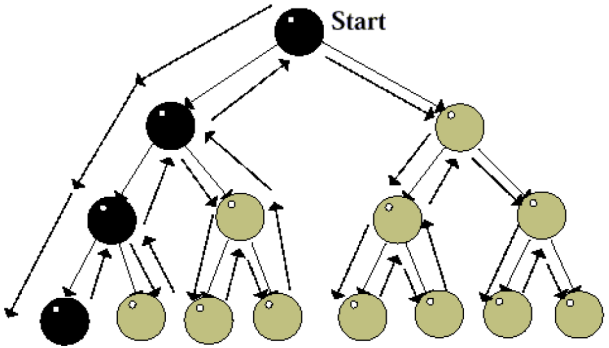
- Expande os nós mais profundos primeiro
- Pode não encontrar a solução
- Complexidade de tempo exponencial
- Complexidade de espaço polinomial

Busca em Profundidade Limitada

- Impõe uma profundidade máxima para a expansão dos nós
- Encontra a solução se esta estiver em uma profundidade menor ou igual ao limite estabelecido
- Complexidade de tempo exponencial
- Complexidade de espaço polinomial

Busca em Profundidade

- Tem este nome pois segue um caminho até a sua maior profundidade, daí começa um novo caminho;
- A busca em profundidade utiliza um método chamado **retrocesso cronológico** para voltar até a árvore de busca, uma vez que um caminho sem saída foi detectado;
- Um exemplo da aplicação da busca em profundidade é para indexação de páginas na Internet;
- Tem a desvantagem quando os caminhos são muito longos ou mesmo infinitos;
- Tem a vantagem quando os caminhos tem mesmo comprimento ou todos os caminhos levam ao estado objetivo;



Busca em Profundidade Iterativa

- Aumenta o limite de profundidade a cada iteração
- Encontra solução ótima
- Complexidade de tempo exponencial
- Complexidade de espaço polinomial

Métodos informados de busca

- Utilizam alguma informação específica do problema para gerar um novo estado
- Função de avaliação que procura estimar o número de passos para chegar à solução
- A heurística utilizada foi a quantidade de peças que estão fora do lugar em relação ao estado final

Busca Gulosa

Busca Gulosa

- Expande o nó que possui a melhor avaliação heurística
- Pode não encontrar a solução
- Complexidades de tempo e espaço exponenciais

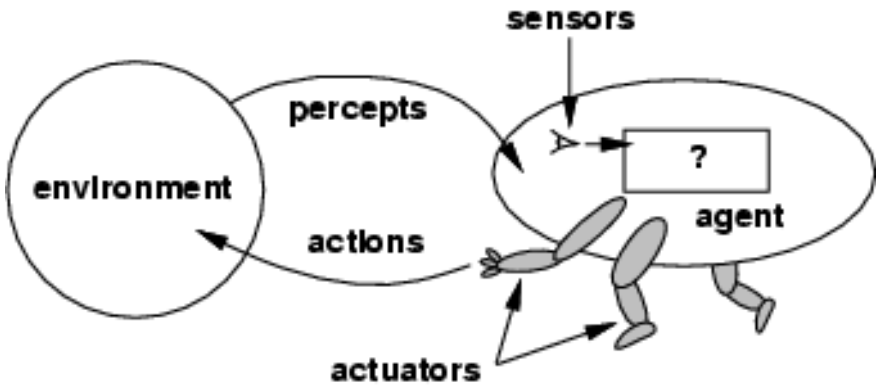
Busca A*

- Expande o nó que possui a melhor função de avaliação (soma do custo e avaliação heurística)
- Encontra a solução ótima
- Complexidades de tempo e espaço exponenciais
- Gera menos nós que os outros métodos de busca ótimos

Agentes

Agentes

- Um **agente** é algo capaz de perceber seu **ambiente** por meio de **sensores** e de agir sobre esse ambiente por meio de **atuadores**.



Exemplos

- Agente humano
  - Sensores: Olhos, ouvidos e outros órgãos.
  - Atuadores: Mãos, pernas, boca e outras partes do corpo.
- Agente robótico
  - Sensores: câmeras e detectores de infravermelho.
  - Atuadores: vários motores.

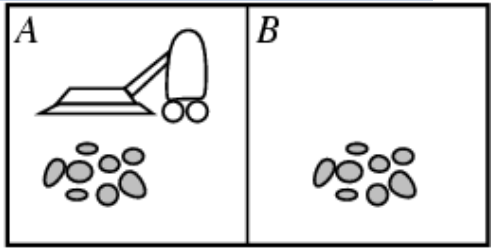
Mapeando percepções em ações

- Sequência de percepções: história completa de tudo que o agente percebeu.
- O comportamento do agente é dado abstratamente pela **função do agente**:

  [f: $\mathcal{P}^{\star} \rightarrow \mathcal{A}$]

  onde é a $\mathcal{P}^{\star}$ é uma

Exemplo:
O mundo do aspirador de pó



- Percepções: local e conteúdo
  - Exemplo: [A, sujo]
- Ações: Esquerda, Direita, Aspirar, NoOp

Uma função para o agente aspirador de pó

Sequência de Percepções	Ação
[A, Limpo]	Direita
[A, Sujo]	Aspirar
[B, Limpo]	Esquerda
[B, Sujo]	Aspirar
[A, Limpo], [A, Limpo]	Direita
[A, Limpo], [A, Sujo]	Aspirar
...	
[A, Limpo], [A, Limpo], [A, Limpo]	Direita
[A, Limpo], [A, Limpo], [A, Sujo]	Aspirar
...	

Programa: Se o quadrado atual estiver sujo, então aspirar, caso contrário mover para o outro lado.

Agentes Racionais

- Como preencher corretamente a tabela de ações do agente para cada situação?
- O agente deve tomar a ação “correta” baseado no que ele percebe para ter sucesso.
  - O conceito de sucesso do agente depende uma **medida de desempenho** objetiva.
    - Exemplos: quantidade de sujeira aspirada, gasto de energia, gasto de tempo, quantidade de barulho gerado, etc.
  - A medida de desempenho deve refletir o resultado realmente desejado.

Agentes Racionais

- Agente racional: para cada sequência de percepções possíveis deve selecionar uma ação que se espera venha a maximizar sua medida de desempenho, dada a evidência fornecida pela seqüência de percepções e por qualquer conhecimento interno do agente.
  - Exercício: para que medida de desempenho o agente aspirador de pó é racional?

Agentes Racionais

- Racionalidade é diferente de perfeição.
  - A racionalidade maximiza o desempenho esperado, enquanto a perfeição maximiza o desempenho real.
  - A escolha racional só depende das percepções até o momento.
- Mas os agentes podem (e devem!) executar ações para coleta de informações.
  - Um tipo importante de coleta de informação é a exploração de um ambiente desconhecido.
- O agente também pode (e deve!) aprender, ou seja, modificar seu comportamento dependendo do que ele percebe ao longo do tempo.
  - Nesse caso o agente é chamado de autônomo.
  - Um agente que aprende pode ter sucesso em uma ampla variedade de ambientes.

PEAS

- Ao projetar um agente, a primeira etapa deve ser sempre especificar o ambiente de tarefa.
  - **P**erformance = Medida de Desempenho
  - **E**nvironment = Ambiente
  - **A**ctuators = Atuadores
  - **S**ensors = Sensores

Exemplo de PEAS: Motorista de Táxi Automatizado

- Medida de desempenho: viagem segura, rápida, sem violações às leis de trânsito, confortável para os passageiros, maximizando os lucros.
- Ambiente: ruas, estradas, outros veículos, pedestres, clientes.
- Atuadores: direção, acelerador, freio, embreagem, marcha, seta, buzina.
- Sensores: câmera, sonar, velocímetro, GPS, hodômetro, acelerômetro, sensores do motor, teclado ou microfone.

Exemplo de PEAS: Sistema de Diagnóstico Médico

- Medida de desempenho: paciente saudável, minimizar custos, processos judiciais.
- Ambiente: paciente, hospital, equipe.
- Atuadores: exibir na tela perguntas, testes, diagnósticos, tratamentos.
- Sensores: entrada pelo teclado para sintomas, descobertas, respostas do paciente.

Exemplo de PEAS: Robô de seleção de peças

- Medida de desempenho: porcentagem de peças em bandejas corretas.
- Ambiente: correia transportadora com peças; bandejas.
- Atuadores: braço e mão articulados.
- Sensores: câmera, sensores angulares articulados.

Exemplo de PEAS: Instrutor de Inglês Interativo

- Medida de desempenho: maximizar nota de aluno em teste.
- Ambiente: conjunto de alunos.
- Atuadores: exibir exercícios, sugestões, correções.
- Sensores: entrada pelo teclado.

Propriedades de ambientes de tarefa

- Completamente observável (versus parcialmente observável)
  - Os sensores do agente dão acesso ao estado completo do ambiente em cada instante.
  - Todos os aspectos relevantes do ambiente são acessíveis.
- Determinístico (versus estocástico)
  - O próximo estado do ambiente é completamente determinado pelo estado atual e pela ação executada pelo agente.
  - Se o ambiente é determinístico exceto pelas ações de outros agentes, dizemos que o ambiente é estratégico.

Propriedades de ambientes de tarefa

- Episódico (versus sequencial)
  - A experiência do agente pode ser dividida em episódios (percepção e execução de uma única ação).
  - A escolha da ação em cada episódio só depende do próprio episódio.
- Estático (versus dinâmico)
  - O ambiente não muda enquanto o agente pensa.
  - O ambiente é semidinâmico se ele não muda com a passagem do tempo, mas o nível de desempenho do agente se altera.

Propriedades de ambientes de tarefa

- Discreto (versus contínuo)
  - Um número limitado e claramente definido de percepções e ações.
- Agente único (versus multi-agente)
  - Um único agente operando sozinho no ambiente.
  - No caso multi-agente podemos ter
    - Multi-agente cooperativo
    - Multi-agente competitivo

Exemplo

	Xadrez com relógio	Xadrez sem relógio	Direção de Táxi
Completamente observável			
Determinístico			
Episódico			
Estático			
Discreto			
Agente único			

- O tipo de ambiente de tarefa determina em grande parte o projeto do agente.
- O mundo real é parcialmente observável, estocástico, seqüêncial, dinâmico, contínuo, multi-agente.

Exemplo

	Xadrez com relógio	Xadrez sem relógio	Direção de Táxi
Completamente observável	Sim		
Determinístico	Sim		
Episódico	Não		
Estático	Semi		
Discreto	Sim		
Agente único	Não		

- O tipo de ambiente de tarefa determina em grande parte o projeto do agente.
- O mundo real é parcialmente observável, estocástico, seqüêncial, dinâmico, contínuo, multi-agente.

Exemplo

	Xadrez com relógio	Xadrez sem relógio	Direção de Táxi
Completamente observável	Sim	Sim	
Determinístico	Sim	Sim	
Episódico	Não	Não	
Estático	Semi	Sim	
Discreto	Sim	Sim	
Agente único	Não	Não	

- O tipo de ambiente de tarefa determina em grande parte o projeto do agente.
- O mundo real é parcialmente observável, estocástico, seqüêncial, dinâmico, contínuo, multi-agente.

Exemplo

	Xadrez com relógio	Xadrez sem relógio	Direção de Táxi
Completamente observável	Sim	Sim	Não
Determinístico	Sim	Sim	Não
Episódico	Não	Não	Não
Estático	Semi	Sim	Não
Discreto	Sim	Sim	Não
Agente único	Não	Não	Não

- O tipo de ambiente de tarefa determina em grande parte o projeto do agente.
- O mundo real é parcialmente observável, estocástico, seqüêncial, dinâmico, contínuo, multi-agente.

Programas e funções de agentes

- Um agente é completamente especificado pela função de agente que mapeia sequências de percepções em acções.
- Uma única função de agente (ou uma única classe de funções equivalentes) é racional.
- Objetivo: encontrar uma maneira de representar a função racional do agente concisamente.

Agente Dirigido por Tabela

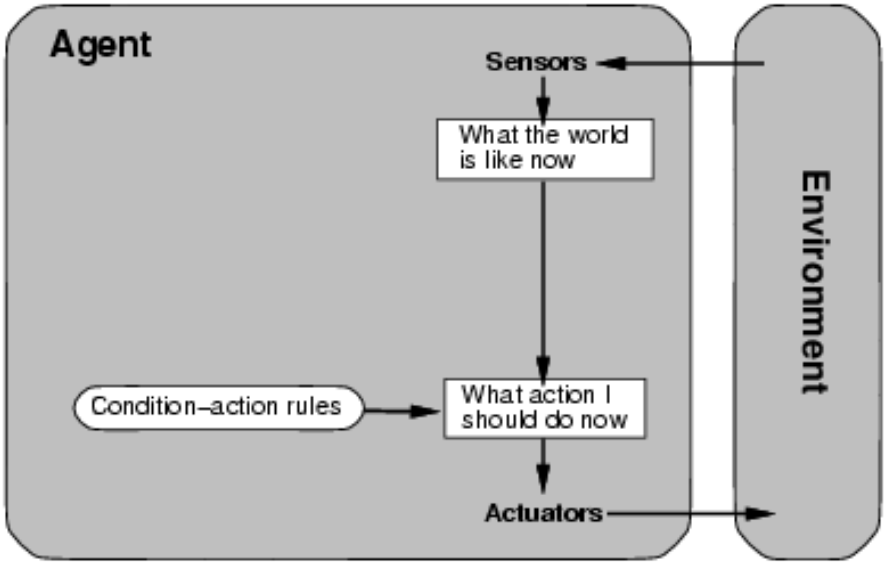
```
Função AGENTE-DIRIGIDO-POR-TABELA(percepção) retorna
  uma ação
      Variáveis estáticas:
        • percepções, uma seqüência, inicialmente vazia
        • tabela, uma tabela de ações, indexada por
          seqüências de percepções, de início
          completamente especificada
      anexar percepção ao fim de percepções
      ação ← ACESSAR(percepções, tabela)
      retornar ação
```

- Desvantagens:
  - Tabela gigante (xadrez = 10^{150} entradas)
  - Tempo longo para construir a tabela
  - Não tem autonomia
  - Mesmo com aprendizado demoraria muito para aprender a tabela.

Tipos básicos de agentes

- Quatro tipos básicos, do mais simples ao mais geral
  - Agentes reativos simples
  - Agentes reativos baseados em modelos
  - Agentes baseados em objetivos
  - Agentes baseados na utilidade

Agente Reativo Simples

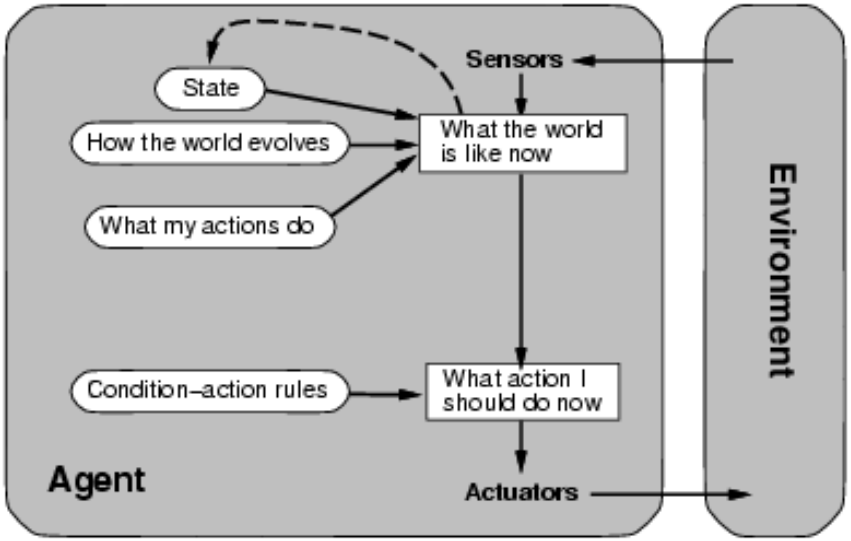


Exemplo: Agente Reativo Simples

```
Função AGENTE-ASPIRADOR-DE-PÓ-
    REATIVO([posição,estado]) retorna uma ação
        se estado = Sujo então retorna Aspirar
        senão se posição = A então retorna Direita
        senão se posição = B então retorna Esquerda
```

- Regras condição-ação (regras se-então) fazem uma ligação direta entre a percepção atual e a ação.
- O agente funciona apenas se o ambiente for completamente observável e a decisão correta puder ser tomada com base apenas na percepção atual.

Agentes reativos baseados em modelos



Agentes reativos baseados em modelo

Função AGENTE-REATIVO-COM-ESTADOS(*percepção*)
retorna uma *ação*

Variáveis estáticas:

estado, uma descrição do estado atual do mundo

regras, um conjunto de regras condição-ação

ação, a ação mais recente, incialmente nenhuma

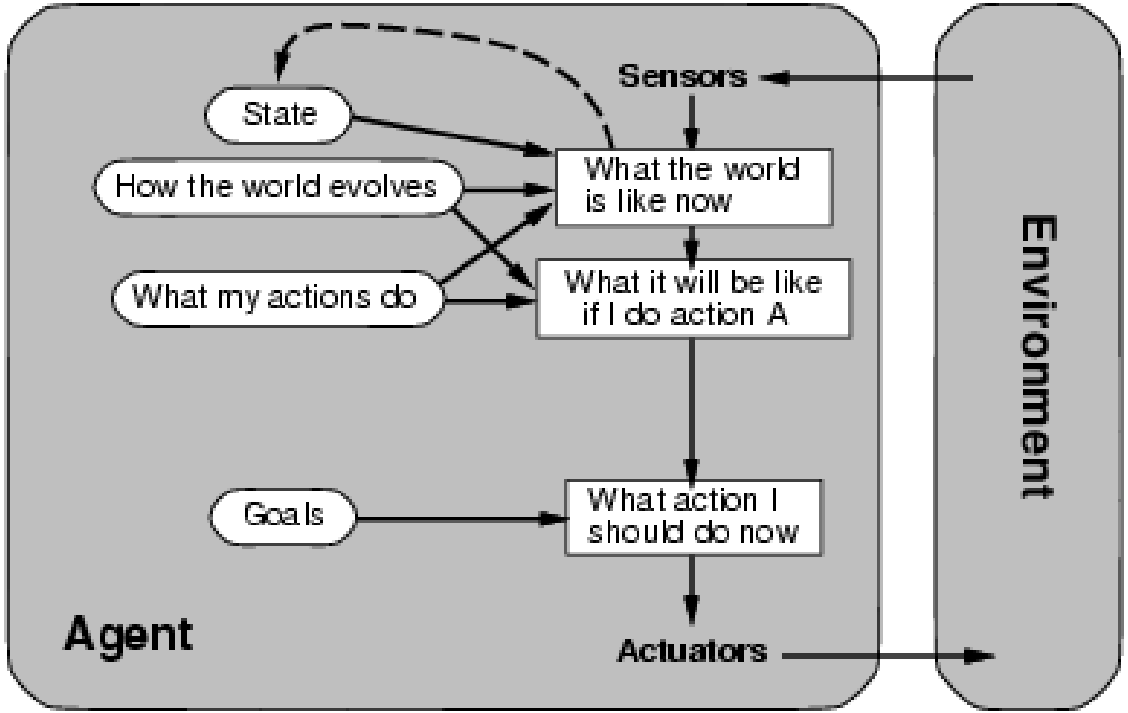
estado ← ATUALIZA-ESTADO(*estado*, *ação*, *percepção*)

regra ← REGRA-CORRESPONDENTE(*estado*, *regras*)

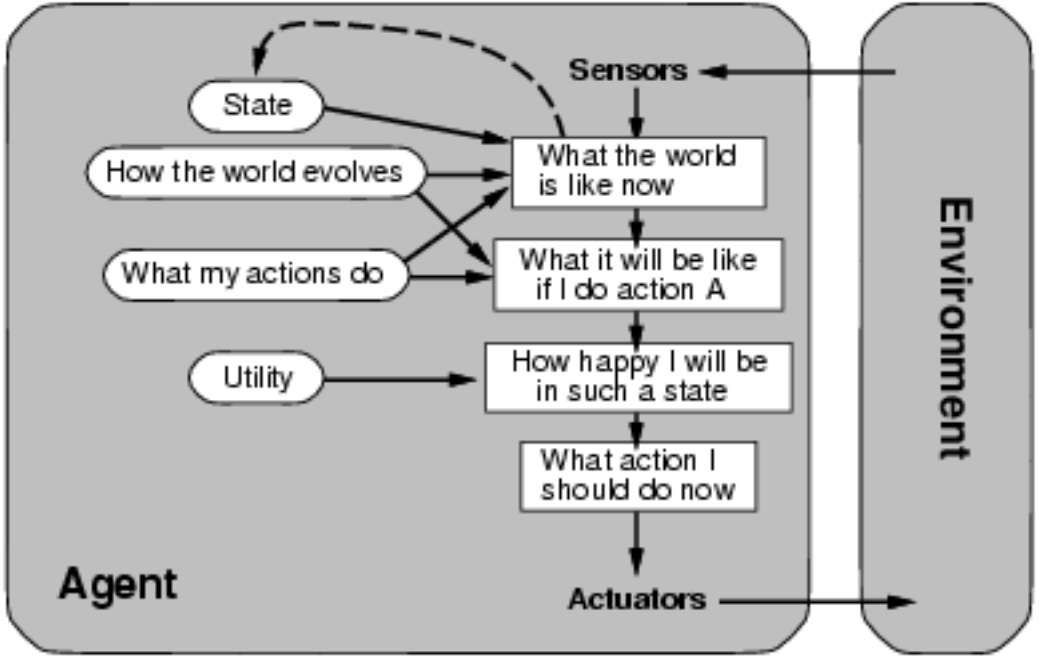
ação ← AÇÃO-DA-REGRA[regra]

retornar *ação*

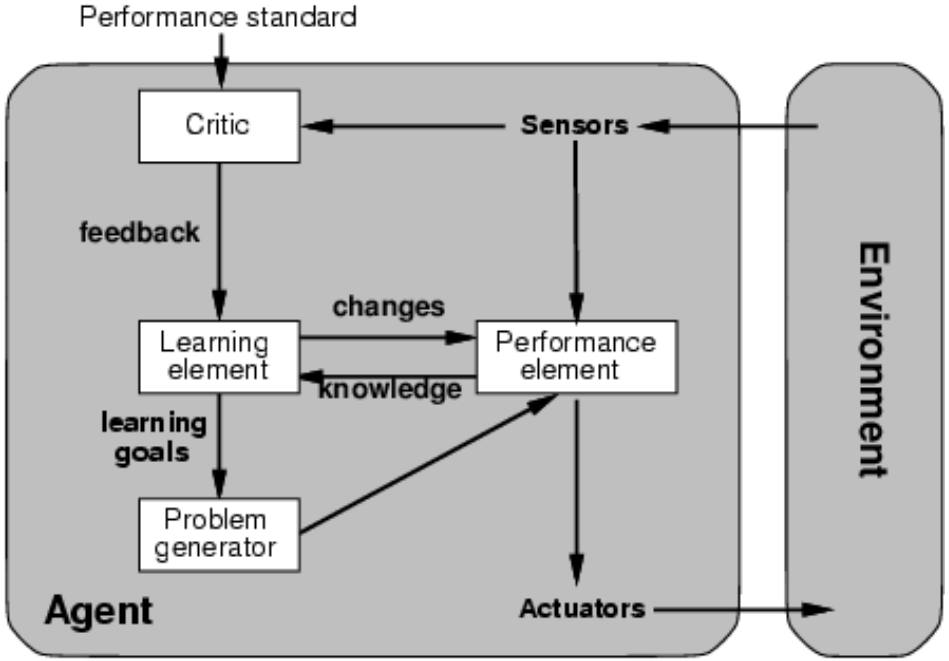
Agentes reativos baseados em objetivos



Agentes reativos baseados na utilidade



Agentes com aprendizagem



OBRIGADO