Tarefa 02: Agente e formulação de problemas de busca

Objetivos de aprendizagem

- compreender a separação entre agente e ambiente (crenças x estado do mundo)
- compreender o conceito de agente com raciocínio off-line e plano armazenado
- compreender os elementos que constituem a formulação de problemas de busca

Método

Equipe

Até 2 pessoas

Objetivo da tarefa

A partir do ambiente labirinto ilustrado abaixo, inclua no **agente** um <u>plano armazenado</u> (<u>=solução de um problema de busca</u>) capaz de levá-lo do estado inicial (S₀) até o estado objetivo (uma posição S_g). Este plano será **codificado pelo programador**. As paredes serão colocadas de acordo com a figura abaixo. Seguem os outros parâmetros:

- Linhas=colunas=9
- $S_0 = A = (8, 0)$
- Sg = G= (2, 8)

	0	1	2 	3	4	5	6	7 ++	8
0	XXX				XXX	XXX		XXX	
1	XXX +			' 		 		' XXX +	
2	 +	 +			XXX	XXX		 ++	G
3	 +	 	 	XXX +	XXX	XXX		XXX ++	ĺ
4	 	 	 	 	 	 	 	 ++	 +
5		XXX 		 	 	XXX		XXX +	
6		XXX 			XXX +			XXX ++	
7	 +	XXX 			XXX +		 	XXX ++	
8	A +	XXX +	XXX 	 +	 	 	 	 ++	

Requisitos

O agente deve TER uma REPRESENTAÇÃO DO PROBLEMA como uma instância da classe *Problema* (esta classe está parcialmente implementada). Os <u>atributos</u> da classe Problema a serem utilizados são:

- estado inicial: atributo inicializado pelo programador
- estado objetivo: atributo inicializado pelo programador
- labirinto: é uma crença do agente sobre o labirinto fornecida pelo programador. Deve ser um objeto da classe Labirinto criado na instanciação do Agente. Esta instância é diferente da que existe na classe Model (criada no método main()). A razão é que o agente não possui um sensor capaz de retornar a posição das paredes porque o ambiente não é completamente observável. O agente somente é capaz de saber sua posição atual por meio de um sensor a ser implementado.

Os métodos da classe Problema a serem utilizados são:

- ações possíveis (acoesPossiveis): método que calcula as ações possíveis a partir de um estado (ações que não o levem para fora do tabuleiro nem de encontro a uma parede).
 É a implementação da função ações (S): S -> A. O agente é capaz de ir para qualquer direção do conjunto {N, NE, L, SE, S, SO, O, NO} exceto quando há paredes.
- teste de objetivo (testeObjetivo): método que testa se o agente atingiu o objetivo deve invocar ao final da execução do plano para verificar se realmente está na posição objetivo

Implementar o método Problema.suc(s,a):

sucessora (suc): método que calcula o estado sucessor s' a partir da execução de uma ação a em um estado s qualquer. Portanto, Problema.suc(s, a) é o método que implementa a função suc(s, a): (s, a) -> s'

Implementar métodos/atributos no código fonte do Agente tal que ele:

- armazene um plano de ações (sequência de ações = solução)
- execute o plano de ações
- calcule o custo do caminho percorrido. Neste caso, vamos supor que as ações N, S, L e
 O têm custo 1 e, as demais, 1,5.

Implementar no método Agente.deliberar() procedimentos para:

- imprimir o estado atual e as ações possíveis no estado corrente
- imprimir a ação escolhida
- executar a ação escolhida
- imprimir custo acumulado até a execução da ação

Então, a cada execução de Agente.deliberar() deve ser mostrado algo como:

```
estado atual: 2,7 açoes possiveis: {NE L SE SO O NO } ct = 10 de 10 ação escolhida=L custo ate o momento: 12.5
```

Para fazer e entregar

- 1. A implementação descrita na seção acima satisfazendo todos os requisitos (carregar somente os códigos fontes modificados)
- 2. Faça um mapeamento do ciclo de raciocínio acima e o apresentado no algoritmo *goal-based-agent* (agente baseado em objetivos) do livro AIMA correspondência entre todas

as *caixas* da figura (inclusive environment, sensors e actuators) com os atributos e/ou métodos do código. Por exemplo:

```
Environment → classes Model e View

Agent → <resposta>

Sensors → método Estado Agente.sensorPosicao()

Actuators → <resposta>

State → <resposta>
... continuar
...
```

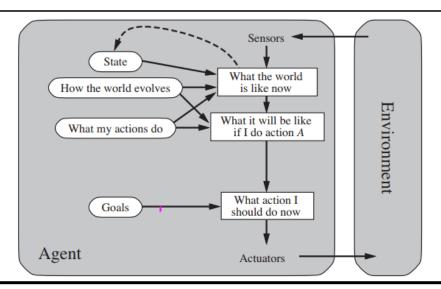


Figure 2.13 FILES: figures/goal-based-agent.eps (Tue Nov 3 16:22:54 2009). A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

- 3. Sobre espaço de estados, planos e crenças, responda as perguntas abaixo e entregue junto com o documento da questão 2.
 - 3.1. Quantos planos de ação são possíveis para sair de S_o e alcançar S_g?
 - 3.2. Qual o tamanho do espaço de estados e como pode ser calculado?
 - 3.3. Quais são os conhecimentos/crenças que o agente deve ter acerca do ambiente para que possa executar o plano?
 - 3.4. Em todo e qualquer problema, as crenças do agente sempre correspondem ao estado real ou simulado do mundo? O que ocorre no caso de divergências entre a representação que o agente possui do ambiente e o estado real do ambiente? De onde podem vir estas divergências?

Avaliação

A tarefa será avaliada por meio de:

- acompanhamento em sala de aula pelo professor (participação dos membros da equipe);
- avaliação pelos pares segundo barema passado pelo professor.

Referências

- slides 010a-Busca-Intro.pdf
- AIMA 3a. ed.:

o seção 2.4 e, especificamente, 2.4.4 Goal-based agents (cap. 3 Russel & Norvig)