Tarefa AAFP: Ambiente, Agente e formulação de problemas de busca

Objetivos de aprendizagem

Compreender

- o conceito de estado (posteriormente utilizado em problemas de busca);
- o conceito de agente situado em um ambiente;
- os tipos de ambientes
- a separação entre agente e ambiente (crenças x estado do mundo)
- o conceito de agente com raciocínio off-line e plano armazenado
- os elementos que constituem a formulação de problemas de busca

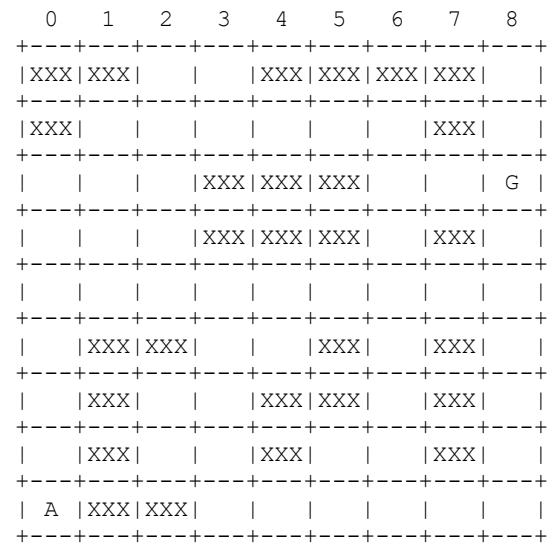
Cenário

Em um ambiente 2D (um labirinto) cuja forma é um grid de tamanho configurável de L x C (linhas por colunas), há um **agente** (dinâmico) **que deve planejar um caminho para sair do estado inicial e atingir o estado objetivo.** No ambiente existem paredes (estáticas), como ilustra a figura abaixo sendo A um agente no estado inicial e G, o estado objetivo.

Objetivo da tarefa

A partir do ambiente abaixo, fazer com que o **agente** siga um plano armazenado (solução de um problema de busca) capaz de levá-lo do estado inicial (S_0) até o estado objetivo (S_g). Este plano será **codificado pelo programador** (não será calculado algoritmicamente neste momento). As paredes estão dispostas conforme figura abaixo. Seguem os outros parâmetros:

- Linhas=colunas=9
- S_0 = A = (8, 0) // índice é um par (linha, coluna)
- Sg = G= (2, 8)



Para fazer

As implementações a serem feitos pela(o) estudante estão marcados com a etiqueta

@TODO T_AAFP

Classe problem: *contém as crenças do agente sobre o problema que ele deve resolver.*
Implemente os métodos que seguem.

- **ações possíveis (*possibleActions*):** 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.
- **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'`
- **teste de objetivo (*goalTest*)**: 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

Classe *agent*: *contém os métodos que permitem ao agente perceber e atuar o/no ambiente e deliberar (escolher) sua próxima ação*.

- *positionSensor(self)*: observe este método que emula um sensor de posição, i.e. que permite ao agente conhecer sua posição no ambiente (que neste caso é simulado);
- *executeGo(self, Direction)*: observe este método que emula um atuador, i.e. que permite ao agente se movimentar no ambiente;
- *na inicialização do agente colocar crenças sobre* o estado inicial, o objetivo e o labirinto:;
- no método *deliberate(self)* fazer:
  - **armazene um plano de ações** (sequência de ações = solução)
  - faça o agente executar o plano:
    Para cada ação do plano faça:
    - imprimir o estado atual e as ações possíveis no estado corrente
    - imprimir a ação escolhida na deliberação (uma ação por chamada do método)
    - executar a ação invocando o método executeGo(self, Direction)
    - imprimir custo <u>acumulado</u> até a execução da ação (incluindo a ação escolhida). Ações N, S, L e O têm custo *1* e, as demais, *1,5*

Então, a cada execução *deliberate(self)* deve ser mostrado algo como:

```
*************** inicio do ciclo ***********************
estado atual: (3,6)
açoes possiveis: {N NE SE S SO }
ct = 10 de 11. Ação escolhida=NE
custo ate o momento (com a acao escolhida): 11.5
**************************************************


--- Estado do AMBIENTE ---
2,7
     0   1   2   3   4   5   6   7   8
   +---+---+---+---+---+---+---+---+---+
 0 |XXX|XXX|   |   |XXX|XXX|XXX|XXX|   |
   +---+---+---+---+---+---+---+---+---+
 1 |XXX|   |   |   |   |   |   |XXX|   |
   +---+---+---+---+---+---+---+---+---+
 2 |   |   |   |XXX|XXX|XXX|   | A | G |
   +---+---+---+---+---+---+---+---+---+
 3 |   |   |   |XXX|XXX|XXX|   |XXX|   |
   +---+---+---+---+---+---+---+---+---+
 4 |   |   |   |   |   |   |   |   |   |
```

```
  +---+---+---+---+---+---+---+---+---+---+
5 |   |XXX|XXX|   |   |XXX|   |XXX|   |   |
  +---+---+---+---+---+---+---+---+---+---+
6 |   |XXX|   |   |XXX|XXX|   |XXX|   |   |
  +---+---+---+---+---+---+---+---+---+---+
7 |   |XXX|   |   |XXX|   |   |XXX|   |   |
  +---+---+---+---+---+---+---+---+---+---+
8 |   |XXX|XXX|   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+---+---+
```

Para fazer e entregar

1. As implementações solicitadas 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 → <resposta>
Actuators → <resposta>
State →  <resposta>
... continuar
...
```

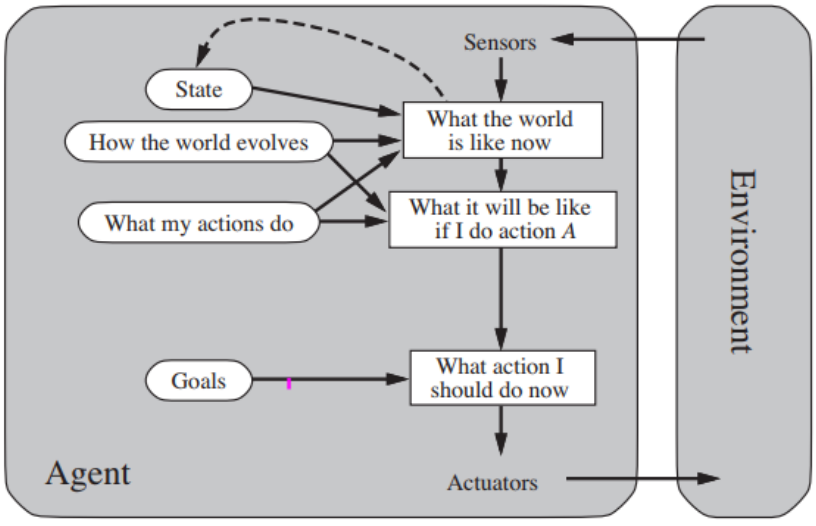


Figure 2.13 FILES: figures/goal-based-agent.eps (Tue Nov 3 16:22:54 2009). A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

3. Sobre o tipo do ambiente, espaço de estados, planos e crenças, responda as perguntas abaixo e entregue junto com o documento da questão 2.
   3.1. Quais as características do ambiente labirinto (ex. discreto, dinâmico, etc.)?
   3.2. Quantos planos de ação são possíveis para sair de S_o e alcançar S_g?
   3.3. Qual o tamanho do espaço de estados e como pode ser calculado?
   3.4. Quais são os conhecimentos/crenças que o agente deve ter acerca do ambiente para que possa executar o plano?

3.5. 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:

- auto-avaliação segundo barema passado pelo professor.

Referências

- slides 010a-Busca-Intro.pdf
- AIMA 3a. ed.:
  - seção 2.4 e, especificamente, 2.4.4 Goal-based agents (cap. 3 Russel & Norvig)