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 Grid 2D construído na tarefa anterior, inclua um **agente** com raciocínio procedural. O agente deve atingir um **objetivo** (no caso, uma posição S_g) a partir de um estado inicial (S_0) por meio da execução de um plano armazenado em sua memória. Este plano será *hard-coded* pelo programador. As paredes serão colocadas livremente pelo programador. Seguem os outros parâmetros:

- N=M=10
- S_0 = posição do agente (0, 0)
- Sg = posição do agente (7, 8)

Requisitos funcionais para o agente

O agente deve:

- ter a representação do **estado inicial** de forma explícita: atributo inicializado pelo programador
- ter a representação do **estado objetivo** de forma explícita: atributo inicializado pelo programador
- ter a representação do ambiente grid com suas paredes fornecida pelo programador (*hard-coded,* estrutura de dados diferente daquela utilizada no ambiente 2D já que o agente não possui um sensor capaz de retornar a posição das paredes; também não é possível perguntar ao ambiente onde estão localizadas!!!)
- ser capaz de **armazenar um plano de ações** (sequência de ações = solução)
- ser capaz de executar o plano de ações
- ter um **atuador** que comanda o **objeto móvel** situado no ambiente. Implemente no agente o método ir(X) tal que X = {N, NE, L, SE, S, SO, O, NO} que, por sua vez, invoca o método ir(X) do objeto móvel.
- ter um **sensor** que lê a posição atual requisitando-a ao **objeto móvel** (o agente invoca o método lerPos() do objeto móvel).
- ser capaz de calcular as **ações possíveis** a partir de um estado (ações que não o levem para fora do tabuleiro nem de encontro a uma parede). Isto equivale a implementar a função `ações(S): S -> A`
- ser capaz de calcular o **estado sucessor** a partir da execução de uma ação A em um estado S qualquer. Isto equivale a implementar a função `suc(S,A): (S,A) -> S'`
- ser capaz de identificar que atingiu o objetivo: **teste de objetivo**
- ser capaz de calcular 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*.

O agente deve ter um método chamado ciclo de raciocínio que, a cada invocação, deve:

- imprimir as crenças do agente acerca do ambiente grid 2D
- imprimir as ações possíveis no estado corrente
- retirar a próxima ação do plano
- imprimi-la
- executá-la

Na classe principal, você deve fazer um loop que
- invoca o método *ciclo de raciocínio* do agente e
- imprimi o estado do ambiente grid 2D.

Para refletir

- Compare o ciclo de raciocínio acima e o apresentado no algoritmo *goal-based-agent* (agente baseado em objetivos) do livro AIMA
- Observe quais são os conhecimentos/crenças que o agente deve ter acerca do ambiente para que possa executar o plano
- As crenças do agente sempre correspondem ao estado do mundo? O que ocorre no caso de divergências?
- Quantos planos de ação são possíveis para sair de S_o e alcançar S_g?
- Qual o tamanho do espaço de estados e como pode ser calculado?

Avaliação

A tarefa será avaliada por meio de:

- acompanhamento em sala de aula pelo professor (participação dos membros da equipe);
- exercícios de múltipla escolha respondidos individualmente em sala de aula. Este questionário conterá perguntas conceituais e práticas sobre formulação de problemas de busca.
- As questões podem versar sobre outra situação problema.

Referências

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