INF1771 - INTELIGÊNCIA ARTIFICIAL

TRABALHO 2 – LÓGICA

Descrição:

*"Após reunir a equipe de programadores para participar do **1° Concurso Mundial de Desenvolvimento de Softwares**, Barbie e seus amigos iniciaram o desenvolvimento de um software revolucionário: **A simulação ultra-realista do Incrível Mundo da Barbie**!*

*Após passarem várias madrugadas programando incansavelmente, Barbie e seus amigos finalmente concluíram a versão beta do seu software. Porém, a simulação ainda está **repleta de bugs**.*

*O principal bug que está afetando a simulação do Incrível Mundo da Barbie é um **denso nevoeiro** que impossibilita que os usuários possam ver o que está acontecendo ao seu redor.*

*Além disso, outros bugs estão gerando perigos ocultos na simulação, como **perigosos buracos** nos quais cairá qualquer um que tentar passar sobre eles, perigosos **vórtices espaciais** que teletransportam quem se aproximar deles para qualquer outro ponto da simulação, além de perigosas **baratas** espalhadas pela simulação.*

Para conseguir corrigir esses bugs, Barbie terá que entrar na simulação e localizar todos os bugs existentes. O seu objetivo é ajudar a Barbie a explorar a perigosa simulação beta do Incrível Mundo da Barbie e corrigir todos os bugs!"

Figura 1. Barbie Programadora.

Figura 2. Bug do Incrível Mundo da Barbie.

O Trabalho 2 consiste em implementar um **agente baseado em conhecimento** capaz de raciocinar de forma inteligente nesse ambiente desconhecido. Você deve implementar

uma interface para representar visualmente esse ambiente e utilizar a linguagem **Prolog** para representar o conhecimento do agente.

O mapa da simulação do Incrível Mundo da Barbie é mostrado na Figura 3.

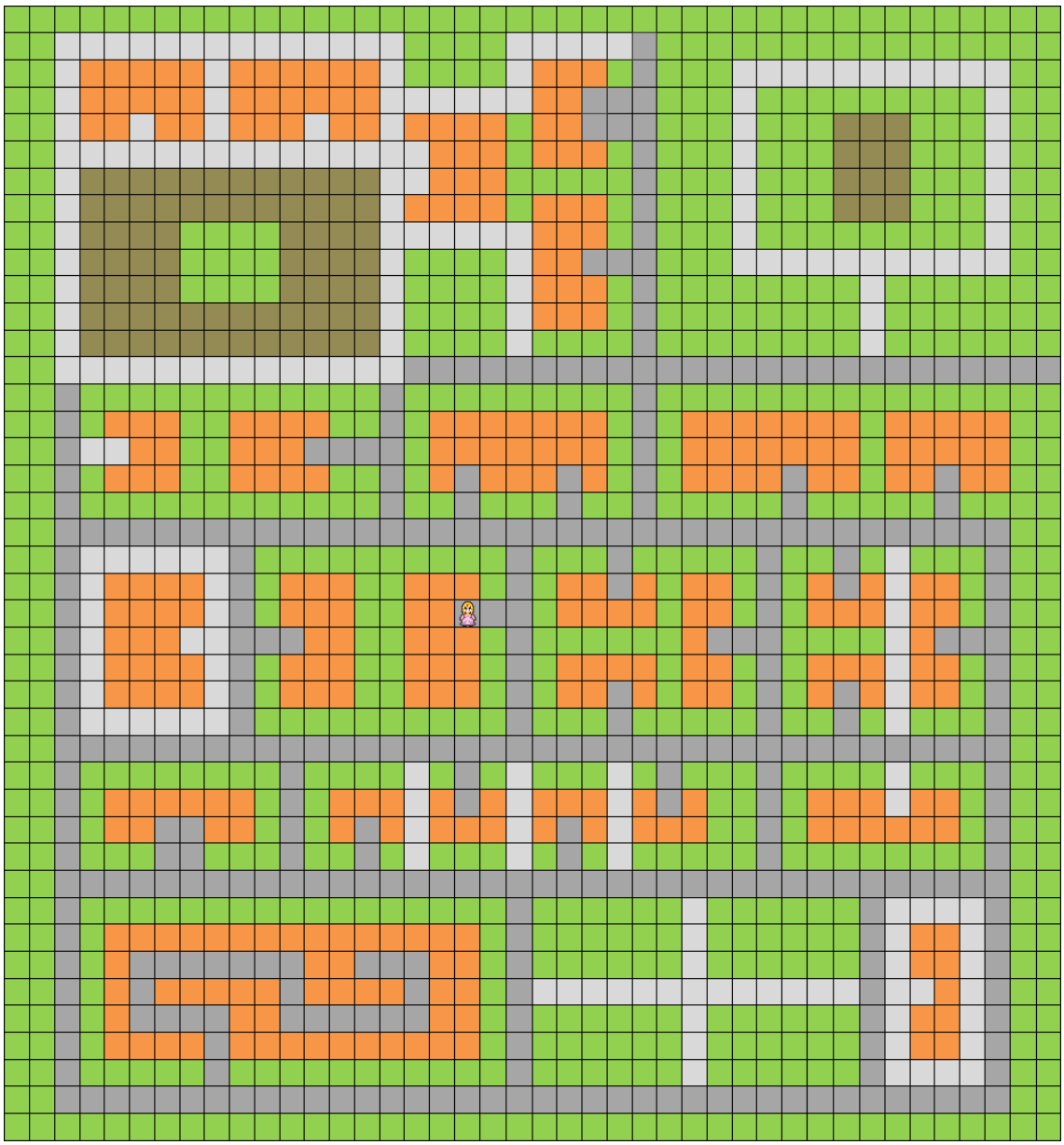

Figura 3. Mapa do Incrível Mundo da Barbie.

A simulação do Incrível Mundo da Barbie é formado por 5 tipos de terrenos: asfalto (região cinza escuro), grama (região verde), terra (região marrom), paralelepípedo (região cinza claro) e edifícios (região laranja). O agente pode passar livremente por todos os tipos de terrenos, exceto regiões de edifícios (região laranja).

Inicialmente o agente não conhece nada sobre a simulação do Incrível Mundo da Barbie, ele deve explorar o ambiente e utilizar seus **sensores** para obter informações.

Informações Adicionais:

- O mapa deve ser representado por uma matriz 42 x 42 (igual à mostrada na Figura 3).

- Na simulação do Incrível Mundo da Barbie existem os seguintes **elementos**:

    - **Buracos** – se o agente cair em um buraco ele morre imediatamente;
    - **Baratas** – o agente morre imediatamente se ele entrar em um local onde existe uma barata;
    - **Vórtices Espaciais** – ao entrar em um vórtice espacial o agente é teletransportado para algum outro local aleatório da simulação;
    - **Kits de Maquiagem** – até mesmo na simulação é importante manter o glamour! Encontrar um kit de maquiagem da pontos extras ao agente;
    - **Bugs** – bugs que estão causando os problemas na simulação. O agente deve encontrar todos os bugs para concluir a sua jornada;

- O agente sempre **inicia** a jornada na **Casa da Barbie** (posição [19, 23] no mapa). A aventura **termina** quando o agente conseguir encontrar e corrigir todos os bugs da simulação.

- O agente pode executar as seguintes **ações**:

    - Mover para Frente;
    - Virar a Direita (rotação de 90°);
    - Virar a Esquerda (rotação de 90°);
    - Atacar – Para jogar o salto-alto na direção em que o personagem estiver olhando (matando as baratas que estiverem no local adjacente);
    - Pegar Kit de Maquiagem – Para pegar o kit de maquiagem existente do local onde o agente estiver. A ação somente pode ser executada uma vez em cada local que exista um kit de maquiagem;
    - Corrigir Bug – Para corrigir um bug existente na simulação. A ação somente pode ser executada em locais que existam bugs;

- Cada ação executada pelo agente possui um **custo**:

    - Mover para Frente = -1;
    - Virar a Direita = -1;
    - Virar a Esquerda = -1;
    - Atacar = -10;
    - Pegar Kit de Maquiagem = +10;
    - Corrigir Bug = +100;
    - Cair em um Buraco = -10000;
    - Ser atacado por uma Barata = -10000;

- O agente não tem acesso a nenhuma informação do mapa, mas ele possui alguns **sensores** para perceber o ambiente. O agente possui os seguintes sensores:

    - Em locais adjacentes a buracos, exceto diagonal, o agente sente uma leve brisa;
    - Em locais adjacentes a baratas, exceto diagonal, o agente ouve as baratas andando;
    - Em locais adjacentes a vórtices espaciais, exceto diagonal, o agente percebe distorções espaciais;
    - Em locais onde existe um bug, o agente percebe códigos binários voando ao seu redor;
    - Em locais onde existe um kit de maquiagem, o agente sente o seu glamour aumentando;

- O mapa tem a estrutura ilustrada na Figura 3. O agente tem acesso à estrutura do mapa, mas **é desconhecida a localização** dos buracos, baratas, bugs, vórtices espaciais e kits de maquiagem. Sabe-se apenas que existem:

    - 100 Baratas;
    - 50 Buracos;
    - 30 Kits de Maquiagem;
    - 20 Vórtices Espaciais;
    - 20 Bugs;

- As posições dos buracos, baratas, bugs, vórtices espaciais e kits de maquiagem devem ser **sorteadas aleatoriamente** no inicio do programa. Mas o agente NÃO PODE ter acesso direto a essas informações.

- Os elementos (buracos, baratas, bugs, vórtices espaciais e kits de maquiagem) não podem estar em regiões de prédios. Além disso, não pode existir mais de um elemento na mesma posição. Durante a geração aleatória da posição dos elementos, o seu programa deve garantir que essas regras sejam respeitadas.

- Ao **entrar em um vórtice espacial** o agente é teletransportado para uma nova posição no mapa. Essa posição deve ser um local com terreno diferente de edificios sorteado aleatoriamente. Podendo ser um local onde existem buracos, baratas ou qualquer outro elemento, inclusive um novo vórtice espacial.

- Enquanto o agente corrige os bugs, gradativamente a **simulação volta a funcionar corretamente**. Após a correção de um determinado número de bugs, as seguintes alterações ocorrem na simulação:

    - 10 bugs corrigidos: os vórtices espaciais desaparecem da simulação;

    - 14 bugs corrigidos: as baratas desaparecem da simulação;
    - 18 bugs corrigidos: os buracos desaparecem da simulação;
    - 20 bugs corrigidos: o nevoeiro desaparece;

- O **jogo acaba** quando o agente conseguir corrigir todos os bugs existentes na simulação, ou quando o agente morrer ao cair em um buraco ou ao ser atacado por uma barata. O jogo deve acabar imediatamente quando o agente corrigir os 20 bugs.

Requisitos:

- O programa deve ser implementado em C/C++ ou Java utilizando a biblioteca do SWI-Prolog que permite acessar diretamente o Prolog. Também é permitido utilizar outras linguagens, mas antes você deve verificar se ela é compatível com o SWI-Prolog. Exemplos:

    - C# (http://www.swi-prolog.org/contrib/CSharp.html)
    - Python (http://code.google.com/p/pyswip/)
    - PHP (http://www.j-paine.org/dobbs/prolog_from_php.html)

- O Prolog deve ser utilizado somente para **representar o conhecimento do agente,** a interface visual e demais controles devem ser implementados em C/C++ ou Java.

- **Não é permitido realizar nenhum processo de tomada de decisão em C/C++ ou Java**, a decisão de quais ações o agente vai realizar deve ser feita exclusivamente pelo Prolog.

- Deve existir uma maneira de **visualizar os movimentos** do agente, mesmo que a interface seja bem simples. Podendo até mesmo ser uma matriz desenhada e atualizada no console.

- **O mapa do Incrível Mundo da Barbie deve ser configurável**, ou seja, deve ser possível modificar o tipo de terreno em cada local. O mapa pode ser lido de um arquivo de texto ou deve ser facilmente editável no código.

- O programa deve exibir um **log das consultas e inserções** realizadas na base de conhecimento Prolog.

- O programa também deve **exibir a pontuação** do agente enquanto ele executa as ações. Assim como a pontuação final.

- O trabalho pode ser feito **individualmente** ou em **grupos** de no máximo 3 pessoas.

- O programa deve ser apresentado durante a aula por **todos os membros do grupo**:

    - O membro do grupo que **não comparecer** receberá nota **zero**;

    - O membro do grupo que **não souber explicar** algo relacionado ao trabalho perderá 5.0 pontos.

Dicas:

- Planeje e defina exatamente quais vão ser os predicados necessários no Prolog para codificar o conhecimento que o agente tem do mundo. Exemplos:

    - em(3, 3). - define a posição atual do agente;

    - buraco(10, 6). - identifica que existe um buraco na posição (10, 6);

    - maquiagem(10, 13). - identifica que existe um kit de maquiagem na posição (10, 13);

- Lembre-se de codificar predicados para identificar locais seguros e também locais visitados.

- A maneira mais simples de codificar a comunicação entre o Prolog e o C/C++ ou Java é definindo um predicado "*melhorAção*" no Prolog. Esse predicado deve retornar a melhor ação para ser executada naquele momento. Comece codificando os comportamentos mais simples, como por exemplo:

    - melhorAcao(pegar_maquiagem(X,Y)) :- em(X,Y), maquiagem(X, Y).

- A ação "*andar*" não necessariamente precisa ser para um local adjacente a posição do agente. Pode ser um "*andar*" para outro local (X, Y) ainda não visitado. Nesse caso, você pode executar o A* para calcular o melhor caminho para chegar até a posição (X, Y) passando por locais seguros, mas lembre-se de tomar cuidado com os elementos existentes no mapa e também de aplicar os custos de movimentação.

Forma de Avaliação:

Será avaliado se:

(1) O trabalho atendeu a todos os requisitos especificados anteriormente;
(2) Os algoritmos foram implementados e aplicados de forma correta;
(3) O código foi devidamente organizado;
(4) O trabalho foi apresentado corretamente em sala de aula;

Bônus:

(1) O agente que conseguir corrigir todos os bugs com o menor custo, dado uma determinada configuração de elementos (buracos, baratas, bugs, vórtices espaciais e kits de maquiagem), receberá 2 pontos extras na nota. Para participar dessa competição é necessário que o programa inclua uma forma simples de definir manualmente a posição dos buracos, baratas, bugs, vórtices espaciais e kits de maquiagem. Em caso de empate, ambos os trabalhos receberão a nota extra.

Data de Entrega:

05/11

Forma de Entrega:

O programa deve ser apresentado na aula do dia 05/11 (quarta) e enviando até o mesmo dia para o email edirlei.slima@gmail.com.

Trabalhos entregues atrasados perderam 0.5 pontos para cada dia de atraso.