Robótica Cognitiva

Felipe Werndl Trevizan (trevisan@ime.usp.br)
Orientadora: Leliane Nunes de Barros (leliane@ime.usp.br)
Departamento de Computação - Instituto de Matemática e Estatística - USP

Apoio: FAPESP

Introdução

Uma das maneiras para desenvolver um agente inteligente é através de sua especificação em um formalismo lógico. Em especial, na área de Robótica Cognitiva, em que os domínios de aplicação envolvem o raciocínio sobre ações e atualização do estado do mundo, a especificação formal do agente se torna mais interessante. Isso acontece porque dada uma especificação em linguagem lógica, juntamente com um mecanismo correto de inferência, é possível obter e provar o comportamento desse agente. Assim, a área de Robótica Cognitiva se preocupa em estudar as capacidades de um agente inteligente cuja implementação se baseia numa especificação lógica. Os problemas de maior interesse de RC envolvem mundos dinâmicos e mundos com informação incompleta, ou seja, onde o agente não conhece *a priori* todas as características do ambiente.

Nesse projeto, foi construído um agente inteligente imerso em um mundo com informação incompleta, usando como formalismo lógico o Cálculo de Situações, uma extensão do Cálculo de Predicados de Primeira Ordem. Com esse formalismo, é possível raciocinar sobre ações e seus efeitos, permitindo a criação de planos de ações e a representação de mundos dinâmicos e com informação incompleta. Outro motivo que levou a escolha desse formalismo lógico foi o fato dele ser a base da linguagem de programação **Golog** [2]: uma linguagem de programação de agentes robóticos proposta por pesquisadores de Robótica Cognitiva [4], e que foi usada para implementar o agente proposto nesse projeto.

Mundo do Wumpus

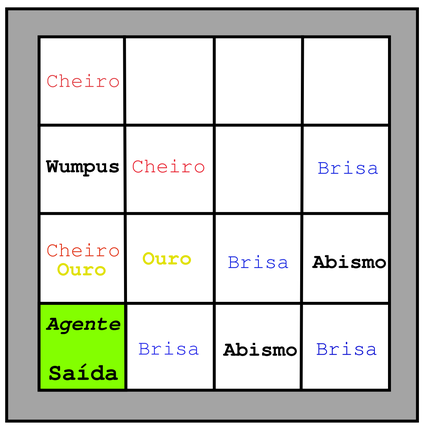

Figura 1: Representação do Mundo do Wumpus

O domínio de exemplo escolhido é o problema clássico do Mundo do Wumpus [1], um domínio com informação incompleta e percepção local. Esse domínio é representado por um reticulado (Figura 1), cercado por paredes que delimitam o modelo da caverna cuja exploração será feita pelo agente. Durante sua jornada, o agente poderá se deparar com o Wumpus - um monstro que o devorará -, um abismo no qual a queda do agente implica na sua morte ou uma pepita de ouro. O objetivo do agente é encontrar o ouro e escalar a saída da caverna (a mesma posição de entrada) sem ser morto pelo Wumpus ou pela queda no abismo. As principais regras para a criação do ambiente (reticulado) são:

- Todo o reticulado deve conter um Wumpus e pelo menos uma pepita de ouro;
- Qualquer posição, exceto a inicial, tem probabilidade de 0.2 de ser um abismo;
- As posições adjacentes ao Wumpus, excluindo as diagonais, serão marcadas como `stench` (Cheiro), indicando sua proximidade. De forma análoga, a adjancência do abismo será marcada como `breeze` (Brisa);
- Quando o agente estiver na mesma posição do ouro, ele perceberá o seu brilho, ou seja, a posição será marcada como `glitter` (Ouro);
- Se o agente tentar caminhar para uma posição que é uma parede, ele perceberá o choque (`bump`) e continuará na mesma posição e
- O único meio de comunicação entre o agente e o ambiente será através da ação `percept`, que retorna um vetor com 4 percepções, `stench, breeze, glitter, bump`, e caso alguma delas não ocorra, sua posição no vetor terá o valor `None`.

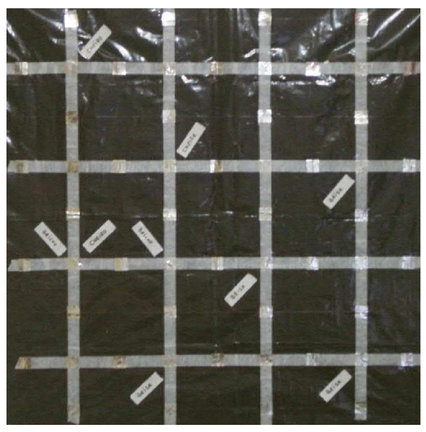

Figura 2: Maquete do Mundo do Wumpus para a Figura 1.

O agente pode executar seis ações diferentes no Mundo do Wumpus:

- `Turn(direction)`: muda a orientação do agente no mapa, onde `direction` pode valer North, South, East ou West;
- `Forward`: faz o agente avançar uma posição na sua atual direção;
- `Grab(object)`: o agente pega e segura o objeto `object`;
- `Release(object)`: solta o objeto `object`;
- `Climb`: faz com que o agente tente escalar a posição atual e
- `Shoot`: atira uma flecha na direção em que o agente está virado.

O robô Lego Mindstorm

Entre as diferentes versões do agente implementado nesse projeto, como o agente em **Prolog** e em **Golog**, o agente também foi implementado em **Legolog** (que será descrito a seguir) permitindo o controle de robôs Lego® MindStorms®. Esses robôs fazem parte do conjunto básico RIS (*Robotics Invention System*^{TM}) e são composto por (Figura 3):

- **bloco RCX** (Robotic Commander Explorer), que contém um microprocessador Hitachi H8/3297 de 16 bits, juntamente com uma máquina virtual implementada para ele em um *firmware*;
- **sensores**: um sensor de luz/contraste, que mede a quantidade de luz refletida por uma superfície e dois botões, que podem ser usados como sensores de toque;
- **atuadores**: dois motores que possuem cinco intensidades de funcionamento diferentes para ambas as direções (frente e trás) e
- **torre de infravermelho**: um transmissor infravermelho que é ligado na porta serial do computador para comunicação com o bloco RCX.

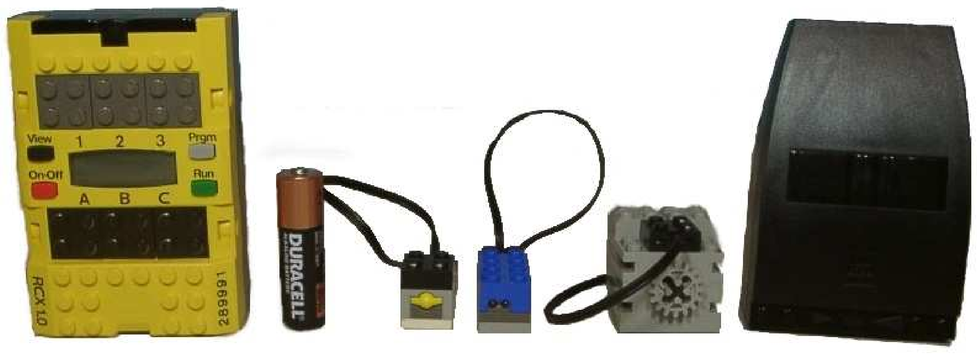

Figura 3: bloco RCX, pilha AA (para referência de tamanho), sensor de toque, sensor de contraste, motor e torre de infravermelho, escala 1:1.

A Linguagem Legolog

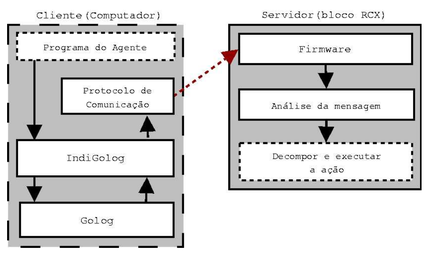

Figura 4: Decomposição do pacote **Legolog**.

Legolog [3] é uma linguagem para programação de robôs Lego® MindStorms® baseada na linguagem **IndiGolog** (uma extensão do **Golog**), proposta pelo grupo de Robótica Cognitiva da Universidade de Toronto[4]. Essa linguagem pode ser usada para modelar aplicações do tipo cliente e servidor (Figura 4), em que o bloco RCX é o servidor de atuadores e sensores, e o computador (com o meta-interpretador **Legolog**) é o cliente. A comunicação entre cliente e servidor é feita através da troca de um número inteiro por um protocolo de mensagens via infravermelho. Dessa forma, um programa em **Legolog** é divido em duas partes:

- **Programa de raciocínio do agente**, armazenado no computador (PC), ele é executado pelo meta-interpretador **Legolog**. fazendo a geração incremental (*online*) de planos de ações primitivas, considerando as percepções do robô e a ocorrência de ações exógenas.
- **Programa de execução de ações**, armazenado no bloco RCX, ele é executado pela máquina virtual que está no seu *firmware*. Esse programa é responsável por implementar as ações especificadas como primitivas no programa em **Legolog**, que podem ser divididas em: (i) as ações de atuação; (ii) ações de sensoriamento (percepções) e (iii) monitoração de ações exógenas.

Representação do Mundo do Wumpus

Para representar as três percepções possíveis para o Mundo do Wumpus (Figura 1), `glitter` (amarelo), `stench` (vermelho) e `breeze` (azul), todas as posição do ambiente foram divididas em quatro partes (Figura 5), usando como delimitador linhas pretas contínuas da Figura 6. Cada um desses quadrantes é responsável por representar uma percepção, seguindo a orientação da Figura 5, onde um dos quadrantes não é utilizado. Como cada percepção está presente ou não (percepção binária) na posição P, as três percepções possíveis foram representadas através de fitas (marcas de percepção), sendo a presença de cada uma denota que essa percepção está presente quando o agente atingir a posição P.

Para ler as percepções de uma posição, foram adicionadas as marcas de giro (quadrados cinzas na intersecção das linhas contínuas e pontilhadas na Figura 6), que são usadas pelo algoritmo de percepção, indicando o local para realizar um giro dentro de uma mesma posição ao procurar por marcas de percepção. Para o deslocamento do robô, foi usado as linhas pretas contínuas (Figura 6), sendo a intersecção entre duas dessas linhas marcadas com quadrados cinzas, representando que o agente atingiu uma nova posição.

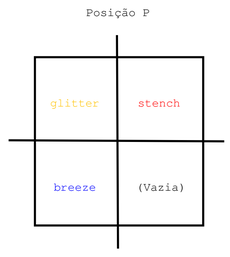

Figura 5: Modelo de cada posição P do Mundo do Wumpus

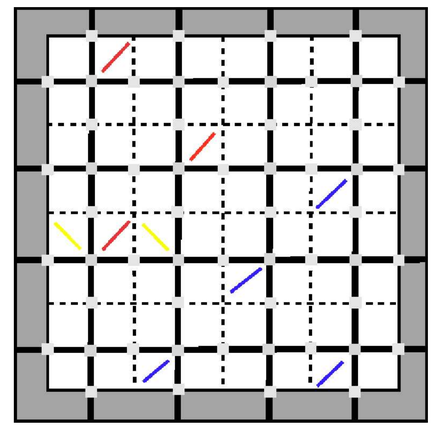

Figura 6: Representação da caverna do mundo do Wumpus ilustrado na Figura 1.

O agente em Legolog

Como **Legolog** é um linguagem baseada em **Golog** (Figura 4), a construção desse agente se resumiu em decompor as ações do agente em **Golog** em ações primitivas entre o meta-interpretador **Legolog** e o bloco RCX, isso é, quais ações serão implementadas diretamente no robô Lego® MindStorms®. Na implementação atual do agente, as seguintes ações são primitivas:

- Captar as percepções de uma posição de forma não ordenada;
- Girar 90° no sentido horário e anti-horário;
- Andar até a próxima posição;
- Pegar o ouro, atirar no Wumpus e escalar a saída.

As ações agrupadas no último item apenas fazem o robô emitir sons, indicando que elas foram executadas e esperando que o ambiente seja atualizado manualmente, ou seja, que as marcas de percepção sejam atualizadas.

Figura 8: Foto do robô Lego® MindStorms® em ação.

Referências

[1] RUSSELL, J. Stuart, NORVIG, Peter. *Artificial Intelligence: Modern Approach.* 1 ed. Prentice Hall, 1995.

[2] LEVESQUE, J. Hector, REITER, Raymond, LESPÉRANCE, Yves., LIN, Fangzhen., SCHERL, B. Richard. 1994. *GOLOG: A Logic Programming Language for Dynamic Domains.*

[3] LEVESQUE, J. Hector, PAGNUCCO, Maurice. 2000. *Legolog: Inexpensive Experiments in Cognitive Robotics.* Berlin, Germany.

[4] Grupo de robótica cognitiva da Universidade de Toronto. `http://www.cs.toronto.edu/cogrobo/`