UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

CAIO ARCE NISHIBE
LUÍS GUILHERME BERGAMINI MENDES

MODELAGEM DE PLANOS NO FUTEBOL DE ROBÔS

TAREFA II

CURITIBA
2011

CAIO ARCE NISHIBE
LUÍS GUILHERME BERGAMINI MENDES

MODELAGEM DE PLANOS NO FUTEBOL DE ROBÔS

Relatório apresentado à Disciplina de Sistemas Inteligentes 2, do Departamento Acadêmico de Informática - DAINF - do Curso Superior de Engenharia de Computação da Universidade Tecnológica Federal do Paraná - UTFPR, como requisito parcial para finalização da disciplina.

Prof. Cesar Augusto Tacla
Prof. Gustavo Giménez Lugo

CURITIBA
2011

SUMÁRIO

1 INTRODUÇÃO

Neste documento, será apresentada uma solução de modelagem de um grupo de agentes inseridos no futebol de robôs. O cenário considerado contém três agentes atacantes e um goleiro. Deve-se elaborar uma sequência de ações para que os atacantes consigam marcar um gol. Nesse processo estão envolvidas técnicas de modelagem e escolhas (**??**).

Especificamente, este documento trata da segunda fase do projeto de implementação de agentes no futebol de robôs. Os planos do goleiro e dos atacantes serão definidos e analisados. Mudanças na modelagem do ambiente foram necessárias e serão discutidas nas seções a seguir.

1.1 Objetivos

Inicialmente será feita a modelagem do ambiente no qual os agentes estão inseridos, assim como suas crenças iniciais e possíveis ações, tanto dos atacantes quanto do goleiro. A ferramenta de modelagem utilizada será o Jason (**??**), integrado com o simulador Tewnta para a liga F180 (**??**).

O ambiente Jason irá se comunicar com o simulador Tewnta, que fará a tradução para as ações dos jogadores na interface gráfica. O contrário também é válido, já que percepções no ambiente do Tewnta deverão ser enviadas ao Jason.

Na sequência, os planos de ação do goleiro e dos atacantes serão definidos e modelados em Jason. Métodos correspondentes em Java serão escritos para atender aos eventos disparados pela modelagem.

2 AMBIENTE

2.1 Descrição do Ambiente

No simulador Tewnta versão 1.3, o campo possui a dimensão de 490 x 338 pixels. Cada robô possui, aproxidamente, um diâmetro de 20 pixels. Com isso, foi criado um grid 25 x 17 em que somente um robô pode ocupar uma das posições no grid, dado um instante de tempo. A bola não é considerada um agente, sendo apenas um objeto do mundo, portanto poderá estar em uma mesma célula que um jogador.

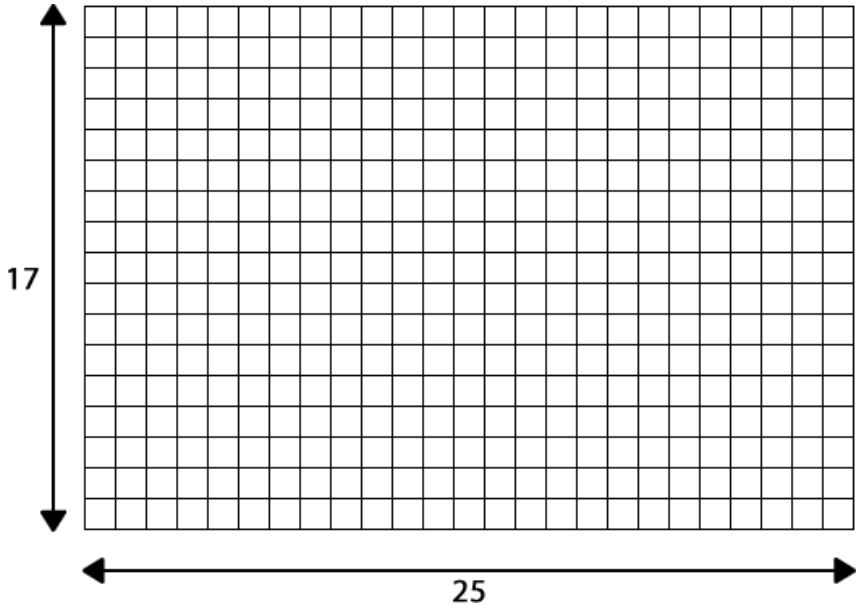

Figura 1: Grid modelado

Durante a fase de implementação dos planos, percebeu-se que a definição do ambiente na forma de grid não trouxe vantagens. Pelo contrário, tornou os agentes demasiadamente presos e sem a precisão necessária para determinadas ações, como por exemplo a determinação correta da posse de bola. Pelos motivos citados, o grid foi removido e os agentes podem possuir qualquer coordenada do campo do Tewnta.

3 AGENTES

3.1 Movimentos Possíveis

Considerando um agente que ocupa uma posição dentro do grid definido no ambiente, existem oito possíveis direções de movimentação.

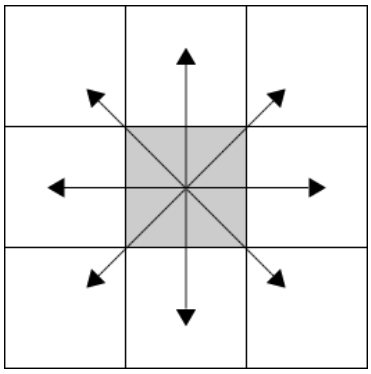

Figura 2: Movimentos possíveis para um agente que ocupa uma célula do grid

Os oito movimentos foram definidos da seguinte forma:

- **UP_CENTER**: verticalmente para cima.
- **UP_RIGHT**: diagonal para cima e para direita.
- **UP_LEFT**: diagonal para cima e para esquerda.
- **DOWN_CENTER**: verticalmente para baixo.
- **DOWN_RIGHT**: diagonalmente para baixo e para direita.
- **DOWN_LEFT**: diagonalmente para baixo e para esquerda.
- **RIGHT**: horizontalmente para direita.
- **LEFT**: horizontalmente para esquerda.

Com a remoção do grid, os agentes não ficam mais presos aos movimentos definidos anteriormente. O agente pode mover-se em qualquer direção, mas para isso deve primeiramente rotacionar até o ponto desejado, em seguida traçar um vetor que o leve até a posição desejada.

3.2 Crenças Iniciais

Quando o sistema é iniciado, o agente necessita de crenças iniciais para que possa começar seus procedimentos de inferência que levem a próximas ações. Nessa modelagem, foram definidas as seguintes crenças iniciais:

- **Posição inicial**: uma coordenada (x,y) do campo do Tewnta.
- **Time**: em qual time o agente está inserido, de forma a poder definir alvos de ataque e defesa.
- **Nome do Agente**: identificador do agente.
- **Entrar em campo**: os agentes são posicionados no campo. A partir daí, no caso específico do goleiro, será iniciado o plano de defesa. No caso do atacante, será iniciado o plano de marcar um gol.

As linhas de código a seguir mostram as implementações de crenças iniciais para o atacante e goleiro, em Jason, assim como seus objetivos iniciais.

3.3 Ações Possíveis

Ambos os agentes, atacante e goleiro, possuem um conjunto básico de ações possíveis. A ação de rotacionar consiste em definir um vetor de direção e apontar para ele, sem sair de sua posição atual. Com a direção definida, é possível praticar a ação de andar em linha reta no intuito de se aproximar do ponto desejado. Em resumo:

- Rotacionar;
- Andar em linha reta;

```
// Agente: Atacante (atacante.asl)

/* Initial beliefs and rules */
posicao(5, 8).
time(team_a).

/* Initial goals */
// Objetivo inicial: entrar em campo
!entrarEmCampo.

// Agente: Goleiro (goleiro.asl)

/* Initial beliefs and rules */
posicaoInicial(24, 8).
time(team_b).

/* Initial goal */
// Objetivo inicial: entrar em campo
!entrarEmCampo.
```

Quadro 1: Definição de crenças iniciais, em Jason

3.3.1 Atacante

No caso específico do atacante, há uma série de possíveis ações. O chute ao gol ocorre após a deliberação do agente ao perceber que está numa situação favorável para chutar ao gol. O passe é o toque da bola a um agente próximo, desde que seja do mesmo time. Há também a marcação, que consiste em acompanhar um jogador adversário no intuito de interceptar a bola. A ação de posicionar-se no campo é necessária, pois um agente pode escolher esperar a bola em um determinado local antes de executar a próxima ação. Há também ações de enviar e receber estratégia, que faz uso das funções de broadcast para informar os outros agentes das intenções atuais. A ação de pedir bola é feita quando o agente infere que está numa posição favorável para chute. Por fim, um agente pode buscar a bola em alguma posição do campo, caso nenhum outro agente esteja com posse. Em resumo:

- Chute ao gol;
- Passe;
- Marcação;
- Posicionar-se;

- Enviar estratégia;
- Receber estratégia;
- Pedir bola;
- Buscar a bola;

3.3.2 Goleiro

O goleiro não poderá sair de sua área em frente ao gol, com exceção de um pequeno avanço ou recuo, caso este posicionamento seja vantajoso para sua estratégia. Sua principal ação é a de defesa, que ocorre quanto a bola possui rota encaminhada ao gol. Após uma defesa, o agente poderá executar a ação de chute, enviando a bola o mais longe possível de sua região, ou de passe, que corresponde a tocar a bola para um agente próximo, que seja do mesmo time. Em resumo:

- Avanço;
- Recuo;
- Defesa;
- Chute;
- Passe;

4 PLANOS

Esta seção corresponde a fase dois da implementação de agentes no futebol de robôs. Serão definidos, em Jason, os planos de ação dos atacantes e do goleiro.

4.1 Atacantes

O primeiro plano define que, sempre que `entrarEmCampo` entrar na base de objetivos do agente, a posição e a que time pertence serão resgatados da sua base de crenças, então será criado um jogador novo com estes parâmetros e o plano `defender` é chamado.

O segundo plano leva em consideração duas percepções. Se o agente é o que está mais próximo da bola e não está com ela, então olha para bola, resgata da sua base de crenças a posição desta e vai em direção a ela para dominá-la.

O terceiro plano leva em consideração duas percepções. Se o agente não está com a posse da bola e está longe do gol, então posicione-se próximo ao gol, utilizando sua base de crenças.

O quarto plano leva em consideração duas percepções. Se o agente não está com a posse da bola e está próximo do gol, então permaneça na posição de ataque.

O quinto plano leva em consideração duas percepções. Se o agente está com a posse da bola e próximo ao gol, então determine a melhor posição para chute, através de uma máquina de inferência Fuzzy.

O sexto plano leva em consideração duas percepções. Se o agente está com a posse da bola, mas longe do gol, então passe para o agente mais próximo, desde que seja do mesmo time.

O sétimo plano funciona como plano *default*. Sempre que o agente não encontrar um plano que contenha um contexto desejável, ele executará o plano de reposicionamento.

O oitavo plano define que, sempre que `olheBola` entrar na base de objetivos do agente,

a posição da bola é resgatada da sua base de crenças e este posiciona sua frente na direção da mesma.

O nono plano define que, sempre que `olharCompanheiro` entrar na base de objetivos do agente, este rotacionará em direção a posição do agente mais próximo.

4.2 Goleiro

O primeiro plano define que, sempre que `entrarEmCampo` entrar na base de objetivos do agente, a posição e a que time pertence serão resgatados da sua base de crenças, então será criado um jogador novo com estes parâmetros e o plano `defender` é chamado.

O segundo plano leva em consideração uma percepção. Se o agente está de posse da bola, no caso de uma defesa bem sucedida, por exemplo, deve chutar a bola para longe.

O terceiro plano, funciona como plano *default*. Sempre que o agente não encontrar um plano que contenha um contexto desejável, ele executará o plano de defesa em si. O agente olha para bola e a partir da sua posição e da posição da bola, executa a manobra de defesa.

O quarto e último plano define que, sempre que `olheBola` entrar na base de objetivos do agente, a posição da bola é resgatada da sua base de crenças e este posiciona sua frente na direção da mesma.

```
// ao entrar em campo, obter a posicao inicial do jogador em campo e o
// time, criar o jogador no tewnta e comecar a rastrear a bola.
+!entrarEmCampo : true
    <-  ?posicaoIni(X,Y); ?time(Z);
       createPlayer(X, Y, Z);
       !atacar.
//caso o atacante seja o jogador mais proximo da bola, então domine a bola
+!atacar: maisPerto(bola) & not com(bola)
<-!!olheBola;
?posBola(X,Y);
irLinhaReta(X,Y);
!atacar.
//se sem bola e longe do gol, vá para perto do gol
+!atacar: not com(bola) & not perto(gol)
<-  !!olheBola;
?posicao(X,Y);
?posicaoIni(A,B);
posicionaAtaque(420,B);
!atacar.
//se sem bola e perto do gol, posicao de ataque
+!atacar: not com(bola) & perto(gol)
<-  !!olheBola;
?posicao(X,Y);
?posicaoIni(A,B);
posicionaAtaque(420,B);
!atacar.
//se com a bola e perto do gol, ache melhor posicao para chute
+!atacar: com(bola) & perto(gol)
<-  posicaoChute;
!atacar.
//se com a bola e longe do gol, passar para o companheiro mais proximo.
+!atacar: com(bola) & not perto(gol)
<-!!olharCompanheiro;
passar;
!atacar.
//caso contrário reposiciona
+!atacar: true
<-  //!!olheBola;
?posicaoIni(X,Y);
irLinhaReta(X,Y);
!atacar.
//Resgate a posicao da bola (percepcao) e rotacione olhando pra bola
+!olheBola : true
<-  ?posBola(X,Y);
rotacionePara(X,Y).
//Olha para o companheiro mais proximo
+!olharCompanheiro:true
<-  ?companheiroMaisProximo(X,Y);
rotacionePara(X,Y).
```

Quadro 2: Código dos planos dos atacantes, em Jason

```
// Ao entrar em campo, obter a posicao inicial do jogador em campo,
// criar o jogador no tewnta e iniciar a defesa.
+!entrarEmCampo : true
    <- ?posicao(X, Y); ?time(Z);
       createPlayer(X, Y,Z);
   !defender.

//caso o goleiro esteja com a bola, chutar
+!defender : com(bola)
<- chutar(100);
!defender.

//caso contrário, defenda o gol
+!defender : true
<-  ?posBola(XBola,YBola);
?posicao(XGoleiro, YGoleiro);
!!olheBola;
defender(XBola, YBola, XGoleiro, YGoleiro);
!defender.


//Resgate a posicao da bola (percepcao) e rotacione olhando pra bola
+!olheBola : true
<- ?posBola(X,Y);
rotacionePara(X,Y).
```

Quadro 3: Código dos planos do goleiro, em Jason

5 TEWNTA E JASON

Para a modelagem das crenças iniciais assim como das possíveis ações de cada agente, foi adotada a linguagem Jason, baseada em AgentSpeak. Para colocar em prática a modelagem em questão, foi usado o simulador Tewnta.

5.1 Código Jason

Inicialmente, deve-se criar um arquivo em Jason (extensão .mas2j) que define o ambiente, agentes e classpaths em Java para executar (**??**). Os arquivos de classpath correspondem aos executáveis do Tewnta. O ambiente corresponde ao nome da classe em Java que implementa as ações definidas para o agente.

```
MAS jasonSoccer {
    infrastructure: Centralised
    environment: SoccerEnv
    agents:

goleiro
/src/agents/goleiro.asl;

atacanteMeio
/src/agents/aMeio.asl;

atacanteDireita
/src/agents/aDireita.asl;

atacanteEsquerda
/src/agents/aEsquerda.asl;


classpath: "lib/tewntaCommons.jar"; "lib/simulator.jar";
}
```

Quadro 4: Código de inicialização em Jason

Os arquivos `goleiro.asl`, `aMeio/aDireita/aEsquerda.asl` contêm as definições de crenças, ações e planos.

5.2 Código Java/Tewnta

Na sequência, a classe SoccerEnv, em Java, implementa a comunicação com o Tewnta através da conexão com o webservice criado pela aplicação. O ambiente modelo é em seguida instanciado, através da classe FieldModel, que contém a definição do grid e métodos de conversão deste grid para medidas do campo. Ações definidas em Jason devem ser acessadas em Java através de uma sintaxe apropriada, que envolve a classe `Literal` ou diretamente por Strings. As linhas a seguir demonstram esse acesso:

```
/* Ações */
private static final Term TERM_GIRE = Literal.parseLiteral("gire");
private static final String FUNCTOR_CREATE_PLAYER = "createPlayer";
private static final String FUNCTOR_ROTACIONE_PARA_BOLA =
    "rotacioneParaBola";
private static final String FUNCTOR_IR_LINHA_RETA = "irLinhaReta";
```

Quadro 5: Conversão de termo Jason em termo Java

Os demais métodos implementados na classe SoccerEnv correspondem ao controle do agente dentro do campo, fazendo uso de técnicas como por exemplo PID, para estabilizar os movimentos.

Para a elaboração deste projeto, o simulador Tewnta passou por alterações no sentido de corrigir e contornar suas limitações. A principal limitação corrigida foi o fato de um agente não segurar a bola enquanto movimenta-se. Com o método funcionando, é possível executar ações que anteriormente necessitavam de métodos de aproximação à bola.

6 CONCLUSÃO

Através da realização desta tarefa, foi possível testar de maneira prática a implementação de um agente que segue o modelo BDI para atingir um objetivo. Durante esta primeira fase, foram modeladas as crenças iniciais dos agentes, assim como suas possíveis ações dentro do ambiente definido. A comunicação do ambiente Jason, que permite escrever sentenças com a linguagem AgentSpeak, foi posta em prática com o simulador Tewnta. Com esta base definida, pode-se partir para fases futuras que incluem a implementação dos comportamentos definidos para os agentes modelados.

Durante a fase de implementação dos planos dos agentes, optou-se pela remoção do grid de campo. Dessa forma, os agentes podem estar em qualquer coordenada (x,y) do campo do Tewnta, assim como podem mover-se livremente em qualquer direção.

REFERÊNCIAS