Universidade Salvador – Unifacs
Núcleo de Pesquisas em Redes de Computadores – NUPERC

O Futebol de Robôs: uma ferramenta para pesquisa e o ensino em Automação e Informática Industrial.

Mini curso apresentado na Semana Universitária
Salvador, 14 a 18 de outubro de 2002

Prof. Augusto Loureiro da Costa

Resumo do Mini Curso apresntado da Semana Universitária da Universidade Salvador.

O Futebol de Robôs: uma ferramenta para pesquisa e o ensino em Automação e Informática Industrial

Prof. Augusto Loureiro da Costa

14 a 18 de outubro de 2002

Área de Concentração: Automação e Informática Industrial.
Palavras Chaves: RoboCup, Futebol de Robôs, Robótica Cooperativa, Sistemas Multiagentes, Agentes Autônomos, Inteligência Artificial.

Resumo : O Futebol de Robôs é uma iniciativa de um grupo internacional de pesquisadores em Inteligência Artificial e Robótica Inteligente, a RoboCup Federation, que propõe um problemas padrão a ser solucionado: um partida de futebol de robôs. O futebol de robôs reúne grande parte dos desafios presentes em problemas eminentemente distribuídos do mundo real, tais como, veículos autônomos, busca de informação em bases de dados distribuídos, planejamento da geração de energia elétrica, recomposição de linhas de transmissão, controle de tráfego aéreo e urbano, etc. Sendo assim, o futebol de robôs apresenta-se como um laboratório para pesquisa e ensino em automação e informática industrial. Neste mini curso é apresentado, o futebol de robôs no formato proposto pela RoboCup Federation, onde os robôs apresentam alto grau de autonomia.

Capítulo 1

Introdução

A Inteligência Artificial (IA), surgiu em meados da década de cinquenta, com o propósito de dotar os computadores digitais da capacidade de representar e manipular conhecimento, além do simples armazenamento e manipulação de dados. Essa nova característica permitiria simular a inteligência humana nos computadores digitais, tornando-os capazes de solucionar problemas que requeriam um comportamento não mais procedural e sim inteligente, tal como jogar xadrez com a habilidade de um enxadrista.

A Inteligência Artificial não possui uma definição formal precisa, mesmo porque isso implicaria uma definição formal para inteligência. Entretanto, foram propostas algumas definições operacionais para a Inteligência Artificial:

- "uma máquina é inteligente se ela é capaz de solucionar uma classe de problemas que requerem inteligência para serem solucionados por seres humanos"[MH69];
- "Inteligência Artificial é um ramo da ciência da computação que compreende o projeto de sistemas computacionais que exibam características associadas, quando presentes no comportamento humano, à inteligência"[BF81];
- "Inteligência Artificial é o estudo das faculdades mentais através do uso de modelos computacionais"[CM85].

Os objetivos centrais da Inteligência Artificial podem ser sintetizados a nível teórico como a criação de teorias e modelos para a capacidade cognitiva e a nível prático como sendo a implementação de sistemas computacionais baseados em tais modelos. Neste sentido, a IA tem uma relação com seus objetos de estudo semelhante a da psicologia, mas com uma importante diferença: os modelos e teorias da IA são implementados em um computador, atribuindo a estes uma certa autonomia. Assim, a validade de um modelo ou de uma teoria de IA dispensa a prova comparativa de seus resultados com o comportamento psíquico humano, como no caso da psicologia, podendo assim ser implementada em um computador e demonstrada através da ação inteligente do programa no mundo [Bit96].

Inicialmente duas linhas de pesquisa destinaram-se à construção de sistemas inteligentes: *Conexionista e Simbólica.*

A linha *Conexionista* visa a modelagem da inteligência humana através da simulação dos componentes do cérebro, os neurônios e suas interligações também conhecidas como as redes neurais ou neuronais. As redes neurais apresentaram grande suscesso realizando tarefas de reconhecimento de padrões, como por exemplo sistemas computacionais de visão, ou na modelagem de sistemas complexos como bolsa de valores [Bar95, Bar97].

A linha *Simbólica* baseia-se na lógica e nos Sistemas de Post [Pos43] e teve McCarthy e Newell como seus principais defensores. Os princípios desta linha de pesquisa foram apresentados no artigo "Physical Symbol System"de Newell [New80]. Na IA Simbólica os chamados *Sistemas Especialistas*, programas computacionais concebidos para simular a perícia de um especialista, ainda que em um domínio restrito, alcançaram enorme sucesso na década de 70 e consolidaram a manipulação simbólica de um grande número de fatos especializados sobre um determinado domínio, como sendo o paradigma corrente para a construção de sistemas inteligentes do tipo simbólico. Esses sistemas apresentavam, entretanto, uma concepção centralizada do conhecimento, caracterizando assim um *Comportamento Individual* em relação ao ambiente.

No final dos anos 70 esse perfil centralizador foi contraposto, por um lado pelos trabalhos da linha conexionista utilizando redes neurais. Estes trabalhos apresentavam distribuição e paralelismo, e paralelamente os sistemas simbólicos baseados no chamado *Modelo de Quadro Negro* (do inglês "Blackboard Model") [HR85], caracterizados por um controle distribuído. O aparecimento desses trabalhos apresentando uma proposta de distribuição em IA, aliado à maturidade alcançada pelas tecnologias das Redes de Computadores e dos Sistemas Distribuídos deram origem a Inteligência Artificial Distribuída, *(IAD).*

A *IAD* representa hoje uma das áreas da IA tradicional que apresenta um grande

potencial em aplicações do mundo real [Dur91]. Esta área dedica-se ao estudo de teorias e implicações práticas a respeito de como reunir um determinado conjunto de Agentes Computacionais em uma Comunidade para solucionar problemas eminentemente distribuídos ou utilizar a robustez das redes de computadores para tratamento de problemas complexos, como por exemplo: controle de tráfego aéreo, monitoramento ambiental, gerenciamento de rede de computadores, distribuição de recursos como água, energia elétrica, etc. Devido a motivos históricos a IAD dividiu-se em dois enfoques: a Solução Distribuída de Problemas (SDP) e os Sistemas Multi-Agentes (SMA). A SDP concentrou seus esforços no problema a ser solucionado e em como utilizar os recursos disponíveis nas redes de computadores para tal fim. Os SMA por outro lado concentram-se no Agente e em suas propriedades, principalmente aquelas que o levam a cooperar [CL87] [Sic96].

Historicamente, devido a diversidade de paradigmas proposto para implementação de sistemas computacionais baseados em modelos para a capacidade cognitiva, a Inteligência Artificial apresentou uma grande dificuldade em comparar tais paradigmas. A adoção de jogos como um problema padrão a solucionado, permitiu essa comparação. O Xadrez foi bastante utilizado para comparar diferentes paradigmas, uma vez que este permite tando a comparação entre dois paradigmas distintos utilizado para implementa a habilidade de um enxadrista, bem como permite confrontrar uma dada implementação com um enxadrista humano.

A exemplo do xadrez, em 1996, um grupo internacional de pesquisadores em Inteligência Artificial e Robótica Inteligente propôs um desafio: uma partida de futebol entre robôs autônomos. O futebol de robôs reúne grande parte dos desafios presentes em problemas eminentemente distribuídos do mundo real, tais como, veículos autônomos, busca de informação em bases de dados distribuídos, planejamento da geração de energia elétrica, recomposição de linhas de transmissão, controle de tráfego aéreo e urbano, etc. Sendo assim, o futebol de robôs apresenta-se como um laboratório para pesquisa e ensino em automação e informática industrial. Dessa iniciativa surgiram duas ligas, a RoboCup (Robot World Cup) Federation e a Fira (Federation of International Robot-soccer Association). Estas duas ligas se diferem basicamente nas condições de contorno do problema proposto.

A Fira propôs inicialmente uma partida de futebol onde os times são compostos por três robôs, chamada MiroSot. Um computador ligado a uma câmera de vídeo, localizada dois metros acima do campo, que fornece uma vista de topo do campo e da posição dos robôs, é utilizado para controlar remotamente os robôs que compõem o time. Outras categorias foram adicionadas posteriormente: NanoSot, times com cinco robôs; SimuroSot robôs simulados; HuroSot robôs humanóides com 40 cm de altura e

K-pheraSot. O futebol de robôs proposto dessa forma favorece as soluções centralizadas onde os robôs possuem baixo grau de autonomia.

A RoboCup propôs inicialmente três diferentes condições de contorno para o problema do futebol de robôs, organizadas em três categorias distintas: robôs simulados, robôs de pequeno porte e robôs de médio porte. Na categoria de robôs simulados os times possuem onze robôs, como no futebol de campo, nas outras duas cada time possui cinco jogadores. Diferentemente da Fira, na RoboCup os robôs são completamente autônomos nas categorias de robôs simulados e de médio porte. Na categoria de robôs de pequeno porte, ainda é permitido a utilização de um sistema de visão global, onde a imagem de câmera é processada e posteriormente transmitida para os robôs. Existem ainda na categoria de robôs de pequeno porte, times que utilizam o sistema de visão embarcado. Outras categorias foram adicionadas posteriormente: robôs com pernas, onde o AIBO (cachorrro robô desenvolvido pela Sony) é utilizado; robôs humanóides com 1,60m de altura. Foi adicionada ainda uma outra modalidade onde o problema em questão é o resgate em escombros realizados por robôs autônomos chamada RoboCup Rescue. O futebol de robôs no formato proposto pela RoboCup Federation, onde os robôs apresentam um grau de autonomia maior, suas relações com as disciplinas e linhas de pesquisa de automação e informática industrial é o objeto de estudo desse mini-curso.

Inicialmente o problema do futebol de robôs é apresentado no capítulo 2, no formato proposto pela RoboCup Federation, e mantendo o foco na categoria de robôs simulados. Em seguida a experiência do projeto UFSC-Team é relatada no capítulo 3. Finalizando, o capítulo 4 apresenta alguns comentário finais e futuros trabalhos.

Capítulo 2

O Futebol de Robôs

2.1 Introdução

Um time de futebol de robôs consiste em uma coleção de veículos autônomos capazes de reconhecer o ambiente onde estão inseridos, no caso o campo de futebol e seus pontos de referência, e os objetos pertencentes a este ambiente, a bola, os outros veículos que compõem o time e os adversários. Estes dispositivos devem ainda ser capazes de representar o ambiente, estabeler metas, planejar e executar ações para atingir tais metas. Em se tratando de um jogo de equipe, além do carater autônomo, os jogadores de um time de futebol de robôs devem ser capazes de interagir com os outros jogadores do time para estabelecer objetivo coletivo, *metas globais*, planejar, alocar tarefas aos demais integrantes do time, sincronizar as ações de forma a imprimir ao time um perfil cooperativo.

A construção de um time de futebol de robôs envolve a integração de diversas tecnologias, como projeto de agentes autônomos, cooperação em sistemas multiagentes, estratégias de aquisição de conhecimento [Bit95], engenharia de sistemas de tempo real, sistemas distribuídos, reconhecimento de padrões, integração de sensores, aprendizado,

controle de processos etc.

O futebol de robôs reúne grande parte dos desafios presentes em problemas eminentemente distribuídos do mundo real, tais como, veículos autônomos, busca de informação em bases de dados distribuídos, planejamento da geração de energia elétrica, recomposição de linhas de transmissão, controle de tráfego aéreo e urbano, etc. Sendo assim o futebol de robôs apresenta-se como um laboratório para pesquisa e ensino em sistemas multiagentes e robótica móvel.

Neste capítulo apresentam-se as características do futebol de robôs que o tornam um importante laboratório para pesquisa na área de sistemas multiagentes. Inicialmente é apresentada uma breve descrição da RoboCup. Em seguida são apresentadas as características do Soccer Server, simulador utilizado pela RoboCup, enfocando os estados do jogo, as informações visuais e auditivas recebidas pelo jogadores, os comandos que podem ser enviados ao simulador para serem aplicados a cada um dos robôs, a temporização e sincronização do simulador e o treinador. Finalizando o capítulo, comentários são realizados relacionando as caractreristicas apresentadas do futebol de robôs e os sistemas multiagentes.

2.2 RoboCup

Anualmente, as soluções desenvolvidas pelos diversos grupos de pesquisadores espalhados pelo mundo, os time de robôs, são confrontadas em uma competição promovida pela RoboCup Federation, a "Robot World Cup" (RoboCup), [Kit97] [KTS+97]. Estas competições acontecem sempre em conjunto com congressos internacionais de grande importância e reconhecimento mundial, tais como IJCAI (International Joint Conference on Artificial Inteligence), ICMAS (International Conference on Multi-Agent Systems) e IROS (Intellinget Robotics Systems). A RoboCup possui diferentes categorias, algumas delas disputadas entre times de robôs reais, de pequeno e de médio porte, robôs com pernas e uma terceira categoria na qual as partidas são disputadas em um simulador, o Soccer Server, disponível na Internet (http://sserver.sourceforge.net/NEW/DownLoad.html). Nessa última categoria, atuam basicamente grupos de pesquisadores que concentram seus trabalhos na área de sistemas multiagentes.

A primeira *Robot World Cup* aconteceu em agosto de 1997, em Nagoya, Japão, durante a IJCAI'97 (Fifteenth International Joint Conference on Artificial Intelligence) e contou com a participação de pelo menos 40 times. Em 1998, a RoboCup aconteceu de 2 a 5 de julho, na *La Cité des Science et de l'Industrie*, Paris, França, em conjunto com o ICMAS-98 (International Conference on Multi-Agent Systems), e com a participação

de 30 times na categoria de simuladores, contando com um representante brasileiro, o *UFSC-Team*, além de 12 times na categoria de robôs de pequeno porte (small size league) e 16 times na categoria de robôs de médio porte (middle-size league). Aconteceram ainda a RoboCup '99 em Estocolmo, Suécia; RoboCup 2000 em Melbourne na Austrália e a última RoboCup 2001 em Seattle, Estados Unidos. A próxima edição da RoboCup deverá acontecer em 2002 no Japão.

2.3 Soccer Server

O *Soccer Server*, utilizado na categoria simuladores da RoboCup, é composto por dois processos: um servidor de conecções *udp/socket*, server e um display gráfico Xwindows onde são mostrados o campo de futebol virtual (108m x 68m) e os robôs de ambos os times (figura 2.1).

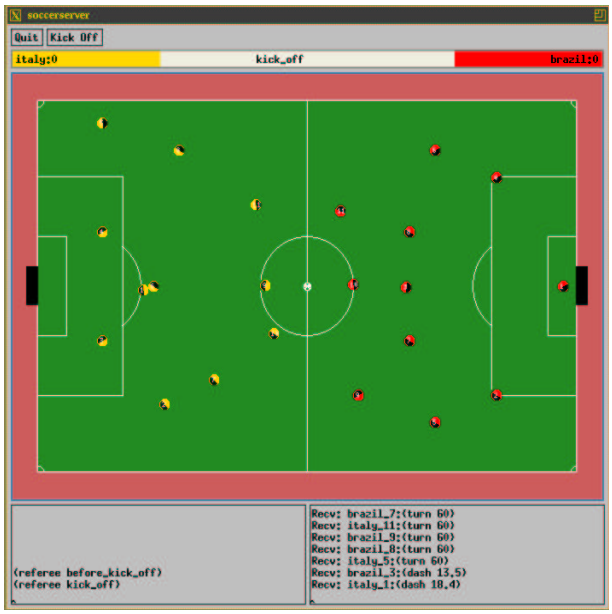

Figura 2.1: Soccer-Sever: Simulador utilizado na RoboCup

O servidor de conecções udp/socket server (figuara 2.2) associa a cada um dos robôs, via conecção por *socket*, um cliente (agente) responsável pelas ações do robô.

Por essas conecções cada um dos agentes recebe as informações visuais (percepção) e auditivas (comunicação) e envia de volta para o simulador os comandos a serem aplicados ao robô. O server possui um modelo numérico do ambiente – o campo de futebol em questão, os robôs e a bola. Esse modelo numérico é responsável pela movimentação dos objetos (os jogadores e a bola), fazendo com ela aconteça da mesma forma que em uma partida de futebol de robôs reais, levando em consideração atrito, inércia, colisões,

ruído, ação do vento, etc. Esse processo assume ainda algumas atribuições de um juiz, responsável pela implementação das seguintes regras de controle do jogo:

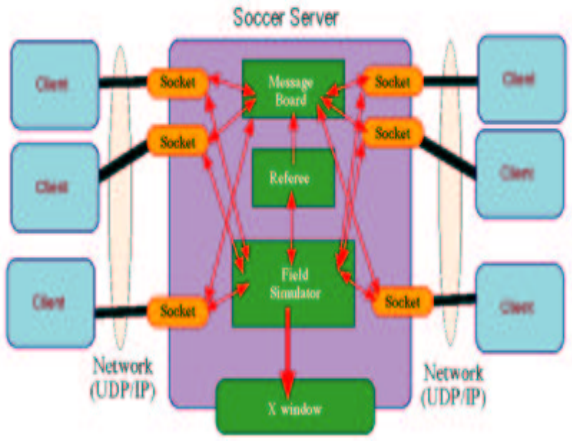

Figura 2.2: Soccer-Sever: servidor de conecções udp/socket

- Gol – Quando a bola se encontra dentro do gol, o juiz implementado no Soccer-Server difunde para todos os agentes uma mensagem informando o acontecimento de um gol. Além disso atualiza o placar, suspende a execução da partida por cinco segundos, move a bola para o centro do campo e muda o estado do jogo para *before_kick_off*, reiniciando a partida. Durante os cinco segundos de interrupção da partida todos os agentes devem voltar para seus respectivos lados do campo.

- Saída de Bola – Durante a saída de bola, início ou reinicio do jogo, todos os jogadores (agentes) devem estar em seu campo defensivo. Caso um jogador se encontre no campo adversário, este será movido para uma posição aleatoriamente escolhida em seu campo defensivo.

- Bola Fora de Jogo – Quando a bola se encontra fora das dimensões do campo, o juiz move a bola para a posição apropriada (lateral do campo, marca de escanteio ou pequena área) e muda o estado do jogo para *kick_in* (reposição de bola), *corner_kick* (escanteio) ou *goal_kick* (tiro de meta).

- Desobstrução – Após o goleiro segurar a bola, ou quando a partida se encontra em um dos seguintes modos, *kick_off*, *thrown_in*, *corner_kick*, *goal_kick* ou *offside*, o juiz remove qualquer jogador adversário num raio de 9.15m da bola, para o perímetro do círculo de mesmo raio (9.15m) centrado na bola.

- Controle do modo de jogo – Quando o jogo se encontra em um dos seguintes estados *kick_ off*, *thrown_ in*, *corner_ kick* ou *goal kick*, o juiz muda o estado do jogo para *play_ on* após a bola ter sido colocada em jogo, por um dos jogadores do time que detém a posse da bola.

- Final de Primeiro Tempo e Fim de Jogo – O juiz suspende o jogo quando o relógio do Soccer Server atinge 3000 ciclos de simulação (5 minutos) decretando o final do primeiro tempo e difunde uma mensagem informando o novo status do jogo. De forma similar, o juiz termina o jogo ao atingir-se 6000 ciclos de simulação.

O processo monitor consiste em um display Xwindow onde são apresentados o campo de futebol virtual nas dimensões (108m x 68m), os jogadores, a bola, um botão para iniciar a partida, um outro para finalizar a execução do soccer server, o placar e o tempo de simulação decorrido (ver figura 2.1). Esse display possui ainda um menu, com as seguintes opções: *Free Kick by Team Left*, *Free Kick by Team Rigth* e *Drop the Ball*. Essas opções permitem a concessão de chutes livres para o time da esquerda ou para o da direita respectivamente, além de permitir que a bola seja colocada em jogo.

2.4 A Partida

As partidas são disputadas em um campo de futebol virtual (108m x 68m), provido pelo simulador Soccer Server, em dois intervalos de cinco minutos (3000 ciclos de simulação). Cada um dos times é composto por onze jogadores. Cada partida pode assumir os seguintes estados:

- **before_kick_off** – A partida encontra-se parada aguardando seu início. Este estado acontece no início do jogo antes do botão kick off do Soccer Server ser acionado ou durante os 5 (cinco) segundos de intervalo dados pelo juiz após a ocorrência de um gol.

- **time_over** – Fim de jogo.

- **play_on** – Quando a bola está em jogo.

- **kick_off_l ou kick_off_r** – Saída de bola para o time da esquerda e da direita respectivamente.

- **kick_in_l ou kick_in_r** – Reposição de bola em jogo para o time da esquerda ou para o da direita respectivamente.

- **free_kick_l ou free_kick_r** – Chute livre para o time da esquerda ou para o da direita respectivamente.
- **corner_kick_l ou corner_kick_r** – Cobrança de escanteio para o time da esquerda ou para o da direita respectivamente.
- **goal_kick_l ou goal_kick_r** – Cobrança de tiro de meta para o time da esquerda e da direita respectivamente.
- **goal_l ou goal_r** – Ocorrência de um gol em favor do time da esquerda ou da direita respectivamente.

2.5 Informação Visual

Cada um dos clientes, agentes responsáveis pelo comportamento dos jogadores, conectados ao Soccerserver via *socket*, recebe periodicamente através dessa conecxão uma mensagem contendo os objetos captados por uma câmera de vídeo que estaria situada no topo do respectivo robô jogador. Essa mensagem possui a seguinte sintaxe:

(ObjName Distance Direction DistChng DirChng BodyDir HeadDir)
ObjName ::= (player *Teamname UniformNumber)*
| (gol [l|r])
| (ball)
| (flag c)
| (flag [l|c|r] [t|b])
| (flag p [l|r] [t|c|b])
| (flag g [l|r] [t|b])
| (flag [l|r|t|b] 0)
| (flag [t|b] [l|r] [10|20|30|40|50])
| (flag [l|r] [t|b] [10|20|30])
| (line [l|r|t|b])

Distance, *Direction*, *DistChng* e *DirChng* são calculados a partir das seguintes expressões:

$$p_{rx} = p_{xt} - p_{xo} \tag{2.1}$$

$$p_{ry} = p_{yt} - p_{yo} \tag{2.2}$$

$$v_{rx} = v_{xt} - v_{xo} \tag{2.3}$$

$$v_{ry} = v_{yt} - v_{yo} \tag{2.4}$$

$$Distance = \sqrt{p_{rx}^2 + p_{ry}^2} \quad (2.5)$$

$$Direction = \arctan(p_{ry}/p_{rx}) - -a_o \quad (2.6)$$

$$e_{rx} = p_{rx}/Distance \quad (2.7)$$

$$e_{ry} = p_{ry}/Distance \quad (2.8)$$

$$DistChng = (v_{rx} * e_{rx}) + (v_{ry} * e_{ry}) \quad (2.9)$$

$$DirChng = [(-(v_{rx} * e_{ry}) + (v_{ry} * e_{rx}))/Distance] * (180/\pi) \quad (2.10)$$

onde (p_{xt}, p_{yt}) é a posição do objeto observado, (p_{xo}, p_{yo}) é a posição do observador, (v_{xt}, v_{yt}) é a velocidade do objeto observado, (v_{xo}, v_{yo}) é a velocidade do observador, a_o é direção absoluta para a qual o observador esta voltado. Adicionalmente, (p_{rx}, p_{ry}) e (v_{rx}, v_{ry}) são respectivamente a posição relativa e a velocidade relativa do objeto observado, e (e_{rx}, e_{ry}) o vetor unitário paralelo ao vetor posição relativa.

2.6 Visibilidade

A informação visual enviada para os agentes corresponde aos objetos identificados pelo robô, no setor visível do jogador. Esse setor visível pode assumir três diferentes ângulos: normal [-45, 45], amplo [-90, 90], direcionado [-22.5, 22.5]. Cada um dos ângulos de visão pode assumir os valores alto e baixo para a qualidade da informação visual.

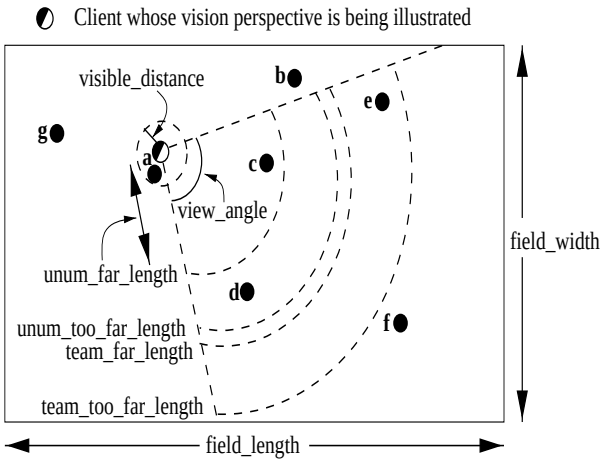

Figura 2.3: Campo visual do jogador

Os valores assumidos pelo ângulo de visão e sua respectiva qualidade influenciam diretamente na taxa de atualização da informação visual enviada pelo simulador. Para o ângulo de visão *normal* e a qualidade da informação *alta*, valor pré-estabelecido, a

atualização da informação visual se dá a cada 150 ms. Pode-se calcular a taxa de atualização da informação visual pela equações 2.11 e 2.12.

$$\text{view\_frequency} = \text{sense\_step} * \text{view\_quality\_factor} * \text{view\_width\_factor} \quad (2.11)$$

onde view_quality_factor é igual a 1 quando ViewQuality é igual a high, e 0.5 para ViewQuality igual a low, view_width_factor é igual a 2 quando ViewWidth é igual a narrow, 1 para ViewWidth igual a normal, e 0.5 para ViewWidth igual a wide.

$$\text{view\_angle} = \text{visible\_angle} * \text{view\_width\_factor} \quad (2.12)$$

onde *view_ width_factor* é igual a 0.5 para *view_ width* é igual a *narrow*, 1 para *view_ width* é igual a *normal*, e 2 para *view_ width* é igual a *wide*.

Existe ainda uma região no raio de 3m do jogador, chamada de *Vizinhança*, tal que quando um objeto nela se encontra, estando fora do setor visível, é apenas identificado o tipo do objeto (bola, gol, etc).

2.7 Localização

O sistema de visão dos robôs é local. As informações visuais recebidas pelo agente, mostram os objetos capturados por uma câmera situada no topo do robôs. Nesse pacote de informação visual, os objetos presentes no campo de visão do robôs aprecem com atributos que determinam sua distância e direção em relação ao centro de massa do robô. Sendo assim, cada um dos robôs possui um sistema de coordenadas polar e local. Faz-se necessário a partir desse sistema de coordenada local implementar mecanismos para permitir ao robô se localizar no universo em questão.

Tais mecanismos podem se contituir de um modelo numérico para o campo de futebol, baseado em coordenadas cartesianas (x, y), adotando-se o centro do gramado como sendo a coordenada $(0, 0)$ e um método para determinar as coopenadas (x_i, y_i) do robô a partir da informação visual recebida pelo agente. Outras abordagens podem ser utilizadas para determinar a localização do robô a partir das informações visuis recebidas utilizando-se redes neurais, lógica difusa ou sistemas baseados em conhecimento.

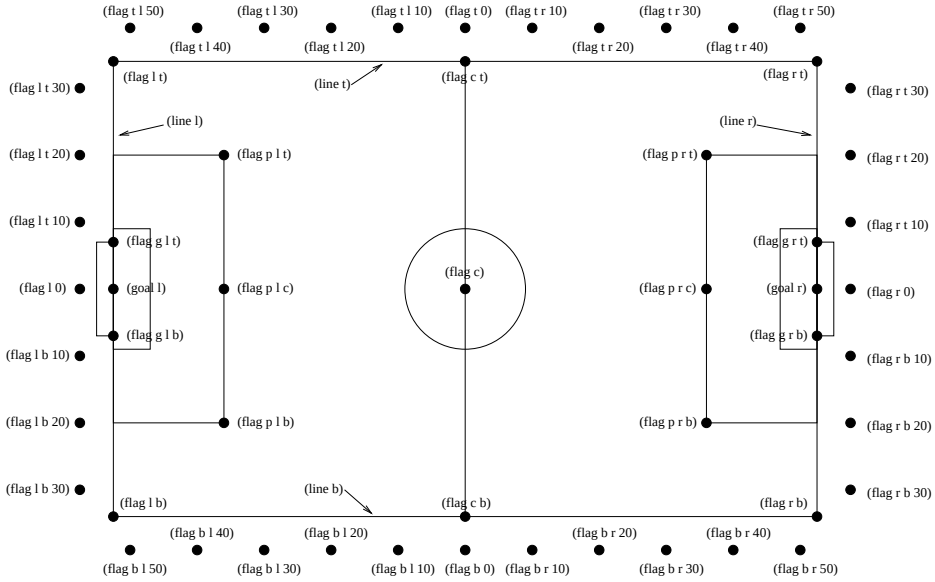

Figura 2.4: Objetos fixos utilizados para localização do robô.

2.8 Informação Auditiva

Além da informação visual, o agente recebe assincronamente via *socket*, mensagens enviadas pelo juiz informando alterações no estado do jogo (por exemplo, ocorrência de um gol) e mensagens enviadas por outros jogadores num raio de 50m.

O jogador poderá processar apenas uma mensagem de cada time a cada dois ciclos de simulação. Essas mensagens são ordenadas de acordo com a ordem de chegada. Existe ainda uma limitação quanto ao comprimento da mensagem (512 caracteres).

2.9 Comandos

Em resposta às informações visuais e auditivas, recebidas pelo agente, deve ser enviado ao Soccer Server comandos que serão responsáveis pela ação do jogador. O Soccer Server aceita um novo comando a cada 20 ms, entretanto existem limitações para a utilização destes comandos. Os comandas disponíveis e suas respectivas limitações são:

- **(turn** *mi***)** – Permite que o jogador execute um giro de [-180°, 180°] graus em torno de seu próprio eixo. Associa-se a este comando um argumento *mi* (momento de inércia) que considera a inércia do objeto.

- **(turn_neck** *ang***)** – Permite que o jogador gire [-180, 180] graus, em torno de seu próprio eixo, a parte superior do robô onde se enconta a câmera.

- **(dash** p**)** – Incrementa a velocidade do jogador na direção atual com a potência $p \in$ [-30, 100] especificada no argumento.
- **(kick** p d**)** – Chuta a bola com a potência $p \in$ [-30, 100], na direção $d \in$ [-180, 180]. Este comando só terá efeito se a bola estiver na margem de chute especificada pelo Soccer Server.
- **(catch** d**)** – Tenta agarrar a bola na direção d. A possibilidade de sucesso é definida no parâmetro do Soccer Server "catch possibility" . A bola deve se encontrar em uma área definida como "goalie cacthable area" , um retângulo de 2m × 1m. Esse comando somente poderá ser utilizado pelo goleiro.
- **(say** *msg***)** – Difunde uma mensagem msg para todos os jogadores num raio de 50m.
- **(change_view** a q**)** – Modifica o ângulo do setor visível para a ([-45, 45], [-22.5, 22.5], [-90, 90]) e a qualidade da informação visual para q (low, high).

2.10 Temporização

A comunicação entre o *Soccer Server* e os agentes envolve mensagens síncronas e assíncronas (figura 2.5).

- **Tempo de Simulação** – Cada ciclo de simulação tem a duração de 100 ms.
- **Aceitação de Comandos** – O Soccer Server aceita um novo comando a cada 20 ms, entretanto apenas um comando turn, kick ou dash é executado a cada ciclo de simulação (100 ms).
- **Informação Visual** – Depende do modo de visão utilizado *(normal, narrow, wild)* e da qualidade da informação visual *(high, low)* utilizada. Para o modo normal com qualidade da informação high, uma nova mensagem contendo a informação visual é recebida a cada 150 ms.
- **Informações Auditivas** – São trocadas assincronamente pelo juiz e pelos demais agentes.

2.11 Sincronização

A sincronização entre o agente e o Soccer Server é mais um dos desafios a serem enfrentados na implementação do agente. Os comandos enviados para o simulador

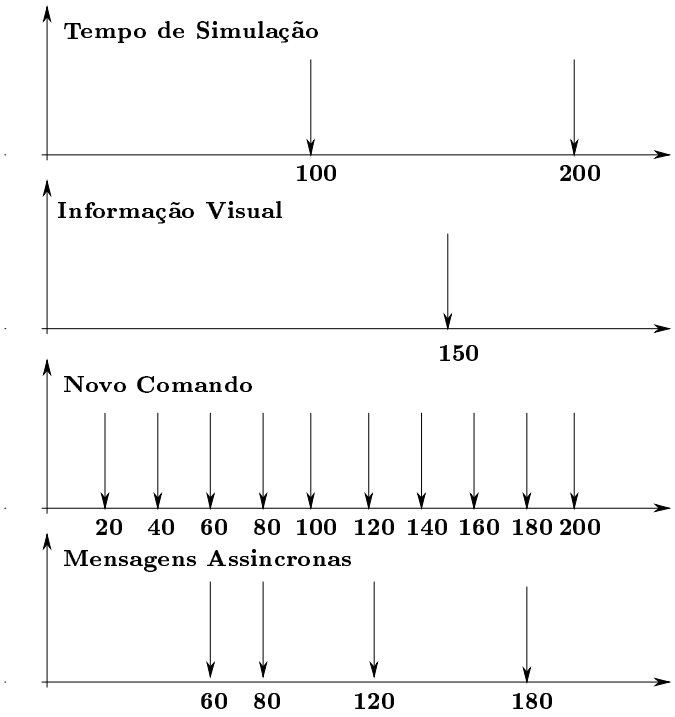

Figura 2.5: Temporização do Soccer Server

possuem um tempo de resposta, e nada garante que na próxima informação visual o efeito do comando enviado seja percebido. Além disso, comandos podem ser perdidos, e a informação enviada pelo simulador pode ser inconsistente. A incorreta sincronização entre o agente e o Soccer Server pode levar o jogador a apresentar um comportamento completamente indesejado.

2.12 O Agente Treinador

Assim como no futebol disputado entre humanos, uma partida decorre sem que interrupções oriundas de fora do campo sejam acatadas. Apenas os jogadores e o juiz da partida podem influenciar e/ou controlar a partida. Entretanto, seria muito interessante ter maior controle sobre o jogo durante o processo de desenvolvimento dos agentes que compõem o time, de modo que fosse possível executar seções de treinamento nas quais certas ações, tais como, dribles, lançamentos e jogadas ensaiadas, são testadas segundo um processo automatizado, dando oportunidade de se utilizar métodos de aprendizado.

Com este objetivo um agente privilegiado chamado treinador foi introduzido. O agente treinador é dotado das seguintes habilidades:

- controlar o status da partida *(play_ mode)*.
- difundir mensagens auditivas.
- mover os objetos (bola e jogadores) para qualquer posição do campo independente do status da partida.
- obter informações a respeito de qualquer dos objetos móveis (jogadores e bola) no campo.

2.13 O Treinador e o Juiz

O Soccer Server provê um módulo interno que automaticamente assume o papel de juiz da partida. Entretanto, é possivel atribuir ao agente treinador o completo controle do jogo desativando assim o módulo juiz da partida. Nesse caso ele passa a ser responsável pelo monitoramento do status partida, efetuando as respectivas mudanças de status *(play-mode)* de acordo com suas próprias regras. É possivel ainda manter o juiz da partida e o treinador ativos, e controlando o jogo. Nesse caso o agente treinador pode ser utilizado para controlar uma sessão de treinamento ou ainda para prover aos jogadores informações a respeito de seu desempenho.

2.14 O Treinador em um Jogo

O agente treinador era utilizado apenas na etapa de desenvolvimento dos agentes até 1998, sendo proibida sua utilização nas competições. A partir de 1999, uma nova categoria foi adicionada à categoria de simuladores onde é permitida a utilização do agente treinador durante a competição. O objetivo é utilizar o agente treinador para obeservar a partida e passar infomações estratégicas aos jogadores. Para tanto as habilidades do agente treinador durante uma partida limitam-se a:

- obter informações a respeito da posição da bola e dos jogadores.
- receber e difundir mensagens auditivas.

Não é permitido ao agente treinador conduzir as ações de um jogador passo a passo, mantendo assim o carater autônomo dos jogadores. Assim como no futebol de humanos, a atuação do treinador deve se limitar a observar a partida, corrigir o posicionamento dos jogadores, promover alterações táticas etc.

2.15 Robôs de Pequeno Porte (F-180)

A RoboCup pssui ainda outras categorias, onde as partidas de futebol de robôs acontecem entres robôs reais, a exemplo da categoria F–180: robôs de pequeno porte.

Na Categoria de robôs de pequeno porte, com cinco robôs em cada time, as competições são disputadas em campo, 152,5cm x 274cm, verde. São permitidos tanto um sistema de visão global, com uma câmera no teto, como sistema distribuído de visão, onde cada um dos robôs tem sua própria câmera embarcada. Entretanto o sistema de visão global deve em breve não mais ser permitido nesta categoria.

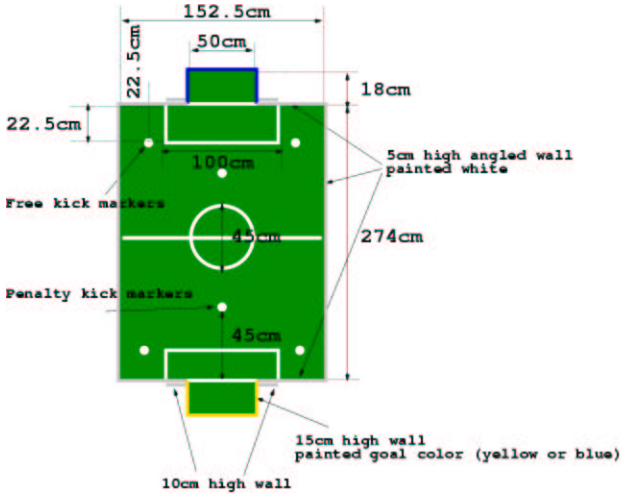

Figura 2.6: Dimensões do campo utilizado na categoria F-180 da RoboCup

Os robôs não devem ultrapassar uma área de 180 cm2. O robô deve caber dentro de um ciclindro de 18 cm de diametro. Em se tratando de um robô cuja área da base assume o formato retangular a diagonal, deve ser inferior a 18 cm. A máxima altura deve ser de 15 cm, se o time optar por um sistema de visão global, ou 22.5 cm, se este utilizar a visão embarcada.

A comunição entre os robôs não sofre qualquer tipo de restrição, o que permite a utilização de estratégias de cooperação mais elaboradas. Nesta categoria, os desafios envolvidos englobam várias áreas da Automação Industrial, Inteligência Artificial, Robótica, Controle de Processos, Reconhecimento de Padrões, Sistemas de Tempo Real, Sistemas Distribuídos, Redes Sem Fio, etc.

Nessa categoria, os desafios presentes na categoria de robôs simulados, implentação de agentes autônomos e sistemas multiagentes, adicionam-se à construção de robôs móveis e a integração destes agentes autônomos aos robôs móveis, tornando-os veículos autônomos capazes de realizar tarefas nos chamados *ambientes abertos*, ambientes que não estão sob total controle do projetista do sistema, como por exemplo inspeção de

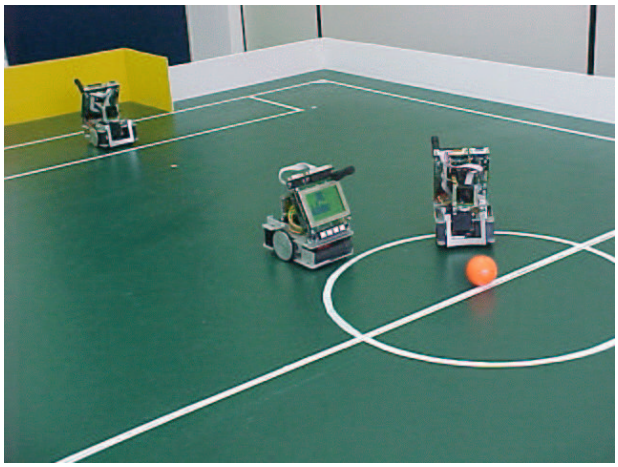

Figura 2.7: Robôs de pequeno porte, categoria F-180 da RoboCup, com sistema de visão embarcado.

plataformas de perfuração de petróleo, inspeção de oleodutos, realização de tarefas em ambientes inóspitos, insalubres ou perigosos.

Este tema apresenta importantes desafios teóricos, representados pelos diversos problemas ainda em aberto a respeito da utilização, integração e avaliação de formalismos para os sistemas robóticos e de informação envolvidos.

2.16 O SoccerServer e Sistemas Multiagentes

Na categoria de robôs simulados, o desafio consiste em implementar sistemas computacionais autônomos, conhecidos como *agentes autônomos* [Mae94], capazes de: reconhecer o ambiente onde encontam-se imersos, o campo de futebol e a bola, o adversário, o gol e as bandeiras de sinalização; representar este ambiente, reconhecer em que parte do campo o jogador se encontra e suas condições de jogo; estabelecer objetivo, planejar uma jogada ou corrigir seu posicionamento; planejar ações e modificar o ambiente para realização de seus objetivos, realizar lançamentos, passar a bola, chutar em gol, desarmar o oponente, etc. Estes agentes autônomos devem interagir como os outros agentes do time estabelecendo e realizando metas globais tais como a execução de uma jogada ensaiada, fazendo emergir um comportamento social inteligente. A implementação de agentes autônomos e a interação destes agentes formando um sistema multiagente, a exemplo do UFSC-Team, são desafios relacionados a Inteligência Artificial.

Na categoria de robôs de pequeno porte, a implementação de um time de futebol de robôs para a categoria de robôs de pequeno porte (f-180) da RoboCup, adiciona aos desafios apresentados em 2.3 problemas relacionados à robótica móvel como os

algorítmos para movimentação e contrôle de trajetória do robô móvel, a navegação autônoma; nas redes sem fio faz-se necessário na implementação de mecanismos para gestão de grupos e macanismos de tolerância à falta; no reconhecimento de padrões, algorítmos para tratamento de imagem e reconhecimeto dos objetos pertencentes ao ambiente e sua localização; nos sistemas operacionais de tempo real o desenvolvimento de um micro kernel de tempo real capaz de ser executado embarcado.

Capítulo 3

UFSC-Team

3.1 UFSC-Team na RoboCup 98

Em sua primeira participação na categoria simuladores da RoboCup'98, o *UFSC-team* apresentou uma arquitetura de agente cognitivo concorrente [CB98b]. A idéia principal dessa primeira arquitetura era implementar percepção, ação, comunicação, cooperação, planejamento e tomada de decisão utilizando a programação concorrente [AS83].

Essa primeira arquitetura concorrente baseava-se em três processos: *Interface, Coordinator* e *Expert*. O processo Interface foi projetado para manipular a percepção e a ação. A interação entre agente e ambiente suportada pelo Soccer Server consiste na troca de mensagens através dos *sockets* no domínio Inet. A função do processo Interface era inicialmente converter as informações visuais recebidas do Soccer Server *(percepção)* e as mensagens recebidas do juiz e dos demais agentes *(comunicação)* na linguagem *Parla* [CB97], A Linguagem para Comunicação de Agentes utilizada pelos agentes do UFSC-Team.

O processo Coordinator responsabilizava-se pela comunicação, pela abertura e pelo gerenciamento dos processos de cooperação entre os agentes do UFSC-Team. Baseado na arquitetura original proposta no ambiente Expert-Coop [BC97] [Cos97], esse processo era responsável pelo gerenciamento da comunicação entre os agentes (por exemplo,

este processo recebia todas as mensagens provenientes dos demais agentes e as manipulava). Entretanto, de acordo com as regras da categoria simuladores da RoboCup, toda a comunicação entre os agentes deve ser efetuada através do Soccer Server. Sendo assim, tanto a percepção do agente quanto as mensagens de comunicação entre os agentes do time são recebidas pelo mesmo *socket* no domínio Inet. Consequentemente, nesta implementação, o processo Interface recebia as mensagens trocadas entre os agentes e as mensagens enviadas pelo juiz, redirecionando-as para o processo Coordinator onde eram manipuladas.

Finalmente, o processo Expert era responsável pelo planejamento e pelo processo decisório do agente. Possuia um sistema baseado em conhecimento encapsulado no qual a percepção, as mensagens enviadas pelo juiz e as mensagens enviadas pelos demais agentes do UFSC-Team eram armazenadas e utilizadas para inferir as decisões apropriadas, de acordo com as regras de um sistema baseado em conhecimento. Estes três processos comunicavam-se através dos *sockets* no domínio do Unix (figura 3.1).

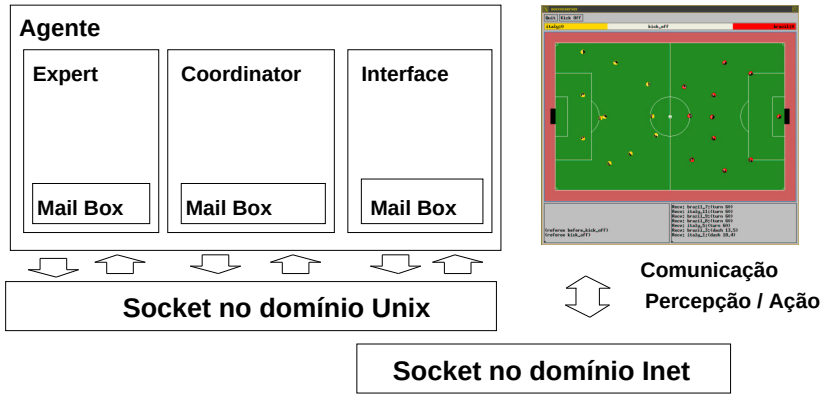

Figura 3.1: Arquitetura do agente utilizado pelo UFSC-Team'98.

Essa primeira implementação concorrente, com o processo decisório do agente centralizado, apresentou alguns problemas de sincronização entre o agente e o ambiente e a resposta em tempo real não foi tão rápida quanto se esperava. Realmente, a melhor resposta em tempo real apresentada pelos agentes do UFSC-Team'98 ficou entre 70 e 80 ms, mesmo utilizando-se o a técnica de Raciocínio Baseado em Casos [All94] para dividir a base de conhecimento em módulos independentes. Além disso, o sistema baseado em conhecimento, responsável pela tomada de decisão do agente, tornou-se complexo. Esse sistema baseado em conhecimento possuía regras destinadas a tratar desde informação de alto nível, como por exemplo que tipo de jogada ensaiada envolvendo outros agentes deveria ser escolhida, ou quais agentes iriam tomar parte em uma dada jogada, até informação de baixo nível, como qual o valor do momento de inércia deveria ser passado no parâmetro do comando turn para se executar um giro ou quais

valores da potência e a direção do chute deveriam ser utilizados.

Visando solucionar os problemas apresentados pelo UFSC-Team'98 na RoboCup'98, migrou-se de uma arquitetura concorrente, com tomada de decisão centralizada, para uma arquitetura de agentes autônomos inspirada no modelo genérico para agentes cognitivos proposto em [Bit97]. O modelo baseado na hipótese que as atividades cognitivas possuem três características principais: auto-organização, natureza evolutiva e dependência histórica. Segundo este modelo, um agente cognitivo apresenta três níveis decisórios: reativo, instintivo e cognitivo (ver figura 3.2).

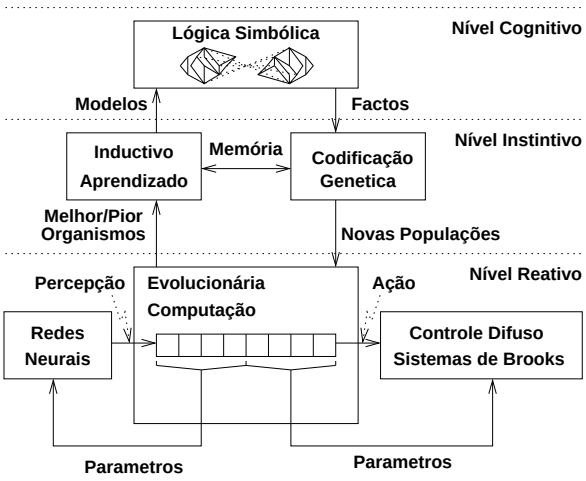

Figura 3.2: Modelo genérico para agentes cognitivos.

Cada um níveis decisórios, juntamente com o nível inferior pretende modelar um agente completo e cada novo nível decisório incrementa a complexidade do comportamento do agente. Este modelo levou a uma descentralização do processo decisório do agente utilizado pelo UFSC-team, dando origem ao Agente Autônomo Concorrente [CB99], onde os três níveis decisórios foram implementados segundo uma abordagem concorrente.

3.2 Agente Autônomo Concorrente

O modelo baseado na hipótese que as atividades cognitivas possuem três características principais: auto-organização, natureza evolutiva e dependência histórica. Segundo este modelo, um agente cognitivo apresenta três níveis decisórios: reativo, instintivo e cognitivo (ver figura 3.2). Cada um níveis decisórios, juntamente com o nível inferior pretende modelar um agente completo e cada novo nível decisório incrementa a complexidade do comportamento do agente. Este modelo levou a uma descentralização do processo decisório do agente utilizado pelo UFSC-team, dando origem ao Agente

Autônomo Concorrente [CB99], onde os três níveis decisórios foram implementados segundo uma abordagem concorrente.

O modelo de concorrência foi mantido com os mesmos três processos: Interface, Coordinator e Expert, mas atualmente cada um desses três processos possui um motor de inferência distinto e é responsável por um dos três níveis decisórios. Ambas as implementações foram escritas utilizando a linguagem de programação C++, e elas integram o ambiente para desenvolvimento de sistemas multiagentes cognitivos sob restrições de tempo real chamado Expert-Coop++. O ambiente Expert-Coop++ suscedeu ao ambiente Expert-Coop [Cos97] [BC97], implementado em LISP.

O motor de inferência do nível reativo é implementado no processo Interface e é responsável pela resposta em tempo real do agente, por exemplo por receber a informação visual do Soccer Server e por enviar os comando de ação adequados. Consiste de um conjunto de controladores difusos. A cada instante apenas um controlador difuso encontra-se ativo, sendo responsável pela escolha dos comandos que devem ser enviados ao Soccer Server e de seus respectivos valores. Essa escolha se baseia na informação visual recebida e nas variavéis de estado dor robô contidas na informação *sense-body*, e é determinada pelas regras do controlador difuso ativo. Cada um dos controladores difusos disponíveis no agente representa um comportamento específico e possui algumas condições associadas que especificam em que situação esse controlador é efetivo.

O motor de inferência do nível instintivo é implementado no processo Coordinator e se responsabiliza por atualizar as variáveis simbólicas utilizadas no nível cognitivo e por escolher o comportamento mais adequado para a situação corrente, isto é, qual o controlador difuso deve ser utilizado no nível reativo para alcançar a meta local em vigor. Uma meta pode ser atingida por uma seqüência de comportamentos que conduzem o agente a uma situação desejada. A escolha dessa seqüência de comportamentos é implementada a partir de um sistema especialista de apenas um ciclo de inferência que escolhe a cada vez que o estado do jogo muda o comportamento reativo mais adequado. Cada estado do jogo é definido por um conjunto de condições que são monitoradas no nível instintivo. Essas condições se referem à percepção e às mensagens enviadas pelo juiz, e são utilizadas como premissas das regras, análogas às do nível reativo. Mas no nível instintivo as implicações das regras consistem em variáveis simbólicas utilizadas para atualizar a base de conhecimento do nível cognitivo e/ou para selecionar um novo comportamento no nível reativo. A cada instante, o comportamento selecionado deve responder aos estímulos do ambiente buscando alcançar a meta, devendo também ter suas condições satisfeitas pelo estado da partida. Uma vez escolhido um comportamento, o nível instintivo se mantém monitorando os requisitos associados a este comportamento. Caso algum desses requisitos não mais se verifique, ele utiliza

seu conjunto de regras para inferir um novo comportamento. Caso não seja possível, a meta corrente está comprometida, e uma nova meta deve ser especificada. O nível instintivo também manipula as mensagens enviadas pelo juiz da partida informando uma mudança no status do jogo. Essas mudanças são tratadas de forma análoga à das mudanças de estado da partida, levando o nível instintivo a escolher o comportamento adequado à nova situação.

Finalmente o motor de inferência do nível cognitivo é implementado no processo Expert e responsabiliza-se por determinar as metas locais e globais do agente. O nível cognitivo não interfere diretamente sobre o nível reativo, ele apenas determina a meta local e a envia para o nível instintivo. Essa meta tem efeito direto nas regras do nível instintivo, que seleciona o comportamento reativo adequado. Enquanto uma meta não é alcançada, ou falha, o nível cognitivo não especificar uma nova meta e esse tempo livre é utilizado para o planejamento estratégico. Esse planejamento consiste em determinar as possíveis metas locais futuras, de acordo com os resultados atuais e as especificações das requisições de cooperação para alcançar as metas globais. Essas requisições são manipuladas pelo processo Coordinator e resultam na adoção de novas metas locais por parte de outros agentes compatíveis com a meta global desejada. O nível cognitivo também é implementado por um sistema especialista, entretanto esse sistema especialista pode ser muito mais complexo do que o implementado no nível instintivo, pois possui um tempo de resposta maior (figura 3.3).

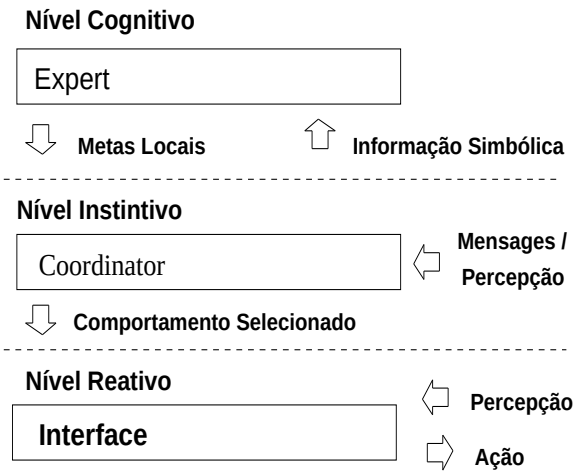

Figura 3.3: Fluxo de Informação no Agente Autônomo Concorrente.

Nessa nova implementação, os três processos que compõem o agente utilizam uma abordagem de programação multi-thread. Essa tecnologia permite particionar o processo e executar concorrentemente as partes resultantes. Em nosso caso, cada processo é composto por dois threads. O primeiro é responsável por manipular a interrupção

SIGIO do Unix, usada para informar que uma nova mensagem foi recebida pelo *socket* e por colocar essa nova mensagem no mailbox. O outro thread, o principal, se responsabiliza pela execução das atividades do processo propriamente dito. A exclusão mútua entre os dois threads é feita utilizando semáforos. Essa implementação consiste em uma abordagem concorrente do clássico problema produtor/consumidor. Isso evita que o processo principal despenda um tempo precioso verificando se existe ou não numa nova mensagem no socket.

3.2.1 Nível Reativo

O motor de inferência do nível reativo é completamente implementado no processo Interface. Esse processo é composto por um mailbox, um conjunto de controladores difusos, um filtro de entrada e um filtro de saída (figura 3.2). O mailbox é responsável pela recepção e pelo ordenamento das mensagem recebidas pelo processo. Todas as mensagens enviadas pelo Soccer Server relativas à percepção (informação visual) serão armazenadas neste mailbox. As mensagens enviadas pelo juiz do jogo e pelos demais agentes são re-direcionadas para o processo Coordinator.

Os controladores difusos são implementados utilizando-se a biblioteca *CNCL* escrita em C++ para auxiliar a implementação de sistemas especialistas difusos ou controladores. Cada um desses controladores difusos é responsável por uma habilidade reativa do agente chamada Comportamento. Inicialmente o seguinte conjunto de Comportamentos foi escolhido para ser implementado nos agentes do UFSC-Team: *Inicializar_jogador, Saída_de_Bola, Passar_a_Bola, Chute_em_Gol, Driblar_Oponente, Conduzir_Bola_Avante, Mover_para_Posição, Mover_para_Bola, Agarrar_Bola, Desarmar_Oponente, Marcar_Oponente, Observar_bola.*

O conjunto de controladores difusos associado a cada agente do UFSC-Team depende do grupo ao qual esse agente pertence: goleiro, defensor, meio-campo, atacante. Realmente, não faz sentido para os agentes que pertencem aos grupos meio-campo e atacante possuir um controlador difuso para Agarrar-Bola, assim como também não faz sentido para o goleiro possuir um controlador difuso para chutar a bola no gol adversário.

O filtro de entrada é responsável por extrair da informação visual o valor das variáveis lingüísticas utilizadas pelo controlador difuso ativo. O filtro de saída, por sua vez, verifica o valor das saídas do controlador difuso e as combina observando os seguintes critérios:

- **Saída Nula** - Se a potência do impulso, *(dash p)*, e o momento de inércia para

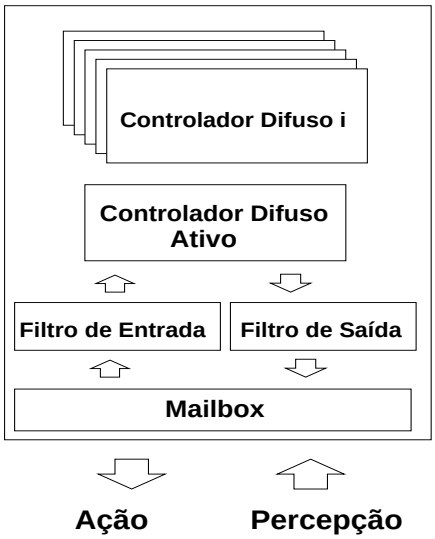

Figura 3.4: Nível Reativo: o Processo Interface.

o giro, *(turn mi)*, apresentarem valor nulo, o respectivo comando não é enviado ao Soccer Server.

- **Impulso e Giro simultâneos** - Caso o controlador difuso apresente simultaneamente valores nas saídas de potência do impulso, *(dash p)* e do momento de inércia para o giro, *(turn mi)*, o comando *turn*, é enviado primeiramente, e somente após um atraso intencional de 20 ms o comando *dash* é enviado para o Soccer Server.

- **Direção e Potência do Chute** - As saídas, direção do chute *(kick direction)* e de potência do chute *(kick power)* são sempre combinadas para compor o comando kick.

A maioria dos controladores difusos possui quatro saídas: *kick direction*, *kick power*, *dash power* e *turn moment*. Os controladores *Passar_a_Bola* e *Chute_em_Goal* possuem apenas as saídas *kick direction* e *kick power* e o controlador difuso *Mover_para_Posição* possui apenas as saídas *turn moment* e *dash power*. As entradas são variáveis lingüistícas, dependendo de qual Comportamento está ativo naquele momento. Cada controlador difuso possui suas próprias variáveis lingüistícas e o filtro de entrada se responsabiliza por extrair das informações visuais recebidas do *Soccer Server* e das informações *sense-body*, também enviada pelo *Soccer Server*, a cada novo ciclo de simulação, os valores destas variáveis lingüistícas.

A utilização de controladores difusos para implementação do nível reativo possui algumas vantagens. Primeiramente, é possível sincronizar o agente simplesmente ajustando-se a relação entre entrada e saída, em outras palavras, ajustando-se o ganho

do controlador. Esse ajuste do ganho é realizado nos conjuntos difusos que representam a entrada e a saída do controlador. Pode-se ainda obter uma sintonia fina do controlador ajustando-se esses conjuntos difusos. A figura 3.5 representa os conjuntos difusos utilizados para a saída turn moment e a respectiva variável lingüistíca direção da bola.

As regras utilizadas para a implementação do controlador difuso podem ser escritas intuitivamente, evitando o dispêdio de um longo tempo para modelar um ambiente extremamente dinâmico. É possível ainda utilizar algoritmos genéticos para evoluir os conjuntos difusos.

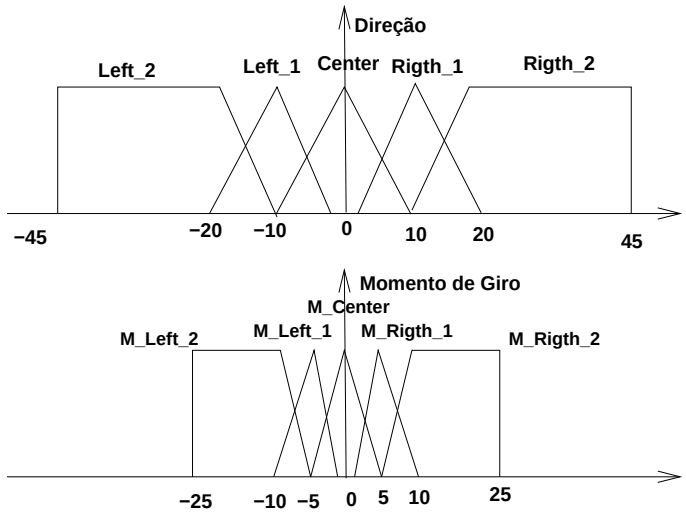

Figura 3.5: Conjuntos difusos utilizados para a implementação para a saída *turn_ moment* e a respectiva variável lingüística *direção_ da_ bola*

Outra importante vantagem da utilização de controladores difusos para se implementar os comportamentos reativos dos agentes é poder assegurar que um dado controlador difuso estará sempre apto a satisfazer os requisitos de tempo real, porque esses controladores difusos são sistemas determinísticos. Além disso, uma vez que o controlador difuso ativo corresponde ao comportamento reativo mais apropriado para uma dada situação, os motores de inferência dos níveis instintivo e cognitivo podem dispor de mais tempo para realizar tarefas mais sofisticadas como extrair informações simbólicas da percepção, planejar, estabelecer metas ou participar de processos de cooperação.

3.2.2 Nível Instintivo

O motor de inferência do nível instintivo é implementado no processo Coordinator e é responsável pela execução das metas locais do agente e pela geração da informação simbólica para atualização da base de conhecimento do nível cognitivo. Consiste de um sistema especialista de um único ciclo de inferência que escolhe, a cada mudança de

estado do jogo, o comportamento reativo mais adequado para a meta local em vigor. Esta meta local em vigor é estabelecida pelo nível cognitivo e determina o conjunto de regras a ser utilizado pelo motor de inferência. Cada estado do jogo é definido por um conjunto de condições a serem verificadas na percepção. Os valores destas condições são determinados experimentalmente.

As entradas do motor de inferência do nível instintivo são a percepção, recebida pelo processo Interface e as mensagens enviadas pelo juiz da partida e pelos demais agentes do UFSC-Team. A percepção consiste na mesma informação visual recebida de forma síncrona pela Interface e proveniente do Soccer Server mas, diferentemente do nível reativo, o nível instintivo possui uma memória. Essa memória consiste em um buffer, onde a percepção é armazenada e cujo tamanho inicial é definido na implementação do agente. Isso torna possível escolher o montante de informação visual (percepção) usado no ciclo de inferência. Por exemplo, assumindo que uma nova informação visual é recebida a cada 150 ms e que o tamanho do buffer é de três, em um dado instante t, o sistema especialista de um ciclo de inferência irá considerar as informações visuais enviadas nos instantes t, $t_1$50 e $t_3$00.

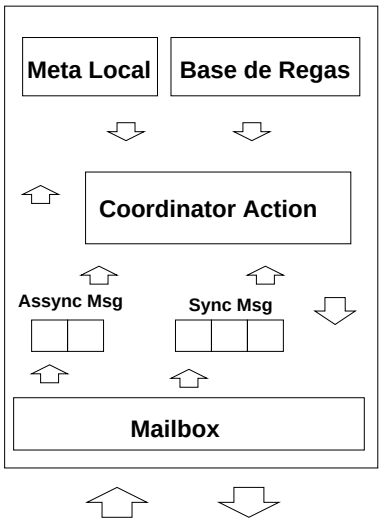

Figura 3.6: Nível Instintivo: o processo Coordinator.

A percepção é armazenada no buffer para mensagens síncronas (Sync Msg) e as mensagens enviadas pelo juiz da partida e pelos demais agentes do UFSC-Team armazenadas num outro buffer, destinado às mensagens assíncronas (Assync Msg) (figura 3.6). Cada vez que um desse dois buffers é atualizado, ou quando uma nova meta local é recebida do nível cognitivo, o sistema especialista é executado. Dada uma entrada, as regras estão aptas a reconhecer as mudanças no estado do jogo. O resultado da execução destas regras pode ser uma atualização da base de conhecimento do nível

cognitivo e/ou a seleção de um novo controlador difuso para conduzir o agente do estado atual até a meta local.

Suponha, por exemplo, que o time oponente possui a posse de bola e que o UFSC-Team está realizando uma jogada defensiva, na qual a meta local é Recuperar_Posse_da_Bola e o comporatmento reativo atual é Marcar_Oponente. Suponha ainda que o jogador do time adversário que possui a posse da bola cometeu um erro e chutou a bola para fora do campo. Então o status do jogo é modificado pelo Soccer Server para kick_in_side, ou seja, reposição de bola para o UFSC-Team; essa mudança é reconhecida por uma mensagem enviada pelo juiz do jogo informando o novo status da partida. Nesse caso, um novo comportamento reativo pode ser selecionado diretamente, por exemplo, Mover_para_Bola e isso significa também que a meta local Recuperar_Posseda_Bola foi alcançada. O nível cognitivo será informado e estabelecerá uma nova meta local. Numa situação como essa tanto a execução, reagindo a um estímulo do ambiente através do nível reativo, quanto o planejamento através do nível cognitivo acontecem concorrentemente.

3.2.3 Nível Cognitivo

O nível decisório cognitivo é implementado no processo Expert. Consiste em um sistema baseado em conhecimento simbólico e orientado a objetos que manipula tanto as informações simbólicas recebidas do nível instintivo quanto as mensagens assíncronas recebidas dos demais agentes do UFSC-Team, gerando metas locais e globais. Esse sistema baseado em conhecimento possui um motor de inferência, uma base de fatos, uma base de regras local e uma base de regras social (figura 3.7).

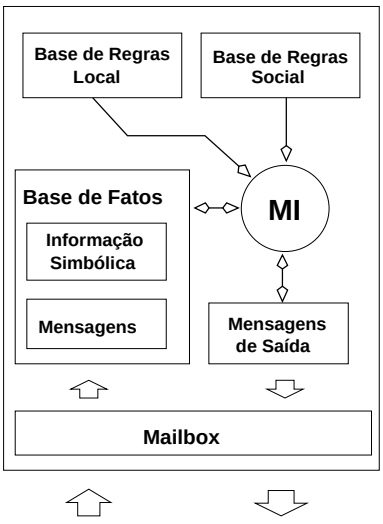

Figura 3.7: Nível Cognitivo: o processo Expert.

A base de fatos armazena as informações simbólicas geradas pelo nível instintivo e pelo motor de inferência e as mensagens recebidas dos demais agentes. Essas informações simbólicas geradas pelo nível instintivo são enviadas para o nível cognitivo através de mensagens locais, trocadas entre os processos que implementam os nivéis decisórios do agente, e são armazenadas utilizando a Lógica como formalismo de representacão de conhecimento segundo o formato objeto, atributo e valor.

```
(logic (<objeto> <atributo> <valor> ))
```

As mensagens armazenadas na base de fatos utilizam o mesmo formato da linguagem para comunicação de agentes *(LCA)* Parla [CB97].

```
((to ...) (from ...) (round ...) (alpha ...) (grade ...)
 (deadline ...) (time-stamp ...)
 (body (<primitiva de comunicação> <re\-pre\-sen\-ta\-ção de conhecimento>)))
```

O motor de inferência permite tanto a manipulação de informação simbólica segundo a representação de conhecimento adotada, a lógica, como a manipulação das mensagens recebidas pelo agente segundo o formato da linguagem Parla, gerando novas informações simbólicas, que são armazenadas na base de fatos e mensagens que, são enviadas a outros agentes ou para o nível instintivo. Basicamente essas mensagens consistem em metas locais a serem enviadas para o nível instintivo, informações a serem usadas em processos de cooperação ou mensagens destinadas a outros agentes.

A entrada do processo Expert é sempre uma mensagem. Esta mensagem pode conter uma informação simbólica gerada pelo nível instintivo, no caso uma mensagem local, ou pode ainda conter uma informação enviada por um outro agente da comunidade. No primeiro caso, o conteúdo da mensagem é armazenado na base de informação simbólica e o motor de inferência utiliza essa base e a base de regras local para gerar novas informações simbólicas. No caso das mensagens enviadas por outros agentes estas são armazendas em uma base de mensagens, e o motor de inferência utiliza tais mensagens, a base de informação simbólica e a base de regras social, para inferir novas mensagens.

O nível cognitivo é responsável pelo estabelecimento de metas locais e pela interação do agente com a comunidade estebelecendo metas globais através de processo de cooperação. Uma importante característica dessa nova arquitetura é que o nível cognitivo pode despender mais tempo com planejamento, estabelecimento de novas metas, etc, uma vez que o nível reativo assim como, em algumas situações, o nível instintivo são responsáveis pela interação com o ambiente respeitando os requisitos de tempo real.

3.3 UFSC-Team F-180

Em setembro de 2000, foram adquirdos três robôs SoccerBot Plus da Universidade do Oeste da Autrália dando início aos trabalhos com os robôs de pequeno porte. O objetivo é transferir a tecnologia já adquida com os robôs simulados para a categoria de robôs de pequeno porte. Em primeira instância estão sendo desenvolvidos trabalhos para portar o processo Interface, capacitando-o a ser executado pelo processador do SoccerBot Plus. A médio prazo, tem-se como objetivo portar todo o Agente Autônomo Concorrente para rodar embarcado no Soccerbot Plus. Serão desenvolvidos ainda trabalhos na área de visão computacional, para otmizar o reconhecimento dos objetos capturados pela câmera; na área de planejamento e controle de trájetoria de veículos autônomos, visando um melhor sistema de navegação; na área de Sistemas de Tempo Real; etc.

O robô SoccerBot III Plus foi desenvolvido pela Universidade do Oeste da Austrália. Trata-se de um robô autônomo concebido segundo as limitações da categoria F-180 da RoboCup, com sistema de visão embarcado e dotado de um potencial computacional suficiente para executar autônomamente algoritmos simples de processamento de imagem, atuar no ambiente em tempo real e comunicar-se com outros robôs, possibilitanto sua utilização em projetos de pesquisa e em atividades de ensino envolvendo robôs autônomos e grupos de robôs para realização de tarefas coletivas. As características do SoccerBot são brevemente apresentadas nesta seção.

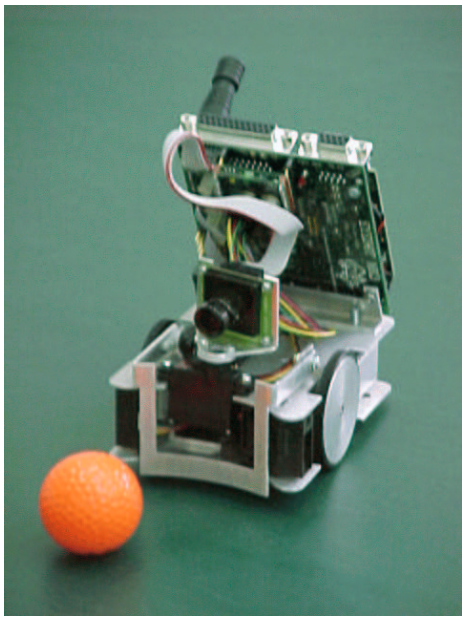

Figura 3.8: Robô Soccerbot III Plus.

- **Dimensões :** O robô SoccerBot, possui as seguintes dimensões: 125x90x150 mm. Projetado para atender as especificadas da categoria F180 da RoboCup e ao mesmo tempo atender a requisito de veículos autônomos.
- **Velocidade :**
- **Autonomia :** 30 minutos.
- **Processsador e Memória :** O processador é Motorola 68332 35MHz 1 MB de RAM 512 de ROM.
- **Modularidade :** A arquitetura SoccerBot apresenta baixa modularidade. Concentrando no módulo base, acionamento dos motores, CPU, entretanto permite adicionar sensores e atuadores a este módulo base. Este módulo inclui três sensores de distância infra-vermelho.Possui ainda um display gráfico LCD.
- **Rádio Enlace :** Velocidade 9.6 Kb/s, frequência 433 MHz, protocolo de tolerancia a falta, configuração automática da rede.
- **Sistema de Visão Embarcado:** Possui uma câmera digital 60x80 pixel com 24 cores. Aquisição, processamento e monitoramento é feito embarcado evitando assim a necessidade de transmitir imagens via rádio enlace.
- **Sensores Adicionais :** Sensores para medição de distância Infra-vermelho.
- **Mecanismo de Chute :** Possui mecanismo de chute frontal.
- **Software para Desenvolvimento :** GNU C, m68k assembly, possui ferramenta para programar estruturas de controle para o robô.

3.4 O Projeto UFSC-Team

O projeto UFSC-Team teve seu início em 1997, em 1998 apresentou na RoboCup'98, o UFSC-Team–98, que trazia uma aquitetura de agente cognitivo onde percepção, ação e comunicação eram implementados segundo uma abordagem concorrente [CB98b]. Essa arquitetura evoluiu para o chamado *Agente Autônomo Concorrente* [CB99], onde o processo decisório foi descentralizado em três níveis: reativo, instintivo e cognitivo. Uma estratégia de cooperação para sistemas multiagentes cognitívos foi tanbém desenvolvida no âmbito desse, projeto chamada *Conhecimento Social Dinâmico* [CB98a, CB00, CB02, Cos01], além da dissertação de mestrado *Otimização de controladores nebulosos e sistemas especialistas reativos utilizando algoritmos genéticos*

[Gon01, GCB01]. Existem atualmente em andamento mais duas dessertações de mestrado e duas teses de doutorado. Uma biblioteca, utilizando a linguagem de programação C++, para auxilar a implementação de sistemas multiagentes cognitívos vem sendo desenvolvida também no âmbito do projeto UFSC-Team. Essa biblioteca é utilizada na implementação dos agentes para o UFSC-Team para a categoria dos robôs simulados. Algumns estudos vem sendo realizados para utilização dessa mesma biblioteca para implementa os agentes para o time de robôs de pequeno porte UFSC-Team F-180.

Capítulo 4

Conclusões e Futuros Trabalhos

O futebol de robôs no formato proposto pela RoboCup Federation, onde os robôs apresentam um alto grau de autonomia, consiste em um ambiente dinâmico onde robôs autônomos precisam interagir para realização de uma tarefa coletiva.

A implementação de um time de futebol de robôs tanto para a categoria de robôs simulados, como para as categorias de robôs reais, requer a integração de diversas tecnólogias inerentes a automação e a informática industrial. Encontram-se presentes neste desafio, diversos problemas que vão desde o controle de servo-mecanismos, reconhecimento de padrões, escalonamento de tarefas em tempo real, até a utilização de técnicas da inteligência artificial para navegação autônoma dos robôs e para fazer estes robôs autônomos interagir de forma cooperativa e apresentando um comportamento socialmente inteligente.

Mais que uma competição mundial a RoboCup se firmou como um fórum para as discussões das soluções propostas pelos pesquisadores. Soluções estas que são disponibilizadas em forma de artigos científicos, tutoriais, ferramentas e bibliotecas de software de domínio público.

Importante também ressaltar o aspecto motivacional gerado pelo tema proposto junto aos aluno de graduação e pós-graduação nas divesas universidades onde grupos de pesquisa adotaram o futebol de robôs e deste problema são identificados desafios que são transformados em teses de doutorado, dissertações de mestrado ou ainda em trabalhos de iniciação científica. Também existem experiências bem sucedidas onde o futebol de robôs foi utilizado como tema para a disciplina de graduação *Programação em Inteligência Artificial* na Univesidade de Linköping (Suécia), de e na disciplina de *Sistemas Autônomos* no MIT (EUA).

O projeto de pesquisa entitulado “AxeBot: Desenvolvimento de time de futebol de robôs para pesquisa e ensino em automação e informática industrial", foi consebido visando a implementação de um times de futebol de robôs. Estes times de futebol de robôs serão utilazados como laboratório, para pesquisa e ensino de diversos tópicos da automação e informática industrial a exemplo de robótica móvel, sistemas de tempo real, sistemas embarcados, agentes autônomos, sistemas multiagentes, dentre outros. Atualmente encontra-se em fase de desenvolvimento o protópo de robô de pequeno porte, completamente autônomo, para a categoria F-180 da RoboCup Federation.

Bibliografia

[All94] B.P. Allen. Case-based reasoning: Business applications. *Communications of ACM*, 37(3):40–42, March 1994.

[AS83] G. R. Andrews and F.B. Schneider. Concepts and notations for concurrent programming. *Computing Surveys*, 15(1), March 1983.

[Bar95] J.M. Barreto. *Redes Neurais Fundamentos e Aplicações.* Minicurso do II Simpósio Brasileiro de Automação Inteligente, CEFET, Curitiba, PR, 13 a 15 de setembro, 1995.

[Bar97] J.M. Barreto. *Introdução as redes neurais artificiais.* V Escola Regional de Informática da SBC Regional Sul, 1997.

[BC97] G. Bittencourt and A. L. da Costa. Expert-coop: An environment for cognitive multi-agent systems. *in pre-printers IFAC/IFIP MCPL'97, Conference on Management and Control of Production and Logistics*, 2:492–497, October 1997.

[BF81] A. Barr and E.A. Feigenbaum, editors. *The Handbook of Artificial Intelligence*, volume I-II. William Kaufmann Inc., Los Altos, California, 1981.

[Bit95] G. Bittencourt. Um ambiente para ensino e desenvolvimento de sistemas especialistas. In *III Workshop sobre Educação em Informática/IV Congresso Íbero-Americano de Educação Superior em Computação*, 1995. 29 de julho a 4 de agosto, Canela, RS.

[Bit96] G. Bittencourt. *Inteligência Artificial - Ferramentas e Teorias.* Instituto de Computação, UNICAMP, Brasil, 1996.

[Bit97] G. Bittencourt. In the quest of the missing link. In *Proceedings of IJCAI 15, Nagoya, Japan, August 23-29*, pages 310–315. Morgan Kaufmann (ISBN 1-55860-480-4), 1997.

[CB97] A. L. da Costa and G. Bittencourt. Parla: A cooperation language for cognitive multi-agent systems. *EPIA'97, 8th Portuguese Conference of Artificial Inteligence*, 1323:207–215, October 1997. Spring-Verlag, Lecture Notes in Artificial Inteligence.

[CB98a] A. L. da Costa and G. Bittencourt. Dynamic social knowledge: A cooperation strategie for cognitive multi-agent systems. *Third International Conference on Multi-Agent Systems (ICMAS'98)*, pages 415–416, Paris, France, July 2-7 1998. IEEE Computer Society.

[CB98b] A. L. da Costa and G. Bittencourt. Ufsc-team: A cognitive multi-agent approach to the robocup'98 simulator league. *RoboCup ´98 Workshop - Team description*, July 1998.

[CB99] A. L. da Costa and G. Bittencourt. From a concurrent architecture to a concurrent autonomous agents architecture. *IJCAI'99, Third International Workshop in RoboCup*, pages 85–90, Sweden, Stockholm, July 31 - August 1999. IJCAI Press.

[CB00] A. L. da Costa and G. Bittencourt. Dynamic social knowledge: A comparative evaluation. *Intenational Join Conference IBREAMIA'2000 / SBIA'2000.*, pages 176–185, Brazil, Atibaia - SP, November 19 - 22 2000. Spring-Verlag, Lecture Notes in Artificial Inteligence, vol. 1952 - Best Paper Track Award.

[CB02] A. L. da Costa and G. Bittencourt. Dynamic social knowledge: The timming evaluation. *XVI Brazilian Symposium on Artificial Intelligence - SBIA'02*, page to appear, Brazil, Porto de Galinhas/Recife - Brazil, November, 11 - 14 2002 2002. Spring-Verlag, Lecture Notes in Artificial Inteligence, vol.

[CL87] P.R. Cohen and H.J. Levesque. Intention = choice + commitment. In *Proceedings of AAAI-87, Seattle*, pages 410–415, 1987.

[CM85] E. Charniak and D. McDermott. *Introduction to Artificial Intelligence.* Addison-Wesley Publishing Company, Reading, MA, 1985.

[Cos97] A. L. da Costa. Expert-coop: Um ambiente para desenvolvimento de sistemas multi-agentes cognitivos. Master's thesis, Universidade Federal de Santa Catarina / Programa de Pós-Graduação em Engernharia Elétrica, Florianópolis, Brasil, Junho 1997.

[Cos01] A. L. da Costa. *Conhecimento Social Dinamico: Uma Estratégia de cooperação para Sistemas Multiagentes Cognitívos.* PhD thesis, Universidade Federal de Santa Catarina / Programa de Pós-Graduação em Engernharia Elétrica, Florianópolis, Brasil, Setembro 2001.

[Dur91] E.H. Durfee. The distributed artificial intelligence melting pot. *IEEE Transactions on Systems, Man and Cybernetics*, 21(6):1301–1306, November 1991. Special Issue on Distributed Artificial Intelligence.

[GCB01] Eder M. N. Gonçalves, A. L. Costa, and G. Bittencourt. Otimização de controladores nebulosos e sistemas especialistas reativos utilizando algoritmos genéticos. In *XXI Congresso da Sociedade Brasileira de Computação*, August 2001.

[Gon01] Eder M. N. Gonçalves. Otimização de controladores nebulosos e sistemas especialistas reativos utilizando algoritmos genéticos. Master's thesis, UFSC - Universidade Federal de Santa Catarina, Fevereiro 2001.

[HR85] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321, July 1985.

[Kit97] H. Kitano. Robocup: The robot world cup initiative. *in Proc. of The First International Conference on Autonomous Agent (Agents-97))*, 1997. Marina del Ray, The ACM Press.

[KTS+97] H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The robocup synthetic agent challenge, 97. *International Joint Conference on Artificial Intelligence (IJCAI97)*, 1997. Nagoya, Japan.

[Mae94] P. Maes. *Designing Autonomous Agents.* MIT Press, Elsevier Science Publishers B. V., 1994.

[MH69] J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In D. Michie and B. Meltzer, editors, *Machine Intelligence 4*, pages 463–502. Edimburgh University Press, Edimbourgh, GB, 1969.

[New80] A. Newell. Physical symbol systems. *Cognitive Science*, 4:135–183, 1980.

[Pos43] E. Post. Formal reductions of the general combinatorial problem. *American Journal of Mathematics*, 65:197–268, 1943.

[Sic96] J.S. Sichman. A model for the decision phase of autonomous belief revesion in open multi-agent system. *Journal of Brazilian Computer Society*, 3(1):40–50, March 1996. ISSN 0104-6500.