UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Conhecimento Social Dinâmico: Uma Estratégia de Cooperação para Sistemas Multiagentes Cognitivos

Tese submetida à Universidade Federal de Santa Catarina
como requisito parcial à obtenção do grau de

Doutor em Engenharia Elétrica

por

Augusto Cesar Pinto Loureiro da Costa

Conhecimento Social Dinâmico:Uma Estratégia de Cooperação para Sistemas Multiagentes Cognitívos.

Augusto Cesar Pinto Loureiro da Costa

Esta tese foi julgada adequada para a obtenção do título de **Doutor em Engenharia** na especialidade **Engenharia Elétrica**, área de concentração **Sistemas de Informação**, e aprovada em sua forma final pelo curso de Pós-Graduação.

Florianópolis, 21 de Setembro de 2001.

Prof. Dr. Guilherme Bittencout, orientador

Prof. Dr. Edson Roberto de Piere
coordenador do curso de Pós-Graduação em Engenharia Elétrica
da Universidade Federal de Santa Catarina.

Banca Examinadora

Prof. Dr. Guilherme Bittencourt, orientador

Prof. Dr. Jaime Sichman

Prof. Dr. Luís Otávio Álvares

Prof. Dr. Marcelo Ricardo Stemmer

Prof. Dr. Ricardo José Rabelo

àquele que sempre esteve perto,

mesmo quando estive longe,

e que sempre apoiou,

mesmo nas decisões mais radicais que já tomrei ...

meu pai.

Agradecimentos

Agradeço primeiramente ao prof. Guilherme Bittencourt pela oportunidade, pela confiança depositada, por todo o trabalho de orientação, pela oportunidade de participar no projeto de cooperação CAPES-DAAD Brasil Alemanha (Probral), e principalmente pela amizade cultivada durante este tempo.

Ao prof. Jacques Calmet pela atenção dispensada, pelas sugestões, e por todo apoio disponibilizado no Institüt für Algorithm und Kognitive Sistem da Univesität Karlsruhe, na Alemanha, onde parte desse trabalho foi realizado no âmbito do projeto CAPES-DAAD PROBRAL.

Aos membros da banca examinadora, que contribuiram opinando e sugerindo, especialmente ao prof. Jaime Sichman que acompanhou boa parte deste trabalho.

À profa. Sandra Sandri pelas contribuições agregadas ao trabalho e pela atenção dispensada.

Ao prof. Edson de Piere, Coordenador do programa de pós-graduação por toda atenção dispensada.

Aos prof. Luis Edmundo Prado de Campos, Cid Gesteira e Caiuby Alves da Costa, pelo incentivo dado no início dessa jornada.

Agradeço, especialmente a Luciano Rottava, Eder Mathes Gonçalves Nunes, Priscila Martins, Sergio Hermesmayer Jr., Alexandre Loçato, Tiago Villaça, com quem tive oportunidade de trabalhar no âmbito do Projeto UFSC-team.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnologico, CNPq, que financiou este trabalho através da concessão de bolsa de doutorado.

A Peter e Martina Kullmann, Otília Werner-Kytoelae, Werner Sailer, Ursulla Dietrich, Christhoph e Brigitte Thomalla e James Hunt, por toda a atenção dispensada no periodo em que morei em Karlsruhe na Alemanha.

A Leoni Schneider, a quem chamo carinhosamente de "minha Frau", que conheci no último ano do doutorado e que compartilhou comigo pacientemente todas as emoções desse último de doutorado.

Agradecimentos especiais a Isabel Tonin e Frederico Freitas, por tudo!

Agradeço aos amigos que fizeram parte desta jornada, em especial a Cristiane Paim, Ruben Cesar Macedo, Alexandre Moreira de Moraes, Christian Koliver, Carlos Brandão, Jerusa Marchi, João Moraes Neto, Sergio Melo, Max Mauro Dias, Lau Lung, Frank Siquera, Michele Wangham e

Luis Magno, José Eduardo Góes, Ângelo Roncale e Carminha, Conrado e Silene Seibel, Udo Fritzke e Carlos Montez. Agradeço também a Mônica Midlej Cardoso, com quem convivi durante boa parte desse trabalho.

Agradeço ainda a meu pai, meus irmãos e meus tios que mesmo tão distante sempre se fizeram tão perto.

Resumo da Tese apresentada a UFSC como parte dos requisitos necessários a Obtenção do grau de Doutor em Engernharia Elétrica.

Conhecimento Social Dinamico: Uma Estratégia de cooperação para Sistemas Multiagentes Cognitívos

Augusto Cesar Pinto Loureiro da Costa

setembro de 2001

Orientador: prof. Guilherme Bittencourt, Dr.
Área de Concentração: Sistemas de Informação.
Palavras Chaves: Sistemas Multiagentes, Inteligência Aritificial Distribuída, Estratgias de cooperação.
Número de Páginas:112

Resumo : Os agentes autônomos possuem um alto grau de determinação, eles podem decidir por motivações próprias, quando e sob que condições suas ações devem ser tomadas. Em muitos casos estes agentes precisam interagir com outros agentes autônomos para atingir seus objetivos, por não possuirem habilidades ou recursos suficientes para solucionar seus problemas sozinhos ou ainda pela interdependencia com outros agentes. Os objetivos destas interações são para fazer outros agentes assunirem um determinado sentido em suas ações (como por exemplo executar um serviço em particular), modificar uma linha de ação planejada, ou ainda atingir uma acordo sobre ações conjuntas. Uma vez que estes agentes não possuem um controle direto sobre os outros, faz-se necessario utilizar uma estratégia de cooperação para aglutinar outros agentes autônomos na realização de ação cooperativa. Diversos pesquisadores têm proposto estratégias de cooperação para sistemas multiagentes nos mais diversos cenários. Para citar algumas dessas estratégias de cooperação já propostas, temos o conhecido *The Contract Net Protocol (CNP)*, O Protocolo Hierárquico, Coalisão Baseada em Dependência, etc. Uma nova estratégia de cooperação para sistemas multiagentes cognitivos com restrições de tempo real, chamada Conhecimento Social Dinâmico é apresentada nesta tese. Esta estratégia de cooperação utiliza o conceito de contratos, similar ao *(CNP)* e adiciona novos conceitos como Estrutura de Contratos, Conjunto de Planos e Base de Conhecimento Social Dinâmica. Esses novos conceitos modificam o papel dos contratos que no CNP são utilizados apenas para alocação de tarefas e passam a assumir também a função de um instrumento para aquisição de conhecimento da comunidade. Além disso, introduz a idéia de construir dinamicamente uma base de conhecimento social direcionada para as metas que os agentes pretendem alcançar.

Thesis abstract presented to UFSC as partial requirements for degree of Doctor in Electrical Engineering

Dynamic Social Knowledge: A Cooperation Strategie for Cognitive Mult-Agent Systems

Augusto Cesar Pinto Loureiro da Costa

September, 2001

Advisor: prof. Guilherme Bittencourt, Dr.
Concentration Area: Sistemas de Informação.
Keywords: Multi-Agent Systems, Distributed Artificial Inteligence, Team-Working, Agent Cooperation Strategie.
Number of Pages :112

Abstract : Autonomous agents have a high degree of self determination, they can decide for theyselves based, when and what under conditions theyer action should be performed. There are many cases where these agents have to interact with another agents to achieve common goals, to performe an action that the need skills are not availeble in that agent or when there is an interdependence amoung the agents.This interaction is done in the sense to convince the involved agent to take a way in you actions, to modify a set of planed actions, or to achieve an agreement about join actions. Once that this agent do not have a direct control unther the others, it is necessary to use a cooperation strategy join others autonomous agent to performe an given cooperative action. Cooperations strategies have been proposed to support cooperation into multi-agent systems , most of them concerned with a unique method of cooperation and the possible negotiation steps within that method. The Contract Net Protocol, desiged to task allocation in Distributed Problem Solving, The Hierarchical Protocol aimed at coordination and large scale of Negociation Strategies to join agents into a coalision like The service-oriented negotiation model, or Time-dependent negotiation, Resource-dependent strategy, etc. The proposed cooperation strategy Dynamic Social Knowledge join some features from Contract Net Protocol, and introduces some new concept like Contract Frame, Plan Set and a social knowledge base built dynamically and narrowed to the cooperation process. It also and make intensive use of rules based system. The most important contributions of this strategy should appear in Open Autonomous Cognitive with real time restrictions that accept the best effort approach. In this kind of agent community, even the number of agents able to cooperate changes dynamically and the environment presents dynamical features.

Conteúdo

Os perigosos entre os subversivos - Podemos dividir os que pretendem uma subversão da sociedade entre aqueles que desejam alcançar algo para si e aqueles que desejam para seus filhos e netos. Esses últimos são os mais perigosos; porque têm a fé e a boa consciência do desinteresse. Os demais podem ser contentados com um osso: a sociedade dominante é rica e inteligente o bastante para isso. O perigo começa quando quando os objetivos se tornam impessoais; os revolucionários movidos por interesse impessoal podem considerar todos os defensores da ordem vigente como pessoalmente interessados, sentindo-se então superior a eles.

Friedrich Nietzsche

Humano, demasiado humano: um livro para espíritos livres.

Capítulo 1

Introdução

Historicamente, os primeiros trabalhos de pesquisa em Inteligência Artificial Distribuída *(IAD)* [Dur91] foram relacionados à Resolução Distribuída de Problemas *(RDP)* [DLC89]. A incorporação de alguns resultados práticos e técnicas provenientes das ciências sociais, da psicologia e da etologia [1], deram origem aos Sistemas Multiagentes *(SMA)* [Jen93]. O panorama resumido nos parágrafos seguintes baseia-se em [Sl97].

As primeiras tentativas de solucionar problemas cooperativamente surgiram nos anos setenta [FG91]. Uma delas foi o projeto HEARSAY-II [Erm80], um sistema para entendimento de diálogos que introduziu *O Modelo Quadro Negro* (do inglês *Blackboard Model* [HR85]). Na mesma época alguns modelos computacionais, como por exemplo os atores de Agha e Hewitt [AH88], foram desenvolvidos visando tratar problemas como compartilhamento de recursos, controles complexos e o aparecimento de comportamentos complexos provenientes de interações bastante simples.

Durante os anos oitenta, a importância de representar explicitamente no agente, um

[1] Ciência que estuda o comportamento dos animais.

conhecimento sobre os demais agentes da sociedade, no contexto do planejamento, foi levantada por Konolige e Nilson [KN80]. A utilização de modelos organizacionais humanos no trabalho cooperativo de agentes computacionais foi discutido por Fox [Fox81], e o *Contract Net Protocol*, um protocolo de cooperação baseado em uma estrutura de mercado, foi desenvolvido por Smith [Smi80].

O conceito de Sistemas Abertos [Hew91], introduzido por Hewitt, foi crucial para as pesquisas em SMA. Finalmente alguns dos primeiros ambientes para implementação e teste de algumas dessas idéias foram desenvolvidos, por exemplo o DVMT [CL83], o DPSK [Car87] e o MACE [Gas87].

Os Sistemas Multiagentes (SMA) baseiam-se na idéia de Comunidade Inteligente [Bit98], ou seja, um sistema cuja inteligência emerge do comportamento social da comunidade. Essa comunidade é formada por agentes autônomos, inseridos em um ambiente comum, capazes de cooperar para alcançar uma meta global. Os membros de uma comunidade inteligente podem ser de baixa complexidade computacional, Agentes Reativos [Bro86], ou extremamente complexos, Agentes Cognitivos ou Inteligentes. Os trabalhos de pesquisa realizados nessa área se concentram basicamente nas propriedades dos agentes, mais precisamente naquelas propriedades que os levam a cooperar com os demais membros da comunidade.

A principal propriedade de uma comunidade de agentes que a faz cooperar para realização de metas globais é a estratégia de cooperação adotada. Muito mais do que prover mecanismos de comunicação, uma estratégia de cooperação para agentes cognitivos provê uma estrutura para projetar padrões de dependência contextual do diálogo, para relacionar as mensagens aos seus respectivos contextos, habilitando os agentes envolvidos na comunicação a manter a compreensão do contexto e de como o diálogo deve ser conduzido [Had96]. Uma vez iniciado o diálogo por um dos agentes da comunidade, utilizando uma determinada estratégia de cooperação, os demais agentes envolvidos no processo de cooperação devem possuir uma cópia da estratégia de cooperação em questão, permitindo a tais agentes responder ao diálogo iniciado com padrões admitidos pela estratégia de cooperação. Conseqüentemente para que haja a cooperação, faz-se necessário que os agentes envolvidos nesse processo utilizem a mesma estratégia de cooperação.

As estratégias de cooperação utilizadas atualmente baseiam-se no Contract Net Protocol *(CNP)* ou em Coalisões Baseadas em Dependência *(CBD)* [Sic98] [IS99]. O CNP consiste em uma espécie de licitação, aberta para alocar uma dada tarefa a um dos agentes da comunidade. Nas estratégias do tipo em CBD, cada um dos agentes envolvidos no processo de cooperação verbaliza suas intenções e contradições e alternam-se propostas até que um acordo seja alcançado.

Uma nova estratégia de cooperação para sistemas multiagentes cognitivos com restrições de tempo real, chamada Conhecimento Social Dinâmico *(CSD)* [CB00] é apresentada nesta tese. Inicialmente no capítulo 2, um problema que vem sendo estudado por uma comunidade internacional de pesquisadores em sistemas multiagentes e robótica inteligente, o futebol de robôs proposto pela RoboCup Federation, é apresentado, focalizando uma das situações em que a utilização de uma estratégia de cooperação representa um ganho significativo. No capitulo 3 é apresentado o UFSC-Team, sistema multiagente para o problema proposto. No capítulo 4, são apresentados os sistemas multiagentes, mantendo a atenção nos sistemas multiagentes cognitivos. No Capítulo 5 a cooperação entre agentes cognitivos é abordada enfocando as estratégias de cooperação: o Contract Net Protocol e a Coalisão Baseada em Dependência. A nova estretégia de cooperação proposta, o Conhecimento Social Dinâmico [CB98a] é apresentada no capítulo 6. No capítulo 7 é apresentada a implementação da biblioteca *Expert-Coop++*, dos sistemas multiagentes para o problema proposto CNP_Team, CBD_Team e CSD_Team e os resultados. Finalmente no capítulo 8 são apresentadas as conclusões desta tese e as perspectivas.

Parte I

Futebol de Robôs

Capítulo 2

O Futebol de Robôs

2.1 Introdução

Uma partida de futebol entre robôs autônomos, este foi o desafio lançado em 1996, por um grupo internacional de pesquisadores em Inteligência Artificial e Robótica Inteligente. Dessa iniciativa surgiram, em 1997 a RoboCup (Robot World Cup), uma competição realizada entre os diversos times de futebol de robôs desenvolvidos por pesquisadores de diversos países, e a RoboCup Federation, uma entidade internacional responsável pela realização dessas competições e pela manutenção de um fórum de discussão sobre os desafios e avanços científicos envolvidos no problema proposto.

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, estabelecer 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 objetivos coletivos, *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.

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* 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).

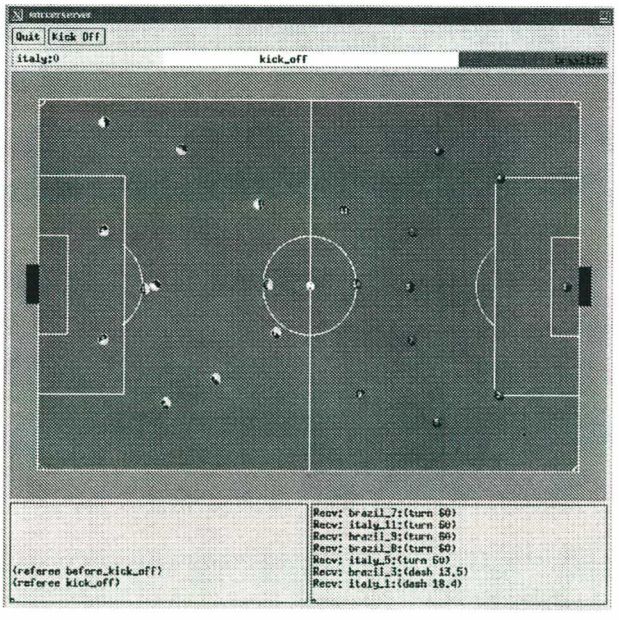

Figura 2.1: Soccer-Sever: Simulador utilizado na RoboCup

O servidor de conecções udp/socket server (figura 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:

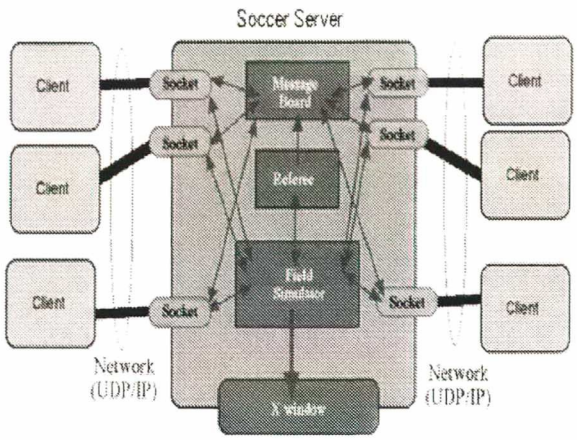

Figura 2.2: Soccer-Sever: servidor de conecções udp/socket

- 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 server 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:

$$p_{rx} = p_{xt} - p_{xo} \quad (2.1)$$

$$p_{ry} = p_{yt} - p_{yo} \quad (2.2)$$

$$v_{rx} = v_{xt} - v_{xo} \quad (2.3)$$

$$v_{ry} = v_{yt} - v_{yo} \quad (2.4)$$

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

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

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

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

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

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

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.

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.

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

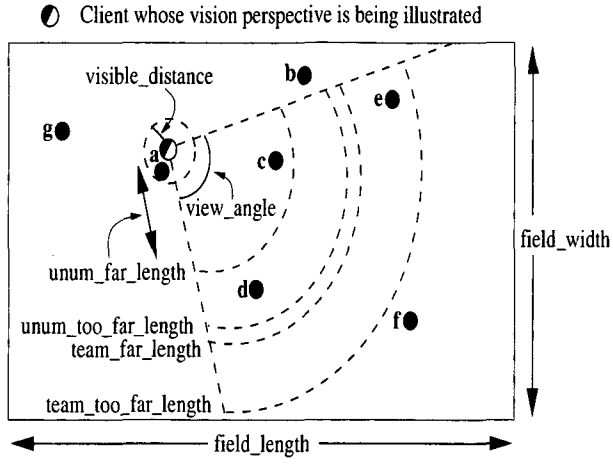

Figura 2.3: Campo visual do jogador

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.

$$\text{view\_angle} = \text{visible\_angle} * \text{view\_width\_factor} \tag{2.12}$$

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 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).

2.8 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).

2.9 Temporização

A comunicação entre o *Soccer Server* e os agentes envolve mensagens síncronas e assíncronas (figura 2.4).

- **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).

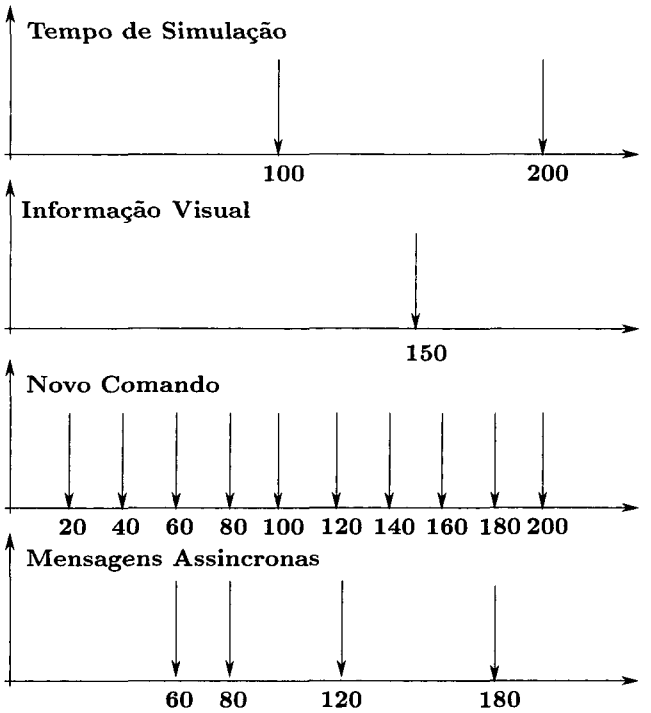

Figura 2.4: Temporização do Soccer Server

- **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.

2.10 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 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.

2.11 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.

2.12 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.

2.13 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 observar 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.

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).

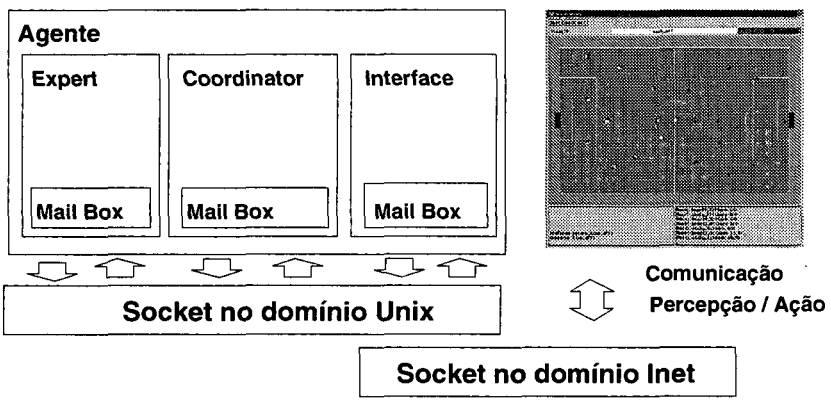

Figura 3.1: Arquitetura do agente utilizado pelo UFSC-Team'98.

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.

3.2 Agente Autônomo Concorrente

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] (ver figura 3.2). 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. Cada um dos 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.

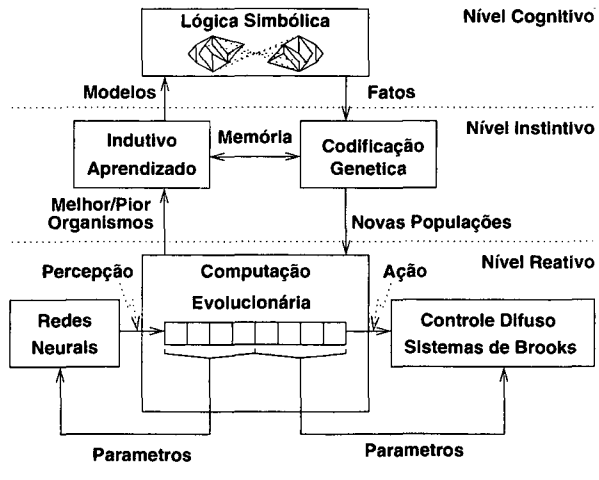

Figura 3.2: Modelo genérico para agentes cognitivos.

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 do 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 à 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 especifica uma nova meta, 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).

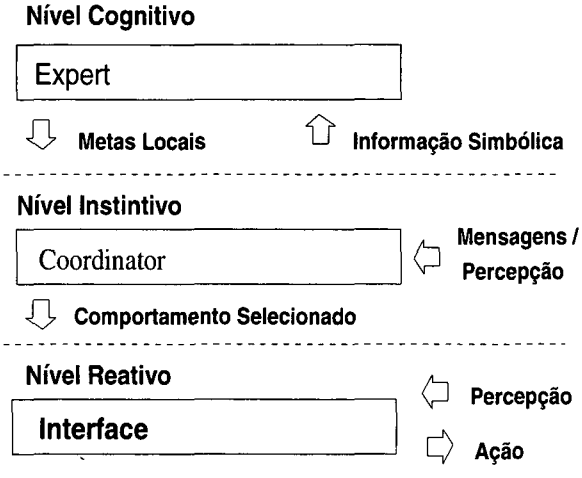

Figura 3.3: Fluxo de Informação no Agente Autônomo Concorrente.

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 (informação auditiva) 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 [JS97]. 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.

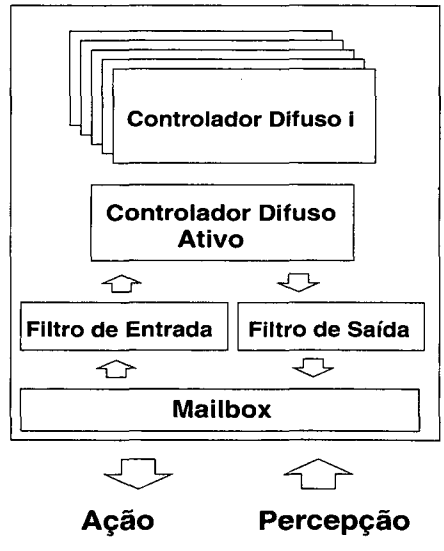

Figura 3.4: Nível Reativo: o Processo Interface.

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 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 enviadas pelo *Soccer Server*, a cada novo ciclo de simulação, os valores destas variáveis lingüísticas.

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 otimizar os conjuntos difusos.

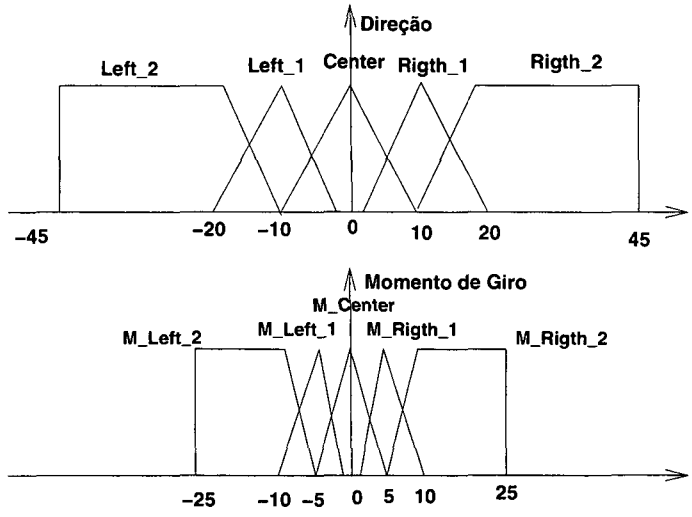

Figura 3.5: Conjuntos difusos utilizados para a implementação para a saída *turn_ moment* e a respectiva variável lingüística *direção_ da_ bola*

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_{150} e t_{300}.

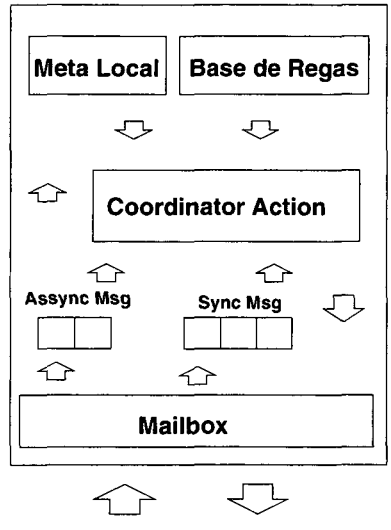

Figura 3.6: Nível Instintivo: o processo Coordinator.

A percepção visual 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 comportamento 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_Posse_da_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).

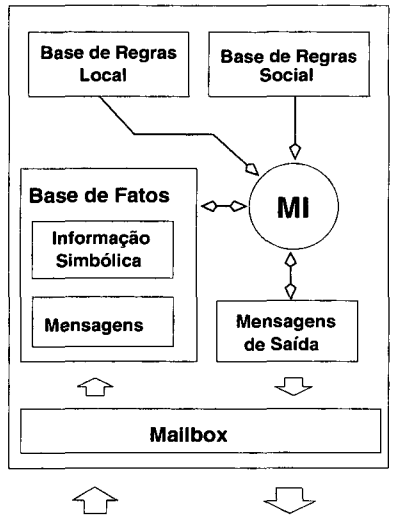

Figura 3.7: Nível Cognitivo: o processo Expert.

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.

Parte II

Sistemas Multiagentes

Capítulo 4

Sistemas Multiagentes

4.1 Introdução

Atualmente não existe uma definição para agente aceita por toda a comunidade de IAD. Uma definição genérica foi proposta por Ferber [FG91]:

"Um agente é uma entidade real, ou virtual, imersa em um dado ambiente onde ela pode tomar algumas ações, estar habilitada para perceber e representar parcialmente este ambiente, podendo ainda comunicar-se com os demais agentes do ambiente. Este agente apresenta um comportamento autônomo que é uma conseqüência de suas observações, do conhecimento armazenado e das interações com os demais agentes do ambiente".

Os SMA baseiam-se na idéia de Comunidade Inteligente [Bit98], ou seja, um sistema cuja inteligência emerge do comportamento social da comunidade. Essa comunidade é formada por agentes autônomos, inseridos em um ambiente comum, capazes de cooperar para alcançarem uma meta global. Os membros de uma comunidade inteligente podem ser de baixa complexidade computacional, Agentes Reativos, ou extremamente complexos, Agentes Cognitivos ou Inteligentes. Os trabalhos de pesquisa realizados nessa área concentram-se basicamente nas propriedades dos agentes, mais precisamente naquelas propriedades que os levam a cooperar com os demais membros da comunidade.

4.2 Agentes Reativos

Os Agentes Reativos são baseados em modelos de organização biológica ou etológica, como por exemplo as colméias, as sociedades de formigas ou de cupins. Embora o comportamento de uma formiga isoladamente não possa ser considerado inteligente, o formigueiro é um exemplo claro de uma comunidade inteligente, uma vez que existe uma atividade coletiva de busca de alimentos e uma posterior estocagem, a reprodução é organizada apresentando berçários, enfermeiras etc.

O modelo de funcionamento de um Agente Reativo é o de Estímulo-Resposta. Geralmente esses agentes não apresentam uma memória das ações tomadas e também não planejam suas ações futuras. Todo o conhecimento a respeito das ações e do comportamento dos demais membros da sociedade é percebido através das modificações sofridas pelo ambiente. Esses agentes não possuem um modelo de comunicação de alto nível, nem uma representação explícita do ambiente ou dos membros da sociedade.

As sociedades de agentes reativos, em geral, são numerosas, podendo apresentar milhares de membros em uma mesma comunidade. Existem diversos trabalhos na literatura sobre SMA baseados em agentes reativos, por exemplo: a arquitetura de subsunção de Brooks [Bro86], o modelo PACO [Dem93], entre qutros.

4.3 Agentes Cognitivos

Os agentes cognitivos são baseados nos modelos de organização social, de sociedades humanas: grupos, hierarquias, mercados etc. Esses Agentes Cognitivos possuem uma representação explícita do ambiente e dos membros da comunidade e podem raciocinar sobre as ações tomadas no passado e planejar as futuras ações. Os Agentes Cognitivos podem ainda interagir com os demais membros da comunidade através de linguagens e protocolos de comunicação complexos, estratégias sofisticadas de negociação etc. Esses agentes normalmente apresentam elevada complexidade computacional e caracterizam-se por apresentar um comportamento inteligente tanto em uma Comunidade de Agentes quanto isoladamente. Estas comunidades geralmente são compostas por um pequeno número de participantes.

Existem diferentes tipos de Sistemas Multiagentes Cognitivos, não mutuamente exclusivos, a maioria deles difere quanto à arquitetura, possibilidades de comunicação e complexidade do agente. Os quatro tipos de arquitetura de Sistemas Multiagentes Cognitivos geralmente utilizados são: Sistemas Multiagentes Federados, Sistemas Multiagentes Democráticos, Sistemas Multiagentes Abertos [SVAB96].

- **Sistemas Multiagentes Federados:** um Sistema Multiagentes Cognitivo é

chamado Federado quando existem agentes especiais, complexos, chamados de Facilitadores, que possuem um conhecimento a respeito das habilidades de outros agentes mais simples. O papel dos Facilitadores é organizar o trabalho entre os agentes mais simples. Uma vez que uma determinada requisição é recebida por um Facilitador, ele se responsabiliza por encontrar um agente mais simples capaz de realizar essa requisição. A principal vantagem dessa abordagem é permitir um gerenciamento eficiente da comunicação entre os agentes. Por outro lado, a centralização de importantes tarefas torna o sistema como um todo sensível ao mau funcionamento do agente facilitador.

- **Sistemas Multiagentes Democráticos:** a principal característica de um Sistema Multiagentes Democrático é o fato de todos os agentes da comunidade possuírem o mesmo nível hierárquico. Nesta abordagem, os agentes realizam ações coletivas assincronamente, e a comunicação é uma dessas ações. Essa comunicação tipicamente obedece às regras de alguma Linguagem de Cooperação adequada, como por exemplo *KQML (Knowledge Query Manipulation Language)* [FLM95], uma linguagem que utiliza primitivas especiais para expressar crenças, necessidades e modalidades de comunicação. Um outro exemplo de Linguagem de Cooperação é a *CooL (Cooperation Language)* [BF94], baseada no *Contract Net Protocol* [Smi80] e que suporta a comunicação de agentes utilizando um conjunto de mensagens chamado Primitivas de Cooperação. As principais vantagens dessa abordagem são a modularidade e a flexibilidade, e a desvantagem é que cada um dos agentes necessita conhecer a identidade e ao menos parcialmente, as habilidades dos demais agentes da comunidade.

- **Sistemas Multiagentes Abertos**: um SMA é dito aberto quando a composição da comunidade não é fixa, permitindo que os agentes possam se subscrever e se desligar da comunidade, dinamicamente. Neste tipo de sistema, alguns serviços podem estar disponíveis ou não, de acordo com a composição da comunidade em um dado momento. Os agentes podem adaptar-se e escolher diferentes objetivos a serem alcançados, planos de execução ou padrões de cooperação, dependendo dos serviços disponíveis na comunidade. A principal vantagem dessa abordagem é a sua robustez. A desvantagem é a excessiva troca de mensagens realizada por um complexo protocolo de comunicação capaz de assegurar o caráter de comunidade aberta.

4.3.1 Comportamento Social

O comportamento dos Agentes Cognitivos pode ser classificado segundo dois critérios: a alocação das tarefas, locais ou globais, e a habilitação para realização das tarefas.

- **alocação das tarefas:** uma tarefa global envolve todos os agentes da comunidade, uma tarefa local envolve apenas um agente;
- **habilitação:** um determinado agente está habilitado a executar uma tarefa se ele possui as aptidões necessárias para a sua realização;

Segundo esses critérios, os Agentes Cognitivos podem apresentar quatro tipos de comportamento quando inseridos em uma sociedade de agentes: coabitação, cooperação, colaboração, distribuição.

- **coabitação:** um agente pode e deve realizar suas tarefas locais. O simples fato de estar imerso em um ambiente comum não o obriga a cooperar com os demais agentes da comunidade;
- **cooperação:** caso um agente esteja impossibilitado de realizar uma tarefa, por não possuir as aptidões necessárias ou por não conseguir realizá-la com a desenvoltura requerida, ele solicita aos demais agentes que cooperem com ele;
- **colaboração:** um dado agente pode ser capaz de executar uma tarefa global sozinho, caso existam mais agentes na comunidade capazes de realizar essa mesma tarefa, um mecanismo eletivo deve ser acionado;
- **distribuição:** algumas tarefas globais precisam ser realizas por um conjunto de ações coletivas, isso ocorre quando nenhum dos agentes pode alcançar o objetivo isoladamente, o que exige a divisão e a alocação das tarefas segundo algum critério.

4.3.2 Descritores Internos e Externos

O conhecimento que um agente cognitivo possui a respeito da suas aptidões e das aptidões dos demais membros da comunidade é crucial para que a comunidade de agentes apresente um comportamento cooperativo / distribuído. Esse conhecimento é armazenado em duas estruturas de representação diferentes: os descritores Internos e Externos.

- **Internos:** contêm as aptidões e os comportamentos do próprio agente. Essa descrição pode ser explícita, possibilitando ao agente conhecer seus procedimentos internos, ou não.

- **Externos:** armazenam as aptidões e os comportamentos dos demais agentes da comunidade. Cada um dos agentes deve ter uma representação externa dos demais.

Basicamente, a informação estática, como por exemplo o conhecimento, e a informação dinâmica, o comprometimento com uma atividade para solucionar um problema, são representados nesses descritores. Um aspecto importante é que dois agentes podem ter o mesmo descritor externo, mesmo que possuam organizações internas distintas. O conteúdo desse descritor externo é um tópico de pesquisa ainda aberto em IAD, estando intimamente relacionado com os protocolos de comunicação utilizados pelos agentes durante os processos de interação.

4.4 Protocolos de Comunicação

Os trabalhos de pesquisa em IAD dão ênfase à importância das ações e interações entre agentes de uma comunidade. A interface dos agentes com o ambiente pode ser dividida em percepção, ação e comunicação.

- **Percepção:** é a capacidade de um agente de observar as transformações sofridas pelo ambiente, como por exemplo enxergar, ouvir, sentir, etc. Isso pode ser obtido em um robô, por exemplo, através de câmeras, microfones, sensores de pressão, etc.

- **Ação:** é a capacidade de um agente em atuar no ambiemte, de modo a transformá-lo, como por exemplo, carregar peça, mover objetos, locomover-se, etc.

- **Comunicação:** é a troca de dados e conhecimentos entre os agentes.

A alta complexidade dessa comunicação leva à definição de protocolos para expressar os conceitos semânticos envolvidos na solução cooperativa de problemas. Uma proposta adotada por vários pesquisadores é a utilização da ação comentada [CD90]. Resumidamente, essa teoria propõe que cada elocução seja vista como um tipo de ação, similar às ações realizadas no ambiente. Realmente, cada ação comentada carrega algum conteúdo proposicional mais uma força de locução, como por exemplo a

solicitação de uma informação, a requisição de um serviço, uma advertência etc. Considere as seguintes frases: *Sincronize o gerador 1.* e *Seria possível sincronizar o gerador 1?*. Ambas possuem o mesmo conteúdo proposicional, mas a primeira frase expressa uma ordem, e a segunda, uma solicitação. Um tópico de pesquisa aberto em IAD é o quão extensiva deve ser a representação desses conceitos em primitivas ou no conteúdo da mensagem propriamente dita.

Uma outra idéia interessante que vem sendo utilizada recentemente nessa área é o conceito de sistemas governados por leis (do inglês *law-governed systems*) [MIN89]. A idéia central é que os protocolos para sistemas distribuídos devem ser "governados", para evitar sua violação. Ou seja, toda comunicação é governada por leis que possuem uma seqüência de regras pré-definidas.

4.5 A Estrutura de Controle de um Agente

As atividades que devem ser realizadas por um agente são:

- percepção do ambiente onde ele está inserido;
- ação no ambiemte onde ele está inserido;
- comunicação com os demais agentes da comunidade;
- planejamento estratégico para realização de tarefas;
- execução de atividades de solução de problemas, cujos resultados podem ser a modificação de seu estado interno, uma ação no ambiente ou uma comunicação com outros agentes.

Uma estrutura de controle precisa estar presente para assegurar um comportamento global coerente. Esse controle pode ser centralizado ou descentralizado. O controle centralizado assemelha-se às organizações humanas altamente hierarquizadas, onde o agente detentor do conhecimento necessário à solução do problema irá dizer a todos os membros da sociedade o que fazer e quando. Por outro lado, no controle descentralizado, os agentes da comunidade interagem para alocar as tarefas de acordo com as habilidades de cada um.

Assim como não há consenso sobre a definição do termo agente, não existe uma estrutura de controle universalmente aceita. Ou melhor, existem duas dimensões diferentes de controle: controle do agente e controle da sociedade. O controle do agente orienta como um agente deve organizar internamente suas atividades. Por outro lado

o controle da sociedade orienta como organizar o conjunto de agentes e como controlar suas interações. Boisser [Boi92] tentou esclarecer esses conceitos quando estava investigando como o controle do agente e o da sociedade poderiam ser divididos hierarquicamente, aplicando-se uma abordagem SMA aos problemas de visão computacional. Esses conceitos tiveram outra tentativa de formalização no trabalho de Levi [Lev90]. Do ponto de vista da Engenharia de Software, Shoham [Sho92] propôs uma estrutura chamada *Agent-Oriented Programming*, definindo os estados internos e os ciclos para um interpretador genérico de agentes. Ishida [Ish92], no âmbito do controle social, propôs um modelo para organizações centralizadas para solução de problemas, no qual são definidas algumas estruturas de controle baseadas na realimentação entre as atividades de solução de problemas e as organizações *auto-design.*

4.6 Linguagem para Comunicação de Agentes

Para atingir objetivos comuns, a interação e entre agentes cognitivos requerem mais do que uma linguagem de comum entendimento entre agentes envolvidos. O comportamento cooperativo em uma comunidade de agentes requer o cumprimento de três requisitos [FLM95]:

1. uma linguagem em comum;
2. um entendimento comum sobre a informação e o conhecimento compartilhados;
3. a habilidade para entender o que esta incluído em (1) e (2).

Em se tratando de comunidades de agentes cognitivos, (2) e (3) referem-se diretamente à base de conhecimento e aos mecanismos para comunicação de conhecimento presentes em cada agente. Nesse caso um formalismo e uma ontologia comuns de representação de conhecimento são determinantes para permitir que esses elementos sejam efetivos.

Os agentes são em sua maioria entidades computacionais residentes em um Nível de Conhecimento [FLM95] e eles não estariam bem servidos pelas linguagens e protocolos desenvolvidos para a computação distribuída. Estas linguagens e protocolos focalizam um processo ao invés de um programa ou uma coleção de programas que constituem um agente. Como resultado, uma linguagem de comunicação deve ser poderosa o suficiente para suportar a comunicação entre programas em um alto nível.

Embora uma linguagem de comunicação não seja um protocolo, a distinção entre ambas é difusa. Um protocolo, como os que são utilizados no contexto das linguagens de comunicação, pode apresentar um dos seguintes significados:

- Um protocolo de transporte como ftp, http, etc.
- Uma estrutura de interação de alto nível, como negociação ou protocolo da teoria de jogos;
- Uma restrição às possíveis trocas válidas de primitivas de comunicação.

Uma linguagem de comunicação pode utilizar protocolos do primeiro tipo como mecanismo de transporte, pode ser usada por protocolos do segundo tipo como uma forma de implementação, e normalmente inclui protocolos do terceiro tipo. Mas definitivamente uma linguagem de comunicação não é meramente um protocolo [FLM95].

Uma Linguagem para Comunicação de Agentes *(LCA)* normalmente consiste em um conjunto de primitivas conhecido por todos os agentes da comunidade e um conjunto de regras de conversação. As primitivas informam o que está sendo compartilhado e o que deve ser feito com esse conhecimento ou informação. As regras de conversação, por sua vez, regulamentam as atitudes adotadas pelo agente durante a comunicação. Uma LCA utiliza mecanismos para comunicação de processos provenientes de sistemas distribuídos. Finin, Labrou e Mayfield sugeriram um conjunto de necessidades para uma LCA em "KQML as an agent communication language" [FLM95]. Essas necessidades são divididas em sete categorias: forma, conteúdo, semântica, implementação, redes de comunicação, ambiente e confiabilidade.

- **Forma** - Uma boa LCA deve ser declarativa, sintaticamente simples e legível por seres humanos. Deve apresentar consistência, ser fácil de analisar e de gerar. Uma linguagem deve ser linear ou facilmente transformável em uma forma linear. Finalmente, deve apresentar uma sintaxe extensível.
- **Conteúdo** - Uma LCA deve ser organizada em camadas visando uma boa adaptação a outros sistemas. Em particular uma distinção deve ser feita entre linguagens de comunicação, que expressam atos comunicativos, e o conteúdo da linguagem, que expressa fatos sobre o domínio. A organização em camadas facilita a integração da linguagem com a aplicação, além de propiciar uma estrutura conceitual para o entendimento da linguagem. A linguagem deve ainda basear-se em um conjunto de atos comunicativos (primitivas).
- **Semântica** - A semântica de uma linguagem de comunicação não deve ser ambígua e deve, se possível, apresentar uma forma canônica, isto é, a similaridade de um significado deve levar à similaridade da representação.

- **Implementação** - A implementação deve ser eficiente, tanto na velocidade quanto no espectro de utilização. Deve proporcionar uma boa integração com as tecnologias de "software" existentes. A interface deve ser de fácil utilização. Finalmente a linguagem deve ser amigável com implementações parciais.

- **Redes de Comunicação** - Uma linguagem para comunicação de agentes deve adaptar-se bem às tecnologias modernas de redes de computadores. As linguagens devem suportar todos os tipos básicos de conexão: *ponto a ponto*, *multicast* e *broadcast*. As conexões síncronas e assíncronas devem ser suportadas. A linguagem deve ser formada por um conjunto de primitivas suficientemente rico para poder servir como um substrato onde linguagens de mais alto nível e protocolos de interação possam ser construídos. Mais ainda, estes protocolos de mais alto nível devem poder ser implementados independentemente do mecanismo de transporte utilizado.

- **Ambiente** - O ambiente onde os agentes inteligentes serão requisitados para trabalhar poderá ser altamente distribuído, heterogêneo e dinâmico. Para propiciar um canal de comunicação nesse ambiente, uma linguagem de comunicação precisa propiciar ferramentas para suportar heterogeneidade e dinamismo. Precisa proporcionar a interoperação com outras linguagens e protocolos. Finalmente deve ser facilmente incorporável aos sistemas herdados.

- **Confiabilidade** - Uma linguagem deve permitir uma comunição entre agente confiável e segura. Precauções em relação à segurança e conversações privadas entre dois agentes devem ser suportadas. A linguagem deve prover um forma de garantir a autenticidade dos agentes. Deve ainda ser robusta o suficiente para suportar mensagens não apropriadas e mal formadas e deve ainda possuir mecanismos para identificação e sinalização de erros e advertências.

Entre as linguagens para comunicação de agentes, as mais conhecidas são a KQML, *Knowledge Query Manipulation Language* [FLM95], que propõe um formato para as mensagens a serem trocadas, e um protocolo para manipulação das mensagens compartilhadas e para interação dos agentes, e a CooL *Cooperation Language* [BF94], uma linguagem baseada na KQML.

4.7 Agentes Autônomos e Sistemas Multiagentes

Os agentes autônomos possuem um alto grau de determinação, eles podem decidir por motivações próprias, quando e sob que condições suas ações devem ser tomadas.

Em muitos casos estes agentes precisam interagir com outros agentes autônomos para atingir seus objetivos, por não possuirem habilidades ou recursos suficientes para solucionar seus problemas sozinhos ou ainda pela interdependência em relação a outros agentes. Os objetivos destas interações são para fazer outros agentes assumirem um determinado sentido de suas ações (como por exemplo executar um serviço em particular), modificar uma linha de ação planejada, ou ainda atingir um acordo sobre ações conjuntas. Uma vez que estes agentes não possuem um controle direto sobre os outros faz-se necessário utilizar uma estratégia de cooperação para aglutinar outros agentes autônomos na realização de uma dada tarefa formando assim um sistema multiagente para solução de problemas através de ação cooperativa.

Capítulo 5

Estratégias de cooperação para Agentes Cognitivos

5.1 Introdução

Um processo de cooperação consiste basicamente em distribuir metas, planos e tarefas em uma comunidade de agentes [JW98]. Para que haja cooperação entre uma comunidade de agentes, estes precisam saber o objetivo desse processo de cooperação e como cooperar eficientemente. Isso é atingido fornecendo-se ao agente uma meta - uma descrição de um estado do ambiente que se deseja alcançar. Baseado em um conjunto de ações que esse agente pode realizar, o agente pode construir uma variedade de planos para alcançar essa meta. O agente se compromete com o melhor plano adicionando ações requisitadas pelo plano para que ocorra uma seqüência de eventos e uma execução do plano de acordo com esse seqüenciamento. A cooperação é atingida não apenas pela condução de ações individuais, mas principalmente pela execução compartilhada dos planos, assim como através de um planejamento conjunto. Para que aconteça esse tipo de cooperação entre agentes cognitivos, faz-se necessário que estes agentes possuam algumas habilidades. Primeiramente, o agente deve ser capaz de modificar o ambiente através de suas ações, de comunicar-se com outros agentes e deve ainda ser capaz de planejar suas ações e conhecer como alcançar suas metas.

A principal propriedade de uma comunidade de agentes que faz com que seus agentes cooperem para a realização de metas globais é a estratégia de cooperação adotada. Muito mais do que prover mecanismos de comunicação, uma estratégia de cooperação para agentes cognitivos provê uma estrutura para projetar padrões de dependência contextual do diálogo, para relacionar as mensagens aos seus respectivos contextos, habilitando os agentes envolvidos na comunicação a manter a compreensão do contexto e de como o diálogo deve ser conduzido [Had96]. Uma vez iniciado o diálogo por um dos agentes da comunidade, utilizando uma determinada estratégia de cooperação, os demais agentes envolvidos no processo de cooperação devem possuir uma cópia da estratégia de cooperação em questão, permitindo a tais agentes responder ao diálogo iniciado com padrões admitidos pela estratégia de cooperação. Conseqüentemente para que haja a cooperação, faz-se necessário que os agentes envolvidos nesse processo utilizem a mesma estratégia de cooperação.

Diversos pesquisadores têm proposto estratégias de cooperação para sistemas multiagentes nos mais diversos cenários. Para citar algumas dessas estratégias de cooperação já propostas, temos o conhecido *Contract Net Protocol (CNP)* [Smi80], O Protocolo Hierárquico [DM90] e a Coalisão Baseada em Dependência [Sic98] [IS00]. Uma outra abordagem destinada à resolução de conflitos entre agentes autônomos, cujas origens apontam para o final dos anos 80 consiste em estratégias de cooperação para agentes autônomos baseadas em processos de negociação. Uma Negociação é o processo pelo qual uma decisão conjunta é realizada por duas (bilateral) ou mais (multilateral) partes. As partes envolvidas primeiramente verbalizam suas demandas contraditórias e então buscam um acordo através de concessões e novas alternativas [Pru81]. Nesta linha de estratégia de cooperação diversos trabalhos de pesquisa têm sido apresentados propondo estratégias de negociação distintas, mas mantendo seu foco na convergência dos processos de negociação. Nesse capítulo as principais estratégias de cooperação para agentes cognitivos são apresentadas.

5.2 Contract Net Protocol

O *Contract Net Protocol (CNP)* [Smi80] é considerado o precursor das estratégias de cooperação para sistemas multiagentes autônomos. Concebido inicialmente como um protocolo de comunicação e controle para fontes de conhecimento geograficamente distribuídas e fracamente acopladas na Solução Distribuída de Problemas [DR94] [DLC89], atingiu grande notoriedade nos anos 80 e foi utilizado com sucesso na implementação de vários sistemas multiagentes. Ainda na década de 80 foi apresentado o Protocolo Hierárquico [DM90], concebido para coordenar a interação de agentes inteligentes, que

a priori não conhecem bem os demais agentes com os quais devem interagir.

O CNP baseia-se na noção de contrato. Um contrato é estabelecido, entre um gerente e contratado, através de um processo de seleção mútua local baseado numa troca de informação bidirecional. Em resumo, os contratados disponíveis avaliam as tarefas anunciadas pelos gerentes e submetem uma proposta para a realização da referida tarefa. Os gerentes avaliam as propostas recebidas para uma dada tarefa, e o agente que depositou a proposta mais apropriada à realização da tarefa em questão é declarado vencedor do contrato. O CNP admite ainda a utilização de rodadas de negociação entre gerentes e contratados em torno da tarefa em questão. O CNP possui um controle distribuído devido ao fato de o processamento e a comunicação não estarem vinculados a um agente especificamente; preferencialmente todos os agentes são contratados em potencial e podem assumir a realização de uma tarefa.

5.2.1 Mensagens e Procedimentos Básicos

No *Contrac Net Protocol*, os agentes envolvidos no processo de cooperação são candidatos a integrar uma rede de contratos e a execução desta tarefa é decidida através do estabelecimento de um contrato entre os agentes. Cada um dos agentes pode assumir um dos seguintes papéis: *gerente* ou *contratado*. O gerente é responsável pelo monitoramento da tarefa e pelo processamento dos resultados. O contratado por sua vez, assume a realização da tarefa. Os agentes não são projetados a priori como gerentes ou contratados. Estes papéis podem ser assumidos dinamicamente por qualquer um dos agentes durante o processo de cooperação.

O CNP possui um conjunto definido de mensagens utilizadas durante o processo de cooperação que serão apresentadas aqui. Serão ainda descritas a informação codificada em cada uma destas mensagens e seu processamento.

- **Task Announcement** - O agente gerente que necessita que uma tarefa seja realizada, inicia o processo de cooperação difundindo para os demais agentes uma mensagem informando a existência de uma tarefa a ser realizada. Essa mensagem pode ser endereçada a todos os agentes *(broadcast)*, a um determinado grupo de agentes *(multicast)* ou a uma agente especificamente *(ponto-a-ponto)*. Uma mensagem do tipo *Task Announcement* possui quatro campos principais: eligibility specification contém os pré-requisitos para que um agente submeta uma proposta de realização da tarefa, *task abstraction* contém uma breve descrição da tarefa, *bid specification* contém um modelo de proposta esperada, *expiration time* contém o prazo final para recebimento de propostas.

- **Processamento da mensagem Task Announcement** - Para cada um dos tipos de tarefas no CNP, os agentes mantêm um lista ordenada das mensagens do tipo *Task Announcement* recebidas e ainda não expiradas. Cada um dos agentes avalia sua elegibilidade para as respectivas tarefas anunciadas. Um vez assegurados os pré-requisitos para uma dada tarefa os agentes estão aptos a formular uma proposta para realização daquela tarefa.

- **Bidding** - A lista ordenada de tarefas anunciadas é processada concorrentemente à execução de tarefas até que o agente candidato a contratado que se encontre processando uma tarefa a finalize e esteja apto a processar uma nova. Nesse momento o agente encontra-se apto a submeter uma proposta *(Bid)* de realização de uma tarefa anunciada. Caso haja mais de uma tarefa na lista ordenada, o agente selecionará a mais recente. Um agente poderá submeter uma proposta para uma tarefa anunciada nas seguinte situações: tendo recebido um novo anúncio de tarefa, ou tendo uma dada tarefa ultrapassado o tempo limite de execução. O campo node abstraction de uma proposta *(Bid)* é preenchido com as habilidades do agente relevantes para a realização da tarefa anunciada.

- **Processamento da mensagem Bidding** - Os contratos são mantidos abertos pelo agente gerente que anunciou a tarefa até que um vencedor tenha sido declarado. O gerente também mantém uma lista ordenada das propostas recebidas para a tarefa em questão. Quando uma proposta é recebida, o gerente avalia e ordena essa proposta em relação às demais propostas já recebidas; caso uma das propostas seja considerada satisfatória para a realização da tarefa o agente que enviou tal proposta é declarado vencedor do contrato. Caso contrário o gerente permanece aguardando outras propostas. Caso o prazo limite para recebimento de propostas tenha sido alcançado, as seguintes atitudes podem ser tomadas: declarar vencedor do contrato o agente cuja proposta seja a mais apropriada à realização da tarefa; difundir um novo *Task Announcement*; aguardar um breve intervalo e transmitir novamente um *Task Announcement*.

- **Award** - Esse tipo de mensagem informa ao agente que ele foi o vencedor de um contrato e se responsabilizará pela execução da referida tarefa.

- **Information** - Normalmente utilizada para troca de informação entre os agentes envolvidos num processo de cooperação.

- **Report** - Utilizada pelo contratado para informar a um gerente que uma dada tarefa sob sua responsabilidade foi executada, por completo ou parcialmente.

- **Termination** - Utilizada pelo gerente para finalizar um contrato.

5.2.2 Política de Cooperação

Na implementação original do CNP, em um processo de cooperação, cada um dos agente envolvidos está habilitado a submeter não mais do que uma proposta por tarefa anunciada. Isso reduz o tráfego de mensagens e a possibilidade de um agente assumir mais tarefas para executar do que permite sua capacidade de processamento. Pela mesma razão apenas os agentes disponíveis estão aptos a submeter propostas. Essas medidas reduzem o tráfego de mensagens e mantêm a natureza distribuída do processo de cooperação. O escalonamento não preemptivo foi escolhido para ser implementado originalmente no CNP.

5.2.3 Complicações e Extensões

O processo de cooperação utilizando o CNP apresentado baseou-se na condição de que cada um dos agentes envolvidos no processo de cooperação somente poderá submeter uma proposta para execução de uma tarefa anunciada se estiver livre para processar tal tarefa. Essa estratégia pode conduzir a certas dificuldades.

Por exemplo, um agente que difundiu o anúncio de uma nova tarefa (*task annoucement*), pode não receber propostas por uma das seguintes razões: não há agentes disponíveis para a realização da tarefa; alguns agentes encontram-se disponíveis e habilitados, mas a tarefa foi ordenada em uma posição inferior da lista ordenada; nenhum dos agentes está habilitado a realizar a tarefa. Nos dois primeiros casos pode ser interessante difundir novamente o anúncio da tarefa em questão até que uma proposta tenha sido recebida de um agente. Por outro lado, na terceira situação, reeditar o anúncio da tarefa não vai ajudar. Em suma, o agente necessita determinar um método para identificar o motivo da ausência de propostas para um determinado contrato.

5.2.4 Proposta de Resposta Imediata

Uma classe de propostas, chamada resposta imediata, propicia um mecanismo para solucionar este problema. Três diferentes tipos de proposta são identificadas, permitindo ao agente indicar que ele é elegível mas encontra-se ocupado *(BUSY)*, ou que não está apto a realizar a tarefa *(INELIGIBLE)*, ou ainda que a tarefa foi escalonada em uma posição baixa da lista ordenada de tarefas *(LOW RANKING)*. O gerente pode especificar que agente irá responder em um desses casos ou em um subconjunto destes casos.

Um vez recebido um anúncio de tarefa *(task annoucement)* por um agente cuja especificação da proposta solicita uma proposta de resposta imediata, esse anúncio é tratado prioritariamente, sendo imediatamente ordenado na lista de tarefas em uma posição elevada e imediatamente respondido.

O mecanismo de resposta imediata permite ao gerente tomar o curso mais apropriado das ações no caso de uma tarefa anunciada não receber propostas. O procedimento normal consiste em reeditar o anúncio da tarefa. Caso esse contrato continue sem receber propostas, o gerente deve especificar uma proposta de resposta imediata. Se a resposta for uniformemente *BUSY*, o gerente aguarda um intervalo de tempo e só então reedita o contrato. Se todos os agentes são *INELIGIBLE*, o gerente deve rever as especificações de elegibilidade para execução da tarefa. Caso todos os agente tenham atribuído uma posição baixa na lista ordenada de tarefas *(LOW RANKING)*, o gerente pode aguardar um intervalo de tempo e reeditar o contrato na esperança de que nessa nova reedição a tarefa ocupe uma posição mais atraente ou que ele mesmo possa realizar a tarefa se estiver disponível.

5.2.5 Contratos Diretos

O processo normal de cooperação do CNP pode ser simplificado em alguns casos, resultando no aumento da eficiência do processo de cooperação. Se um gerente sabe exatamente qual dos agentes é o mais apropriado para realizar uma determinada tarefa, este pode realizar um contrato direto. Nesse caso não há necessidade de difundir o anúncio da tarefa, e conseqüentemente não serão recebidas propostas de outros agentes. Nesse caso uma mensagem do tipo direct award é enviada ao agente que deverá realizar a tarefa.

Caso o agente que recebeu uma mensagem do tipo direct award não atenda aos critérios de elegibilidade ou não esteja apto a realizar a tarefa, ele enviará uma mensagem para o gerente recusando a execução da tarefa. Tal mensagem possui uma justificativa. Uma vez recebida uma mensagem informando uma recusa de execução de tarefa, o gerente deverá rever as especificações para a realização da tarefa.

5.2.6 Requisições e Mensagens Informativas

Se uma simples transferência de informação é requerida, então uma simples seqüência requisição-informação pode ser utilizada. Uma mensagem de requisição pode ser utilizada para solicitar algum tipo de informação a um outro agente, a qual é respondida com uma mensagem informativa. Essa mensagem informativa pode também

ser utilizada para a comunicação entre gerente e contratado durante um processo de cooperação.

5.2.7 Agentes Disponíveis

Quando o processamento de tarefas é alto na comunidade de agentes, alguns anúncios de contratos podem ficar sem receber propostas devido à não existência de agentes disponíveis para processar as novas tarefas. O CNP porém, possui uma mensagem para informar que um determinado agente se encontra agora disponível para execução de novas tarefas. Essa mensagem informa ainda as habilidades do agente em questão e por quanto tempo ele se encontra disponível.

Uma vez recebida uma mensagem informando que um novo agente encontra-se disponível para realizar tarefas, o gerente que recebeu tal mensagem verifica se é possível associar ao agente em questão um dos contratos abertos e ainda sem um vencedor. Primeiramente, o gerente verifica quais dos contratos abertos podem ser executados pelo agente voluntário; caso haja algum, o gerente verifica se o voluntário se interessa pela realização do contrato. Caso haja o interesse, um contrato direto é realizado.

No CNP, cada um dos agentes aptos a assumir o papel de gerente mantém uma lista atualizada de mensagens informando a disponibilidade dos demais agentes. Antes de optar pelo anúncio de uma nova tarefa ele tenta associá-la a um dos agentes disponíveis nessa lista.

Um agente pode assumir a realização de tarefas de duas formas. Ele pode esperar por um anúncio de uma tarefa a ser realizada e submeter uma proposta ou enviar uma mensagem informando sua disponibilidade para executar uma tarefa e realizar um contrato direto.

5.2.8 Distribuição da Informação

O *Contract Net Protocol* possibilita distribuir dinamicamente a informação (procedimentos e dados) por três formas diferentes. Primeiramente, um agente pode transmitir uma requisição direta a um outro agente para transferir a informação desejada. A resposta é a informação requerida. O agente pode ainda difundir uma mensagem anunciando uma tarefa *(task annoucement)* na qual a tarefa na verdade é a transferência de informação. Uma proposta para a tarefa anunciada representa que o proponente possui a informação desejada e pretende enviá-la. Finalmente, um agente pode perceber que, em uma proposta ou tarefa, uma informação específica é necessária para a execução da tarefa. Nesse caso o gerente enviará a informação requerida ao agente vencedor do

contrato. A distribuição dinâmica de informação permite um maior aproveitamento dos recursos computacionais e facilita a inserção de novos agentes na comunidade.

5.2.9 Extended Contract Net

O *Extended Contract Net*, um refinamento do clássico CNP, previne a comunicação desnecessária no caso de um contrato apresentar uma falha. Nessa evolução do CNP, o gerente mantém as propostas referentes a uma dada tarefa até que esta seja finalizada pelo agente que venceu o contrato. A diferença em relação do CNP aparece quando o agente que venceu o contrato e se responsabilizou pela execução da tarefa falha. Nesse caso o gerente transfere a execução da tarefa para o agente cuja proposta ocupa uma posição imediatamente inferior à da proposta vencedora. A desvantagem dessa abordagem é que as propostas enviadas para uma dada tarefa permanecem armazenadas por um tempo maior, e às vezes desnecessariamente.

5.2.10 O Processo de cooperação

O processo de cooperação do *CNP* consiste em quatro passos: *difusão da meta*, *anúncio dos contratos*, *acolhimento de propostas* e *confirmação*. O primeiro passo, difusão da meta, não está definido na proposta original do CNP, porque o CNP foi originalmente concebido para alocar tarefas a nós de processamento fortemente acoplados, na Solução Distribuída de Problemas, onde as metas são normalmente parte da definição dos agentes. Entretanto, em SMA que operam em um ambiente aberto, os agentes precisam saber a motivação do processo de cooperação, implicando assim a inclusão desse passo.

- **Difusão da Meta :** a meta ou motivação pela qual o processo de cooperação está sendo aberto é difundida pelo agente que assumiu o papel de *gerente* do CNP, para todos os *contratados* em potencial;

- **Anúncios dos Contratos :** a tarefa J que requer n agentes para ser realizada, é particionada pelo *gerente* em sub-tarefas $j_1, j_2, ...j_n$, e para cada uma das sub-tarefas j_i, um contrato c_i é aberto e anunciado para todos os *contratados* em potencial;

- **Acolhimento de Propostas :** nesse estágio os *contratados* respondem ao anúncio do contrato em questão c_i, submetendo uma proposta. Cada uma destas propostas contém uma nota que expressa o quão bem o agente pode realizar aquela tarefa;

- **confirmação :** Uma lista ordenada de propostas recebidas é mantida pelo *gerente* até que uma proposta considerada satisfatória seja recebida. Uma vez recebida uma proposta considerada satisfatória, o contrato é fechado e o agente *contratado*, que submeteu tal proposta doravante denominado é contemplado vencedor do contrato em questão.

Os conflitos surgidos durante o processo de cooperação são solucionados buscando novas alternativas para realização da tarefa em questão. Isto é feito através da abertura de sucessivos contratos para realização da tarefa, onde esta pode ser divida em sub-tarefas, ou ainda pode ter o grau de satisfação para sua realização reduzido. Esta abertura de sucessivos contratos podem acarretar uma sobrecarga na comunicação.

5.2.11 Formalizando o CNP

Duas situações são consideradas para a formalização do CNP: a melhor e a pior situação. Dada a meta g_i que pode ser alcançada através da ação cooperativa J, divisível em $\{j_1, j_2, ..., j_l\}$, envolvendo n agentes $\{a_1, a_2, a_3, ..., a_n\}$. A melhor situação acontece quando, para cada uma das l ações, a primeira proposta recebida pelo gerente é satisfatória para realizar a tarefa e o contrato é imediatamente contemplado (veja figura 5.1).

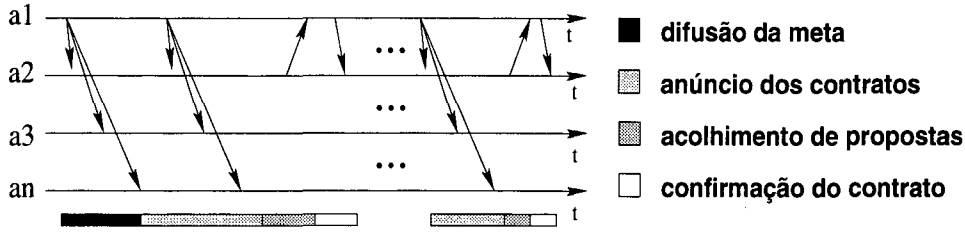

Figura 5.1: Mensagens trocadas durante a melhor situação no CNP.

O montante de mensagens $M_{CNP,melhor}$ pode ser representado pela seguinte expressão:

$$M_{CNP,melhor} = \textit{difusão da meta} + \textit{anúncio} + \textit{acolhimento} + \textit{confirmação}$$

$$M_{CNP,melhor} = (n-1) + (n-1)l + l + l \tag{5.1}$$

em um caso particular onde $l = 1$, então:

$$M_{CNP,melhor} = 2n \tag{5.2}$$

A superfície $M_{CNP,melhor} = f(l, n)$ no caso particular, quando $l = 1$ é um plano, e o crescimento de $M_{CNP,melhor}$ depende apenas de n – o número de agentes envolvidos no processo de cooperação. Entretanto, quando $l \neq 1$ a superfície $M_{CNP,melhor}$ depende tanto de n quanto de l, quantos agentes e ações estão envolvidos no processo de cooperação (veja 5.2).

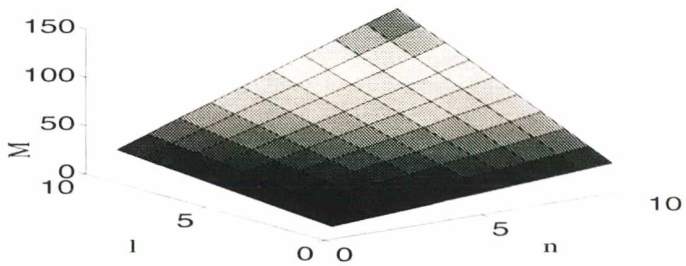

Figura 5.2: Montante de mensagens trocadas na melhor situação do CNP.

O crescimento de $M_{CNP,melhor}$ pode ser expresso por $\frac{\partial M_{CNP,melhor}}{\partial n}$ e $\frac{\partial M_{CNP,melhor}}{\partial l}$.

$$\frac{\partial M_{CNP,melhor}}{\partial n} = 1 + l \tag{5.3}$$

$$\frac{\partial M_{CNP,melhor}}{\partial l} = 1 + n \tag{5.4}$$

O pior caso acontece quando diversos contratos (w vezes) são abertos buscando por alternativas, para todas as l ações, sem sucesso, e um ciclo de negociação, com k_{CNP} ciclos, acontece para alguns atributos do contrato em questão, permitindo ao gerente achar um contratado para poder contemplar o contrato (veja figura 5.3).

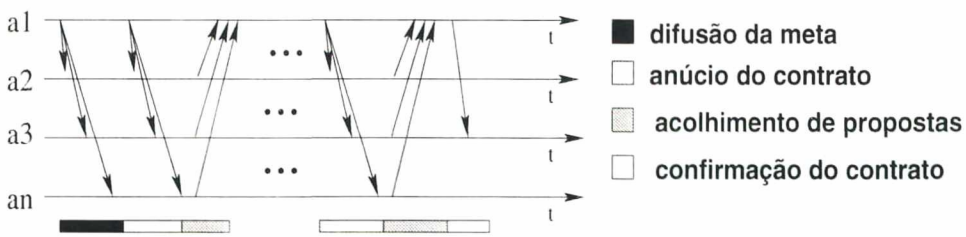

Figura 5.3: Mensagens trocadas na pior situação do CNP.

$$M_{CNP,pior} = \textit{difusão da meta} + \textit{anúncio} + \textit{acolhimento} + \textit{negociação} + \textit{confirmação}$$

$$M_{CNP,pior} = (n-1) + (n-1)lw + (n-1)lw + 2lk_{CNP} + l \tag{5.5}$$

onde k_{CNP} pode ser representado por:

$$k_{CNP} = \sum_{i=1}^{l} \alpha k_i$$

A superfície $M_{CNP,pior} = f(n, l)$ pode ser desenhada assumindo k_{CNP} como uma pequena constante, uma vez que ela representa algumas mensagens trocadas entre um gerente e um contratado negociando alguns atributos do contato. Faz-se necessário ainda atribuir valores a w, por exemplo 1, 2 e 3, representando quantos contratos foram abertos para a ação j_i. É possível ver que $M_{CNP,pior}$ pode atingir um grande número de mensagens trocadas para um pequeno crescimento de w (veja figura 5.4).

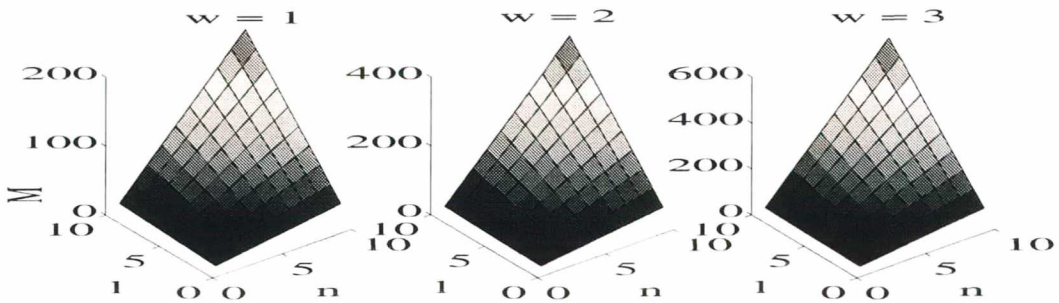

Figura 5.4: Montante de mensagens trocadas na pior situação do CNP.

O crescimento de $M_{CNP,pior}$ representado por $\frac{\partial M_{CNP,pior}}{\partial n}$ e $\frac{\partial M_{CNP,pior}}{\partial l}$ é :

$$\frac{\partial M_{CNP,pior}}{\partial n} = 1 + 2lw \tag{5.6}$$

$$\frac{\partial M_{CNP,pior}}{\partial l} = 2nw + 2w + 2k_{CNP} + 1 \tag{5.7}$$

5.3 Coalisão Baseada em Dependência

O modelo de Coalisão Baseada em Dependência (CBD) [Sic98] [IS00] é um modelo de organização dinâmica baseado na Teoria do Poder Social [Cas90], que utiliza o conceito de relações de dependência. Esse mecanismo de raciocínio social é considerado essencial para a formação de uma sociedade de agentes autônomos, imersos em um contexto de sistema multiagentes aberto, onde os agentes podem dinamicamente entrar ou sair da sociedade sem que haja um mecanismo global de controle. As relações de dependência permitem ao agente saber quais das suas metas são atingíveis e quais dos seus planos são viáveis a cada momento, permitindo ao agente a adaptação a um dado cenário de interesse. Dessa forma um agente pode dinamicamente escolher uma meta

para perseguir e um plano para realizá-la assegurando-se que cada habilidade necessária para a execução do plano está disponível na sociedade. O modelo de formação de coalisão introduz a noção de *situação de dependência*, a qual permite ao agente avaliar a susceptibilidade dos demais agentes em adotar suas metas.

Na CBD, os agentes necessitam de um certo montante de informação sobre os outros agentes da sociedade, suas metas, planos, recursos e ações, antes de escolher um parceiro em potencial para formar uma coalisão. Esta informação é obtida durante a fase inicial de apresentação. A cada entrada de um agente na sociedade, ele deve se apresentar aos demais, enviando-lhes as informações necessárias para que eles o considerem um parceiro em potencial nas futuras coalisões. Os outros agentes, por sua vez, enviam para o novo agente as informações recíprocas. Da mesma forma, quando um agente sai da sociedade, ele deve informar aos demais que esta deixando a sociedade, visando deixar os demais agentes cientes de que alguns recursos ou ações não mais estão disponíveis.

Após a fase de apresentação, são iniciados os ciclos de resolução. São chamados de ciclos de resolução, os períodos em que o agente necessita para alcançar uma de suas metas. Este agente é chamado de *agente ativo*. O ciclo de resolução inicia quando um agente ativo escolhe uma meta a ser atingida e um plano para ser realizado. O plano escolhido pode ser um plano autônomo, um plano que o agente pode executar sozinho, ou um plano dependente, um plano no qual o agente necessita da ajuda de outros agentes para executar ao menos uma ação ou liberar o controle de um dado recurso. Caso o agente tenha escolhido um plano autônomo, o agente ativo pode executar tal plano sozinho sem a necessidade de formar qualquer coalisão. Entretanto, se o agente optar por um plano dependente, o processo para formação de uma coalisão é iniciado.

O agente ativo avalia as relações de dependência entre os parceiros em potencial e ele próprio. O parceiro em potencial preferido é aquele cuja susceptibilidade de cooperar é a mais alta. Esta susceptibilidade é capturada pela noção de situação de dependência, que modela o comportamento dos outros agentes. O agente ativo constrói uma lista ordenada dos possíveis parceiros. O agente ativo inicia então uma fase de questionamento aos parceiros em potencial da lista, perguntando-lhes se estão dispostos a integrar uma possível coalisão. O processo termina quando alguns agentes aceitam tomar parte na coalisão e o agente ativo lhes envia uma mensagem confirmando a coalisão.

5.3.1 Coalisão Baseada em Dependência*

Uma versão modificada do modelo de Coalisão Baseada em Dependência, chamada CBD*, foi adotada neste trabalho para fins de comparação com a proposta do autor.

Essa versão modificada apresenta as seguintes alterações em relação à versão original:

- **Ciclo de Apresentação Inicial :** este ciclo foi suprimido, admitindo-se que o agente ativo, conhece os agentes que potencialmente podem integrar a coalisão, conhecendo suas habilidades, que tipo de ações ou recursos um dado agente pode contribuir em um processo de coalisão. Entretanto, o agente ativo desconhece a disponibilidade e as impressões dos agentes a respeito da meta global.

- **Novo Ciclo de Apresentação :** um novo ciclo de apresentação é inserido logo após a difusão da meta global que motiva o processo de cooperação. Nesse novo ciclo de apresentação todos os agentes da comunidade difundem suas impressões sobre aquela meta global. Estas impressões, afinidades e contradições, a respeito da meta global definem quais os agentes que participarão no processo de cooperação.

- **Escolha do Plano :** o agente ativo escolherá não um plano p_i, mas um conjunto de planos P_i, composto por uma lista de planos $p_1, p_2, ..., p_t$, sendo p_1 o plano ótimo, tentando inicialmente realizar p_1, e em seguida os demais planos $p_2, p_3, ..., p_t$.

5.3.2 Mensagens Básicas

O CBD* possui um conjunto definido de mensagens utilizadas durante o processo de cooperação que serão apresentadas aqui. Serão ainda descritos a informação codificada em cada uma destas mensagens e seu processamento.

- **Mensagem de Difusão da Meta** - O agente que necessita que uma tarefa cooperativa seja realizada, inicia o processo de cooperação difundindo para os demais agentes uma mensagem informando a meta global desejada.

- **Mensagem de Apresentação** - Após o recebimento de uma Mensagem de Difusão da Meta, o agente avalia a meta global proposta e difunde para todos os agentes suas impressões sobre a meta proposta. Esta mensagem define quais são os agentes que participarão das fases seguintes do processo de cooperação.

- **Mensagem de Proposta** - propõe a um determinado agente, que se manifestou favorável a integrar o processo de cooperação, a realização de uma tarefa, ação ou liberação de um dado recurso.

- **Mensagem de Aceitação** - manifesta favoravelmente à realização de uma tarefa, ação ou liberação de um dado recurso, proposto pelo agente ativo.

- **Mensagem de Declinação** - rejeita a realização de uma tarefa, ação ou liberação de um dado recurso, proposto pelo agente ativo.
- **Mensagem de Confirmação** - confirma a realização de uma tarefa, ação ou liberação de um dado recurso.
- **Mensagem de Convergência** - informa aos agentes envolvidos, a convergência do processo de cooperação.

5.3.3 O Processo de cooperação

O processo de cooperação do *CBD** consiste em quatro passos: *difusão da meta, apresentação, ciclo de negociação* e *acordo*.

- **Difusão da Meta :** a meta ou motivação g_i pela qual o processo de cooperação está sendo aberto é difundida pelo agente ativo, para todos os parceiros em potencial.
- **Apresentação :** os agentes da comunidade difundem suas impressões sobre a meta global proposta g_i, externando suas disponibilidades em integrar o processo de cooperação. Os agentes que se manifestarem favoráveis à meta global em questão g_i, serão considerados pelo agente ativo parceiros em potencial. Para os demais agentes, a manifestação favorável àquela meta global, implica a disponibilidade das habilidades, ações e ou recursos do agente em questão para o processo de cooperação em andamento.
- **Ciclos de Negociação :** o agente ativo e os parceiros em potencial trocam mensagens apresentando propostas, fazendo concessões ou buscando alternativas para a alocação de uma tarefa, realização de uma ação ou liberação de um recurso.
- **Acordo :** a alocação de uma tarefa, realização de uma ação ou liberação de um recurso, e quando isso deve acontecer.

5.3.4 Formalização da CBD*

Duas situações são também consideradas aqui para formalizar a CBD*: a melhor e a pior situação. Dada a meta g_i que pode ser atingida através da ação cooperativa J divisível em $\{j_1, j_2, ..., j_l\}$, envolvendo n agentes, a melhor situação acontece quando após a etapa de apresentação, para cada uma das l ações, o agente ativo escolhe um outro agente para realizar a coalisão, envia-lhe uma proposta, e o respectivo agente concorda (veja figura 5.5).

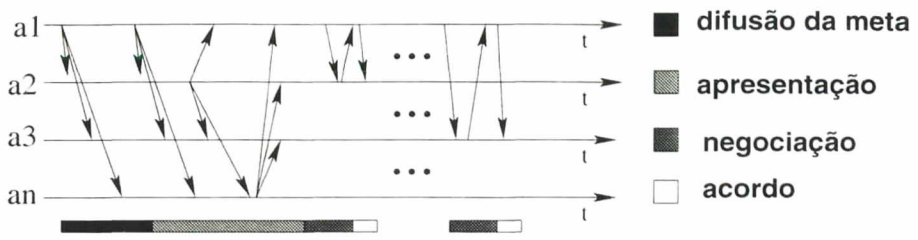

Figura 5.5: Mensagens trocadas no melhor caso da CBD*

O montante de mensagens trocado na melhor situação $M_{CBD^*,melhor}$ pode ser expresso pela seguinte equação:

$$M_{CBD^*,melhor} = difusão\ da\ meta + apresentação + negociação + acordo$$

$$M_{CBD^*,melhor} = (n-1) + n(n-1) + 2l + l \tag{5.8}$$

A superfície $M_{CBD^*,melhor} = f(l,n)$, em um caso particular, quando $l = 1$, é também um plano que depende apenas de n, e cresce muito mais do que o mesmo caso particular para o CNP, $M_{CNP,melhor}$. Entretanto quando $l \neq 1$ a superfície $M_{CBD*,melhor}$ é menor que $M_{CNP,melhor}$ quando $l > n(n-1)/(n-2)$. Quando l e n crescem $M_{CBD^*,melhor}$ é maior que $M_{CNP,melhor}$ (veja 5.6).

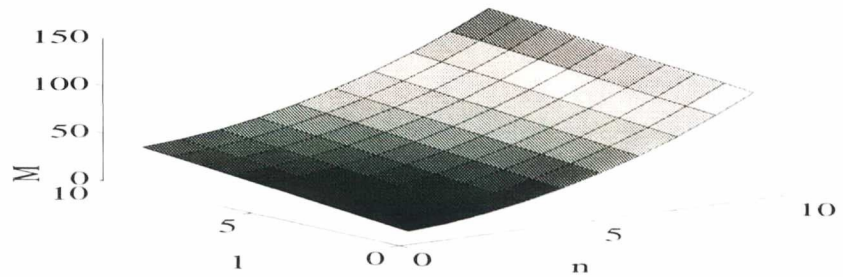

Figura 5.6: Mensagens trocadas no melhor caso da CBD*.

O crescimento da superfície $M_{CBD^*,melhor}$ pode ser expresso por $\frac{\partial M_{CBD^*,melhor}}{\partial n}$ e $\frac{\partial M_{CBD^*,melhor}}{\partial l}$.

$$\frac{\partial M_{CBD^*,melhor}}{\partial n} = 2n \tag{5.9}$$

$$\frac{\partial M_{CBD^*,melhor}}{\partial l} = 3 \tag{5.10}$$

O pior caso na CBD* acontece quando, para todas as l ações, diversas propostas e contra-propostas são trocadas, k_{CBD^*}, pelos n agentes durante o ciclo de negociação,

buscando por alternativas ou fazendo concessões para alcançar um acordo (veja figura 5.7).

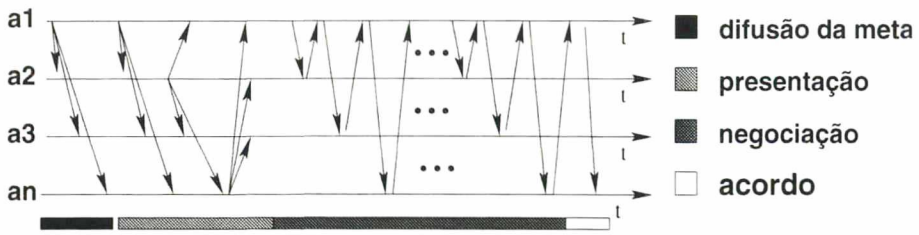

Figura 5.7: Mensagens trocadas durante o pior caso da CBD*

$$M_{CBD^*,pior} = difusão\ da\ meta + apresentação + negociação + acordo$$

$$M_{CBD^*,pior} = (n-1) + n(n-1) + 2nlk_{CBD^*} + l \tag{5.11}$$

onde k_{CBD^*} pode ser definido por:

$$k_{CBD^*} = \sum_{j=1}^{l} \sum_{i=1}^{n} \alpha k_i$$

A superfície $M_{CBD^*,pior}$, como $f(n,l)$, pode ser obtida atribuindo os valores 1, 2 e 3 para k_{CBD*}, uma vez que k_{CBD^*} é uma constante que assume diferentes valores dependendo de quantos ciclos de negociação são necessários para alcançar o acordo. É possível ver que a superfície $M_{CBD^*,pior}$ é maior que $M_{CNP,pior}$ para as seguintes situações: $w = k_{CBD^*} = 1$, $w = k_{CBD^*} = 2$ e $w = k_{CBD^*} = 3$.

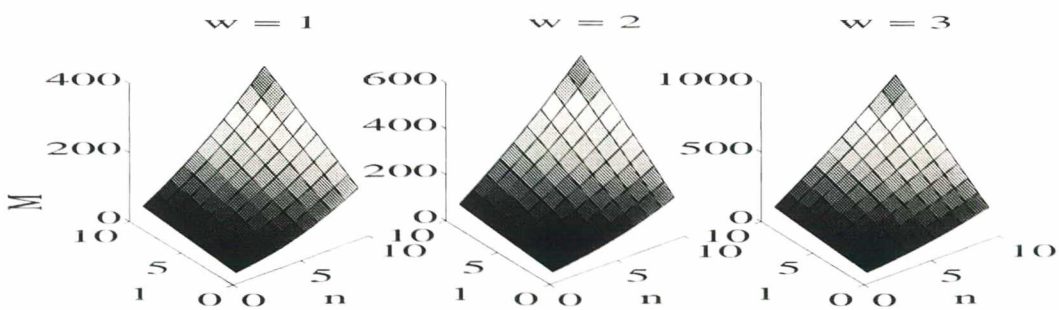

Figura 5.8: Mensagens trocadas na pior situação CBD*.

O crescimento de $M_{CBD^*,pior}$ é expresso por $\partial M_{CBD^*,pior}/\partial n$ and $\frac{\partial M_{CBD^*,pior}}{\partial l}$ is:

$$\frac{\partial M_{CBD^*,pior}}{\partial n} = 2n + 2lk_{CBD^*} \tag{5.12}$$

$$\frac{\partial M_{CBD^*,pior}}{\partial l} = 2nk_{CBD^*} + 1 \tag{5.13}$$

5.4 Considerações sobre o CNP e CBD*

O CNP consiste em uma estratégia de cooperação extremamente voltada para alocação de tarefa. Mostra-se bastante flexível pois permite que a comunidade de agentes assuma um comportamento cooperativo mesmo conhecendo-se muito pouco sobre os demais agentes envolvidos no processo de cooperação. Utilizar o CNP como estratégia de cooperação é dividir a tarefa que se deseja realizar em sub-tarefas e abrir um contrato para cada uma desta sub-tarefas direcionado a todos os agentes ativos naquele instante. Permite ainda que os contratos relativos às sub-tarefas sejam abertos paralelamente. Por um outro lado, conhecendo-se qual o melhor agente para realizar uma dada tarefa, o CNP permite a abertura de um contrato direto. Entretanto as limitações do CNP, como estratégia de cooperação, começam a aparecer quando a solução ótima não está disponível. Nesses casos o processo de cooperação pode ser conduzido a aberturas sucessivas de contratos para realização de uma mesma tarefa.

As estratégias de cooperação baseadas em processos de coalisão concentram seus esforços na solução de conflitos existentes entre os agentes da comunidade envolvidos em um processo de cooperação. Esses conflitos são solucionados através de ciclos de negociação buscando-se planos alternativos, através de um conjunto contendo propostas e contra-propostas, e que se finaliza com a aceitação ou recusa do plano. As diversas estratégias de negociação visam minimizar a troca de propostas e contra-propostas otimizando a convergência. A utilização de negociação como processo de cooperação requer uma base de conhecimento social, em que todas as informações relevantes a respeito dos demais agentes envolvidos no processo de cooperação sejam mantidas atualizadas. Quando uma tarefa cooperativa precisa ser realizada o agente irá procurar em sua base de conhecimento social por um agente específico para realizar esta tarefa. Essa estratégia permite que um agente convença um outro agente ajudá-lo em uma tarefa cooperativa, permite a formação de grupos de agentes para realização de uma tarefa específica e permite ainda alcançar um acordo quando a solução ótima não está disponível. Entretanto, utilizar uma estratégia de cooperação baseada em negociação implica ter um conhecimento social que pode assumir grandes dimensões dependendo do número de agentes pertencentes à comunidade, das suas habilidades e principalmente do quão freqüentemente as características dessa comunidade assumem novos valores. Infelizmente a atualização do conhecimento social de um agente em uma comunidade aberta normalmente implica utilizar significativamente o canal de comunicação além

do tempo necessário para a convergência do processo de cooperação.

Uma nova estratégia de cooperação chamada Conhecimento Social Dinâmico é apresentada no capítulo 6. Essa estratégia utiliza o conceito de contratos, similar ao CNP e adiciona novos conceitos como Estrutura de Contratos, Conjunto de Planos e Base de Conhecimento Social Dinâmica. Esses novos conceitos modificam o papel dos contratos que no CNP são utilizados apenas para alocação de tarefas e passam a assumir também a função de um instrumento para aquisição de conhecimento da comunidade. Além disso, introduz a idéia de construir dinamicamente uma base de conhecimento social direcionada para as metas que os agentes pretendem alcançar.

Capítulo 6

Conhecimento Social Dinâmico

6.1 Introdução

A estratégia de cooperação proposta, Conhecimento Social Dinâmico (CSD), introduz o conceito de uma base de conhecimento social dinamicamente construída e direcionada para alcançar metas específicas. A contribuição mais importante desta estratégia de cooperação pode ser percebida com mais relevância em sistemas multiagentes autônomos e com restrições de tempo real. Neste tipo de comunidade de agentes, tanto o número de agentes aptos a integrar um processo de cooperação como as características do ambiente apresentam um comportamento dinâmico. A estratégia proposta permite alcançar metas globais com processos de cooperação curtos, evitando a troca de um número excessivo de mensagens. Permite ainda a utização de uma abordagem cognitiva na condução do processo de coooperação além de permitir uma implementação concorrente baseada no modelo de agente autônomo concorrente apresentado no capítulo 3. Esta implementação concorrente, assegura resposta em tempo real e permite que os processos de cooperação aconteçam concorrentemente às respostas do agente aos estímulos do ambiente. Antes de iniciar a descrição da estratégia de cooperação proposta, três novas definições,; Estrutura de Contrato, Conjunto de Planos e

Base de Conhecimento Social Dinâmico; são introduzidas.

Definição 1: Uma Estrutura de Contrato C_i é composto por uma meta g_i, um grupo de agentes $a_1, a_2, ..., a_n$, um corte α que expressa o grau de satisfação desejado para a realização de g_i, *alpha-cut*, um conjunto de contatos de contratos $c_1, c_2, ..., c_n$, uma lista para armazenar os contratos já contemplados, um campo onde é armazenado o agente ou o grupo de agentes vencedor do contrato *winners* e um prazo final para o aceitação de proposta *dl*. A sintaxe do Suporte de Contratos é apresentado a seguir:

```
((goal  gi)
 (agent-group ( a1, a2, ...,an ))
 (alpha-cut  alpha )
 (deadline dl)
 (winners (winners))
 (contract-set (c1, c2, ..., cn ))
 (award-contracts (())))
```

Definição 2: *Um Conjunto de Planos* P_i é composto por uma meta g_i, um identificador *plan_set_id* e uma lista de planos $p_1, p_2, ..., p_t$, que satisfazem a meta g_i, ordenados segundo a optimalidade dos planos, onde p_1 é o plano ótimo.

Definição 3: *Base de Conhecimento Social Dinâmica* é uma base de conhecimento construída a partir das propostas recebidas pelo agente para uma Estrutura de Contrato C_i específica. Esta base de conhecimento é mantida aberta e atualizada até que um agente ou grupo de agentes seja declarado vencedor para a Estrutura de Contrato.

6.2 O Processo de Cooperação

O processo de cooperação tem início quando uma tarefa cooperativa precisa ser realizada para alcançar o estado descrito pela meta g_i. A meta g_i é difundida para os agentes envolvidos no processo de cooperação $a_1, a_2, ..., a_n$. Um conjunto de planos P_i compatível com a meta g_i é selecionado. Para cada uma das tarefas $j_1, j_2, j_3, ..., j_n$ pertencentes aos planos $p_1, p_2, ..., p_t$ que compõem o conjunto de planos P_i, um contrato é criado. Existindo tarefas múltiplas, apenas um contrato é criado e a *multiplicidade* desse contrato é incrementada. Os contratos criados a partir de P_i, a meta g_i em conjunto com o corte α *alpha-cut* e um prazo final para encerramento do contrato *dl*, dão origem a uma Estrutura de Contratos C_i.

Uma vez criado C_i, os diferentes contratos armazenados no *contract-set*, $c_1, c_2, ..., c_n$, são anunciados para os agentes envolvidos no processo de cooperação agent-group.

Estes agentes irão avaliar a tarefa anunciada e responder a cada um dos contratos anunciados $c_1, c_2, ..., c_n$, aceitando ou rejeitando a realização de cada um dos contratos.

Nos casos em que o agente envia uma proposta aceitando a realização de uma tarefa, um quantificador *grade*, expressando quão bem o agente pode realizar aquela tarefa é associado a proposta. Este *grade* é utilizado para selecionar um vencedor para o contrato. O agente é mantido aceitando novas propostas, e armazenado-as em C_i, até que uma das seguintes situações se verifique:

- **Situação 1 : Premiação Direta** - todos os contratos pertencentes ao plano p_1, plano ótimo de P_i, apresentam um vencedor. Nesta situação a meta g_i pode ser alcançada pela realização das tarefas, pertencentes ao plano ótimo p_1, e o processo de cooperação conduzido para o estágio decontemplação do contrato é finalizado.

- **Situação 2** - O agente responsável pela Estrutura de Contrato C_i, já recebeu uma proposta de cada um dos agentes envolvidos no processo de cooperação $a_1, a_2, ..., a_n$. Os contratos que permitem a realização do plano ótimo p_1 não apresentam vencedor. Neste caso, a Estrutura de Contratos C_i será utilizada para construir a Base de Conhecimento Social Dinâmico.

- **Situação 3** - O prazo para finalizar os contratos se expirou. Nesta situação a Estrutura de Contratos C_i será utilizada para construir a Base de Conhecimento Social Dinâmico.

A Base de Conhecimento Social Dinâmico, referida nas situações 2 e 3, é construída a partir das propostas recebidas dos demais agentes envolvidos no processo de cooperação. Esta base de conhecimento é utilizada para conduzir o processo de cooperação. Neste caso as seguintes possibilidades podem acontecer, de acordo com o conhecimento armazenado no agente e a Base de Conhecimento Social Dinâmico:

1. O corte α da Estrutura de Contrato C_i é reduzido, de forma que um dos agentes possa ser declarado vencedor para este novo corte α.

2. Um grupo de agentes capaz de realizar um dos planos pertencentes a lista de planos $\{p_1, p_2, ..., p_t\} \in P_i$ é organizado e declarado vencedor da Estrutura de Contrato C_i.

3. Um novo *contrac-set* $c_1, c_2, ..., c_n$ é gerado e anunciado para os demais agentes envolvidos no processo de cooperação.

4. O processo de cooperação é fechado sem um vencedor.

Um conjunto de regras automaticamente gerado a partir do conjunto de planos P_i para a Base de Conhecimento Social Dinâmico conduzirá o processo de cooperação, determinando qual destas possibilidades será de fato adotada.

6.3 Formalizando o CSD

Considera-se a melhor e a pior situação que podem acontecer em um processo de cooperação, utilizando a estratégia de cooperação CSD, dada uma meta g_i que pode ser atingida pela ação cooperativa J, divisível nas sub-tarefas $\{j_1, j_2, ..., j_l\}$, envolvendo n agentes. No CSD um outro parâmetro é introduzido, chamado l_{CSD}, que representa as tarefas possíveis assumidas pelos agentes para atingir g_i. A melhor situação acontece quando, para as l tarefas em J a proposta satisfatória para cada um dos contratos $c_1, c_2, , c_l$ que satisfazem o plano ótimo p_1 são as primeiras propostas recebidas (veja figura 6.1).

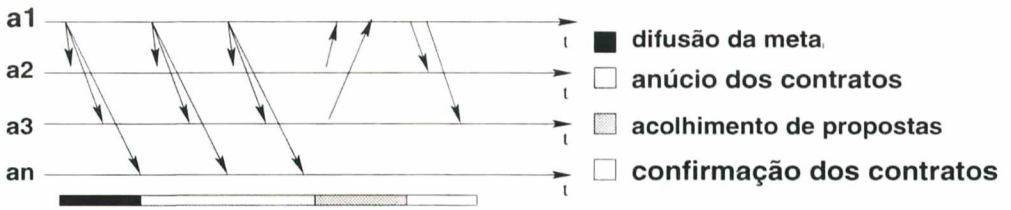

Figura 6.1: Mensagens trocadas durante a melhor situação do CSD.

O montante de mensagens trocadas na melhor situação pode ser representado por:

$$M_{CSD,melhor} = difusão\ da\ meta + anúncio + acolhimento + contemplação$$

$$M_{CSD,melhor} = (n-1) + (n-1)l_{CSD} + l_{CSD} + l \tag{6.1}$$

$$l_{CSD} = \alpha l$$

De forma similar ao CNP, existe um caso particular onde $l = 1$ que irá conduzir $M_{CSD,melhor}$ para o mesmo plano representado pela equação 5.2. Entretanto existe uma outra situação não tão particular, quando ao menos uma tarefa $j_i \in J$ é a mesma que outra tarefa $j_p \in J$, ou seja $\alpha < 1$ então $M_{CSD,melhor} < M_{CNP,melhor}$. Em uma situação extrema quando todas as $j_i \in J$ são as mesmas ações, então $\alpha = 1/n$ e $M_{CSD,melhor}$ é muito menor que $M_{CNP,melhor}$ e $M_{CBDmelhor}$.

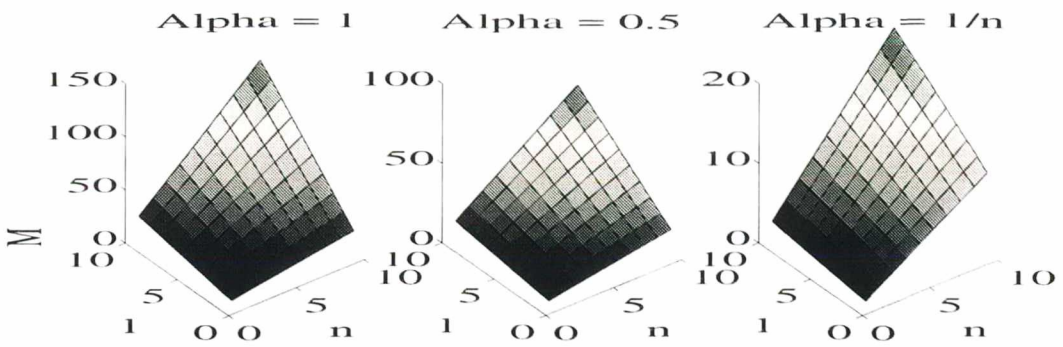

Figura 6.2: Mensagens trocadas na melhor situação do CSD.

O crescimento da superfície $M_{CSD,melhor}$ pode ser representado por $\frac{\partial M_{CSD,melhor}}{\partial n}$ e $\frac{\partial M_{CSD,melhor}}{\partial l}$.

$$\frac{\partial M_{CSD,melhor}}{\partial n} = 1 + \alpha l \tag{6.2}$$

$$\frac{\partial M_{CSD,melhor}}{\partial l} = 1 + \alpha n \tag{6.3}$$

Por outro lado a pior situação acontece quando nenhum dos contratos pertencentes ao plano ótimo apresenta uma proposta satisfatória. Neste caso a Base CSD e alguns ciclos de negociação (k_{CSD} vezes) são necessários para concluir o processo de cooperação.

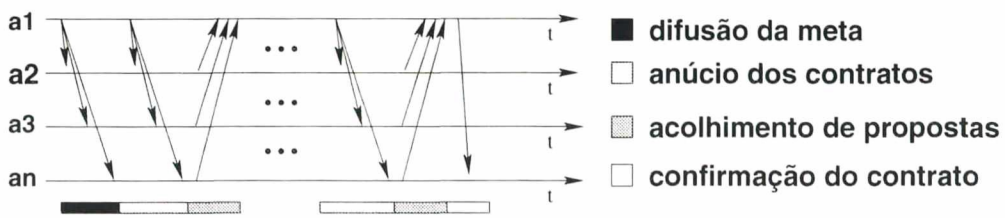

Figura 6.3: Mensagens trocadas na melhor situação do CSD.

e na pior situação:

$M_{CSD,pior}$ = *difusão da meta + anúncio + acolhimento + negociação + comfirmação*

$$M_{CSD,pior} = (n-1) + (n-1)l_{CSD} + (n-1)l_{CSD} + 2k_{CSD} + l \tag{6.4}$$

onde k_{CSD} pode ser definida por:

$$k_{CSD} = \sum_{h=1}^{l_CSD} \sum_{i=1}^{n} \alpha k_i$$

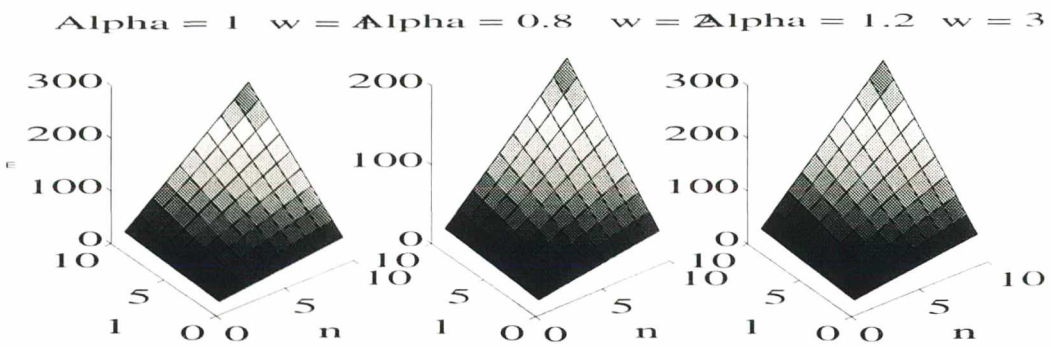

Figura 6.4: Montante de mensagens trocadas na melhor situação do CSD.

O crescimento de $M_{CSD,pior}$ expressado por $\partial M_{CSD,pior}/\partial n$ e $\partial M_{CSD,pior}/\partial l$ é:

$$\partial M_{CSD,pior}/\partial n = 1 + 2\alpha l \tag{6.5}$$

$$\partial M_{CSD,pior}/\partial l = 2n\alpha + 2\alpha k_{CSD} + 1 \tag{6.6}$$

6.4 Considerações sobre o Conhecimento Social Dinâmico

No caso particular onde a ação cooperativa J é composta por sub-tarefas $\{j_1, j_2, ..., j_l\}$ de mesma descrição, apenas um contrato c_1 é criado e difundido. Uma vez recebidas l propostas satisfatórias, são feitas as contemplações. Isso reduz significativamente o montante de mensagens trocadas pelos agentes durante o processo de cooperação.

Dois outros pontos importantes devem ser considerados. Cada vez que $\alpha < 0$, indicando a ocorrência de múltiplas $j_a = j_b = j_c$ e $k_{CSD} \leq k_{CNP}$ então $M_{CSD,pior} < M_{CNP,pior} < M_{CBD^*,pior}$. Outro ponto importante é que é possível obter α ligeiramente maior que 1, como por exemplo 1.2, de forma a reduzir significativamente k_{CSD} implicando em $M_{CSD^*,pior} < M_{CNP,pior} < M_{CBD,pior}$. Em outras palavras, é possível utilizar os anúncios de contrato para explorar diferentes possibilidades para as ações envolvidas no processo de cooperação e utilizar um sistema baseado em regras para combinar tais possibilidades e reduzir os ciclos de interação entre os agentes visando a convergência mais rápida.

Implementações do CSD, CNP e CBD* e uma análise comparativa entre as implementações são apresentada no capítulo 7.

Parte III

Implementação e Testes

Capítulo 7

Implementação e Resultados

7.1 Introdução

A implementação desta tese consiste em duas partes: a biblioteca *Expert-Coop++* e os sistemas multiagentes CNP_Team, CBD*_Team e CSD_Team. A *Expert-Coop++* é uma biblioteca Orientada a Objetos, implementada na linguagem de programação C++, destinada a auxiliar o desenvolvimento de sistemas multiagentes sob restrições de tempo real do tipo melhor esforço. CNP_Team, CBD*_Team e CSD_Team são sistemas multiagentes que utilizam as estratégias de cooperação disponibilizadas na biblioteca Expert-Coop++ e apresentadas nesta tese: CNP, CBD* e CSD.

Os sistemas multiagentes implementados foram submetidos a um mesmo estado inicial e com uma mesma meta global a ser atingida em duas situações distintas: quando todos os agentes necessários estão aptos a cooperar e são capazes de realizar as ações cooperativas satisfatoriamente e quando apenas os agentes estritamente necessários para que a meta seja atingida estão aptos a realizar as ações cooperativas satisfatoriamente.

Neste capítulo serão brevemente apresentadas na seção 7.2 as classes pertencentes à biblioteca Expert-Coop++, que implementam o conhecimento social dos agentes. A implementação dos sistema multiagentes é apresentada na seção 7.5, o problema proposto na seção 7.4 e os resultados obtidos na seção 7.6. Na seção 7.6 são apresentados ainda, nas tabelas 7.24 e 7.25 os tempos de convergência e o montante de

mensagens trocadas em cada um dos processos de cooperação. Finalmente uma análise dos resultados é apresentada.

7.2 Expert-Coop++

O Expert-Coop++ foi concebido para facilitar a implementação de sistemas multiagentes cognitivos, abertos e democráticos, sob restrições de tempo real do tipo melhor esforço e baseados em uma arquitetura pré-determinada. O Expert-Coop++ oferece uma arquitetura de agente, o Agente Autônomo Concorrente (apresentado no capítulo 3), uma linguagem de comunicação de agentes Parla [CB97], e três estratégias de cooperação, CNP, CBD* e CSD apresentadas nos capítulos 5 e 6.

Fica a cargo do projetista a definição das regras que determinam os comportamentos social, cognitivo e instintivo. Para o nível reativo existem já implementados, a partir da biblioteca CNCL [JS97], alguns comportamento reativos destinados a sistemas multiagentes para a categoria de robôs simulados da RoboCup. Outros comportamentos reativos podem ser implementados. Faz-se necessário também especificar os planos para atingir as metas globais admitidas pelo sistema multiagente.

7.2.1 Base de Conhecimento Social

A classe Social_Base provê o conhecimento social necessário a um processo cooperativo:

- a motivação pela qual os agentes devem integrar o processo de cooperação, isto é, a meta global;
- uma lista de agentes que podem integrar o processo de cooperação;
- um conjunto de planos que satisfazem a meta global em questão;
- o plano que está sendo apreciado pelos agentes que integram o processo de cooperação, *o Plano Ativo*;
- e finalmente uma lista de conjuntos de planos, que define para quais metas globais o agente está apto a abrir um processo de cooperação.

Um plano p consiste em uma lista de tarefas $t_1, t_2, ..., t_j$, que uma vez realizadas por uma comunidade de agentes $a_1, a_2, ..., a_n$, implica a satisfação da meta global em questão g_i. Os planos $p_1, p_2, ..., p_t$, que satisfazem a mesma meta g_i, são armazenado em um *Conjunto de Planos P*, onde o primeiro plano é o plano ótimo.

7.2.2 Base CNP

O *Contract Net Protocol* é uma das estratégias de cooperação disponíveis na biblioteca Expert-Coop++. Implementada a partir da classe CNP_Base, filha da classe Social_Base, permite instanciar uma base de Conhecimento Social cuja estratégia de cooperação utilizada é o CNP.

A classe CNP_Base, por sua vez, provê o conhecimento necessário para a utilização do Contract Net Protocol como estratégia de cooperação: uma lista de planos, proveniente de um conjunto de planos; uma *lista de contratos* que contém um contrato para cada uma das tarefas pertencentes ao plano ativo; uma lista com os contratos contemplados; e o contrato sob apreciação. O processo de cooperação implementado em CNP_Base é gerido pelo método *CNP_Cooperative_Behavior*.

O processo de cooperação é iniciado a partir da requisição de uma meta global g_i. O conjunto de planos referente a g_i, P_i, é selecionado, a meta g_i é difundida para os agentes $a_1, a_2, ..., a_n$ pertencentes à lista de agentes, e para cada uma das tarefas $t_1, t_2, ..., t_j$ pertencentes ao plano ótimo da lista de planos armazenada em P_i, um contrato c_i é aberto e armazenado na lista de contratos $c_1, c_2, ..., c_n$. O primeiro contrato da lista torna-se ativo e é anunciado aos agentes $a_1, a_2, ..., a_n$. Este contrato é mantido aberto até que uma das seguintes situações aconteça:

- uma proposta pr_i, que satisfaça o contrato c_i, tenha sido recebida. O contrato c_i é então contemplado, o agente a_i é informado da contemplação. O contrato c_i é armazenado na lista de contratos contemplados e um novo contrato oriundo da lista de contratos torna-se ativo.

- todos os agentes da comunidade $a_1, a_2, ..., a_n$ tenham respondido ao anúncio do contrato com suas respectivas propostas pr_i e nenhuma delas satisfaz ao contrato ativo c_i, implicando o insucesso do plano ativo. Neste caso, o plano ativo é abortado e o próximo plano pertencentes à lista de planos é tornado ativo.

O processo de cooperação converge quando todos os contratos $c_1, c_2, ..., c_n$ de um mesmo plano pertencente ao conjunto de planos P_i, apresentarem um agente contratado para assumir a execução da tarefa. Por sua vez, quando todos os planos pertencentes a P_i falham, a convergência não pode ser atingida.

7.2.3 Base CBD

A classe CBD_Base, por sua vez, provê o conhecimento necessário para a utilização do CBD* como estratégia de cooperação corrente. Uma lista de planos, proveniente de

um conjunto de planos P_i; uma *lista de contratos* que contém um contrato para cada uma das tarefas pertencentes ao plano ativo; uma lista com os contratos contemplados; e o contrato sob apreciação. O processo de cooperação implementado em CBD_Base é gerido pelo método *CBD_Cooperative_Behavior.*

O processo de cooperação é iniciado a partir da requisição de uma meta global g_i. O conjunto de planos referente a g_i, P_i, é tornado ativo, a meta g_i é difundida para os agentes $a_1, a_2, ..., a_n$ pertencentes à lista de agentes, e para cada uma das tarefas $t_1, t_2, ..., t_j$ pertencentes ao plano ótimo da lista de planos armazenada em P_i, um contrato c_i é aberto e armazenado na lista de contratos $c_1, c_2, ..., c_k$.

Os agentes $a_1, a_2, ..., a_n$, por sua vez manifestam para os demais agentes suas impressões a respeito da meta g_i. Os agentes que se manifestarem favoráveis à meta g_i são incluídos na lista de parceiros em potencial $a_1, a_2, ..., a_p$.

O primeiro contrato da lista torna-se ativo e o agente ativo tenta inicialmente uma coalisão com um dos agentes presentes na lista de parceiros em potencial $a_1, a_2, ..., a_p$. Inicialmente o agente ativo busca pelo agente indicado como preferencial pelo plano. Caso este agente não esteja presente na lista, outro é escolhido para tentar a coalisão. Iniciam-se então os ciclos de negociação para tenta alocar os contratos $c_1, c_2, ..., c_k$ aos agentes $a_1, a_2, ..., a_p$. Uma das seguintes situações pode acontecer :

- um contrato c_i, seja aceito. O contrato c_i é então finalizado, o agente a_i informado que sua proposta foi aceita. O contrato c_i é armazenado na lista de contratos contemplados e um novo contrato oriundo da lista de contratos torna-se ativo.

- todos os agentes parceiros em potencial $a_1, a_2, ..., a_p$, tenham rejeitado a coalisão para o contrato ativo c_i, implicando o insucesso do plano ativo. Neste caso, o plano ativo é abortado e o próximo plano pertencente à lista de planos é tornado ativo.

O processo de cooperação converge quando todos os contratos $c_1, c_2, ..., c_n$ de um mesmo plano pertencente ao conjunto de planos P_i apresentarem um agente acordado para assumir a execução da tarefa, ação ou liberação do recurso. Por sua vez, quando todos os planos pertencentes a P_i falham, a convergência não pode ser atingida.

7.2.4 Base CSD

A estratégia de cooperação Conhecimento Social Dinâmico é implementada a partir da classe CSD_Base, filha da classe Social_Base, que permite instanciar uma base de Conhecimento Social cuja estratégia de cooperação utilizada é o CSD.

A classe CSD_Base, por sua vez, provê o conhecimento necessário para a utilização do CSD como estratégia de cooperação. Uma lista de planos, proveniente de um conjunto de planos P_i; uma *lista de contratos* que contém um contrato para cada uma das tarefas pertencentes ao plano ativo; uma lista com os contratos contemplados; e o contrato sob apreciação. O processo de cooperação implementado em CSD_Base é gerido pelo método *CSD_Cooperative_Behavior.*

O processo de cooperação é iniciado a partir da requisição de uma meta global g_i. O conjunto de planos referente a g_i, P_i, torna-se ativo, a meta g_i é difundida para os agentes $a_1, a_2, ..., a_n$ pertencentes à lista de agentes e, para cada uma das tarefas $t_1, t_2, ..., t_j$ pertencentes ao plano ótimo da lista de planos armazenada em P_i, um contrato c_i é aberto e armazenado na lista de contratos. Todos os contratos da lista são anunciados aos agentes $a_1, a_2, ..., a_n$. Estes contratos são mantidos abertos até que uma das seguintes situações aconteça:

- uma proposta pr_i, que satisfaça o contrato c_i, tenha sido recebida. O contrato c_i é então contemplado, o agente a_i é informado que sua proposta foi aceita. O contrato c_i é armazenado na lista de contratos contemplados.

- todos os agentes da comunidade $a_1, a_2, ..., a_n$ tenham respondido ao anúncio do contrato com suas respectivas propostas pr_i, e nenhuma delas satisfaz ao contrato ativo c_i, implicando o insucesso do plano ativo. Neste caso, o plano ativo é abortado e a base de conhecimento social irá determinar qual dos planos pertencente a P_i pode ser realizado como sucesso.

O processo de cooperação converge quando todos os contratos $c_1, c_2, ..., c_n$ de um mesmo plano pertencente ao conjunto de planos P_i, apresentarem um agente contratado para assumir a execução da tarefa. Por sua vez, quando todos os planos pertencentes a P_i falham, a convergência não pode ser atingida.

7.3 A Linguagem Parla

A linguagem para comunicação de agentes, **Parla**, foi desenvolvida para ser utilizada pelo Expert-Coop[Cos97], propondo um formato de mensagem alternativo e uma nova organização da estrutura de camadas permitindo que as atribuições, que seriam realizadas pelo *agente especial* na proposta da *KQML*, possam ser reduzidas e estes novos requisitos passem a ser atribuições do suporte de comunicação do *agente.* Este suporte pode ser implementado utilizando uma ferramenta de sistemas distribuídos como *ISIS* [Bir93].

7.3.1 Padrão de Mensagem

O padrão de mensagem adotado, apresentado abaixo, consiste em uma estrutura linear que contém os seguintes atributos :*From, To, Time-Stamp, Body, Grade, Round, Alfa* e *Simtime.* Dentre estes atributos,*From, To, Time-Stamp, Round* e *Body* são obrigatórios. Grade e alfa são facultativos. O Simtime foi adicionado ao formato inicial visando aplicações destinadas à categoria de robôs simulados da RoboCup.

```
((From agent-1)( To agent-2)( Time-Stamp 13) (Simtime 300)
 (Round 13) (Alpha 0.9) (grade 9)
 ( Body ((<primitiva>  <representação de conhecimento>))))
```

- **From** contém o nome do agente, ou processo no caso das mensagens entre processos de um mesmo *agente.*
- **To** armazena o nome do agente ou processo a que se destina a mensagem.
- **Time-Stamp** contém o "time stamp" da mensagem [Lam78].
- **Round** é utilizado para associar a mensagem a um dado processo licitatório em andamento.
- **Body** contém a mensagem propriamente dita.
- **Grade** permite atribuir um valor numérico, por exemplo um número *fuzzy*, a uma proposta.
- **Alfa** permite estabelecer um grau de satisfação para realização da tarefa.
- **Simtime** contém o tempo de simulação em que foi gerada a mensagem enviada pelo Soccer Serever.

7.3.2 Primitivas

As primitivas de comunicação que compõem a linguagem Parla informam ao agente que tipo de informação ou conhecimento contém a mensagem recebida e que ação deve ser tomada em relação à informação ou ao conhecimento em questão. As primitivas suportadas pela linguagem são: *ANNOUNCE, ACCEPT, CONFIRM, INFORM, RECALL, REQUEST, REPLY, PRESENT, PROPOSE, SUBSCRIBE* e *UNSUBSCRIBE.* As primitivas PRESENT, PROPOSE, foram introduzidas para possibilitar a implementação da estratégia de cooperação CBD*.

- **ANNOUNCE:** Utilizada por um *agente* para informar aos demais *agentes da comunidade* a abertura de um contrato. Neste caso o objeto do contrato será representado no formalismo, e passado como parâmetro. O atributo *Alfa* poderá ser utilizado para estabelecer um grau de satisfação mínimo para o contrato e *Round*, obrigatoriamente, conterá o identificador do contrato.

- **ACCEPT:** Utilizada por um ou mais *agentes da comunidade* para enviar uma proposta, aceitando realizar a tarefa anunciada pelo *agente* que abriu o contrato. Essa proposta terá o mesmo *ROUND* da mensagem referente ao *ANNOUCE* e um quantificador numérico *Grade* que expressa o quão bem o agente pode realizar a tarefa ou ação anunciada.

- **INFORM:** Utilizada para enviar uma informação a um processo ou *agente*.

- **CONFIRM:** Utilizada pelo agente responsável pelo contrato ou pelo agente ativo que iniciou a tentativa de coalisão para informar ao agente vencedor do contrato que sua proposta foi a vencedora, nos casos onde a CNP e o CSD são utilizados, e para confirmar uma coalisão no caso onde a estratégia de cooperação adotada é a CBD*. Essa mensagem terá o mesmo *ROUND* da mensagem referente ao anúncio do contrato *(ANNOUCE)*, ou da proposta de coalisão *(PROPOSE)*.

- **RECALL:** Utilizada pelo agente responsável pela licitação para informar aos demais agentes que suas respectivas propostas não obtiveram êxito.

- **REFUSE:** Utilizada por um ou mais agentes da comunidade para enviar uma proposta, rejeitando realizar a tarefa anunciada pelo *agente* que abriu o contrato, ou para rejeitar uma proposta de coalisão. Essa proposta terá o mesmo *ROUND* da mensagem referente anúncio do contrato *(ANNOUCE)* ou da proposta de coalisão *(PROPOSE)*.

- **REQUEST:** Utilizada para solicitar uma informação a um *agente* ou processo.

- **REPLY:** Utilizada para responder uma solicitação (REQUEST).

- **PRESENT:** Utilizada durante o período de apresentação da estratégia de cooperação CBD*, quando os agentes se apresentam e manifestam suas impressões sobre a meta global previamente difundida.

- **PROPOSE:** Utilizada pelo agente ativo na estratégia de cooperação CBD* para propor a um parceiro em potencial uma coalisão.

- **SUBSCRIBE:** Utilizada por um agente que não pertence à comunidade para solicitar sua inclusão. Esta solicitação pode ser feita a qualquer membro da comunidade e no formalismo de representação de conhecimento passado como parâmetro, deve aparecer o nome do agente e seu "host". Uma vez recebida tal solicitação o agente inclui o novo agente na lista de membros e difunde a nova lista de participantes da comunidade.
- **UNSUBSCRIBE:** Utilizado para solicitar a saída da comunidade.

7.3.3 Regras de Comunicação

As primitivas de comunicação da linguagem *Parla* estão sujeitas às seguintes regras:

- As primitivas **ANNOUNCE** e **PROPOSE** devem ser respondidas com **ACCEPT** ou **REFUSE**.
- A primitiva **REQUEST** deve ser respondida com **REPLY**.
- As primitivas **SUBSCRIBE** e **UNSUBSCRIBE** devem invocar uma atualização dos *agentes ativos*.
- As primitivas **INFORM**, **CONFIRM** e **RECALL** são utilizadas para uma comunicação unidirecional.
- A primitiva **PRESENT** deve ser utilizada em resposta à difusão da meta global proposta.

7.4 O Problema Proposto

O problema proposto consiste em uma situação extraída do futebol de robôs, onde uma ação cooperativa precisa ser realizada para atingir um objetivo comum.

Trata-se de um estado, que aqui consideramos como o estado inicial no instante t_0, cujos agentes envolvidos são os jogadores de meio campo e atacantes *7, 8, 9, 10* e *11*. Estes jogadores encontram-se realizando uma formação defensiva, com dois jogadores de meio campo posicionados no campo defensivo *8* e *10* e os outros três, *7, 9* e *11*, nas proximidades do meio campo, posicionados no campo ofensivo (ver figura 7.1).

Nesse instante, a posse da bola é recuperada pelo jogador *8* que abre um processo de cooperação para realizar a meta global g_i *(logic (global_goal description rws_attack_playset))*: realizar uma jogada de linha de fundo pela lateral direita do

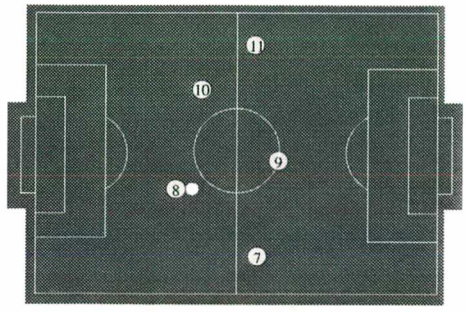

Figura 7.1: Estado inicial para o problema proposto.

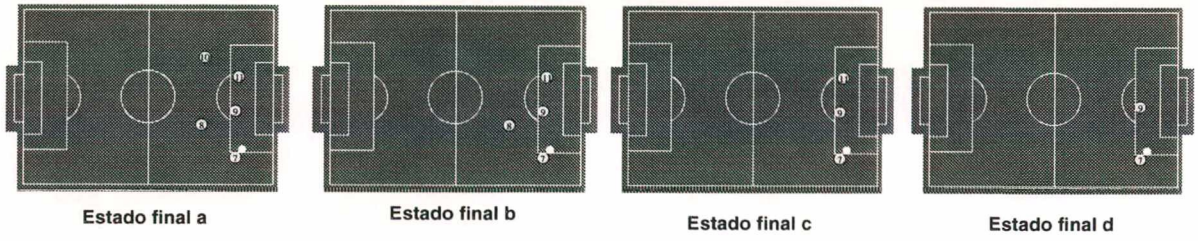

Figura 7.2: Estados finais para o problema proposto.

campo. A meta global g_i pode ser alcançada conduzindo o jogo para um dos quatro estados em $t_0 + \Delta t$, representados abaixo:

- Estado a : uma jogada de linha de fundo envolvendo cinco jogadores.
- Estado b : uma jogada de linha de fundo envolvendo quatro jogadores.
- Estado c : uma jogada de linha de fundo envolvendo três jogadores.
- Estado d : uma jogada de linha de fundo envolvendo dois jogadores.

7.5 CNP_Team, CBD*_Team e CSD_Team

Foram implementados três sistemas multiagentes *CNP_Team*, *CBD*_Team* e *CSD_Team*, um para cada estratégia de cooperação presente na Expert-Coop++ (CNP, CSD e CBD*). Cada um desses sistemas multiagentes é formado por cinco agentes: *UFSC-Team_7*, *UFSC-Team_8*, *UFSC-Team_9*, *UFSC-Team_10*, *UFSC-Team_11*. A implementação dos agente se deu a partir da implementação dos processos que compõem os agentes.

Uma pré-implementação dos processos Interface, Coordinator e Expert, utilizado as classes da biblioteca Expert-Coop++, foi concebida para facilitar a implementação dos agentes. Nesta pré-implementação, são instanciados das classes da Expert-Coop++ os objetos necessários aos níveis decisórios que compõem a arquitetura do agente

autônomo concorrente (apresentado no capítulo 3), *Reativo, Instintivo* e *Cognitivo* nos processos *Interface, Coordinator* e *Expert*, respectivamente.

Cada um dos três processos que compõem o agente é composto por dois "threads", segundo o clássico problema *produtor-consumidor*. O primeiro thread lê as mensagens recebidas pelos *socket's* e as coloca no *mailbox*; o segundo thread lê as mensagens armazenadas no mailbox e as processa implementando o nível decisório relativo ao processo. O *mailbox* e os dois threads já se encontram pré-implementados. Faz-se necessário então editar os arquivos que contêm esta pré-implementação instanciando e inicializando os objetos necessários para compor o processo; definir as regras a serem utilizadas pelos sistemas baseados em conhecimento utilizados nos níveis instintivo e cognitivo; e especificar os planos necessários para alcançar as metas globais.

7.5.1 Processo Interface

O processo Inteface pré-implementado no arquivo já possui, devidamente instanciados, os controladores difusos *Watch_Ball, Move_To_Ball, Kick_To_Goal, Pass_Ball, Move_To_Direction* e *Drive_Ball_Fwd* [RdSHJLdCB00], armazenados em uma lista de controladores difusos *fc_list*. São também instanciados nesta pré-implementação as variáveis lingüísticas de entrada e de saída *input_lv* e *output_lv* e principalmente o objeto responsável pela manipulação destes objetos aqui apresentados e cujos métodos implementam o nível reativo propriamente dito *reactive_level*. De forma sucinta a descrição do comportamento reativo implementado no processo interface é a seguinte:

1. Lê a mensagem armazenada no mailbox;
2. Se a mensagem foi enviada pelo Coordinator, então um dos seguintes métodos podem ser acionados: trocar o controlador ativo, ativar ou desativar a saída do processo.
3. Caso a mensagem seja procedente do Soccer Server, as variáveis lingüísticas são atualizadas, o controlador difuso ativo é acionado, as variáveis linguísticas de saída são defuzificadas e convertidas em comandos a serem enviados para o Soccer Server.

7.5.2 Processo Coordinator

O processo Coordinator pré-implementado já possui devidamente instanciados uma base de fatos *visual_kb*, um conjunto de regras *instintive_base* e um motor de inferência

inference_ engine.

Na base de fatos são armazenadas as informações visuais recebidas do processo Interface, a meta local enviada pelo processo Expert e o nome do comportamento ativo no processo Interface.

O conjunto de regras é lido a partir do arquivo especificado em INSTINTIVE_RULES_FILE que deve ser implementado pelo usúario e que determina o comportamento do processo. As regras armazenadas no arquivo INSTINTIVE_RULES_FILE são utilizadas para: classificar os estados do jogo, gerar a informação simbólica que é utilizada no nível cognitivo e escolher qual comportamento reativo deve ser utilizado no processo Interface, dada a meta local e o estado identificado a partir das informações visuais e auditivas recebidas. Um exemplo de regra utilizada na implementação do sistema multiagente em questão é apresentado a seguir:

```
(rule_001
 (if (frame ( ?xst1  ((goal r) (distance ?x1))))
     (frame ( ?xst1  ((goal r) (direction ?x2))))
     (logic ( player position ?x3 )))
 (filter ( >  ?x1  0.0 )
  ( <  ?x1 22.0 )
  ( >  ?x2 -45.0 )
  ( <  ?x2 45.0 )
          ( != ?x3 attack_goal_area ))
 (then (logic ( player position attack_goal_area ))
       (message ((to Expert) (from Coord) (deadline 0)
          (grade 0.0) (alpha 0.0) (round 0.0)
  (body (INFORM ((logic (player position attack_goal_area )))))))))
```

A regra apresentada identifica se o jogador se encontra na grande área do adversário, informando para o nível cognitivo a posição do jogador.

O motor de inferência é capaz de manipular fatos lógicos, frames contendo as informações visuais e mensagens no formato da linguagem Parla. Durante o processo de inferência, previamente ajustado para apenas um ciclo, são utilizados os fatos da *visual_kb* as mensagens recebidas do Expert e do juiz do jogo e o conjunto de regras carregado a partir do arquivo INSTINTIVE_RULES_FILE. O funcionamento do processo Coordinator pode ser brevemente descrito por:

1. Lê a mensagem armazenada no mailbox.

2. Armazena na base de fatos.

3. Carrega fatos e regras no motor de inferência.

4. Envia mensagens para os processos Interface e/ou Expert.

7.5.3 Processo Expert

O processo Expert pré-implementado já possui devidamente instanciados uma base de fatos locais *local_base*, uma base de conhecimento social *social_base*, um conjunto de regras locais *cognitive_rules*, um conjunto de regras sociais *social_rules*, um motor de inferência *inference_engine* e uma lista conjunto de planos *agent_plan_set_list*.

Na base de fatos locais são armazenadas as informações simbólicas geradas no nível instintivo, as informações enviadas por outros agentes e informações geradas a partir de processos de cooperação. Esta mesma base é utilizada tanto nos processos de inferência locais quanto nos sociais.

A base de conhecimento social instanciada determina o tipo de estratégia de cooperação escolhida para ser utilizada no agente: CNP, CBD* e CSD. Nela estão armazenadas as informações que serão utilizadas nos processo de cooperação bem como os métodos que implementam a estratégia de cooperação utilizada.

As regras locais, armazenadas em *cognitive_rules* e lidas a partir do arquivo COGNITIVE_RULES_FILE, determinam o comportamento cognitivo local do agente. Um exemplo de regra local, que identifica a recuperação da posse da bola pelo UFSC_Team, modifica a meta local para *manter o controle da bola* e solicita a abertura de um processo de cooperação para realzar *uma joada de linha de fundo pela lateral direita* é mostrado a seguir:

```
(rule_003
  (if (logic ( local_goal current get_ball_control ))
      (logic ( player position back_midfild ))
      (logic ( ball control UFSC-Team_8 )))
  (then
   (message
    ((to UFSC-Team_8) (from Expert) (deadline 0) (grade 0.0)
     (alpha 0.0) (round 0.0)
     (body
      (REQUEST ((logic ( global_goal description  rws_attack_play )))))))
   (message
    ((to Coord) (from Expert) (deadline 0) (grade 0.0)
     (alpha 0.0) (round 0.0)
```

```
    (body (INFORM ((logic (local_goal current keep_ball_control )))))))
  (logic ( local_goal current keep_ball_control ))))
```

As regras sociais determinam o comportamento do agente nos processo de cooperação determinando quais contratos ou propostas de coalisão serão aceitos e sob que condições. Um exemplo de regra social, que instrui ao agente quando aceitar a realização da ação de *conduzir a bola pela lateral direita*, implementada para uma base de conhecimento social instanciada a partir da classe CSD_Base é mostrada a seguir:

```
(rule_001
 (if (message ((to UFSC-Team_7) (from ?x1) (deadline ?x2) (grade ?x3)
               (alpha ?x4) (round ?x5)
      (body (ANNOUNCE ((logic (drive_ball_rws agent x_agent)))))))
     (logic ( global_goal description rws_attack_play ))
     (logic ( player position attack_rws_midfild ))
     (logic ( sense_body stamina high ))
     (logic ( player availability free )))
 (then
  (message ((to ?x1) (from UFSC-Team_7) (deadline ?x2) (grade 1.0)
            (alpha 0.0) (round ?x5)
    (body (ACCEPT ((logic (drive_ball_rws agent UFSC-Team_7)))))))))
```

As bases de conhecimento social e local se comunicam através de mensagens enviadas ao mailbox.

O motor de inferência utilizado tanto para as inferências locais quanto sociais é o mesmo. Este mesmo motor de inferência é utilizado no nível instintivo. Entretanto no processo Expert, o motor de inferência é previamente ajustado para admitir múltiplos ciclos de inferência.

A lista de conjunto de planos determina as ações cooperativas que podem ser realizadas pela sociedade de agentes. Cada conjunto de planos P_i possui uma meta global g_i, uma lista de planos $\{p_1, p_2, p_3, ..., p_t\}$ que satisfazem a meta g_i, sendo p_1 o plano ótimo. Cada plano p_i contém uma lista de tarefas $\{t_1, t_2, t_3, ..., t_m\}$ que devem ser alocadas para que o plano p_i seja realizado. Um exemplo de conjunto de planos é mostrado a seguir:

```
/*
*-------------------------------------------------
```

```
* Task Statement
*-------------------------------------------------
*/

task_1 = Task("rightwingside",
               Logic_Pattern ("drive_ball_rws", "agent",
                      "UFSC-Team_7", 0),
        0.8, 5000 );
...

/*
*---------------------------------
*    Plan 1
*---------------------------------
*/

task_list_1.push_back(task_5);
task_list_1.push_back(task_4);
task_list_1.push_back(task_3);
task_list_1.push_back(task_2);
task_list_1.push_back(task_1);
plan_1= Plan( "attack_rws_1",
               Logic_Pattern ("global_goal", "description",
                      "rws_attack_play", 0),
       task_list_1, 5000 );
...

/*
*-------------------------------------
* Adding Global Plans
*-------------------------------------
*/

rws_plan_list.push_back(plan_1);
rws_plan_list.push_back(plan_2);
rws_plan_list.push_back(plan_3);
rws_plan_list.push_back(plan_4);
```

```
rws_planset = Plan_Set("rightwingside",
                        Logic_Pattern("global_goal",  "description",
              "rws_attack_play", 0),
         rws_plan_list );

social_base.Add_Plan_Set(rws_planset);
```

Cabe lembrar que no caso da estratégia de cooperação CSD, os conjuntos de planos dão origem às regras que são utilizadas pela Base de Conhecimento Social Dinâmico, responsável por conduzir o processo de cooperação no caso de não ser possível a premiação direta da Estrutura de Contratos.

7.6 Resultados

Uma vez implementados os sistemas multiagentes *CNP_Team*, *CBD*_Team* e *CSD_Team* estes foram submetidos ao problema proposto em duas situações:

1. Todos os cinco agentes estão aptos a realizar a ação cooperativa, assumindo as ações pertencentes ao plano ótimo.

2. Todos os cinco agentes estão aptos a realizar a ação cooperativa, entretanto apenas os agentes *UFSC-Team_7* e *UFSC-Team_9* estão aptos a assumir a realização da tarefa com o grau de satisfação desejado, viabilizando apenas o plano 4.

Esta duas situações escolhidas caracteriram respectivamente, o caso ótimo e o caso crítico, para realização da meta global g_i, que visa realizar uma jogada de linha de fundo pela lateral esquerdo campo.

Esses dois ensaios foram realizados em uma única máquina(um Atlhon 700 MHz com 196 MBytes de RAM). Foi observado o tempo de processamento de cada mensagem, este tempo sendo obtido através da função *clock()*, da linguagem de programação C. Esta função retorna o tempo utilizado pelo processo desde o momento em que o processo foi iniciado até o instante em que a função foi invocada. O intervalo entre duas chamada da função *clock()* nos dá o número de pulsos do relógio do processador entre as duas chamadas. Dividindo-se pela constante *CLOCKS_PER_SECOUNDS*, tem-se o tempo decorrido em segundos entre as duas chamadas.

Nas tabelas e gráficos apresentados a seguir o instante em que foi iniciado o processamento de cada mensagem é expresso em milisegundos *ms*. O instante *0* (zero)

corresponde ao instante em que foi processada pelo agente a mensagem requisitando o processo de cooperação. Foram observadas as mensagens processadas pelo agente que abre e gerencia o processo de cooperação, UFSC-Team_8.

7.6.1 CNP_Team situação 1

Os resultados para o sistema multiagente *CNP_Team* submetido à situação onde os cinco agentes estão aptos a realizar a ação cooperativa e assumir as ações pertencentes ao plano ótimo é apresentado a seguir.

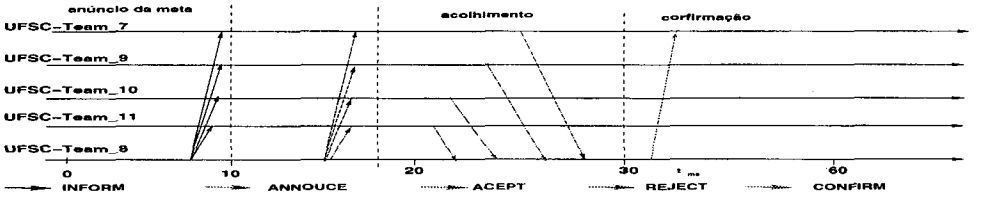

Figura 7.3: Mensagens trocadas no CNP_Team situação 1, difusão da meta, anúncio do contrato identificado pelo round 12, acolhimento de propostas e confirmação.

A figura 7.3 mostra as mensagens trocadas pelos agentes do CNP_Team envolvidos no processo de cooperação para realizar uma jogada de linha de fundo pela lateral direita do campo adversário. As mensagens recebidas pelo agente UFSC-Team_8 são mostradas na tabela 7.1. Estas mensagens correspondem à difusão da meta global, *(logic (global_goal description rws_attack_play))*, ao anúncio do contrato identificado pelo *round 12*, relativo à alocação da tarefa de conduzir a bola pela lateral do campo até a linha de fundo, representada pela representação de conhecimento *(logic (drive_ball_rws agent UFSC-Team_8)))*. Nesse caso apenas o agente *UFSC-Team_7* aceita realizar a tarefa *(ACCEPT ((logic (drive_ball_rws agent UFSC-Team_7))))*. Uma mensagem de confirmação *(CONFIRM ((logic (drive_ball_rws agent UFSC-Team_7))))* é enviada para o agente *UFSC-Team_7* informando-lhe que ele ficará responsável pela realização da tarefa em questão.

Uma vez alocada a primeira tarefa do plano ativo, identificado pelo *round* 12, o agente tenta alocar a tarefa seguinte do plano ativo, anunciando um novo contrato, identificado pelo *round* 13, visando alocar a tarefa de posicionar-se na grande área do adversário para aguardar um cruzamento da ponta direita que deverá ser realizado pelo agente UFSC-Team_7. Essa tarefa á representada por *(ANNOUNCE ((logic (main_area_position agent x_agent))))* . As mensagens relativas ao anúncio do contrato identificado pelo *round* 13 e as propostas recebidas podem ser visualizadas na figura 7.4.

Message Body	Rd	clock
(INFORM ((logic (global_goal description rws_attack_play))))		10
(ANNOUNCE ((logic (drive_ball_rws agent x_agent))))	12	20
(REFUSE ((logic (drive_ball_rws agent UFSC-Team_8))))	12	30
(REFUSE ((logic (drive_ball_rws agent UFSC-Team_11))))	12	30
(REFUSE ((logic (drive_ball_rws agent UFSC-Team_10))))	12	30
(REFUSE ((logic (drive_ball_rws agent UFSC-Team_9))))	12	30
(ACCEPT ((logic (drive_ball_rws agent UFSC-Team_7))))	12	30

Tabela 7.1: UFSC-Team_8 : difusão da meta global, anúncio e recebimento de propostas do contrato identificado pelo round 12.

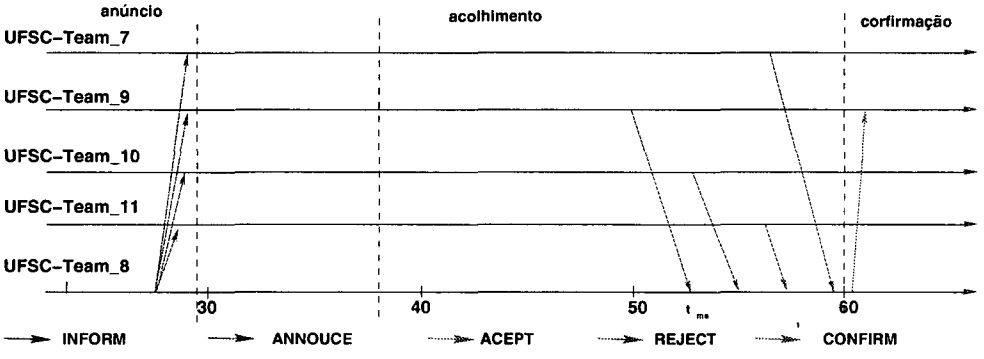

Figura 7.4: Mensagens trocadas no CNP_Team situação 1, anúncio do contrato identifiado pelo *round* 13, acolhimento de propostas e contemplação.

As mensagens recebidas pelo agente UFSC-Team_8, relativas ao anúncio do contrato identificado pelo *round* 13, são apresentadas na tabela 7.2. A realização da tarefa foi aceita pelos agentes UFSC-Team_9 e UFSC-Team_11, em ambos os casos com um grau de safistação acima do específicado. O agente UFSC-Team_9 foi escolhido, pois a mensagem enviada por ele foi processada primeiro, e este recebe a mensagem de confirmação *(CONFIRM ((logic (main_area_position agent UFSC-Team_9))))*.

Finalizado o contrato identificado pelo *round* 13, um novo contrato, *round* 14, é aberto para alocar a tarefa seguinte pertencente ao plano, cuja descrição é a mesma da tarefa identificado pelo *round* 13. As mensagens recebidas pelo agente podem ser visualizadas na tabela 7.3.

Apenas o agente UFSC-Team_11 enviou uma proposta aceitando realizar a tarefa, os demais declinaram. Sendo assim o agente UFSC-Team_11 é contemplado e recebe a mensagem de confirmação *(CONFIRM ((logic (main_area_position agent UFSC-Team_11))))*.

A tarefa seguinte do plano ativo é anunciada, buscando um agente para assumir a posição de cobertura, próximo a grande área *(ANNOUNCE ((logic (cover_position*

Message Body	Rd	clock
(ANNOUNCE ((logic (main_area_position agent x_agent))))	13	30
(ACCEPT ((logic (main_area_position agent UFSC-Team_9))))	13	60
(REFUSE ((logic (main_area_position agent UFSC-Team_10))))	13	60
(ACCEPT ((logic (main_area_position agent UFSC-Team_11))))	13	60
(REFUSE ((logic (main_area_position agent UFSC-Team_7))))	13	60
(REFUSE ((logic (main_area_position agent UFSC-Team_8))))	13	60

Tabela 7.2: UFSC-Team_8 : Difusão da meta global, anúncio e recebimento de propostas do contrato identificado pelo round 13.

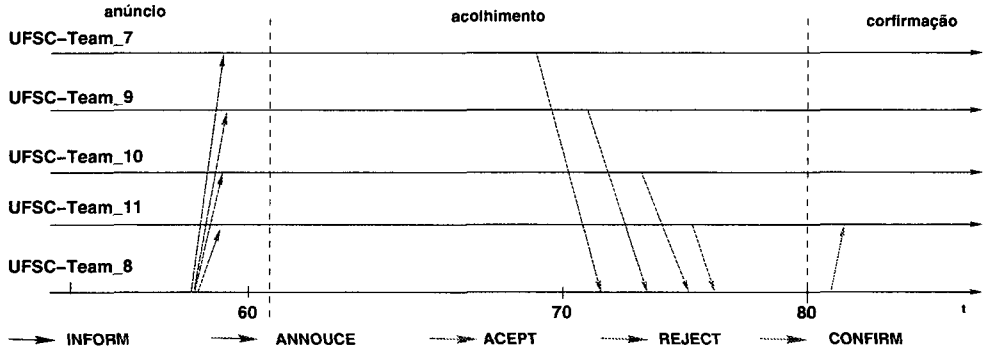

Figura 7.5: Mensagens trocadas no CNP_Team situação 1, anúncio do contrato identificado pelo round 14, acolhimento de propostas e confirmação.

agent x_ agent)))) (ver figura 7.6).

As mensagens recebidas pelo agente *UFSC-Team_ 8* podem ser visualizadas na tabela 7.4. Foram recebidas duas propostas aceitando a realização da tarefa em questão, enviadas pelos agentes UFSC-Team_10 e UFSC-Team_8, sendo o agente UFSC-Team_10 contemplado com a mensagem de confirmação *(CONFIRM ((logic (cover_position agent UFSC-Team_ 10))))*.

A tarefa seguinte possui a mesma descrição da tarefa recém alocada, buscando um segundo agente para assumir também uma posição de cobertura próximo à grande área do adversário. Um novo contrato é anunciado para alocar essa tarefa, identificada pelo *round* 16. A troca de mensagens relativa ao anúncio, acolhimento e confirmação é mostrada na figuara 7.7. A finalização do processo de cooperação também é representado uma vez que todas as tarefas relativas ao plano ativo foram alocadas.

As mensagens recebidas são mostradas em na tabela 7.5. Apenas o agente UFSC-Team_8 manifestou-se favorável à realização da tarefa anunciada. Neste caso, a mensagem *(CONFIRM ((logic (cover_position agent UFSC-Team_ 8))))* aparece na tabela 7.5, por ser o agente *UFSC-Team_ 8* o vencedor do contrato identidicado pelo *Round*

Message Body	Rd	clock
(ANNOUNCE ((logic (main_area_position agent x_agent))))	14	60
(REFUSE ((logic (main_area_position agent UFSC-Team_7))))	14	80
(REFUSE ((logic (main_area_position agent UFSC-Team_9))))	14	80
(REFUSE ((logic (main_area_position agent UFSC-Team_10))))	14	80
(ACCEPT ((logic (main_area_position agent UFSC-Team_11))))	14	80
(REFUSE ((logic (main_area_position agent UFSC-Team_8))))	14	80

Tabela 7.3: UFSC-Team_8 : difusão da meta global, anúncio e recebimento de propostas do contrato identificado pelo round 14.

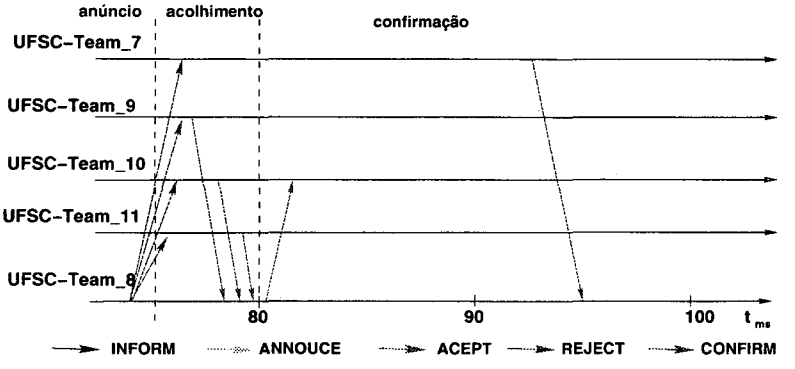

Figura 7.6: Mensagens trocadas no CNP_Team situação 1, anúncio do contrato identificado pelo round 15, acolhimento de propostas e contemplação.

15, e este recebe a mensagem de confirmação. Em se tratando da última tarefa pertencente ao plano ativo uma mensagem informando aos agentes envolvidos a convergência do processo de cooperação é difundida. Essa mensagem inicia a execução do plano multiagente assumido para a realização da meta.

7.6.2 CBD*_Team situação 1

Os resultados para o sistema multiagentes *CBD*_Team* submetido à situação na qual os cinco agentes estão aptos a realizar a ação cooperativa e assumir as ações pertencentes ao plano ótimo são apresentados a seguir. As mensagens trocadas durante os ciclos de difusão da meta global e apresentação estão representados na figura 7.8.

A tabela 7.6, mostra as mensagens recebidas pelo agente UFSC-Team_8. A meta global e uma mensagem enviada por cada um dos agentes da sociedade expressando suas impressões sobre a meta global proposta. Neste caso todos os agentes se manifestaram favoráveis a integrar o processo de cooperação, tornando-se parceiros em potencial para realizar uma coalisão.

Uma vez recebidas as mensagem de apresentação, os agentes que se manifestaram

Message Body	Rd	clock
(ANNOUNCE ((logic (cover_position agent x_agent))))	15	80
(REFUSE ((logic (cover_position agent agent UFSC-Team_9))))	15	80
(ACCEPT ((logic (cover_position agent UFSC-Team_10))))	15	80
(REFUSE ((logic (cover_position agent UFSC-Team_11))))	15	80
(ACCEPT ((logic (cover_position agent UFSC-Team_8))))	15	100
(REFUSE ((logic (cover_position agent UFSC-Team_7))))	15	100

Tabela 7.4: UFSC-Team_8 : difusão da meta global, anúncio e recebimento de propostas do contrato identificado pelo round 15.

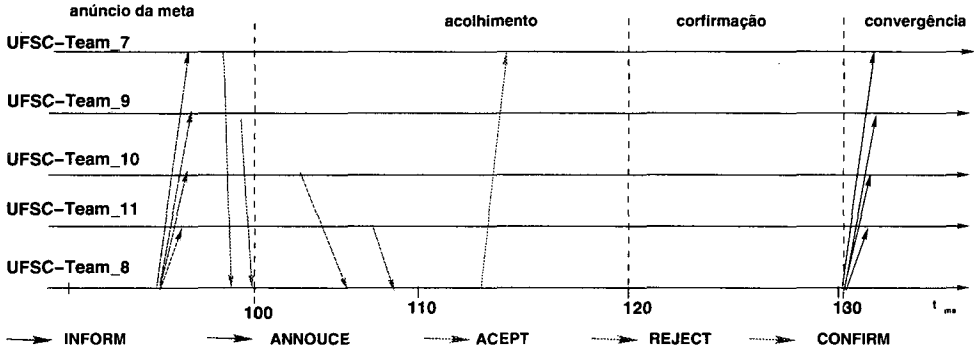

Figura 7.7: Mensagens trocadas no CNP_Team situação 1, anúncio do contrato identifiado pelo round 16, acolhimento de propostas, contemplação e convergência do processo de cooperação.

favoráveis a integrar o processo de cooperação para realizar a meta proposta são incluídos na lista de parceiros em potencial. Os ciclos de negociação têm início visando formar uma coalisão com os parceiros em potencial para a realização das tarefas que compõem o plano ativo. Inicialmente, o agente ativo verifica se o agente preferêncial, indicado pelo plano ativo, foi incluído na lista dos parceiros em potencial. Em caso afirmativo, uma coalisão é proposta a esse agente para realizar a tarefa em questão, caso o agente preferencial não esteja incluído na lista de parceiros em potencial ou recuse a coalisão os demais agentes da lista de parceiros em potencial são consultados sobre a formação da coalisão (ver figura 7.9).

As mensagens recebidas pelo agente ativo, UFSC-Team_8, são mostradas na tabela 7.7. Nesse caso, os agentes indicados pelo plano como preferenciais para a realização das tarefas identificadas pelos *Round's* 12, 13, 14, e 15 aceitaram as propostas de coalisão levando o processo de cooperação à convergência.

Message Body	Rd	clock
(ANNOUNCE ((logic (cover_position agent x_agent))))	16	100
(REFUSE ((logic (cover_position agent agent UFSC-Team_7))))	16	100
(REFUSE ((logic (cover_position agent UFSC-Team_9))))	16	100
(REFUSE ((logic (cover_position agent UFSC-Team_11))))	16	100
(REFUSE ((logic (cover_position agent UFSC-Team_10))))	16	110
(ACCEPT ((logic (cover_position agent UFSC-Team_8))))	16	110
(CONFIRM ((logic (cover_position agent UFSC-Team_8))))	16	120
(INFORM ((logic (global_goal converged rws_attack_play))))		130

Tabela 7.5: UFSC-Team_8 : difusão da meta global, anúncio, recebimento de propostas do contrato identificado pelo round 16 e covergência do processo de cooperação aberto para a meta global proposta.

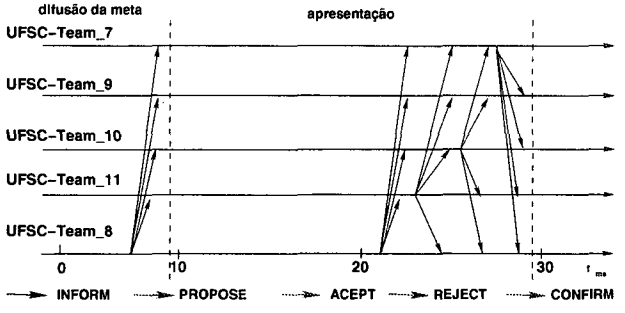

Figura 7.8: Mensagens trocadas no CBD*_Team situação 1 difusão da meta global e apresentação.

7.6.3 CSD_Team situação 1

Os resultados para o sistema multiagentes *CSD_ Team* submetido à situação onde os cinco agentes estão aptos a realizar a ação cooperativa e assumir as ações pertencentes ao plano ótimo é apresentado a seguir. As mensagens trocadas durante os ciclos de difusão da meta global e anúncio dos contratos pertencentes à estrutura de contratos criado para o processo de cooperação em questão são mostradas na figura 7.10.

As mensagens recebidas pelo agente UFSC-Team_8 durante os ciclos de difusão da meta global e anúncio dos contratos são mostradas na tabela 7.8.

Message Body	Rd	clock
(INFORM ((logic (global_goal description rws_attack_play))))		10
(PRESENT ((logic (rws_attack_play join UFSC-Team_11))))		10
(PRESENT ((logic (rws_attack_play join UFSC-Team_10))))		30
(PRESENT ((logic (rws_attack_play join UFSC-Team_9))))		30
(PRESENT ((logic (rws_attack_play join UFSC-Team_7))))		30
(PRESENT ((logic (rws_attack_play join UFSC-Team_8))))		30

Tabela 7.6: UFSC-Team_8 : difusão da meta global, apresentação dos parceiros em potencial para as coalisões.

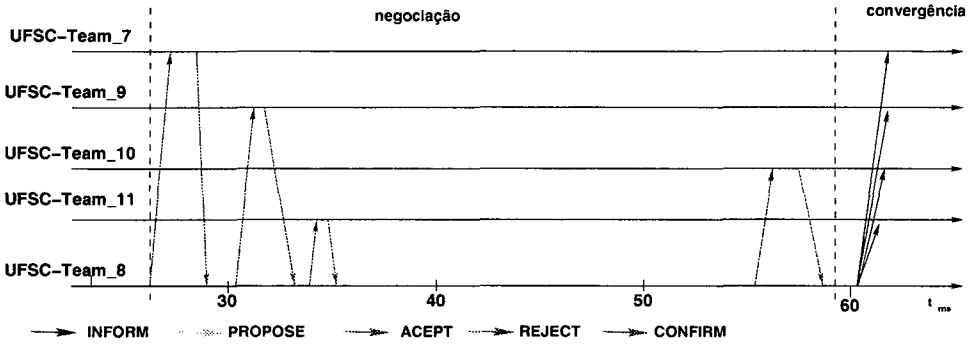

Figura 7.9: Mensagens trocadas no CBD*_Team situação 1 ciclos de negociação .

Observe que apenas três contratos foram abertos para a realização de cinco tarefas. Isso acontece porque duas das três tarefas possuem a mesma descrição, passando a ter *multiplicidade* 2. Ou seja é necessário que dois agentes assumam a execução da mesma tarefa.

As propostas enviadas pelos demais agentes da sociedade, em resposta aos contratos anuciados são mostradas na figura 7.11.

As mensagens recebidas podem ser visualizadas na tabela 7.9. Nesse caso, foram recebidas cinco propostas. Uma enviada pelo UFSC-Team_7, aceitando conduzir a bola pela lateral do campo, duas outras, enviadas pelos agentes UFSC-Team_9 e UFSC-Team_11 aceitando se posicionar na grande área para aguardar o cruzamento e mais duas mensagens aceitando a posição de cobertura da jogada, posicionando-se próximos à grande área, enviadas pelos agentes UFSC-Team_8 e UFSC-Team_10. Estas mensagens possibilitam a confirmação direta da estrutura de contratos levando o processo de cooperação a convergir. São enviadas mensagens de confirmação para os cinco agentes. Foram ainda enviadas outras mensagens declinando a realização das tarefas.

Message Body	Rd	clock
(ACCEPT ((logic (drive_ball_rws agent UFSC-Team_7))))	12	30
(ACCEPT ((logic (main_area_position agent UFSC-Team_9))))	13	40
(ACCEPT ((logic (main_area_position agent UFSC-Team_11))))	14	40
(PROPOSE ((logic (cover_position agent x_agent))))	15	40
(ACCEPT ((logic (cover_position agent UFSC-Team_8))))	15	50
(CONFIRM ((logic (cover_position agent UFSC-Team_8))))	14	50
(ACCEPT ((logic (cover_position agent UFSC-Team_10))))	14	60
(INFORM ((logic (global_goal converged rws_attack_play))))	14	60

Tabela 7.7: UFSC-Team_8 : negociação e convergência.

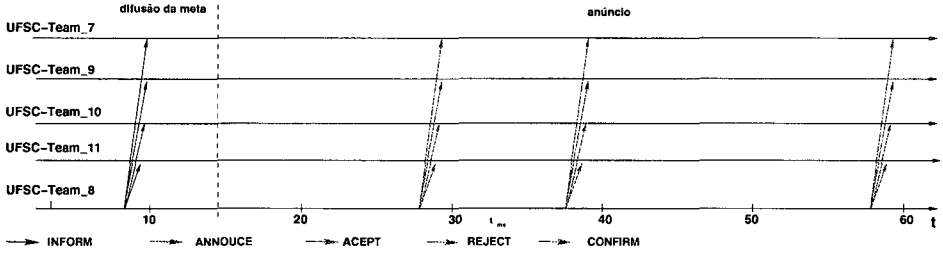

Figura 7.10: Mensagens trocadas durante a abertura do processo de cooperação: difusão da meta e anúncio dos contratos que compõem a estrutura de contratos.

7.6.4 CNP_Team situação 2

Os resultados para o sistema multiagente *CNP_Team* submetido à situação 2, são apresentados a seguir. Nesta situação os cinco agentes estão aptos a realizar a ação cooperativa, entretanto apenas os agentes *UFSC-Team_7* e *UFSC-Team_9* , estão aptos a assumir a realização da tarefa com o grau de satisfação desejado, viabilizando apenas o plano 4.

A figura 7.12 mostra as mensagens trocadas pelos agentes do CNP_Team envolvidos no processo de cooperação para realizar uma jogada de linha de fundo pela lateral direita do campo adversário. As mensagens recebidas pelo agente UFSC-Team_8 são mostradas na tabela 7.10. Estas mensagens correspondem à difusão da meta global, *(logic (global_goal description rws_attack_play))*, ao anúncio do contrato identificado pelo *round 12*, relativo à alocação da tarefa se posicionar no interior da grande área para receber o cruzamento, utilizando a representação de conhecimento *(logic (main_area_position agent x_agent)))*.

Nesse caso apenas o agente *UFSC-Team_11* aceita realizar a tarefa *(ACCEPT ((ogic (main_area_position agent UFSC-Team_11))))*. Uma mensagem de confirmação *(CONFIRM ((logic (main_area_position agent UFSC-Team_11))))* é enviada

Message Body	Rd	clock
(INFORM ((logic (global_goal description rws_attack_play))))		10
(ANNOUNCE ((logic (cover_position agent x_agent))))	12	30
(ANNOUNCE ((logic (main_area_position agent x_agent))))	13	40
(ANNOUNCE ((logic (drive_ball_rws agent x_agent))))	14	60

Tabela 7.8: UFSC-Team_8 : difusão da meta global, anúncio contratos pertencenets ao suporte de contratos.

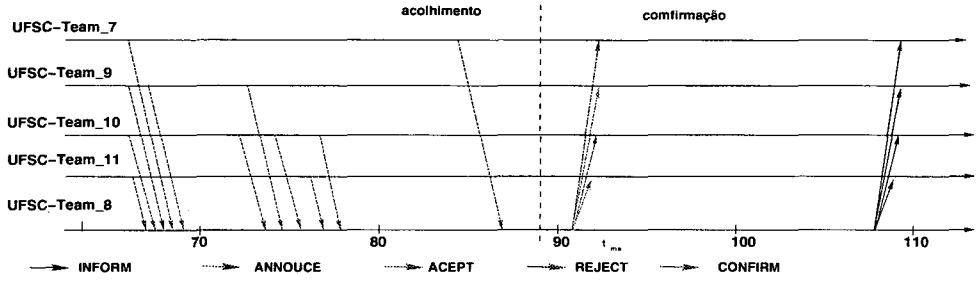

Figura 7.11: recebimento de propostas , contenplação e convergência do processo de cooperação

para o agente *UFSC-Team_11* informando-lhe que ele ficará responsável pela realização da tarefa em questão.

Uma vez alocada a primeira tarefa do plano ativo, identificado pelo *round* 12, o agente tenta alocar a tarefa seguinte do plano ativo, anunciando um novo contrato, identificado pelo *round* 13, visando alocar a tarefa para que o agente assuma uma posição próxima a grande área visando aproveitar um eventual rebote. Essa tarefa á representada por *(ANNOUNCE ((logic (cover_position agent x_agent))))* . As mensagens relativas ao anúncio do contrato identificado pelo *round* 13 e as propostas recebidas podem ser visualizadas na figura 7.13.

As mensagens recebidas pelo agente UFSC-Team_8, relativas ao anúncio do contrato identificado pelo *round* 13, são apresentadas na tabela 7.11. Nesse caso nenhum dos agentes está apto a realizar a tarefa em questão inviabilizando o plano ativo p_1, considerado plano ótimo. Um novo plano p_2, armazenado oriundo do conjunto de planos P_i torna-se ativo. Uma nova lista contratos é criada, sendo um contrato para cada ação pertencente a p_2. O primeiro contrato da nova lista torna-se ativo e um anúcio identificado pelo *Round* 20 é difundido (veja tabela 7.12).

Novamente nenhum dos agente encontra-se apto a realizar a ação anunciada inviabilizando o plano p_2. Um novo plano, p_3 oriundo do conjunto de planos P_i torna-se ativo, Uma nova lista contratos é criada, sendo um contrato para cada ação pertencente

(REFUSE ((logic (cover_position agent UFSC-Team_11)	12	70
(ACCEPT ((logic (cover_position agent UFSC-Team_10))))	12	70
(REFUSE ((logic (cover_position agent UFSC-Team_9))))	12	70
(REFUSE ((logic (cover_position agent UFSC-Team_7))))	12	70
(ACCEPT ((logic (main_area_position agent UFSC-Team_9))))	13	70
(REFUSE ((logic (main_area_position agent UFSC-Team_10))))	13	80
(REFUSE ((logic (main_area_position agent UFSC-Team_10))))	13	80
(ACCEPT ((logic (main_area_position agent UFSC-Team_11))))	13	80
(REFUSE ((logic (drive_ball_rws agent UFSC-Team_9))))	14	80
(ACCEPT ((logic (cover_position agent UFSC-Team_8))))	12	80
(REFUSE ((logic (main_area_position agent UFSC-Team_8))))	14	80
(REFUSE ((logic (drive_ball_rws agent UFSC-Team_10))))	14	80
(ACCEPT ((logic (drive_ball_rws agent UFSC-Team_7))))	14	90
(REFUSE ((logic (main_area_position agent UFSC-Team_8))))	13	90
(REFUSE ((logic (drive_ball_rws agent UFSC-Team_8))))	14	90
(CONFIRM ((logic (cover_position agent UFSC-Team_8))))	12	90
(INFORM ((logic (global_goal converged rws_attack_play))))	12	100

Tabela 7.9: UFSC-Team_8 : recebimento de propostas dos contratos.

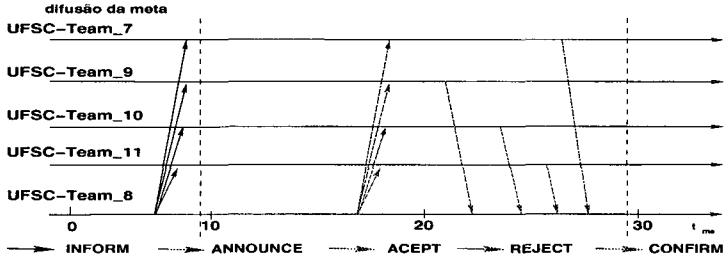

Figura 7.12: Mensagens trocadas no CNP_Team situação 1, difusão da meta, anúncio do contrato identificado pelo round 12, acolhimento de propostas e confirmação.

a p_3. O primeiro contrato da nova lista torna-se ativo e um anúcio identificado pelo *Round* 25 é difundido (veja tabela 7.13).

Mais uma vez nenhum dos agentes encontra-se apto a realizar a ação anunciada inviabilizando o plano p_3. Um novo plano, p_4 oriundo do conjunto de planos P_i torna-se ativo, Uma nova lista contratos é criada, sendo um contrato para cada ação pertencente a p_3. O primeiro contrato da nova lista torna-se ativo e um anúcio identificado pelo *Round* 29 é difundido (veja tabela 7.14).

As mensagens recebidas são mostradas em na tabela 7.14. Apenas o agente UFSC-Team_11 manifestou-se favorável à realização da tarefa anunciada. Neste caso, o vencedor do contrato identidicado pelo *Round* 29, e este recebe a mensagem de confirmação. Uma vez alocada a ação identificada pelo *Round* 29, um novo contrato torna-se ativo

Message Body	Rd	clock
(INFORM ((logic (global_goal description rws_attack_play))))		10
(ANNOUNCE ((logic (main_area_position agent x_agent))))	12	20
(REFUSE ((logic (main_area_position agent UFSC-Team_9))))	12	30
(REFUSE ((logic (main_area_position agent UFSC-Team_10))))	12	30
(ACCEPT ((logic (main_area_position agent UFSC-Team_11))))	12	30
(REFUSE ((logic (main_area_position agent UFSC-Team_7))))	12	30
(REFUSE ((logic (main_area_position agent UFSC-Team_8))))	12	30

Tabela 7.10: UFSC-Team_8 : difusão da meta global, anúncio e recebimento de propostas do contrato identificado pelo round 12.

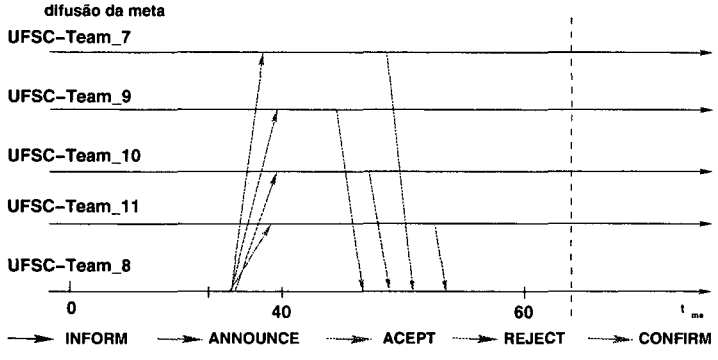

Figura 7.13: Mensagens trocadas no CNP_Team situação 1, anúncio do contrato identifiado pelo *round* 13, acolhimento de propostas e contemplação.

para tentar alocar a última ação do plano ativo p_4 identificada pelo *Round* 30.

As mensagens recebidas pelo agente UFSC-Team_8 podem ser vistas na tabela 7.15. Neste caso o agente UFSC-Team_7 assumiu a realização da ação identificada pelo *Round 30.* Em se tratando da última tarefa pertencente ao plano ativo uma mensagem informando aos agentes envolvidos a convergência do processo de cooperação é difundida. Essa mensagem inicia a execução do plano multiagente assumido para a realização da meta.

7.6.5 CBD*_Team situação 2

A seguir, apresentam-se os resultados para o sistema multiagente *CBD*_Team* submetido à situação onde os cinco agentes estão aptos a realizar a ação cooperativa, entretanto apenas os agentes *UFSC-Team_7* e *UFSC-Team_9* estão aptos a assumir a realização da tarefa com o grau de satisfação desejado, viabilizando apenas o plano 4. Os ciclos de difusão da meta e apresentação repetem os mesmos resultados apresentados na figura 7.8 e na tabela 7.6. Entretanto os ciclos de negociação apresentam resultados

Message Body	Rd	clock
(ANNOUNCE ((logic (cover_position agent x_agent))))	13	30
(ACCEPT ((logic (cover_position agent UFSC-Team_9))))	13	40
(REFUSE ((logic (cover_position agent UFSC-Team_10))))	13	40
(ACCEPT ((logic (cover_position agent UFSC-Team_7))))	13	40
(REFUSE ((logic (cover_position agent UFSC-Team_8))))	13	40
(REFUSE ((logic (cover_position agent UFSC-Team_11))))	13	40

Tabela 7.11: UFSC-Team_8 : Difusão da meta global, anúncio e recebimento de propostas do contrato identificado pelo round 13.

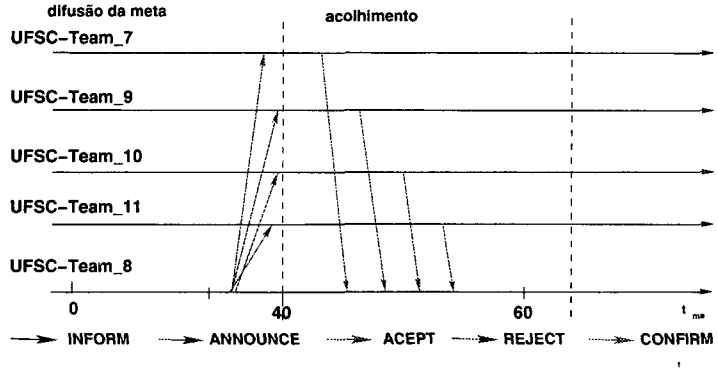

Figura 7.14: Mensagens trocadas no CNP_Team situação 1, anúncio do contrato identificado pelo round 20, acolhimento de propostas e confirmação.

distintos. Inicialmete tenta-se uma coalisão para achar um agente disposto a assumir a posição de cobertura descrita no plano ativo. As tentativas de coalisão são mostradas na figura 7.17.

As mensagens recebidas podem ser vistas na tabela ??. Nesse caso nenhum dos agentes aceitou formar uma coalisão para realizar a tarefa proposta representada segundo a representação de conhecimento: *(PROPOSE ((logic (cover_position agent x_ agent))))*. Isso inviabiliza a execução do plano ativo.

Sendo assim o plano seguinte, pertencente ao mesmo conjunto de planos é tornado ativo e um novo ciclo de negociação é iniciado. As mensagens trocadas para o segundo plano são mostradas na figura 7.18.

As mensagens recebidas, (ver tabela 7.17), mostram que, de forma similar, nenhum dos agentes aceitou formar uma coalisão para realizar a tarefa proposta. Isso inviabiliza mais uma vez o plano ativo.

O plano seguinte, pertencente ao mesmo conjunto de planos torna-se ativo, e um novo ciclo de negociação é iniciado. As mensagens trocadas para o segundo plano são mostradas na figura 7.19.

Message Body	Rd	clock
(ANNOUNCE ((logic (cover_position agent x_agent))))	20	40
(REFUSE ((logic (cover_position agent UFSC-Team_7))))	20	60
(REFUSE ((logic (cover_position agent UFSC-Team_9))))	20	60
(REFUSE ((logic (cover_position agent UFSC-Team_10))))	20	60
(REFUSE ((logic (cover_position agent UFSC-Team_11))))	20	60
(REFUSE ((logic (cover_position agent UFSC-Team_8))))	20	60

Tabela 7.12: UFSC-Team_8 : difusão da meta global, anúncio e recebimento de propostas do contrato identificado pelo round 20.

Message Body	Rd	clock
(ANNOUNCE ((logic (main_area_position agent x_agent))))	25	60
(REFUSE ((logic (main_area_position agent agent UFSC-Team_9))))	15	70
(REFUSE ((logic (main_area_position agent UFSC-Team_10))))	25	70
(REFUSE ((logic (main_area_position agent UFSC-Team_11))))	25	80
(REFUSE ((logic (main_area_position agent UFSC-Team_8))))	25	80
(REFUSE ((logic (main_area_position agent UFSC-Team_7))))	25	80

Tabela 7.13: UFSC-Team_8 : difusão da meta global, anúncio e recebimento de propostas do contrato identificado pelo round 25.

As mensagens recebidas, (veja tabela 7.18), mostram que, de forma similar, nenhum dos agentes aceitou formar uma coalisão para realizar a tarefa proposta. Isso inviabiliza mais uma vez o plano ativo.

O plano seguinte, pertencente ao mesmo conjunto de planos é tornado ativo e um novo ciclo de negociação é iniciado. As mensagens trocadas para o segundo plano são mostradas na figura 7.20.

Pode-se observar na tabela 7.19 que apenas o último agente consultado aceita realizar a tarefa em questão. Uma nova tentativa de coalisão é realizada para alocar a tarefa seguinte do plano ativo(ver figura 7.21).

Novamente apenas o último agente consultado aceita realizar a tarefa em questão. A aceitação da tarefa em questão implica a satisfação do plano ativo e conseqüentemente na convergência do processo de cooperação como mostram as mensagens recebidas (veja tabela 7.20). Uma mensagem é difundida informando que processo de cooperação convergiu e os agentes UFSC-Team_9 e UFSC-Team_7 iniciam a realização do plano acordado.

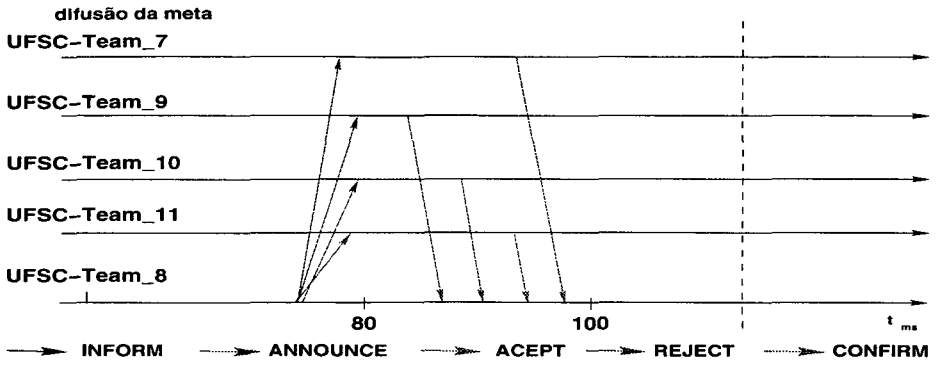

Figura 7.15: Mensagens trocadas no CNP_Team situação 1, anúncio do contrato identifiado pelo round 29, acolhimento de propostas, contemplação e convergência do processo de cooperação.

Message Body	Rd	clock
(ANNOUNCE ((logic (main_area_position agent x_agent))))	29	80
(REFUSE ((logic (main_area_position agent agent UFSC-Team_9))))	29	100
(REFUSE ((logic (main_area_position agent UFSC-Team_10))))	29	100
(ACCEPT ((logic (main_area_position agent UFSC-Team_11))))	29	100
(REFUSE ((logic (main_area_position agent UFSC-Team_7))))	29	100
(REFUSE ((logic (main_area_position agent UFSC-Team_8))))	29	100

Tabela 7.14: UFSC-Team_8 : difusão da meta global, anúncio, recebimento de propostas do contrato identificado pelo round 29 e covergência do processo de cooperação aberto para a meta global proposta.

7.6.6 CSD_Team situação 2

A seguir apresentam-se os resultados para o sistema multiagente CSD_Team submetido à situação onde os cinco agentes estão aptos a realizar a ação cooperativa, entretanto apenas os agentes *UFSC-Team_7* e *UFSC-Team_9* estão aptos a assumir a realização da tarefa com o grau de satisfação desejado, viabilizando apenas o plano 4. Os ciclos de difusão da meta e anúncio dos contratos podem ser visualizados na figura 7.22.

gência

As mensagem recebidas são similares às apresentadas na figura 7.10 e na tabela 7.21. Entretanto as mensagens recebidas diferem da situação apresentada na figura 7.10 e na tabela 7.21. As propostas recebidas pelo agente UFSC-Team_8 durante o ciclo de acolhimento de propostas podem ser vistos na figura 7.23

As propostas recebidas (ver tabela 7.22) não são suficientes para permitir a premiação direta da estrutura de contatos. Sendo assim a estrutura de contratos é utilizada

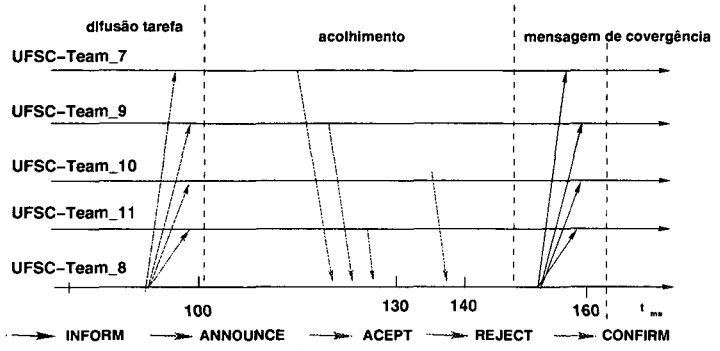

Figura 7.16: Mensagens trocadas no CNP_Team situação 2, anúncio do contrato identifiado pelo round 30, acolhimento de propostas, contemplação e convergência do processo de cooperação.

Message Body	Rd	clock
(ANNOUNCE ((logic (drive_ball_rws agent x_agent))))	30	100
(ACCEPT ((logic (drive_ball_rws agent agent UFSC-Team_7))))	30	130
(REFUSE ((logic (drive_ball_rws agent UFSC-Team_9))))	30	130
(ACCEPT ((logic (drive_ball_rws agent UFSC-Team_10))))	30	130
(REFUSE ((logic (drive_ball_rws agent UFSC-Team_11))))	30	130
(REFUSE ((logic (drive_ball_rws agent UFSC-Team_8))))	30	140
(INFORM ((logic (global_goal converged rws_attack_play))))	30	160

Tabela 7.15: UFSC-Team_8 : difusão da meta global, anúncio, recebimento de propostas do contrato identificado pelo round 29 e covergência do processo de cooperação aberto para a meta global proposta.

para construir a Base de Conhecimento Social Dinâmico *(Base CSD)*.

Uma vez construída a Base CSD o plano 4 é escolhido e os agentes UFSC-Team_7 e UFSC-Team_9 são premiados e assumem a execução deste plano (ver figura 7.24 e tabela 7.23).

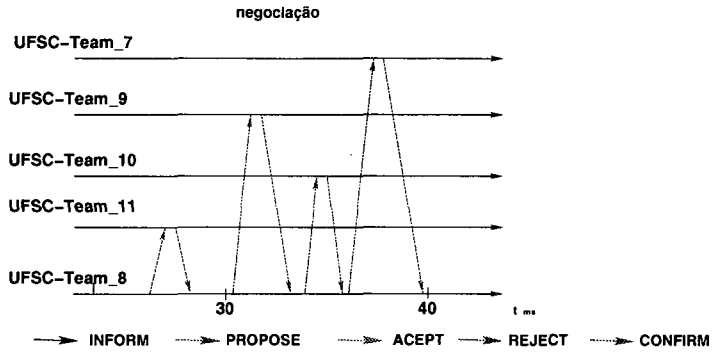

Figura 7.17: Mensagens trocadas no CBD*_Team situação 2 ciclo de negociação em busca de um parceiro pra formar uma coalisão, para a realização da tarefa associada ao round 12.

Message Body	Rd	clock
(REFUSE ((logic (cover_position agent UFSC-Team_11))))	12	30
(PROPOSE ((logic (cover_position agent x_agent))))	12	30
(REFUSE ((logic (cover_position agent UFSC-Team_8))))	12	40
(REFUSE ((logic (cover_position agent UFSC-Team_9))))	12	40
(REFUSE ((logic (cover_position agent UFSC-Team_10))))	12	40
(REFUSE ((logic (cover_position agent UFSC-Team_7))))	12	40

Tabela 7.16: UFSC-Team_8 : negociando a coalisão para a ação identificada pelo round 12.

7.7 Análise Comparativa

A tabela 7.24 apresenta os tempos de convergência para os sistemas multiagente *CNP_Team*, *CBD*_Team* e *CSD_Team* submetidos as situações 1 e 2 descritos na seção 7.6.

Na situação 1, o *CBD*_Team* apresentou o menor tempo de convergência, verificando-se o chamado *melhor caso* formalizado no capítulo 5 subseção 5.3.4. Entretanto os sistemas multiagentes *CNP_Team* e *CSD_Team*, quando expostos a mesma situação 1, não apresentaram suas situações de optimalidade devido ao não determinismo da estratégia de cooperação em questão. Ainda assim o *CSD_Team* apresentou um tempo de convergência menor que *CNP_Team* mas superior ou do *CBD*_Team*.

Na situação 2 o *CSD_Team*, apresentou um tempo de convergência menor que o *CBD*_Team* mesmo não se verificando o chamado *pior caso* formalizado no capítulo 5 subseção 5.3.4. Uma vez que no *round* 36 a quarta tentativa, foi realizada a coalisão.

O montante de mensagens trocadas pelos sistemas multiagentes *CNP_Team*, *CBD*_Team* e *CSD_Team* submetidos as situações 1 e 2 são apresentados na tabela

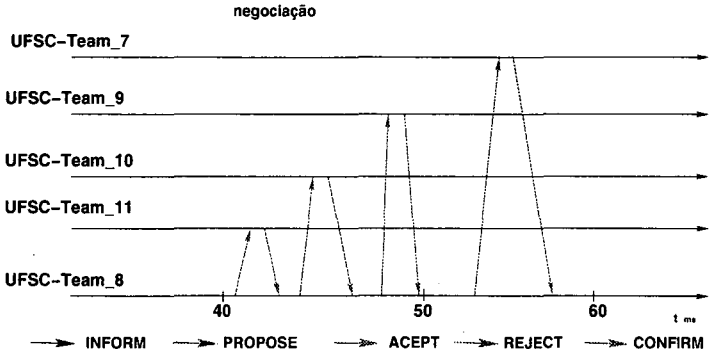

Figura 7.18: Mensagens trocadas no CBD*_Team situação 2 ciclo de negociação em busca de um parceiro para formar uma coalisão, para a realização da tarefa associada ao round 20.

Message Body	Rd	clock
(PROPOUSE ((logic (cover_position agent x_agent))))	20	40
(REFUSE ((logic (cover_position agent UFSC-Team_8))))	20	40
(REFUSE ((logic (cover_position agent UFSC-Team_11))))	20	50
(REFUSE ((logic (cover_position agent UFSC-Team_10))))	20	50
(REFUSE ((logic (cover_position agent UFSC-Team_9))))	20	50
(REFUSE ((logic (cover_position agent UFSC-Team_7))))	20	60

Tabela 7.17: UFSC-Team_8 : negociando a coalisão para a ação identificada pelo round 12.

7.25. O *CNP_Team* apresentou o maior montante de mensagens trocadas durante o processo de cooperação tanto na situação 1 quanto na situação 2. Entretanto, cabe salientar que a situação ótima não se verificou.

Ainda na situação 1, o *CSD_Team* apresentou um montante de mensagens levemente superior ao *CBD*_Team*. Entretanto a situação de optimalidade se verificou para o *CBD*_Team* e não aconteceu para o *CSD_Team* devido ao não determinismo. Em outras palavras, em se verificando a situação de optimalidade para o *CSD_Team* exposto a situação 1, este apresentará um montante de mensagens menor que o *CBD*_Team*. O *CSD_Team* apresentou ainda o menor montante de mensagens trocadas durante o processo de cooperação dos sistemas multiagentes quando submetidos à situação 2 enquanto *CNP_Team* e o *CBD*_Team* paresentaram valores bem superiores.

Os resultados experimentais obtidos a partir da implementação dos sistemas multiagentes CNP_Team, CBD*_Team e CSD_Team, apresentaram o menor tempo de convergência para estratégia de cooperação CBD* na situação em que o plano ótimo pode ser realizado, chamada situação 1. Cabe salientar que nas implementações do

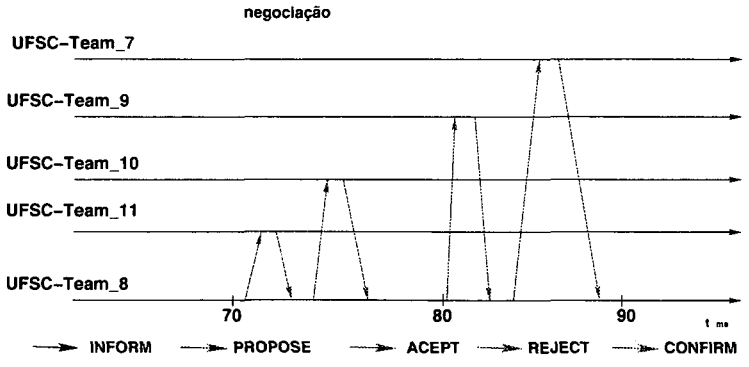

Figura 7.19: Mensagens trocadas no CBD*_Team situação 2 ciclo de negociação em buca de um parceiro pra formar uma coalião, para a realização da tarefa associada ao round 29.

Message Body	Rd	clock
(PROPOUSE ((logic (cover_position agent x_agent))))	29	70
(REFUSE ((logic (cover_position agent UFSC-Team_8))))	29	80
(REFUSE ((logic (cover_position agent UFSC-Team_11))))	29	80
(REFUSE ((logic (cover_position agent UFSC-Team_10))))	29	80
(REFUSE ((logic (cover_position agent UFSC-Team_9))))	29	90
(REFUSE ((logic (cover_position agent UFSC-Team_7))))	29	90

Tabela 7.18: UFSC-Team_8 : negociando a coalisão para a ação identificada pelo round 29.

CNP e do CSD, mesmo sendo possível a execução do plano ótimo não se verificou o chamado melhor caso considerado nos capítulos 5 e 6, onde a primeira proposta recebida satisfaz o contrato, implicando a sua confirmação. Isso acontece devido ao paralelismo não determinístico do processo de cooperação CNP e CSD. Nessa mesma situação o montante de mensagens trocadas pelo CSD e pelo CBD* foram próximos com uma pequena vantagem para o CBD*. Entretanto quando o plano ótimo não está disponível, tanto o tempo de convergência quanto o montante de mensagens trocadas foram menores para a estratégia da cooperação CSD.

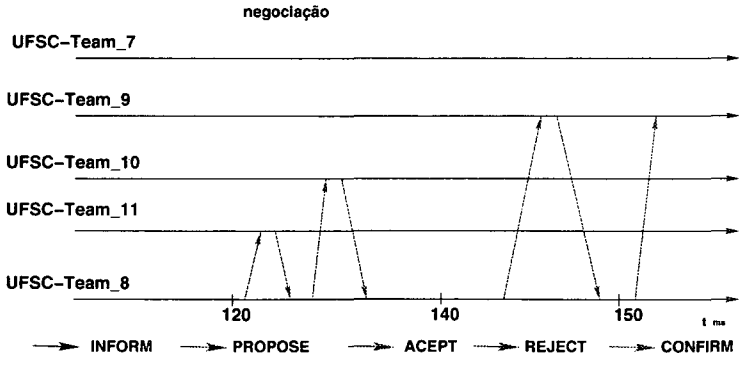

Figura 7.20: Mensagens trocadas no CBD*_Team situação 2 ciclo de negociação em buca de um parceiro pra formar uma coalião, para a realização da tarefa associada ao round 36.

Message Body	Rd	clock
(PROPOUSE ((logic (main_area_position agent x_agent))))	36	90
(REFUSE ((logic (main_area_position agent UFSC-Team_8))))	36	110
(REFUSE ((logic (main_area_position agent UFSC-Team_11))))	36	110
(REFUSE ((logic (main_area_position agent UFSC-Team_10))))	36	110
(ACCEPT ((logic (main_area_position agent UFSC-Team_9))))	36	120

Tabela 7.19: UFSC-Team_8 : negociaçando a coalisão para a ação identificada pelo round 36.

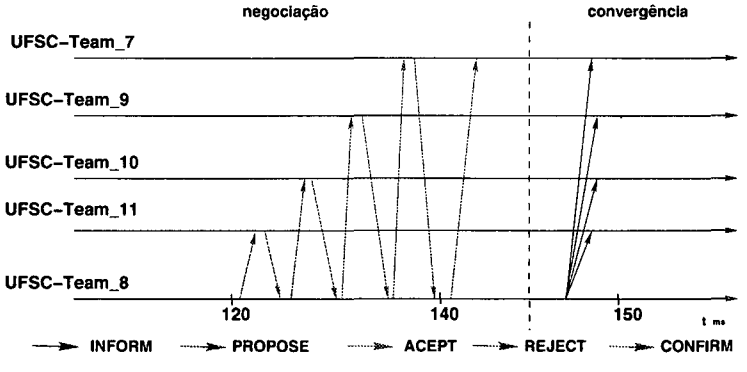

Figura 7.21: Mensagens trocadas no CBD*_Team situação 2 ciclo de negociação em buca de um parceiro pra formar uma coalião, para a realização da tarefa associada ao round 38 e a convergência do processo de cooperação.

Message Body	Rd	clock
(PROPOUSE ((logic (drive_ball_rws agent agent x_agent))))	38	120
(REFUSE ((logic (drive_ball_rws agent agent UFSC-Team_8))))	38	140
(REFUSE ((logic (drive_ball_rws agent agent UFSC-Team_11))))	38	140
(REFUSE ((logic (drive_ball_rws agent agent UFSC-Team_10))))	38	140
(REFUSE ((logic (drive_ball_rws agent agent UFSC-Team_9))))	38	140
(ACCEPT ((logic (drive_ball_rws agent UFSC-Team_7))))	38	140
(INFORM ((logic (global_goal converged rws_attack_play))))	38	150

Tabela 7.20: UFSC-Team_8 : negociando a coalisão para a ação identificada pelo round 38.

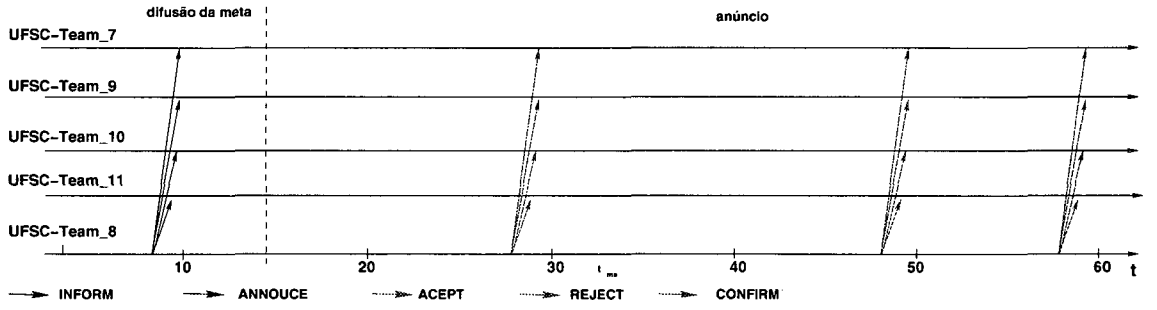

Figura 7.22: Mensagens trocadas durante a abertura do processo de cooperação: difusão da meta e anúncio dos contratos que compõem o suporte de contratos.

Message Body	Rd	clock
(INFORM ((logic (global_goal description rws_attack_play))))		10
(ANNOUNCE ((logic (cover_position agent x_agent))))	12	20
(ANNOUNCE ((logic (main_area_position agent x_agent))))	13	30
(ANNOUNCE ((logic (drive_ball_rws agent x_agent)))	14	50

Tabela 7.21: UFSC-Team_8 : difusão da meta global, anúncio dos contratos identificados pelos *round's* 12, 13 e 14.

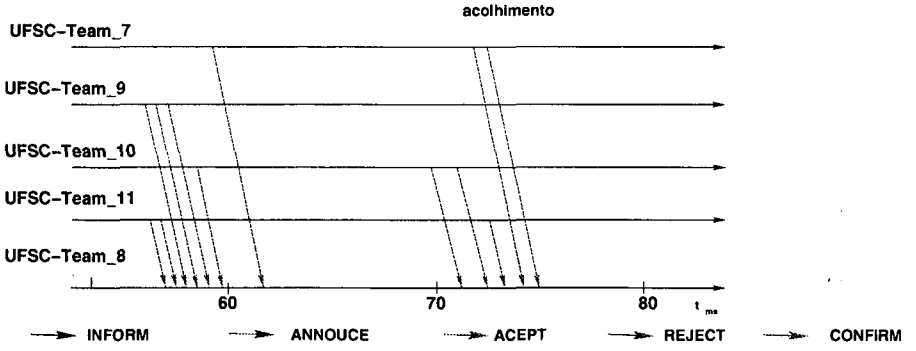

Figura 7.23: Mensagens o processo de cooperação: difusão da meta e anúncio dos contratos que compõem o suporte de contratos.

Message Body	Rd	clock
(REFUSE ((logic (cover_position agent UFSC-Team_11))))	12	60
(ACCEPT ((logic (main_area_position agent UFSC-Team_11))))))	13	60
(REFUSE ((logic (cover_position agent UFSC-Team_9))))	12	60
(ACCEPT ((logic (main_area_position agent UFSC-Team_9))))	13	60
(REFUSE ((logic (drive_ball_rws agent UFSC-Team_9))))	14	60
(REFUSE ((logic (cover_position agent UFSC-Team_10))))	12	60
(REFUSE ((logic (cover_position agent UFSC-Team_7))))	12	70
(REFUSE ((logic (cover_position agent UFSC-Team_8))))	12	70
(REFUSE ((logic (main_area_position agent UFSC-Team_10))))	13	80
(REFUSE ((logic (drive_ball_rws agent UFSC-Team_10))))	14	80
(REFUSE ((logic (drive_ball_rws agent UFSC-Team_11))))	14	80
(REFUSE ((logic (main_area_position agent UFSC-Team_7))))	13	80
(ACCEPT ((logic (drive_ball_rws agent UFSC-Team_7))))	14	80
(REFUSE ((logic (main_area_position agent UFSC-Team_8))))	13	130

Tabela 7.22: UFSC-Team_8 : recebimento de propostas do contrato identificados pelos *round's* 12, 13 e 14.

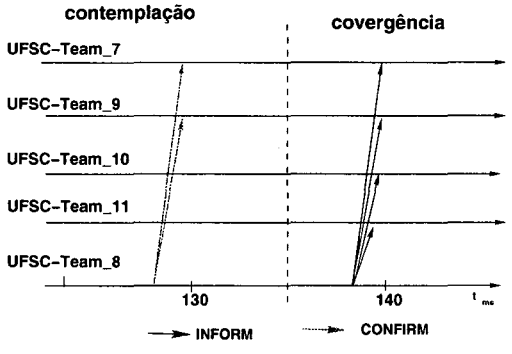

Figura 7.24: Mensagens trocadas durante contemplação e convergência dos processo de cooperação.

Message Body	Rd	clock
(INFORM ((logic (global_goal converged rws_attack_play))))	12	130

Tabela 7.23: UFSC-Team_8 : difusão da meta global, anúncio e recebimento de propostas do contrato identificado pelo round 12.

Estratégia de cooperação	Situação 1	Situação 2
CNP	130	160
CBD*	60	150
CSD	100	130

Tabela 7.24: Tempos de Convergência, em milesegundos (ms) para as estratégias de cooperação CNP, CBD* e CSD.

Estratégia de cooperação	Situação 1	Situação 2
CNP	49	66
CBD*	32	64
CSD	34	34

Tabela 7.25: Montante de mensagem trocadas.

Capítulo 8

Conclusões e Perspectivas

8.1 Conclusões

Um nova estratégia de cooperação para agentes cognitivos submetidos a restrições de tempo real do tipo melhor esforço, chamada Conhecimento Social Dinâmico (CSD), é apresentada nesta tese. Essa estratégia introduz novos conceitos como Estrutura de Contratos e Conjunto de Planos, além de utilizar uma metodologia nova para viabilizar a cooperação em sistemas multiagentes.

Os modelos matemáticos desenvolvidos nos capítulos 5 e 6 apontam para o CSD como sendo um bom equilíbrio entre o número de mensagens trocadas no processo de cooperação e o tempo de convergência. No melhor caso considerado, o CSD pode oferecer um montante menor de mensagens trocadas para atingir a convergência do processo de cooperação e esse montante diminui a medida que aumenta a multiplicidade das sub-tarefas envolvidas no processo de cooperação. No pior caso considerado, o CSD mostrou que é possível utilizar os anúncios de contratos para explorar diferentes possibilidades de realização de uma ação cooperativa, e utilizando um sistema baseado em conhecimento, combinar estas possibilidades para viabilizar a execução de uma ação cooperativa, sem comunicação adicional, reduzindo os ciclos de interação entre os

agentes quando a solução ótima não está disponível e reduzindo também o tempo de convergência.

Os resultados experimentais obtidos a partir da implementação dos sistemas multiagentes CNP_Team, CBD*_Team e CSD_Team, apontam para a estratégia de cooperação Conhecimento Social Dinâmico, que ofereceu um bom equilíbrio entre tempo de convergência do processo de cooperação e o montante de mensagens trocadas pelos agente.

Ainda com relação aos resultados experimentais cabe salientar que as medições foram realizadas pelo relógio do agente UFSC-Team_8, não foi utilizado o mecanismo do relógio lógico o que permitiria um ordenamento global das mensagens devido ao fato desse mecanismo ter como pré-requisito que todos os processos recebem todas as mensagens. Tal fato colocaria o CBD* em evidente desvantagem, uma vez que esta estratégia de cooperação utiliza basicamente comunicação ponto a ponto. O mecanismo de relógios lógicos está implementado na biblioteca Expert-Coop++, respeitado os pré-requisitos estabelecidos para este mecanismo e cabe ao usuário optar pela sua utilização.

8.2 Contribuições

As contribuições trazidas por este trabalho são as seguintes:

- **Agente Autônomo Concorrente:** uma arquitetura de agente cognitivo para aplicações sujeitas a restrições de tempo real, com um processo decisório descentralizado que permite ao agente desempenhar atividades complexas, como estabelecimento de metas, planejamento e participação em processos de cooperação, sem comprometer a resposta em tempo real.

- **Conhecimento Social Dinâmico:** uma estratégia de cooperação para agentes cognitivos que paraleliza a busca por alternativas para a realização de atividades cooperativas e utiliza um sistema baseado em conhecimento para reduzir o tempo de convergência e o montante de mensagens trocadas nos processos de cooperação.

- **Expert-Coop++:** uma biblioteca orientada a objetos, implementada na linguagem de programação C++, que permite o desenvolvimento de sistemas multiagentes utilizando a arquitetura do Agente Autônomo Concorrente e as estratégias de cooperação Conhecimento Social Dinâmico, Contract Net Protocol e Coalisão Baseada em Dependência*.

- **UFSC-Team:** um time de futebol de robôs que vem sendo implementado adotando a arquitetura do Agente Autônomo e desenvolvido a partir da biblioteca

Expert-Coop++ Concorrente

8.3 Perspectivas

- Implementação de sistemas multiagentes cognitivos baseados no Agente Autônomo Concorrente e utilizando a estratégia de cooperação Conhecimento Social Dinâmico, o Contrac Net Protocol ou Coalisão Baseada em Dependência* para aplicações sujeitas a restrições de tempo real do tipo melhor esforço.
- Utilização do Agente Autônomo Concorrente para navegação de veículos autônomos.
- Utilização de algoritmos genéticos para otimização dos Conjuntos de Planos.
- Utilização de aprendizagem de máquina para otimizar os estados reconhecidos pelo nível instintivo e a informação simbólica gerada.
- Implementação de um avaliador de desempenho simbólico para algoritmos genéticos destinado a otimizar os comportamentos reativos ou instintivos.

Bibliografia

[AH88] G. Agha and C. Hewitt. Concurrent programming using actors: exploiting large-scale paralelism. In *Readings in distributed artificial intelligence*, 1988. San Mateo, Morgan Kaufmann Publishers Inc.

[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.

[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.

[BF94] M. Barbuceanu and M. S. Fox. Cool:a language for describimg coordination in multi agent. *EIL working paper*, pages 1–15, 1994.

[Bir93] K. Birman. The process group approach to realiable distributed computing. *Communications of the ACM*, 36(12):37–53, 1993.

[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.

[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.

[Bit98] G. Bittencourt. *Inteligência Artificial : Ferramentas e Teorias.* Editora da Universidade Federal de Santa Catarina, 1998.

[Boi92] O. Boissier. *Le problème du contrôle dans un système de vision integrè.* PhD thesis, Institut National Polytechnique de Grenoble, 1992.

[Bro86] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):435–453, March 1986.

[Car87] E. Cardozo. *DPSK: a kernel for distributed problem solving.* PhD thesis, CAED, Carniege Mellon University, 1987.

[Cas90] C. Castelfranchi. Social power: A point missed in multi-agent, dai and hci. In Y. Demazeuu and J. P. Muller, editors, *Decentralizad A. I.*, 1990. Amsterdam, Elsevier Science Publishers B. V. p. 49-62.

[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*, 1952:176–185, November, from 19 to 22 2000. Spring-Verlag, Lecture Notes in Artificial Inteligence.

[CD90] J. A. Campbell and M. P. D'Inverno. Knowledge interchange protocol. In Y. Demazeuu and J. P. Muller, editors, *Decentralizad A. I.*, 1990. Amsterdam, Elsevier Science Publishers B. V. p. 63-80.

[CL83] D. D. Corkill and V. R. Lesser. The distributed vehicle monitoring testbed: a tool for investigating distributed problem solving network. *AI Magazine*, 4(3):15–33, Fall 1983.

[Cos97] A.L. da Costa. *Expert-Coop: Um Ambiente para Desenvolvimento de Sistemas Multi-Agentes Cognitivos.* Universidade Federal de Santa Catarina, Florianópolis, Brasil, 1997. Dissertação de Mestrado.

[Dem93] Y. Demazeau. La plate-forme paco et ses applications. In Yves Demazeau and Anne Collinot, editors, *Actes de la 2ème Journée Nationale du PRC-GDR Intelligence Artificielle, Montpellier, France*, December 1993.

[DLC89] E.H. Durfee, V.R. Lesser, and D.D. Corkill. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, 1(1):63–83, March 1989.

[DM90] E.H. Durfee and T. A. Montgomery. A hierarchical protocol for coordinating multi-agent behavior. In *Proceedings of AAAI-90*, July 1990.

[DP80] D. Dubois and H. Prade. *Fuzzy Sets and Systems: Theory and Applications.* Academic Press, 1980.

[DR94] E.H. Durfee and J.S. Rosenschein. Distributed problem solving and multi-agent systems: Comparisons and examples. In *Proceedings of the International Workshop on Distributed Artificial Intelligence*, July 1994.

[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.

[Erm80] L. D. Erman. The hearsay-ii speech-understanding system: integrating knowledge to resolve uncertainty. *Computer Surveys*, 12(2):213–253, Jane 1980.

[FG91] J. Ferber and L. Gasser. Intelligence artificielle distribuée, 1991. Tutorial Notes of the 11^th Conference on Expert Systems and their Applications, Avignon'91, France.

[FLM95] T. Finin, Y. Labrou, and J. Mayfield. *KQML as an Agent Communication Language.* MIT Press, Cambridge, 1995.

[Fox81] M. S. Fox. An organizational view of distributed systems. *IEEE Transactions on Systems, Man and Cybernetics*, 11(1):70–80, January 1981.

[Gas87] L. Gasser. Mace: A flexible testbed for distributed ai research. In M. N. Hunhs, editor, *Distributed Artificial Intelligence.* Los Altos, Morgan Kaufmann Publishers Inc., 1987.

[Had96] A. Haddadi. *Communication and Cooperation in Agent Systems: A pragmatic Theory.* Springer, 1996. Lecture Notes in Artificial Inteligence, LNAI 1056.

[Hew91] C.E. Hewitt. Open information system semantics for distributed artificial intelligence. *Artificial Intelligence (Special Volume Foundations of Artificial Intelligence)*, 47(1-3):79–106, January 1991.

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

[IS99] M. Ito and J.S. Sichman. Uma análise comapativa do fluxo de mensagens entre os modelos da rede contractual e coalisões baseadas em dependência. *Anais do XIX Cogresso Nacional da Sociedade Brasileira de Computação*, 4:273–286, julho 1999. ENIA - Encontro Nacional de Inteligência Artificial.

[IS00] M. Ito and J.S. Sichman. Dependence based coalitions and contract net: A comparative analysis. *Intenational Join Conference IBREAMIA'2000 / SBIA'2000*, 1952:106–115, November 2000. Spring-Verlag, Lecture Notes in Artificial Inteligence.

[Ish92] T. Ishida. The tower of babel: towards organization-centered problem solving. In *International Workshop on Distribured Artificial Intelligence*, pages 141–153, 1992. Glen Arbor, Michigan.

[Jen93] N.R. Jennings. Commitments and conventions: the foudations of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.

[JS97] M. Junius and M. Stepple. *CNCL Reference Manual.* Universität Aachen, 1997. http://www.comnets.rwth-aachen.de/cnroot_engl.html.

[JW98] N. R. Jennings and M. J. Wooldridge. *Agent Technology: Foundations, Applications, and Markets (Eds).* Springer Verlag, 1998.

[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.

[KN80] K. KONOLIGE and N.J. NILSSON. Multi-agent planning systems. In *NATIONAL CONFERENCE OF THE AMERICAN ASSOCIATION FOR ARTIFICIAL INTELIGENCE, 1, Stanfor, Proceedings. Menlo Park, AAAI p. 138-141*, March 1980.

[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.

[Lam78] L. Lamport. Time, clocks, and ordering of events. *Communications of ACM*, 21(7):558–565, July 1978.

[Lev90] P. Levi. Architectures of individual and distributed autonomous agents. In T. Kanade, F.C.A Groen, and L.O. Hertzberger, editors, *Intelligent Autonomous Agents 2*, pages 315–324. IAS, Amsterdam, NL, 1990.

[MIN89] N.H. MINSKY. The imposition of protocols over open distributed systems. *Internal Report, Rutgeers University*, LCSR-TR-154:75–94, 1989.

[Pru81] D. G. Pruitt. *Negociation Behavior.* Academic Press, 1981.

[RdSHJLdCB00] L. Rottava da Silva, S.G. Hermesmeyer Jr, A. Loureiro da COSTA, and G. Bittencourt. Implementação de controladores nebulosos em uma equipe de futebol robótico. In *Anais do XIII Congresso Brasileiro de Automática (CBA'2000)*, Florianópolis, SC, Brasil, 11 a 14 de setembro, 2000.

[SDB92] J.S. Sichman, Y. DEMAZEAU, and O. BOISSER. When can knowledge-based systems be called agents? *Anais do IX Seminario Brasileiro de Inteligencia Artificial*, pages 172–185, Outubro 1992. ISSN 0104-6500.

[Sho92] Y. Shoham. Agent oriented programming. *International Workshop on Distributed Artificial Inteligence*, pages 345–353, 1992. 11, Glenn Arbor, Michigan.

[Sic98] J.S. Sichman. Depint: Dependence-based coalition formation in an open multi-agent scenario. *Journal of Artificial Societies and Social Simulation*, 1(2):<http://www.soc.surrey.ac.uk/JASSS/1/2/3.html>, March 1998.

[Sl97] J.S. Sichman and L.O. Álvares. Introdução aos sistemas multiagentes. *XIV JAI - Jornada de Atualização em Informática*, pages 1–37, agosto 1997.

[Smi80] R.G. Smith. The contract net protocol:hifh-level communication and control in a distributed problem solving. *IEEE Transactions on Computers*, 29(12):1104–1113, December 1980.

[SVAB96] E.E. Scalabrin, L. Vandenberghe, H. Azevedo, and Barthes. A generic model of cognitive agent to develop open systems. In *Lecture Notes in Artificial Intelligence - Advances in Artifical Intelligence*, volume 1159, pages 61–70, October 1996. Proceedings of the 13th Brazilian Symposium on Artificial Intelligence.